FAIRINO Collaborative Robot User Manual

Release 3.9.3

FAIR Innovation (Suzhou) Robotic System Co.,Ltd.

Mar 13, 2026

USER MANUAL

1 Collaborative robot **3**

1.1 Product matrix . . . 3
1.2 Quick start . . . 3
1.2.1 Install and power on the robot . . . 3
1.2.1.1 Install the robot arm . . . 3
1.2.1.2 Connect the control box . . . 4
1.2.1.3 Know the button box and end LED . . . 5
1.2.1.3.1 The button box . . . 5
1.2.1.3.1.1 60 Button Box (POE) (BX01) . . . 5
1.2.1.3.1.2 60 Button Box (POE) (BX02)-V1.0 . . . 5
1.2.1.3.1.3 60 Button Box (POE) (BX02)-V2.0 . . . 8
1.2.1.3.2 The end LED . . . 9
1.2.1.4 Power on enable . . . 9
1.2.1.5 Power Off . . . 9
1.2.2 WebApp access login . . . 10
1.2.2.1 Access and log in to the WebApp interface . . . 10
1.2.2.2 Simple understanding of WebApp interface . . . 11
1.2.2.2.1 Control area . . . 11
1.2.2.2.2 Status Bar . . . 13
1.2.3 Robot parameter setting . . . 16
1.2.3.1 Set the installation method . . . 16
1.2.3.2 Set end load . . . 17
1.2.3.3 Set tool coordinates . . . 20
1.2.4 Robot manual teaching . . . 23
1.2.4.1 Manual teaching and recording of teaching points . . . 23
1.2.4.2 View teaching point information . . . 23
1.2.5 Robot Quick Programming . . . 24
1.2.5.1 Introduction to Simple Motion Instructions . . . 24
1.2.5.2 Operate on program files . . . 25
1.2.5.3 Write and run a program . . . 28
1.3 Manual . . . 29
1.3.1 Foreword . . . 29
1.3.1.1 What is installed in the box . . . 29
1.3.1.2 Important security description . . . 30
1.3.1.3 How to use this manual . . . 30
1.3.1.4 Follow the relevant standards . . . 30
1.3.2 Robot brief introduction . . . 30
1.3.2.1 Basic parameters . . . 30
1.3.2.2 Motion . . . 32
1.3.2.3 Robot coordinate system . . . 42

1.3.2.4 Robot Denavit–Hartenberg parameters . . . 42
1.3.2.5 DH parameter download . . . 46
1.3.3 Installation . . . 46
1.3.3.1 Instructions for security . . . 46
1.3.3.1.1 Brief introduction . . . 46
1.3.3.1.2 Personnel safety . . . 47
1.3.3.1.3 Danger recognition . . . 47
1.3.3.1.4 Nameplate information . . . 48
1.3.3.1.5 Effectiveness and responsibility . . . 50
1.3.3.1.6 Limited responsibility . . . 54
1.3.3.1.7 The warning signs in this manual . . . 54
1.3.3.1.8 Pre-use evaluation . . . 54
1.3.3.1.9 Emergency stop . . . 55
1.3.3.1.10 Power-free movement . . . 57
1.3.3.2 Equipment transportation . . . 57
1.3.3.2.1 Transportation . . . 57
1.3.3.2.2 Carry . . . 57
1.3.3.2.3 Storage . . . 58
1.3.3.3 Maintenance, inspection, and scrapping . . . 58
1.3.3.3.1 Maintenance disposal . . . 58
1.3.3.3.2 Inspection Plan . . . 58
1.3.3.3.2.1 Preface . . . 58
1.3.3.3.2.2 Safety Notice . . . 58
1.3.3.3.2.3 Effectiveness and accountability . . . 58
1.3.3.3.2.4 Limited Liability . . . 59
1.3.3.3.2.5 Warning signs in this manual . . . 59
1.3.3.3.3 Digital Input/Output Description of Control Box . . . 60
1.3.3.3.3.1 Precautions When Switching Digital Related Functions of Control Box . . . 60
1.3.3.3.3.2 Digital Input Description of Control Box . . . 60
1.3.3.3.3.3 Summary of Digital Input of Control Box . . . 60
1.3.3.3.3.4 Supported Types of Digital Input of Control Box . . . 62
1.3.3.3.3.5 Wiring Diagram of Digital Input of Control Box . . . 62
1.3.3.3.3.6 Software Settings Related to Digital Input of Control Box . . . 64
1.3.3.3.3.7 Digital Output Description of Control Box . . . 65
1.3.3.3.3.8 Summary of Digital Output of Control Box . . . 65
1.3.3.3.3.9 Supported Types of Digital Output of Control Box . . . 67
1.3.3.3.3.10 Wiring Diagram of Digital Output of Control Box . . . 68
1.3.3.3.3.11 Software Settings Related to Digital Output of Control Box . . . 69
1.3.3.3.3.12 Inspection maintenance plan . . . 70
1.3.3.3.3.13 Robotic arm . . . 70
1.3.3.3.3.14 Control box, teaching device, button box . . . 72
1.3.3.3.3.15 Check program registration card . . . 75
1.3.3.3.4 Robot waste disposal . . . 76
1.3.3.4 Installation specifications . . . 76
1.3.3.4.1 Robot arm installation . . . 76
1.3.3.4.1.1 Installation requirements for FR3/FR3-WMS/FR3-WML/FR3-C robot . . . 77
1.3.3.4.1.2 Installation requirements for FR5 robot . . . 80
1.3.3.4.1.3 Installation requirements for FR10&FR16 robot . . . 82
1.3.3.4.1.4 Installation requirements for FR20&FR30&FR30L robot . . . 83
1.3.3.4.2 Tool end installation . . . 85
1.3.3.4.3 Installation environment . . . 88
1.3.3.4.4 Floor carrier capacity . . . 88

1.3.3.4.5 Load curves for all FR series models . . . 88
1.3.3.4.5.1 Overview . . . 88
1.3.3.4.5.2 Parameter Description . . . 88
1.3.3.4.5.3 FR3 Model Collaborative Robot Load Curve . . . 89
1.3.3.4.5.4 FR3-WMS Model Collaborative Robot Load Curve . . . 89
1.3.3.4.5.5 FR3-WML Model Collaborative Robot Load Curve . . . 90
1.3.3.4.5.6 FR3-C Model Collaborative Robot Load Curve . . . 90
1.3.3.4.5.7 FR5 Model Collaborative Robot Load Curve . . . 92
1.3.3.4.5.8 FR5-WML Model Collaborative Robot Payload Curve . . . 92
1.3.3.4.5.9 FR10 Model Collaborative Robot Load Curve . . . 93
1.3.3.4.5.10 FR16 Model Collaborative Robot Load Curve . . . 93
1.3.3.4.5.11 FR20 Model Collaborative Robot Load Curve . . . 95
1.3.3.4.5.12 FR30 Model Collaborative Robot Load Curve . . . 95
1.3.3.5 Control connection . . . 96
1.3.3.5.1 Controller interface . . . 96
1.3.3.5.2 Controller I/O panel . . . 97
1.3.3.5.3 RJ45 network interface group . . . 99
1.3.3.5.4 End plate . . . 99
1.3.3.5.5 Ground . . . 99
1.3.3.5.6 The common specifications of all digital I/O . . . 102
1.3.3.5.7 Safety I/O . . . 104
1.3.3.5.8 Universal digital amount I/O . . . 106
1.3.3.5.9 Digital input from the button . . . 106
1.3.3.5.10 Interact with other devices or PLC . . . 107
1.3.3.5.11 Simulation I/O . . . 107
1.3.3.5.12 FR3MT&3C Optional Modules . . . 109
1.3.3.5.12.1 Preface . . . 109
1.3.3.5.12.2 Safety Notice . . . 110
1.3.3.5.12.3 Validity and Responsibility . . . 110
1.3.3.5.12.4 Limited Liability . . . 110
1.3.3.5.12.5 Safety Warning Signs . . . 110
1.3.3.5.12.6 FR3MT&3C Base and Module Interface Definitions . . . 111
1.3.3.5.12.7 Base Interface Definitions . . . 111
1.3.3.5.12.8 Power Module Interface Definitions . . . 115
1.3.3.5.12.9 Extension Module Interface Definitions . . . 116
1.3.3.5.12.10 FR3MT&3C Application Scenarios . . . 117
1.3.3.5.12.11 Basic Application . . . 117
1.3.3.5.12.12 Safety Extended . . . 117
1.3.3.5.12.13 Independent Power . . . 117
1.3.3.5.12.14 Full Function . . . 121
1.3.3.5.12.15 Optional Parts List . . . 121
1.3.3.6 Demonstrate and end LED . . . 122
1.3.3.6.1 Introduction to the button box . . . 122
1.3.3.6.1.1 60 Button Box (POE) (BX01) . . . 122
1.3.3.6.1.2 60 Button Box (POE) (BX02)-V1.0 . . . 124
1.3.3.6.1.3 60 Button Box (POE) (BX02) - V2.0 . . . 125
1.3.3.6.2 FR-HMI Teach pendant introduction . . . 126
1.3.3.6.3 End LED definition . . . 126
1.3.4 Quick start . . . 127
1.3.4.1 Install the robot arm and control box . . . 127
1.3.4.2 Demonstrate startup control robot . . . 128
1.3.4.3 Button box control robot movement . . . 128
1.3.4.3.1 Not paired with teaching pendant . . . 128
1.3.4.3.2 Matching the teaching pendant . . . 132

1.3.4.4 Teaching pendant control robot movement . . . 133
1.3.5 Teaching pendant software . . . 134
1.3.5.1 Basic information . . . 134
1.3.5.1.1 Introduction . . . 134
1.3.5.1.2 Robot's first activation . . . 134
1.3.5.1.3 Start software . . . 135
1.3.5.1.4 User login and permission update . . . 135
1.3.5.1.5 Multilingual settings function . . . 136
1.3.5.2 System initial interface . . . 138
1.3.5.2.1 Control area . . . 140
1.3.5.2.2 Status Bar . . . 141
1.3.5.2.3 Menu Bar . . . 145
1.3.5.3 3D simulation robot . . . 145
1.3.5.3.1 3D scene operation bar . . . 145
1.3.5.3.1.1 3D visualization display of robot coordinate system . . . 145
1.3.5.3.1.2 3D virtual trajectory and import tool model . . . 147
1.3.5.3.2 Robot body operation bar . . . 147
1.3.5.3.2.1 TCP . . . 147
1.3.5.3.2.2 Joint Jog . . . 150
1.3.5.3.2.3 Move . . . 152
1.3.5.3.3 Robot supporting function bar . . . 153
1.3.5.3.3.1 Teaching point record . . . 153
1.3.5.3.3.2 I/O . . . 154
1.3.5.3.3.3 TPD (Teach-in programming) . . . 155
1.3.5.3.3.4 Eaxis move . . . 157
1.3.5.3.3.5 FT . . . 158
1.3.5.3.3.6 Telecentric fixed point . . . 159
1.3.5.3.4 Robot and supporting function status bar . . . 159
1.3.5.3.4.1 Robot . . . 159
1.3.5.3.4.2 Program . . . 160
1.3.5.3.4.3 I/O . . . 160
1.3.5.3.4.4 ExAxis . . . 161
1.3.5.3.4.5 Gripper . . . 162
1.3.5.3.4.6 FT . . . 163
1.3.5.3.4.7 Convery . . . 164
1.3.5.3.4.8 Servo . . . 165
1.3.5.3.4.9 Polish . . . 166
1.3.5.3.4.10 Weld . . . 167
1.3.5.3.4.11 Board I/O . . . 168
1.3.6 Base . . . 169
1.3.6.1 Installation . . . 169
1.3.6.1.1 Robot installation method setting and display . . . 169
1.3.6.2 Coordinate . . . 170
1.3.6.2.1 TCP . . . 170
1.3.6.2.2 Ext. TCP . . . 173
1.3.6.2.3 Workpiece . . . 175
1.3.6.3 Payload . . . 177
1.3.6.3.1 End payload . . . 177
1.3.6.4 Joint . . . 180
1.3.6.4.1 Soft limit . . . 180
1.3.6.4.1.1 Joint Soft Limit Protection . . . 181
1.3.6.4.1.2 Overview . . . 181
1.3.6.4.1.3 Joint Soft Limit Protection . . . 182
1.3.6.4.1.4 Soft Limit Configuration and Function Control . . . 182

1.3.6.4.2 Collision level . . . 184
1.3.6.4.2.1 Post-collision response strategy function . . . 184
1.3.6.4.2.2 Collision strategy . . . 185
1.3.6.4.2.3 Gravitational moment mode . . . 185
1.3.6.4.2.4 Vibration response mode . . . 186
1.3.6.4.2.5 Collision rebound mode . . . 186
1.3.6.4.2.6 FT_Guard Command . . . 188
1.3.6.4.2.7 Static collision detection . . . 189
1.3.6.4.2.8 Torque Detection Function Before Dragging . . . 189
1.3.6.4.2.9 Overview . . . 189
1.3.6.4.2.10 Torque Detection Before Dragging . . . 189
1.3.6.4.2.11 False Alarm Detection Function . . . 191
1.3.6.4.2.12 Overview . . . 191
1.3.6.4.2.13 Collision Level Setting . . . 191
1.3.6.4.3 Friction compensation . . . 192
1.3.6.4.4 Friction Compensation Coefficient Adjustment Function . . . 193
1.3.6.4.4.1 Overview . . . 193
1.3.6.4.4.2 Friction Compensation Coefficient Adjustment . . . 193
1.3.6.4.4.3 Friction Compensation Coefficient Configuration . . . 193
1.3.6.4.5 Dragging Force Compensation . . . 194
1.3.6.4.5.1 Overview . . . 194
1.3.6.4.5.2 Robot Dragging Force Optimization . . . 194
1.3.6.4.5.3 Dragging Force Optimization Configuration . . . 194
1.3.6.5 I/O setup . . . 195
1.3.6.5.1 Pausing LUA Program . . . 195
1.3.6.5.2 I/O configuration . . . 196
1.3.6.5.2.1 DI Configuration . . . 197
1.3.6.5.2.2 Adding Configurable Functions to End CI . . . 199
1.3.6.5.2.3 Overview . . . 199
1.3.6.5.2.4 Operation Procedure . . . 199
1.3.6.5.2.5 DO configuration . . . 203
1.3.6.5.3 Control box DO high and low effective configurable function . . . 206
1.3.6.5.3.1 Overview . . . 206
1.3.6.5.3.2 Operation steps . . . 206
1.3.6.5.4 Alias . . . 207
1.3.6.5.5 Filter . . . 208
1.3.6.5.6 Output reset . . . 209
1.3.6.5.7 Pause/Resume DO Reset State Configurable Function . . . 210
1.3.6.5.7.1 Overview . . . 210
1.3.6.5.7.2 Operation Process . . . 210
1.3.6.6 Joint origin . . . 213
1.3.6.7 Photoelectric Sensor TCP Auto-Calibration Function . . . 214
1.3.6.7.1 Overview . . . 214
1.3.6.7.2 Operation Procedure . . . 214
1.3.6.8 TCP Calibration based on flatbed . . . 217
1.3.6.8.1 Overview . . . 217
1.3.6.8.2 2The operation flow of TCP calibration function based on flatbed . . . 218
1.3.6.9 Control Box Analog Feedback Arc Tracking Function . . . 221
1.3.6.9.1 Overview . . . 221
1.3.6.9.2 Control Box Analog AI Configuration Process . . . 222
1.3.6.9.3 Control Box Analog AO Configuration Process . . . 225
1.3.6.10 Linear Rack Guideway Collision Detection . . . 229
1.3.6.10.1 Overview . . . 229
1.3.6.10.2 Linear Rack Guideway Collision Detection Function . . . 229

1.3.6.10.2.1 Linear Rack Guideway Parameter Setting and Enabling 229
1.3.6.10.2.2 Enabling Linear Rack Guideway Collision Detection Function . . 233
1.3.6.11 Force Sensor Loaded Zeroing and Open Posture Compliance Admittance Parameters 236
1.3.6.11.1 Overview 236
1.3.6.11.2 Force Sensor Loaded Zeroing 236
1.3.6.11.3 Open Posture Compliance Admittance Parameters 239
1.3.7 Safety 240
1.3.7.1 Stop mode 240
1.3.7.1.1 Configurable Dual-Channel Safety Stop + Reduced Mode 241
1.3.7.1.1.1 Overview 241
1.3.7.1.1.2 Operation Procedure 241
1.3.7.2 Safe speed 242
1.3.7.3 I/O safety 242
1.3.7.4 Emergency stop 244
1.3.7.4.1 Safety Stop Recovery Optional Auto Enable Function 245
1.3.7.4.1.1 Overview 245
1.3.7.4.1.2 Operation Process 246
1.3.7.5 Protective stop 247
1.3.7.6 Interference zone configuration 247
1.3.7.6.1 Safety Callback Function for Force Sensor-Assisted Dragging into Shaft interference Zone 251
1.3.7.6.1.1 Overview 251
1.3.7.6.1.2 Operation Procedure 252
1.3.7.6.1.3 Joint Limit Ring 252
1.3.7.6.1.4 Shaft interference Configuration 252
1.3.7.6.1.5 Entering Shaft interference Zone with Force Sensor Assistance . . 256
1.3.7.6.1.6 Cuboid Interference Configuration 257
1.3.7.6.1.7 Safety plane 261
1.3.7.6.2 Cube Interference Function 262
1.3.7.6.2.1 Overview 262
1.3.7.6.2.2 Operation Procedure 263
1.3.7.7 Reduction Mode 266
1.3.7.8 Safety plane 267
1.3.7.9 Daemon 270
1.3.7.10 Direction limit (Only used in Linux systems) 271
1.3.7.11 Robot limit (Only used in Linux systems) 271
1.3.7.12 Power detection (Only used in QX systems) 272
1.3.7.13 Motion Configuration 273
1.3.7.13.1 T-Shaped Velocity Optimization + Blending Smoothing Function 273
1.3.7.13.1.1 Overview 273
1.3.7.13.1.2 Operation Process 273
1.3.7.13.1.3 Using Lua Commands 273
1.3.7.13.1.4 Using Motion Configuration Switch 275
1.3.7.13.2 FIR Adaptive Parameter Function + FIR Pause/Resume Function 278
1.3.7.13.2.1 Overview 278
1.3.7.13.2.2 Operation Process 278
1.3.8 Peripherals 281
1.3.8.1 End-Effector Lua Custom Open Protocol 281
1.3.8.1.1 Overview 281
1.3.8.1.2 Operation Steps 281
1.3.8.2 Gripper 286
1.3.8.2.1 Pre-Adapted Devices 286
1.3.8.2.1.1 Gripper Program Teaching 287
1.3.8.2.2 Gripper Lua End-Effector Protocol Configuration 289

1.3.8.2.2.1 Example of a Gripper Peripheral's Lua End-Effector Peripheral Protocol . . . 292
1.3.8.2.2.2 Device Enable . . . 296
1.3.8.2.2.3 Multiple Grippers . . . 296
1.3.8.2.2.4 Rotary Gripper . . . 296
1.3.8.3 Force Sensor . . . 304
1.3.8.3.1 Pre-Adapted Devices . . . 304
1.3.8.3.2 Force Sensor End-Effector Lua Protocol . . . 305
1.3.8.3.3 Welding Handle End Lua Protocol . . . 307
1.3.8.3.3.1 Automatic Generation of End Lua Protocol . . . 307
1.3.8.3.3.2 SmartTool Program Generation Template Import . . . 309
1.3.8.3.3.3 SmartTool Motion Instruction Point Configuration . . . 309
1.3.8.3.3.4 SmartTool Anti-Mistouch Mode . . . 310
1.3.8.3.3.5 Example of Lua End Peripheral Protocol for Welding Handle . . . 310
1.3.8.3.4 Sensor Load Identification . . . 311
1.3.8.3.5 Force Sensor Assisted Dragging . . . 312
1.3.8.3.6 Force/Torque Sensor Collision Detection . . . 315
1.3.8.3.7 Force/Torque Sensor Force Control Motion . . . 316
1.3.8.3.8 Force/Torque Sensor Spiral Insertion . . . 317
1.3.8.3.9 Force/Torque Sensor Rotation Insertion . . . 318
1.3.8.3.10 Force/Torque Sensor Linear Insertion . . . 318
1.3.8.3.11 Force/Torque Sensor Surface Finding . . . 320
1.3.8.3.12 Force/Torque Sensor Center Finding . . . 321
1.3.8.3.13 Custom Open Protocol . . . 321
1.3.8.4 Welding Pendant . . . 322
1.3.8.4.1 Adapted Devices . . . 322
1.3.8.4.1.1 Configuration Steps . . . 322
1.3.8.4.2 Welding Handle End-Effector Lua Protocol . . . 326
1.3.8.4.2.1 Protocol Management . . . 326
1.3.8.4.2.2 Example of Combined Device Lua End-Effector Peripheral Protocol 327
1.3.8.4.2.3 Open Protocol Template . . . 328
1.3.8.4.2.4 Instructions Supported by Open Protocol . . . 330
1.3.8.4.2.5 Configurable Parameters in Open Protocol . . . 331
1.3.8.5 Spray Gun . . . 332
1.3.8.5.1 Spray Gun Peripheral Configuration Steps . . . 332
1.3.8.5.2 Spraying Program Teaching . . . 334
1.3.8.6 Welding Machine . . . 334
1.3.8.6.1 Welding Torch Installation . . . 335
1.3.8.6.2 Welding Machine Parameter Configuration . . . 335
1.3.8.6.2.1 "Controller IO" Welding Control Configuration . . . 335
1.3.8.6.2.2 Welding IO Signal Configuration . . . 336
1.3.8.6.2.3 Welding Process Parameter Configuration . . . 341
1.3.8.6.2.4 Setting the Relationship Diagram Between Welding Current/Voltage and Analog Output . . . 344
1.3.8.6.2.5 Welding Machine Debugging . . . 346
1.3.8.6.2.6 "Digital Communication Protocol (UDP)" Welding Control Configuration . . . 346
1.3.8.6.2.7 Welding IO Signal Configuration . . . 348
1.3.8.6.2.8 Welding Process Parameter Configuration . . . 351
1.3.8.6.2.9 Welding Machine Debugging . . . 353
1.3.8.6.3 Welding Program Writing . . . 353
1.3.8.6.3.1 Writing Programs Using Welding Process Curves . . . 353
1.3.8.6.3.2 Writing Programs Without Using Welding Process Curves . . . 358
1.3.8.6.4 Welding Interruption and Recovery . . . 359

1.3.8.6.4.1 Welding Interruption Recovery Parameter Configuration 361
1.3.8.6.4.2 Welding Interruption Recovery Application 363
1.3.8.6.5 Attachment 1: Robot UDP Communication Protocol 364
1.3.8.6.5.1 Robot Controller -> PLC 364
1.3.8.6.5.2 PLC -> Robot Controller 366
1.3.8.6.6 Digital Communication Protocol (Modbus TCP) 367
1.3.8.6.6.1 Protocol Configuration 368
1.3.8.6.6.2 Welder Open Protocol 368
1.3.8.6.6.3 Welder Open Protocol Example 368
1.3.8.6.6.4 Welder Open Protocol Analysis 370
1.3.8.6.6.5 Welding Machine Open Protocol Upload and Load 372
1.3.8.6.6.6 Welding Machine Debugging 373
1.3.8.6.6.7 Welding Program Writing 373
1.3.8.6.6.8 Welding Machine Open Protocol Unload 378
1.3.8.7 Extended Axis Configuration 379
1.3.8.7.1 Extended Axis Coordinate System 379
1.3.8.7.2 Controller + PLC (UDP Communication) 400
1.3.8.7.2.1 UDP Communication Configuration 401
1.3.8.7.2.2 UDP Extended Axis 402
1.3.8.7.2.3 Extended Axes with Laser Tracking Welding Teaching Program . 411
1.3.8.7.2.4 Positioning Completion Time 413
1.3.8.7.3 Two-DOF Trolley Test 413
1.3.8.7.4 Controller + Servo Drive (485 Communication) 417
1.3.8.7.4.1 Hardware Wiring 417
1.3.8.7.4.2 Communication Configuration 420
1.3.8.7.4.3 Configured Servo Drives 422
1.3.8.7.4.4 Servo Control Mode and Enable 423
1.3.8.7.4.5 Servo Homing 424
1.3.8.7.4.6 Servo Motion 424
1.3.8.7.4.7 Advanced Settings 426
1.3.8.7.4.8 Extended Axis Programming 426
1.3.8.7.4.9 Example Program for Coordinated Motion between Extended Axis and Robot 430
1.3.8.7.4.10 Summary 430
1.3.8.8 Line Laser Sensor 431
1.3.8.8.1 Hardware Connection 431
1.3.8.8.2 Sensor Configuration 431
1.3.8.8.3 Sensor Calibration 433
1.3.8.8.4 Laser Sensor Application 438
1.3.8.8.4.1 Laser Sensor Teaching Points 438
1.3.8.8.4.2 Laser Seam Finding + Tracking 441
1.3.8.8.4.3 Writing Seam Finding Instructions 441
1.3.8.8.4.4 Writing Instruction to Move to Seam Finding Point 449
1.3.8.8.4.5 Writing Laser Tracking Instructions 449
1.3.8.8.4.6 Writing Seam Finding Start Point and Tracking End Point Instructions 453
1.3.8.8.4.7 Writing Welding Related Instructions 455
1.3.8.8.4.8 Laser Trajectory Recording + Trajectory Replay 456
1.3.8.8.4.9 Writing Robot Trajectory Recording Instructions 456
1.3.8.8.4.10 Writing Instruction for Robot to Move to Trajectory Recording Start Point 459
1.3.8.8.4.11 Writing Laser Sensor Trajectory Replay Instructions 459
1.3.8.8.4.12 Writing Welding Related Instructions 459
1.3.8.8.5 Laser Sensor Adaptation Controller Peripheral Open Protocol 461
1.3.8.9 Polishing 461

1.3.8.9.1 Adapted Devices . . . 465
1.3.8.9.2 Peripheral Open Protocol . . . 465
1.3.8.10 Auxiliary Sensors . . . 472
1.3.8.10.1 Adapted Devices . . . 472
1.3.8.11 Combined Devices (SmartTool + Force Sensor Combination) . . . 472
1.3.8.11.1 Adapted Devices . . . 472
1.3.8.11.1.1 FR . . . 475
1.3.8.11.2 Combined Device End-Effector Lua Protocol . . . 479
1.3.8.12 Array Suction Cups . . . 480
1.3.8.12.1 Overview . . . 480
1.3.8.12.1.1 Hardware Description . . . 480
1.3.8.12.2 Function Configuration . . . 481
1.3.8.12.2.1 Unicast Mode Configuration . . . 482
1.3.8.12.2.2 Unicast Mode Protocol Download . . . 484
1.3.8.12.2.3 Broadcast Mode . . . 488
1.3.8.12.3 Array Suction Cup LUA Program Application . . . 491
1.3.8.12.3.1 Suction Cup Control Instruction Addition . . . 492
1.3.8.12.3.2 Unicast Mode Control Instruction Addition . . . 492
1.3.8.12.3.3 Adding Broadcast Mode Control Commands . . . 493
1.3.8.12.3.4 Adding Suction Cup Status Acquisition Command . . . 494
1.3.8.12.3.5 Adding Wait for Suction Cup Adsorption Status Command . . . 495
1.3.8.12.3.6 Application Example . . . 496
1.3.8.13 CNC Function Package Based on FOCAS (For Linux Systems Only) . . . 498
1.3.8.13.1 Overview . . . 498
1.3.8.13.2 Related Operation Instructions . . . 498
1.3.8.13.2.1 Establishing FOCAS Communication . . . 498
1.3.8.13.2.2 Network Configuration . . . 499
1.3.8.13.2.3 Open Protocol File Configuration . . . 499
1.3.8.13.2.4 CNC Status Feedback Description . . . 500
1.3.8.13.2.5 CNC Control Description . . . 504
1.3.8.13.2.6 CNC Teach Program Description . . . 504
1.3.8.14 Virtual Wall Configuration Based on Force Sensor . . . 507
1.3.8.14.1 Force Sensor Installation and Configuration . . . 507
1.3.8.14.2 Virtual Wall Configuration . . . 509
1.3.8.14.3 Force Sensor Assisted Dragging . . . 510
1.3.8.14.4 Six-Axis Force and Joint Impedance Hybrid Dragging Function . . . 519
1.3.8.14.4.1 Overview . . . 519
1.3.8.14.4.2 Force Sensor Installation Configuration and Zeroing Operation . . . 519
1.3.8.14.4.3 Six-Axis Force and Joint Impedance Hybrid Dragging . . . 520
1.3.8.15 Extended Axis with Laser Point Tracking Function . . . 524
1.3.8.15.1 Robot Extended Axis with Laser Point Tracking System Composition . . . 524
1.3.8.15.2 Extended Axis Communication Configuration . . . 524
1.3.8.15.3 Weld Seam Tracking Laser Sensor Connection Configuration . . . 525
1.3.8.15.4 Welder Connection Configuration . . . 525
1.3.8.15.5 Tool Coordinate System and Laser Sensor Coordinate System Calibration 530
1.3.8.15.6 Extended Axis and Laser Point Tracking Function . . . 530
1.3.8.15.6.1 Extended Axis Coordinate System Calibration . . . 530
1.3.8.15.6.2 Synchronized Laser Tracking of Extended Axis and Robot . . . 530
1.3.8.15.6.3 Laser Data with Transformation Method . . . 530
1.3.8.15.6.4 Laser Data without Transformation Method . . . 543
1.3.8.16 Laser Seam Finding Point Position Acquisition Function . . . 544
1.3.8.16.1 Robot Laser Seam Finding Point Position Acquisition System Composition 544
1.3.8.16.2 Laser Sensor Communication Configuration . . . 547
1.3.8.16.3 Laser Seam Finding Point Position Acquisition Function . . . 548

1.3.8.17 DARU DFC Force Control Polishing Head Application . . . 552
1.3.8.17.1 Overview . . . 552
1.3.8.17.1.1 Hardware Description . . . 552
1.3.8.17.2 Function Configuration . . . 553
1.3.8.17.2.1 Pre-adapted Device Configuration . . . 553
1.3.8.17.2.2 Peripheral Open Protocol Download . . . 555
1.3.8.17.3 DFC Polishing Head LUA Program Application . . . 556
1.3.8.17.3.1 Polishing Head Control Instruction Addition . . . 558
1.3.8.17.3.2 Polishing Head Status Acquisition Instruction Addition . . . 558
1.3.8.17.3.3 Application Example . . . 559
1.3.9 Coding . . . 560
1.3.9.1 Introduction . . . 560
1.3.9.2 Tool bar . . . 562
1.3.9.3 Program command . . . 565
1.3.9.4 Logic Command Interface . . . 566
1.3.9.4.1 While command . . . 566
1.3.9.4.2 if…else command . . . 571
1.3.9.4.3 Goto command . . . 572
1.3.9.4.4 Wait command . . . 573
1.3.9.4.5 Pause command . . . 577
1.3.9.4.6 Dofile command . . . 578
1.3.9.4.7 Var command . . . 579
1.3.9.5 Motion command interface . . . 581
1.3.9.5.1 PTP command . . . 581
1.3.9.5.1.1 Point-to-Point Relative Motion . . . 582
1.3.9.5.2 Lin command . . . 584
1.3.9.5.2.1 Linear Relative Motion . . . 585
1.3.9.5.2.2 LIN command joint overspeed processing function . . . 590
1.3.9.5.2.3 Angular velocity adjustable function . . . 596
1.3.9.5.3 Arc command . . . 602
1.3.9.5.4 Circle command . . . 605
1.3.9.5.4.1 Circle command addition . . . 607
1.3.9.5.4.2 Circle trajectory offset . . . 613
1.3.9.5.5 Spiral command . . . 618
1.3.9.5.6 N-Spiral command . . . 619
1.3.9.5.6.1 Uniform Speed Per Helix Circle Setting Function . . . 620
1.3.9.5.6.2 Overview . . . 620
1.3.9.5.6.3 Operation Process . . . 620
1.3.9.5.7 H-Spiral command . . . 622
1.3.9.5.8 Spline command . . . 623
1.3.9.5.9 N-Spline command . . . 623
1.3.9.5.10 Weave command . . . 625
1.3.9.5.10.1 Slope serration weave function . . . 626
1.3.9.5.11 TPD command . . . 629
1.3.9.5.12 Offset command . . . 630
1.3.9.5.13 Servo Command . . . 631
1.3.9.5.13.1 Servo Motion in Cartesian Space . . . 632
1.3.9.5.13.2 Servo Motion in Joint Space . . . 635
1.3.9.5.13.3 Extended Axis Debugging Instructions . . . 637
1.3.9.5.13.4 Overview . . . 637
1.3.9.5.13.5 Operation Process . . . 637
1.3.9.5.14 Trajctory command . . . 639
1.3.9.5.15 TrajctoryJ command . . . 639
1.3.9.5.16 DMP command . . . 640

1.3.9.5.17 WPTrsf command . . . 641
1.3.9.5.18 Tool conversion command . . . 642
1.3.9.6 Control command interface . . . 644
1.3.9.6.1 IO command . . . 644
1.3.9.6.2 AI command . . . 646
1.3.9.6.3 Vir-IO command . . . 648
1.3.9.6.4 Aux-IO command . . . 649
1.3.9.6.5 MoveDO command . . . 650
1.3.9.6.6 MoveAO command . . . 652
1.3.9.6.7 ToolList command . . . 655
1.3.9.6.8 Mode command . . . 656
1.3.9.6.9 Collision command . . . 657
1.3.9.6.10 Acc command . . . 658
1.3.9.7 Peripheral Command Interface . . . 659
1.3.9.7.1 Gripper command . . . 659
1.3.9.7.2 Spray command . . . 660
1.3.9.7.3 EAxis command . . . 661
1.3.9.7.4 Convey command . . . 663
1.3.9.7.5 Polish command . . . 664
1.3.9.8 Welding command interface . . . 665
1.3.9.8.1 Weld command . . . 666
1.3.9.8.2 Segment command . . . 667
1.3.9.8.2.1 Segment welding command added . . . 668
1.3.9.8.2.2 Changes in segment welding motion trajectory and attitude . . . 670
1.3.9.8.2.3 Actual scene of segment welding . . . 679
1.3.9.8.3 Laser command . . . 680
1.3.9.8.3.1 Laser Sensor Fixed-Point Tracking Function . . . 682
1.3.9.8.3.2 Overview . . . 682
1.3.9.8.3.3 Custom Tracking Time Tracking Operation Process . . . 683
1.3.9.8.3.4 IO Trigger Tracking Operation Process . . . 686
1.3.9.8.4 LT-Rec command . . . 689
1.3.9.8.5 W-Search command . . . 690
1.3.9.8.6 Weld-Trc command . . . 691
1.3.9.8.7 Robot Arc Tracking System Composition . . . 693
1.3.9.8.7.1 Welding machine model and setting . . . 696
1.3.9.8.7.2 PLC model and settings . . . 696
1.3.9.8.7.3 Arc tracking function . . . 696
1.3.9.8.8 Adjust command . . . 707
1.3.9.9 Force control command interface . . . 708
1.3.9.9.1 F/T command . . . 709
1.3.9.9.1.1 Force Control Rotation Insertion Optimization Function . . . 709
1.3.9.9.1.2 Overview . . . 709
1.3.9.9.1.3 Operation Process . . . 710
1.3.9.9.2 Torque command . . . 711
1.3.9.10 Visual command interface . . . 712
1.3.9.10.1 3D command . . . 713
1.3.9.11 Palletizing command interface . . . 714
1.3.9.11.1 Pallet command . . . 714
1.3.9.12 Communication command interface . . . 717
1.3.9.12.1 Modbus command . . . 718
1.3.9.12.2 Xmlrpc command . . . 719
1.3.9.13 Auxiliary command interface . . . 721
1.3.9.13.1 Thread command . . . 721
1.3.9.13.2 Function command . . . 722

1.3.9.13.3 PT-Mode command . . . 723
1.3.9.14 Teaching program is not saved for verification . . . 724
1.3.9.15 Teaching program encryption . . . 726
1.3.9.16 Local teaching point . . . 729
1.3.9.17 Current program backup . . . 730
1.3.9.18 Modbus TCP Communication . . . 731
1.3.9.18.1 ModbusTCP master . . . 731
1.3.9.18.1.1 Add ModbusTCP master . . . 732
1.3.9.18.1.2 ModbusTCP master add register . . . 736
1.3.9.18.1.3 ModbusTCP master communication test . . . 739
1.3.9.18.2 Write ModbusTCP master program . . . 743
1.3.9.18.3 ModbusTCP slave . . . 761
1.3.9.18.3.1 ModbusTCP slave communication parameter configuration . . . 762
1.3.9.18.3.2 ModbusTCP slave communication test . . . 765
1.3.9.18.3.3 ModbusTCP slave programming . . . 769
1.3.9.18.3.4 ModbusTCP slave robot state feedback and control . . . 787
1.3.9.19 Robot Backgrounder Function . . . 790
1.3.9.19.1 Robot Backgrounder Function . . . 790
1.3.9.19.1.1 Save the robot background program . . . 790
1.3.9.19.1.2 Robot background program management . . . 792
1.3.9.19.2 Use of the robot user variables . . . 793
1.3.9.19.2.1 Robot user variable management . . . 793
1.3.9.19.2.2 Robot user variable use . . . 793
1.3.9.20 XY horizontal constant . . . 795
1.3.9.20.1 Overview . . . 795
1.3.9.20.2 Operation process of lateral constant force grinding function in XY direction 795
1.3.9.21 Automatic Singularity Avoidance Trajectory . . . 798
1.3.9.21.1 Overview . . . 798
1.3.9.21.2 The trajectory automatically avoids the operation process of the singularity function . . . 799
1.3.9.22 Singularity crossing function in automatic mode . . . 805
1.3.9.22.1 Overview . . . 805
1.3.9.22.2 Operation Flow . . . 805
1.3.9.22.3 Precision Impact Table . . . 808
1.3.9.23 Real-time forward trajectory planning function . . . 810
1.3.9.23.1 Overview . . . 810
1.3.9.23.2 Operation process . . . 810
1.3.9.24 Swing amplitude monotonous gradual arc tracking function . . . 814
1.3.9.24.1 Introduction . . . 815
1.3.9.25 Offset arc tracking function . . . 820
1.3.9.25.1 Sampling bias arc tracking . . . 820
1.3.9.25.2 Percentage bias arc tracking . . . 824
1.3.9.26 Custom collision detection threshold function . . . 827
1.3.9.26.1 Overview . . . 827
1.3.9.26.2 Function setting description . . . 827
1.3.9.26.2.1 Joint detection threshold function setting instructions . . . 828
1.3.9.26.2.2 Parameter setting instructions . . . 828
1.3.9.26.2.3 TCP detection threshold function setting instructions . . . 831
1.3.9.26.2.4 Parameter setting instructions . . . 831
1.3.9.26.2.5 Joint and TCP detection threshold function setting instructions . . . 834
1.3.9.26.2.6 Parameter setting instructions . . . 834
1.3.9.26.2.7 Recommended settings for detection thresholds . . . 838
1.3.9.26.2.8 Joint detection thresholds . . . 838
1.3.9.26.2.9 TCP detection threshold . . . 838

1.3.9.27 T-Shape Velocity Characteristic Optimization + Blending Smoothing Function . . . 838
1.3.9.27.1 Overview . . . 838
1.3.9.27.2 Operation Procedures . . . 839
1.3.9.27.2.1 PTP-PTP Blending . . . 839
1.3.9.27.2.2 Using Lua Instruction Mode . . . 839
1.3.9.27.2.3 Using Motion Configuration Switch Mode . . . 840
1.3.9.27.2.4 PTP-LIN Blending . . . 843
1.3.9.27.2.5 Using Lua Instruction Mode . . . 843
1.3.9.27.2.6 Using Motion Configuration Switch Mode . . . 845
1.3.9.27.2.7 PTP-ARC Blending . . . 848
1.3.9.27.2.8 Using Lua Instruction Mode . . . 848
1.3.9.27.2.9 Using Motion Configuration Switch Mode . . . 850
1.3.9.27.2.10 PTP-CIRCLE Blending . . . 853
1.3.9.27.2.11 Using Lua Instruction Mode . . . 853
1.3.9.27.2.12 Using Motion Configuration Switch Mode . . . 855
1.3.9.27.2.13 LIN-PTP Blending . . . 858
1.3.9.27.2.14 Using Lua Instruction Mode . . . 858
1.3.9.27.2.15 Using Motion Configuration Switch Mode . . . 860
1.3.9.27.2.16 LIN-LIN Blending . . . 863
1.3.9.27.2.17 Using Lua Instruction Mode . . . 863
1.3.9.27.2.18 Using Motion Configuration Switch Mode . . . 865
1.3.9.27.2.19 LIN-ARC Blending . . . 868
1.3.9.27.2.20 Using Lua Instruction Mode . . . 868
1.3.9.27.2.21 Using Motion Configuration Switch Mode . . . 870
1.3.9.27.2.22 LIN-CIRCLE Blending . . . 873
1.3.9.27.2.23 Using Lua Instruction Mode . . . 873
1.3.9.27.2.24 Using Motion Configuration Switch Mode . . . 875
1.3.9.27.2.25 ARC-PTP Blending . . . 878
1.3.9.27.2.26 Using Lua Instruction Mode . . . 878
1.3.9.27.2.27 Using Motion Configuration Switch Mode . . . 880
1.3.9.27.2.28 ARC-LIN Blending . . . 883
1.3.9.27.2.29 Using Lua Instruction Mode . . . 883
1.3.9.27.2.30 Using Motion Configuration Switch Mode . . . 885
1.3.9.27.2.31 ARC-ARC Blending . . . 888
1.3.9.27.2.32 Using Lua Instruction Mode . . . 888
1.3.9.27.2.33 Using Motion Configuration Switch Mode . . . 890
1.3.9.27.2.34 ARC-CIRCLE Blending . . . 893
1.3.9.27.2.35 Using Lua Instruction Mode . . . 893
1.3.9.27.2.36 Using Motion Configuration Switch Mode . . . 895
1.3.9.27.2.37 CIRCLE-PTP Blending . . . 898
1.3.9.27.2.38 Using Lua Instruction Mode . . . 898
1.3.9.27.2.39 Using Motion Configuration Switch Mode . . . 900
1.3.9.27.2.40 CIRCLE-LIN Blending . . . 903
1.3.9.27.2.41 Using Lua Instruction Mode . . . 903
1.3.9.27.2.42 Using Motion Configuration Switch Mode . . . 905
1.3.9.27.2.43 CIRCLE-ARC Blending . . . 908
1.3.9.27.2.44 Using Lua Instruction Mode . . . 908
1.3.9.27.2.45 Using Motion Configuration Switch Mode . . . 910
1.3.9.27.2.46 CIRCLE-CIRCLE Blending . . . 913
1.3.9.27.2.47 Using Lua Instruction Mode . . . 913
1.3.9.27.2.48 Using Motion Configuration Switch Mode . . . 915
1.3.9.27.2.49 Blending for Extended Axis Asynchronous Motion . . . 918
1.3.9.27.2.50 Using Lua Command Method . . . 918
1.3.9.27.2.51 Using Motion Configuration Switch . . . 920

1.3.9.27.2.52 Blending for Extended Axis Synchronous Motion . . . 923
1.3.9.27.2.53 Using Lua Command Method . . . 923
1.3.9.27.2.54 Using Motion Configuration Switch . . . 926
1.3.9.28 Swing Tilt Angle Function . . . 928
1.3.9.28.1 Overview . . . 928
1.3.9.28.2 Operation Procedure . . . 928
1.3.9.29 Welding process parameter gradient function (current, voltage, travel speed along weld seam) . . . 935
1.3.9.29.1 Overview . . . 935
1.3.9.29.2 Current/Voltage Parameter Gradual Change Process . . . 935
1.3.9.29.2.1 Current Parameter Gradual Change . . . 935
1.3.9.29.2.2 Voltage Parameter Gradual Change . . . 939
1.3.9.29.3 Travel Speed Gradual Change Process . . . 942
1.3.9.30 Robot ModbusRTU Communication . . . 947
1.3.9.30.1 Overview . . . 947
1.3.9.30.2 Robot ModbusRTU Master Operation Instructions . . . 948
1.3.9.30.2.1 Adding a ModbusRTU Master . . . 948
1.3.9.30.2.2 Adding Registers to ModbusRTU Master . . . 951
1.3.9.30.2.3 ModbusRTU Master Communication Test . . . 952
1.3.9.30.2.4 Master Read Register Test . . . 953
1.3.9.30.2.5 Master Write Register Test . . . 954
1.3.9.30.2.6 Writing a ModbusRTU Master Program . . . 956
1.3.9.30.2.7 Write Single Coil . . . 958
1.3.9.30.2.8 Write Multiple Coils . . . 959
1.3.9.30.2.9 Read Coils and Discrete Inputs . . . 960
1.3.9.30.2.10 Read Holding Registers and Input Registers . . . 962
1.3.9.30.3 Robot ModbusRTU Slave Operation Instructions . . . 964
1.3.9.30.3.1 ModbusRTU Slave Communication Parameter Configuration . . . 964
1.3.9.30.3.2 ModbusRTU Slave Communication Test . . . 966
1.3.9.30.3.3 General-Purpose Digital Inputs (Coils) . . . 966
1.3.9.30.3.4 General-Purpose Digital Outputs (Discrete Inputs) . . . 966
1.3.9.30.3.5 Analog Inputs (Holding Registers) . . . 967
1.3.9.30.3.6 Analog Outputs (Input Registers) . . . 968
1.3.9.30.3.7 Writing a ModbusRTU Slave Program . . . 969
1.3.9.30.3.8 Write Single Digital Output DO (Discrete Input) . . . 972
1.3.9.30.3.9 Write Multiple Digital Outputs DO (Discrete Inputs) . . . 974
1.3.9.30.3.10 Read Single Digital Output DO (Discrete Input) . . . 975
1.3.9.30.3.11 Read Multiple Digital Outputs DO (Discrete Inputs) . . . 976
1.3.9.30.3.12 Read Digital Input DI (Coils) . . . 977
1.3.9.30.3.13 Read/Write Operations for Analog Output AO (Input Registers) and Analog Input AI (Holding Registers) . . . 979
1.3.9.30.3.14 Wait for Digital Input . . . 982
1.3.9.30.3.15 Wait for Analog Input . . . 983
1.3.9.30.3.16 Robot Status Feedback and Control via ModbusRTU Slave . . . 986
1.3.9.31 Protection Based on 6-Axis Force Sensor Posture Compliance Function . . . 992
1.3.9.31.1 Overview . . . 992
1.3.9.31.2 Operation Procedure . . . 992
1.3.9.32 Socket Communication Interface Function . . . 995
1.3.9.32.1 Socket Configuration . . . 995
1.3.9.32.1.1 Communication Parameter Settings . . . 997
1.3.9.32.1.2 Heartbeat Detection Mechanism . . . 997
1.3.9.32.1.3 Reconnection Mechanism . . . 997
1.3.9.32.1.4 Custom Protocol Parsing . . . 997
1.3.9.32.2 Socket Connection . . . 999

1.3.9.32.2.1 Interface Connection Display . . . 999
1.3.9.32.2.2 Connection Command Module . . . 1002
1.3.9.32.3 Socket Communication . . . 1003
1.3.9.32.3.1 Communication Testing . . . 1003
1.3.9.32.3.2 Communication Command Module . . . 1005
1.3.9.33 Impedance Control Function During Robot Motion . . . 1006
1.3.9.33.1 Overview . . . 1006
1.3.9.33.2 Impedance Control Function . . . 1007
1.3.9.33.2.1 Impedance Control Configuration and Function Start/Stop in Cartesian Space . . . 1007
1.3.9.34 Custom Weaving Welding Function . . . 1012
1.3.9.34.1 Overview . . . 1012
1.3.9.34.2 Operational Procedure for Custom Weaving Welding Function . . . 1012
1.3.9.35 Teach point configuration . . . 1017
1.3.9.35.1 End-point dot automatic overwrite update Lua program function . . . 1017
1.3.9.35.1.1 End-point dot configuration . . . 1017
1.3.9.35.1.2 End-button dot automatic update Lua program . . . 1018
1.3.9.35.1.3 Function usage example . . . 1019
1.3.9.36 Main program configuration . . . 1021
1.3.9.37 Robot Extended Axis Intersecting Line Welding . . . 1023
1.3.9.37.1 System Configuration . . . 1023
1.3.9.37.2 Extended Axis Communication Configuration . . . 1023
1.3.9.37.3 Welder Connection Configuration . . . 1025
1.3.9.37.4 Tool Coordinate System Calibration . . . 1026
1.3.9.37.5 Intersecting Line Welding Function . . . 1029
1.3.9.37.5.1 Extended Axis Coordinate System Calibration . . . 1029
1.3.9.37.5.2 Intersecting Line Trajectory Welding . . . 1031
1.3.9.37.5.3 Method Without Using a Positioner . . . 1032
1.3.9.37.5.4 Method Using a Two-Degree-of-Freedom L-Type Positioner . . . 1035
1.3.10 Graphical programming . . . 1038
1.3.10.1 Introduction . . . 1038
1.3.10.2 Logic Graphical Programming Commands . . . 1039
1.3.10.2.1 If/Else Judgment Instruction . . . 1039
1.3.10.2.2 While Instruction . . . 1040
1.3.10.2.3 Jump instruction . . . 1040
1.3.10.3 Variable class graphical programming command . . . 1041
1.3.10.3.1 Variable instruction . . . 1041
1.3.10.4 Function class graphical programming command . . . 1042
1.3.10.4.1 Function method instruction . . . 1042
1.3.10.5 Motion graphic programming commands . . . 1043
1.3.10.5.1 Point-to-point instruction . . . 1043
1.3.10.5.2 Straight line instruction . . . 1044
1.3.10.5.3 Straight line (adjustable angular velocity at transition point) instruction . . . 1045
1.3.10.5.4 Straight line (seamPos) command . . . 1046
1.3.10.5.5 Arc command . . . 1046
1.3.10.5.6 Full circle command . . . 1047
1.3.10.5.7 Spiral instruction . . . 1048
1.3.10.5.8 New spiral instruction . . . 1049
1.3.10.5.9 Horizontal spiral instruction . . . 1050
1.3.10.5.10 Spline instruction . . . 1051
1.3.10.5.11 New spline instruction . . . 1051
1.3.10.5.12 Swing Instruction . . . 1052
1.3.10.5.13 Point offset instruction . . . 1053
1.3.10.5.14 Servo command . . . 1053

1.3.10.5.15 Trajectory command . . . 1054
1.3.10.5.16 TrajectoryJ command . . . 1055
1.3.10.5.17 Track reproduction command . . . 1056
1.3.10.5.18 DMP Command . . . 1056
1.3.10.5.19 Tool conversion instruction . . . 1057
1.3.10.5.20 Workpiece conversion instruction . . . 1058
1.3.10.6 Control class graphical programming commands . . . 1058
1.3.10.6.1 Wait Command . . . 1059
1.3.10.6.2 Mode switching command . . . 1061
1.3.10.6.3 Pause instruction . . . 1062
1.3.10.6.4 Coordinate system command . . . 1062
1.3.10.6.5 Simulation AI instruction . . . 1063
1.3.10.6.6 Digital IO Instructions . . . 1065
1.3.10.6.7 Motion DO instruction . . . 1066
1.3.10.6.8 Motion AO instruction . . . 1067
1.3.10.6.9 Collision level command . . . 1068
1.3.10.6.10 Acceleration command . . . 1068
1.3.10.7 Peripheral graphical programming commands . . . 1069
1.3.10.7.1 Gripper commands . . . 1070
1.3.10.7.2 Spray gun instruction . . . 1072
1.3.10.7.3 Extended axis instruction (controller+PLC) . . . 1073
1.3.10.7.4 Extended axis instruction (controller+servo drive) . . . 1076
1.3.10.7.5 Conveyor belt command . . . 1080
1.3.10.7.6 Polishing instruction . . . 1081
1.3.10.8 Welding Graphical Programming Commands . . . 1087
1.3.10.8.1 Segment welding instructions . . . 1087
1.3.10.8.2 Welding instruction . . . 1089
1.3.10.8.3 Laser tracking command . . . 1091
1.3.10.8.4 Laser recording instruction . . . 1096
1.3.10.8.5 Welding wire positioning instruction . . . 1097
1.3.10.8.6 Arc tracking instruction . . . 1100
1.3.10.8.7 Posture adjustment instruction . . . 1101
1.3.10.9 Force control graphical programming commands . . . 1102
1.3.10.9.1 Force control commands . . . 1103
1.3.10.9.2 Torque recording instruction . . . 1107
1.3.10.10 Communication graphic programming commands . . . 1109
1.3.10.10.1 Modbus Commands . . . 1110
1.3.10.11 Advanced Graphical Programming Commands . . . 1118
1.3.10.11.1 Folding Instructions . . . 1118
1.3.10.11.2 Call subroutine instruction . . . 1119
1.3.10.11.3 Auxiliary thread instruction . . . 1119
1.3.10.11.4 Point table instruction . . . 1120
1.3.10.11.5 Focus follow instruction . . . 1121
1.3.10.12 Graphical programming command usage example . . . 1121
1.3.10.12.1 Modularization of graphical programming code blocks . . . 1122
1.3.10.12.2 Graphical programming same name overwrite . . . 1124
1.3.10.12.3 Graphical programming program not saved verification . . . 1125
1.3.11 Node graph programming . . . 1126
1.3.11.1 Basic information . . . 1126
1.3.11.1.1 Introduction . . . 1126
1.3.11.1.2 Toolbar . . . 1126
1.3.11.2 Node graph operations . . . 1128
1.3.11.2.1 Node program . . . 1128
1.3.11.3 If/Else instruction . . . 1129

1.3.11.4 While instruction . . . 1130
1.3.11.5 Goto instruction . . . 1130
1.3.11.6 Wait instruction . . . 1131
1.3.11.7 Pause instruction . . . 1133
1.3.11.8 Dofile instruction . . . 1134
1.3.11.9 Set system variable command . . . 1135
1.3.11.10 PTP instruction . . . 1136
1.3.11.11 LIN instruction . . . 1137
1.3.11.12 LIN(seamPos) instruction . . . 1137
1.3.11.13 ARC instruction . . . 1138
1.3.11.14 Circle instruction . . . 1139
1.3.11.15 Spiral instruction . . . 1140
1.3.11.16 N-Spiral instruction . . . 1141
1.3.11.17 H-Spiral instruction . . . 1142
1.3.11.18 Spline instruction . . . 1142
1.3.11.19 N-Spline instruction . . . 1143
1.3.11.20 Weave instruction . . . 1144
1.3.11.21 TPD instruction . . . 1145
1.3.11.22 Offset instruction . . . 1145
1.3.11.23 ServoCart instruction . . . 1146
1.3.11.24 Trajectory instruction . . . 1147
1.3.11.25 TrajectoryJ instruction . . . 1148
1.3.11.26 DMP instruction . . . 1148
1.3.11.27 WPSTrsf instruction . . . 1149
1.3.11.28 ToolTrst instruction . . . 1150
1.3.11.29 Digital IO instruction node . . . 1150
1.3.11.30 Simulate AI commands . . . 1152
1.3.11.31 Virtual IO command node . . . 1153
1.3.11.32 Extended IO command node . . . 1154
1.3.11.33 MoveDO instruction . . . 1156
1.3.11.34 ToolList instruction . . . 1156
1.3.11.35 Mode instruction . . . 1157
1.3.11.36 Collision instruction . . . 1158
1.3.11.37 Acc instruction . . . 1159
1.3.11.38 Gripper instruction . . . 1159
1.3.11.39 Spray instruction . . . 1161
1.3.11.40 Extended axis instructions (controller + PLC) . . . 1163
1.3.11.41 Extended axis instructions (controller + servo drive) . . . 1166
1.3.11.42 Conveyor belt instruction . . . 1169
1.3.11.43 Polish instruction . . . 1171
1.3.11.44 Weld command . . . 1175
1.3.11.45 Segment instruction . . . 1178
1.3.11.46 Laser instruction . . . 1179
1.3.11.47 Laser recording instructions . . . 1184
1.3.11.48 W-Search instruction . . . 1185
1.3.11.49 Weld-Trc instruction . . . 1188
1.3.11.50 Attitude adjustment instructions . . . 1189
1.3.11.51 F/T Instruction . . . 1190
1.3.11.52 Torque recording command . . . 1195
1.3.11.53 Modbus instruction . . . 1197
1.3.11.54 Application scenario usage examples . . . 1203
1.3.12 Teaching Points . . . 1205
1.3.13 Status . . . 1210
1.3.13.1 System log . . . 1210

1.3.13.2 Status Query . . . 1210
1.3.13.2.1 Function Usage . . . 1210
1.3.13.2.2 Chart Export . . . 1213
1.3.13.2.3 Data View Display . . . 1214
1.3.13.2.4 Data Filtering . . . 1215
1.3.14 Applications . . . 1216
1.3.14.1 Robot Packing . . . 1216
1.3.14.2 Data Backup . . . 1218
1.3.14.2.1 Backup Package Integrity Verification Function . . . 1218
1.3.14.2.1.1 Backup Package MD5 Verification Function . . . 1219
1.3.14.2.1.2 Backup Package Key Parameter Verification Function . . . 1219
1.3.14.2.1.3 Failure/Power Failure Exception Rollback Function . . . 1220
1.3.14.3 10s Data Recording . . . 1221
1.3.14.4 End-Effector LED . . . 1221
1.3.14.5 Drag Locking . . . 1222
1.3.14.5.1 Normal Collision Protection Triggering Under Force Sensor Assisted Dragging . . . 1223
1.3.14.5.1.1 Overview . . . 1223
1.3.14.5.1.2 Collision Protection . . . 1224
1.3.14.5.1.3 Parameters Calibration of Joint Torque Sensor on the Whole Machine 1227
1.3.14.5.1.4 Overview . . . 1227
1.3.14.5.1.5 Parameter Calibration . . . 1228
1.3.14.5.1.6 External Force Estimation and Torque Compensation Based on Momentum Observer . . . 1229
1.3.14.5.1.7 Overview . . . 1229
1.3.14.5.1.8 Operation Process . . . 1229
1.3.14.5.1.9 Assisted Drag Optimization Function Based on Joint Torque Sensor 1231
1.3.14.5.1.10 Overview . . . 1231
1.3.14.5.1.11 Assisted Drag Optimization Function Based on Joint Torque Sensor 1231
1.3.14.5.1.12 Zero Point Calibration and Sensitivity Calibration . . . 1231
1.3.14.5.1.13 Assisted Drag Function - Mode One . . . 1233
1.3.14.5.1.14 Assisted Drag Function - Mode Two . . . 1233
1.3.14.5.1.15 Assisted Drag Function - Mode Three . . . 1235
1.3.14.6 Intersection Point Generation (Laser Point Capture Motion) . . . 1235
1.3.14.6.1 Laser Point Capture Motion Function Operation Process . . . 1235
1.3.14.6.2 Finding Intersection Coordinates Using Three-Point and Four-Point Seam Finding . . . 1237
1.3.14.6.2.1 Command-based Intersection Calculation . . . 1237
1.3.14.6.2.2 Three-Point Intersection Calculation . . . 1237
1.3.14.6.2.3 Four-Point Intersection Calculation . . . 1240
1.3.14.6.2.4 Lua Script for Intersection Seam Finding Motion . . . 1241
1.3.14.6.2.5 Three-Point Intersection Calculation . . . 1241
1.3.14.6.2.6 Four-Point Intersection Calculation . . . 1248
1.3.14.7 Peripheral Protocol . . . 1250
1.3.14.8 G-code to Robot Trajectory Planning Function . . . 1252
1.3.14.8.1 Function Overview . . . 1252
1.3.14.8.2 Operation Process . . . 1252
1.3.15 Process Package . . . 1258
1.3.15.1 Welding Expert Library . . . 1258
1.3.15.1.1 Straight Welding . . . 1258
1.3.15.1.2 Arc Welding . . . 1265
1.3.15.1.3 Multi-Layer Multi-Pass Welding . . . 1268
1.3.15.1.4 Posture Adjustment . . . 1278
1.3.15.1.4.1 Posture Adaptive Configuration Steps . . . 1278

1.3.15.1.4.2 Posture Adaptive Combined with Extended Axis and Laser Tracking Welding Teaching Program . . . 1281
1.3.15.2 Palletizing System Configuration . . . 1281
1.3.15.2.1 Palletizing System Configuration Steps . . . 1281
1.3.15.3 Conveyor Tracking . . . 1288
1.3.15.3.1 Conveyor Tracking Configuration Steps . . . 1288
1.3.15.3.2 Conveyor Tracking Teaching Program . . . 1295
1.3.15.3.3 Robot Conveyor Tracking System Composition . . . 1295
1.3.15.3.3.1 Conveyor Encoder Data Communication Connection Method . . . 1295
1.3.15.3.3.2 Conveyor Configuration . . . 1296
1.3.15.3.3.3 Tracking Coordinate System Configuration . . . 1296
1.3.15.3.3.4 Conveyor Tracking Chase Motion Function . . . 1297
1.3.15.3.3.5 Conveyor Tracking Chase Motion Function Introduction . . . 1297
1.3.15.3.3.6 Chase Motion Program Teaching . . . 1298
1.3.16 System . . . 1305
1.3.16.1 General settings . . . 1305
1.3.16.1.1 Network settings . . . 1305
1.3.16.1.1.1 No-Login Operation . . . 1306
1.3.16.1.1.2 Functional Overview . . . 1306
1.3.16.1.1.3 Instructions for Use . . . 1306
1.3.16.1.2 Teach pendant touch screen calibration . . . 1307
1.3.16.1.3 Peripheral industrial computer configuration . . . 1308
1.3.16.1.4 System Language . . . 1308
1.3.16.1.4.1 Language Import . . . 1308
1.3.16.1.4.2 Language Export . . . 1309
1.3.16.1.4.3 Language Application . . . 1309
1.3.16.1.4.4 System safe mode recovery . . . 1310
1.3.16.1.5 Fault data . . . 1311
1.3.16.1.6 Timeout Logout Time Setting . . . 1311
1.3.16.1.7 System Settings . . . 1311
1.3.16.2 Account settings account settings . . . 1312
1.3.16.2.1 User Management . . . 1312
1.3.16.2.2 Authority management . . . 1314
1.3.16.2.3 Import/Export . . . 1319
1.3.16.3 About . . . 1320
1.3.16.3.1 Software Upgrade . . . 1320
1.3.16.3.1.1 Operation Preparation . . . 1320
1.3.16.3.1.2 Important Notes . . . 1320
1.3.16.3.1.3 Operation Steps . . . 1321
1.3.16.4 Custom information . . . 1324
1.3.16.4.1 Robot Model . . . 1324
1.3.16.4.2 Parameter range configuration . . . 1324
1.3.16.4.3 Robot License Usage Time . . . 1325
1.3.16.5 Robot model configuration . . . 1328
1.3.17 Teach pendant . . . 1329
1.3.17.1 Teach pendant activation . . . 1329
1.3.17.2 Teach pendant multi-language setting . . . 1330
1.3.17.2.1 Input method switch . . . 1331
1.3.17.2.2 The language of the teaching pendant and webApp is inconsistent . . . 1333
1.3.17.3 Controller and Teach Pendant IP Reset Function . . . 1333
1.3.17.3.1 Function Overview . . . 1333
1.3.17.3.2 Webrecovery Interface IP Reset . . . 1335
1.3.17.3.3 Physical Teach Pendant F1 Key Custom IP Reset . . . 1335
1.3.17.3.4 Physical Teach Pendant F2 and F4 Key Combination IP Reset . . . 1336

1.3.17.4 Teach Pendant Key Customization Function . . . 1336
1.3.17.4.1 Function Overview . . . 1336
1.3.17.4.2 Operation Instructions . . . 1336
1.3.17.4.2.1 Function Configuration . . . 1336
1.3.17.4.2.2 Key Custom Set to Drag . . . 1337
1.3.17.4.2.3 F1-F4 Key Function Customization . . . 1339
1.3.18 Custom protocol slave commands . . . 1339
1.3.18.1 Overview . . . 1339
1.3.18.2 Industrial Bus Protocol Integration for Robot Motion Control . . . 1339
1.3.18.3 Environment Configuration . . . 1340
1.3.18.3.1 FRH-PCIeN-EC/EIP/CC/PN-RJ-V10 board hardware environment setup . 1340
1.3.18.3.2 FRJ-PCIeN Board Hardware Setup . . . 1342
1.3.18.3.3 Software environment setup . . . 1346
1.3.18.3.4 PLC Environment construction . . . 1348
1.3.18.3.4.1 Siemens Profinet . . . 1348
1.3.18.3.4.2 Mitsubishi CC-Link IEF Basic . . . 1353
1.3.18.3.4.3 Inovance EtherCAT Configuration . . . 1356
1.3.18.3.5 HMI setting (CC-Link IEF Basic emulation) . . . 1360
1.3.18.3.6 HMI setting (Profinet emulation) . . . 1374
1.3.18.4 Robot Slave Mode Operation Manual . . . 1376
1.3.18.4.1 Loading Slave Mode . . . 1376
1.3.18.5 Board Communication Cycle Configuration . . . 1383
1.3.18.6 Appendice . . . 1385
1.3.18.6.1 Instruction List . . . 1385
1.3.19 Appendix . . . 1385
1.3.19.1 Appendix 1: Motion controller errors and handling methods . . . 1385
1.3.19.2 Appendix 2: Servo driver fault code table . . . 1391
1.3.19.3 Appendix 3: End plate 485 upgrade . . . 1396
1.3.19.4 Appendix 4: Control Box 485 Upgrade . . . 1399
1.3.19.5 Appendix 5: List of Spare Parts and Vulnerable Parts . . . 1399
1.3.20 Term . . . 1399
1.4 Changelog . . . 1401
1.4.1 Version V3.9.3 . . . 1401
1.4.2 Version V3.9.2 . . . 1401
1.4.3 Version V3.9.1 . . . 1403
1.4.4 Version V3.9.0 . . . 1404
1.4.5 Version V3.8.7 . . . 1405
1.4.6 Version V3.8.6 . . . 1405
1.4.7 Version V3.8.5 . . . 1406
1.4.8 Version V3.8.4.1 . . . 1407
1.4.9 Version V3.8.4 . . . 1407
1.4.10 Version V3.8.3 . . . 1408
1.4.11 Version V3.8.2 . . . 1408
1.4.12 Version V3.8.1 . . . 1410
1.4.13 Version V3.8.0 . . . 1410
1.4.14 Version V3.7.8 . . . 1411
1.4.15 Version V3.7.7 . . . 1412
1.4.16 Version V3.7.6 . . . 1413
1.4.17 Version V3.7.5 . . . 1415
1.4.18 Version V3.7.4 . . . 1415
1.4.19 Version V3.7.3 . . . 1416
1.4.20 Version V3.7.2 . . . 1416
1.4.21 Version V3.7.1 . . . 1416

2 SDK Manual **1419**
2.1 C++ . . . 1419
2.1.1 Version Update Description . . . 1421
2.1.2 Data structure specification . . . 1422
2.1.2.1 Interface call return value type . . . 1422
2.1.2.2 Joint position data type . . . 1422
2.1.2.3 Cartesian spatial location data type . . . 1422
2.1.2.4 Euler Angle attitude data type . . . 1422
2.1.2.5 Cartesian space pose data type . . . 1423
2.1.2.6 Extension axis position data type . . . 1423
2.1.2.7 Torque sensor data type . . . 1423
2.1.2.8 Spiral parameter data type . . . 1423
2.1.2.9 feedback packet of robot controller state . . . 1424
2.1.3 Basics . . . 1427
2.1.3.1 Instantiate Robot . . . 1427
2.1.3.2 Establish Communication with Controller . . . 1427
2.1.3.3 Close Communication with Controller . . . 1427
2.1.3.4 Query SDK Version . . . 1427
2.1.3.5 Get Controller IP . . . 1427
2.1.3.6 Control Robot to Enter/Exit Drag Teaching Mode . . . 1428
2.1.3.7 Check if Robot is in Drag Mode . . . 1428
2.1.3.8 Enable/Disable Robot . . . 1428
2.1.3.9 Switch Between Manual/Auto Mode . . . 1428
2.1.3.10 Shut Down Robot OS . . . 1428
2.1.3.11 Set Reconnection Parameters . . . 1429
2.1.3.12 Shut down the robot operating system . . . 1429
2.1.3.13 Initialize Log Parameters . . . 1429
2.1.3.14 Set Log Filter Level . . . 1429
2.1.3.15 Basic Robot Control Example . . . 1430
2.1.3.16 Get Robot Software Version Example . . . 1430
2.1.3.17 Get Robot Hardware Version . . . 1431
2.1.3.18 Get Robot Firmware Version . . . 1431
2.1.3.19 Get Robot Software/Firmware Version Example . . . 1431
2.1.4 Robot Motion . . . 1432
2.1.4.1 JOG Motion . . . 1432
2.1.4.2 JOG Deceleration Stop . . . 1433
2.1.4.3 JOG Immediate Stop . . . 1433
2.1.4.4 Robot JOG Control Example . . . 1433
2.1.4.5 Joint Space Motion . . . 1434
2.1.4.6 Joint Space Motion (Automatic Forward Kinematics Calculation) . . . 1435
2.1.4.7 Cartesian Space Linear Motion . . . 1435
2.1.4.8 Cartesian Space Linear Motion (Automatic Inverse Kinematics Calculation) . . . 1436
2.1.4.9 Cartesian Space Circular Motion . . . 1436
2.1.4.10 Cartesian Space Circular Motion (Automatic Inverse Kinematics Calculation) . . . 1437
2.1.4.11 Cartesian Space Full Circle Motion . . . 1438
2.1.4.12 Cartesian Space Full Circle Motion (Automatic Inverse Kinematics Calculation) . . . 1439
2.1.4.13 Cartesian Space Point-to-Point Motion . . . 1439
2.1.4.14 Robot Basic Motion Instruction Code Example . . . 1440
2.1.4.15 Cartesian Space Spiral Motion . . . 1441
2.1.4.16 Cartesian Space Spiral Motion (Automatic Inverse Kinematics Calculation) . . . 1442
2.1.4.17 Spiral Motion Code Example . . . 1442
2.1.4.18 Servo Motion Start . . . 1443
2.1.4.19 Servo Motion End . . . 1443
2.1.4.20 Joint Space Servo Mode Motion . . . 1443

2.1.4.21 Joint Space Servo Mode Motion Example . . . 1444
2.1.4.22 Joint Torque Control Start . . . 1445
2.1.4.23 Joint Torque Control . . . 1445
2.1.4.24 Joint Torque Control End . . . 1445
2.1.4.25 Joint Torque Control Example . . . 1445
2.1.4.26 Joint Torque Control Code Example with Overspeed Protection . . . 1446
2.1.4.27 Cartesian Space Servo Mode Motion . . . 1447
2.1.4.28 Cartesian Space Servo Mode Motion Code Example . . . 1447
2.1.4.29 Spline Motion Start . . . 1448
2.1.4.30 Joint Space Spline Motion (Auto Forward Kinematics Calculation) . . . 1448
2.1.4.31 Spline PTP Motion . . . 1449
2.1.4.32 Spline Motion End . . . 1449
2.1.4.33 Spline Motion Example . . . 1449
2.1.4.34 New Spline Motion Start . . . 1450
2.1.4.35 New Spline Command Point . . . 1450
2.1.4.36 New Spline Command Point (Auto Inverse Kinematics Calculation) . . . 1451
2.1.4.37 New Spline Motion End . . . 1451
2.1.4.38 New Spline Motion Example . . . 1451
2.1.4.39 Stop Motion . . . 1453
2.1.4.40 Pause Motion . . . 1453
2.1.4.41 Resume Motion . . . 1453
2.1.4.42 Motion Pause, Resume, Stop Example . . . 1453
2.1.4.43 Point Global Offset Start . . . 1454
2.1.4.44 Point Global Offset End . . . 1454
2.1.4.45 Point Offset Example . . . 1454
2.1.4.46 Control Box AO Flying Start . . . 1455
2.1.4.47 Control Box AO Flying Stop . . . 1456
2.1.4.48 End AO Flying Start . . . 1456
2.1.4.49 End AO Flying Stop . . . 1456
2.1.4.50 AO Flying Example . . . 1456
2.1.4.51 Start Ptp Motion FIR Filter . . . 1457
2.1.4.52 Close Ptp Motion FIR Filter . . . 1458
2.1.4.53 Start LIN, ARC Motion FIR Filter . . . 1458
2.1.4.54 Close LIN, ARC Motion FIR Filter . . . 1458
2.1.4.55 FIR Filter Example . . . 1458
2.1.4.56 Acceleration Smoothing Start . . . 1459
2.1.4.57 Acceleration Smoothing End . . . 1459
2.1.4.58 Acceleration Smoothing Example . . . 1460
2.1.4.59 Specified Pose Speed Start . . . 1460
2.1.4.60 Specified Pose Speed End . . . 1461
2.1.4.61 Robot Specified Pose Speed Example . . . 1461
2.1.4.62 Start Singular Pose Protection . . . 1462
2.1.4.63 Stop Singular Pose Protection . . . 1462
2.1.4.64 Robot Singular Pose Protection Example . . . 1462
2.1.4.65 Clear the motion command queue . . . 1463
2.1.4.66 Move to Intersecting Line Start Point . . . 1463
2.1.4.67 Intersecting Line Motion . . . 1464
2.1.4.68 Robot Intersecting Line Motion Code Example . . . 1464
2.1.4.69 Stationary Air Motion . . . 1466
2.1.4.70 Stationary Air Motion Code Example . . . 1466
2.1.5 IO . . . 1466
2.1.5.1 Set Control Box Digital Output . . . 1466
2.1.5.2 Set Tool Digital Output . . . 1467
2.1.5.3 Set Control Box Analog Output . . . 1467

2.1.5.4 Set Tool Analog Output . . . 1467
2.1.5.5 Code Example for Setting Digital and Analog Outputs . . . 1467
2.1.5.6 Get Control Box Digital Input . . . 1469
2.1.5.7 Get Tool Digital Input . . . 1469
2.1.5.8 Get Control Box Analog Input . . . 1469
2.1.5.9 Get Tool Analog Input . . . 1469
2.1.5.10 Get Robot End Point Record Button Status . . . 1470
2.1.5.11 Get Robot End DO Output Status . . . 1470
2.1.5.12 Get Robot Controller DO Output Status . . . 1470
2.1.5.13 Code Example for Getting Robot DI and DO Status . . . 1470
2.1.5.14 Wait for Control Box Digital Input . . . 1471
2.1.5.15 Wait for Control Box Multi-Channel Digital Input . . . 1471
2.1.5.16 Wait for Tool Digital Input . . . 1472
2.1.5.17 Wait for Control Box Analog Input . . . 1472
2.1.5.18 Wait for Tool Analog Input . . . 1472
2.1.5.19 Code Example for Waiting for Control Box Digital and Analog Input Signals . . . 1473
2.1.5.20 Set Whether to Reset Control Box DO Output After Stop/Pause . . . 1473
2.1.5.21 Set Whether to Reset Control Box AO Output After Stop/Pause . . . 1474
2.1.5.22 Set Whether to Reset End Tool DO Output After Stop/Pause . . . 1474
2.1.5.23 Set Whether to Reset End Tool AO Output After Stop/Pause . . . 1474
2.1.5.24 Set Whether to Reset Extension DO Output After Stop/Pause . . . 1474
2.1.5.25 Set Whether to Reset Extension AO Output After Stop/Pause . . . 1475
2.1.5.26 Set Whether to Reset SmartTool Output After Stop/Pause . . . 1475
2.1.5.27 Example Code for Setting LUA Program Output Reset After Stop/Pause . . . 1475
2.1.6 Common Settings . . . 1476
2.1.6.1 Set Tool Reference Point - Six-Point Method . . . 1476
2.1.6.2 Calculate Tool Coordinate System . . . 1476
2.1.6.3 Set Tool Reference Point - Four-Point Method . . . 1476
2.1.6.4 Calculate Tool Coordinate System . . . 1477
2.1.6.5 Calculate Tool Coordinate System from Points . . . 1477
2.1.6.6 Set Tool Coordinate System . . . 1477
2.1.6.7 Set Tool Coordinate System List . . . 1477
2.1.6.8 Get Current Tool Coordinate System . . . 1478
2.1.6.9 Robot Tool Coordinate System Operation Example . . . 1478
2.1.6.10 Set External Tool Reference Point - Six-Point Method . . . 1479
2.1.6.11 Calculate External Tool Coordinate System . . . 1480
2.1.6.12 Set External Tool Coordinate System . . . 1480
2.1.6.13 Set External Tool Coordinate System List . . . 1480
2.1.6.14 Robot External Tool Coordinate System Operation Example . . . 1480
2.1.6.15 Set Workpiece Reference Point - Three-Point Method . . . 1481
2.1.6.16 Calculate Workpiece Coordinate System . . . 1481
2.1.6.17 Set Workpiece Coordinate System . . . 1482
2.1.6.18 Set Workpiece Coordinate System List . . . 1482
2.1.6.19 Calculate Workpiece Coordinate System from Points . . . 1482
2.1.6.20 Get Current Workpiece Coordinate System . . . 1483
2.1.6.21 Robot Workpiece Coordinate System Operation Example . . . 1483
2.1.6.22 Set Global Speed . . . 1484
2.1.6.23 Set Robot Acceleration . . . 1484
2.1.6.24 Get Robot Default Speed . . . 1484
2.1.6.25 Set End Load Weight . . . 1484
2.1.6.26 Set End Effector Payload Center of Gravity Coordinates . . . 1485
2.1.6.27 Get Current Load Weight . . . 1485
2.1.6.28 Get Current Load Center of Gravity . . . 1485
2.1.6.29 Set Robot Installation Method . . . 1485

2.1.6.30 Set Robot Installation Angle . . . 1486
2.1.6.31 Get Robot Installation Angle . . . 1486
2.1.6.32 Set System Variable Value . . . 1486
2.1.6.33 Get System Variable Value . . . 1486
2.1.6.34 Robot Common Settings Example . . . 1487
2.1.6.35 Joint Friction Compensation Switch . . . 1488
2.1.6.36 Set Joint Friction Compensation Coefficient - Standard Installation . . . 1488
2.1.6.37 Set Joint Friction Compensation Coefficient - Wall Installation . . . 1488
2.1.6.38 Set Joint Friction Compensation Coefficient - Ceiling Installation . . . 1488
2.1.6.39 Set Joint Friction Compensation Coefficient - Free Installation . . . 1489
2.1.6.40 Robot Joint Friction Compensation Example . . . 1489
2.1.6.41 Query Robot Error Code . . . 1490
2.1.6.42 Error State Clear . . . 1490
2.1.6.43 Robot Fault State Acquisition and Error Clear Example . . . 1490
2.1.6.44 Set wide voltage control box temperature and fan current monitoring parameters . . . 1491
2.1.6.45 Obtain wide voltage control box temperature and fan current monitoring parameters 1491
2.1.6.46 Wide voltage control box temperature and fan current state acquisition code example 1491
2.1.6.47 Calculate Focus Calibration Results . . . 1492
2.1.6.48 Set Focus Position . . . 1492
2.1.6.49 Enable Focus Following . . . 1493
2.1.6.50 Disable Focus Following . . . 1493
2.1.6.51 Robot Focus Following Code Example . . . 1493
2.1.6.52 Open the joint torque sensor sensitivity calibration function . . . 1494
2.1.6.53 Data acquisition of joint torque sensor sensitivity . . . 1495
2.1.6.54 Obtain joint torque sensor sensitivity calibration results . . . 1495
2.1.6.55 Get Joint Torque Sensor Hysteresis Error . . . 1495
2.1.6.56 Get Joint Torque Sensor Repeatability . . . 1495
2.1.6.57 Set Joint Force Sensor Parameters . . . 1495
2.1.6.58 An example of joint torque sensor sensitivity auto-calibration code . . . 1496
2.1.6.59 Gets 8 slave error frames . . . 1499
2.1.6.60 Error frame zeroing at slave port . . . 1499
2.1.6.61 Gets an example of a slave port error frame code . . . 1499
2.1.6.62 Set feed-forward coefficients for each axis . . . 1501
2.1.6.63 Obtain the feed-forward coefficients of each axis . . . 1501
2.1.6.64 Example of robot velocity feed-forward coefficient code . . . 1501
2.1.6.65 Photoelectric Sensor TCP Calibration - Compute Tool RPY . . . 1502
2.1.6.66 Photoelectric Sensor TCP Calibration - Compute Tool XYZ . . . 1502
2.1.6.67 Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position . . . 1502
2.1.6.68 Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position . . . 1503
2.1.6.69 Photoelectric Sensor TCP Calibration - Get Tool Center Point Position . . . 1503
2.1.6.70 Photoelectric Sensor TCP Calibration . . . 1503
2.1.6.71 Photoelectric Sensor TCP Calibration Code Example . . . 1503
2.1.7 Security settings . . . 1504
2.1.7.1 Set Collision Level . . . 1504
2.1.7.2 Set Post-Collision Strategy . . . 1504
2.1.7.3 Custom Collision Detection Threshold Start . . . 1504
2.1.7.4 Custom Collision Detection Threshold End . . . 1505
2.1.7.5 Robot Collision Level Setting Code Example . . . 1505
2.1.7.6 Set Positive Limit . . . 1506
2.1.7.7 Set Negative Limit . . . 1506
2.1.7.8 Get Joint Soft Limit Angles . . . 1506
2.1.7.9 Robot Limit Setting Code Example . . . 1507
2.1.7.10 Set Robot Collision Detection Method . . . 1507
2.1.7.11 Set Static Collision Detection On/Off . . . 1507

2.1.7.12 Robot Collision Detection Method Code Example . . . 1508
2.1.7.13 Joint Torque Power Detection . . . 1508
2.1.7.14 Joint Torque Power Detection Code Example . . . 1508
2.1.8 Status query . . . 1509
2.1.8.1 Get Current Joint Positions (Degrees) . . . 1509
2.1.8.2 Get Joint Feedback Speed . . . 1509
2.1.8.3 Get Joint Feedback Acceleration . . . 1510
2.1.8.4 Get TCP Command Composite Speed . . . 1510
2.1.8.5 Get TCP Feedback Composite Speed . . . 1510
2.1.8.6 Get TCP Command Speed . . . 1510
2.1.8.7 Get TCP Feedback Speed . . . 1511
2.1.8.8 Get Current Tool Pose . . . 1511
2.1.8.9 Get Current Tool Frame Number . . . 1511
2.1.8.10 Get Current Work Object Frame Number . . . 1511
2.1.8.11 Get Current End Flange Pose . . . 1512
2.1.8.12 Get Current Joint Torque . . . 1512
2.1.8.13 Get System Time . . . 1512
2.1.8.14 Check If Robot Motion Is Complete . . . 1512
2.1.8.15 Query Robot Motion Queue Length . . . 1512
2.1.8.16 Get Robot Emergency Stop Status . . . 1513
2.1.8.17 Get SDK-Robot Communication Status . . . 1513
2.1.8.18 Get Safety Stop Signal . . . 1513
2.1.8.19 Get Robot Joint Driver Temperature(°C) . . . 1513
2.1.8.20 Get Robot Joint Driver Torque(Nm) . . . 1513
2.1.8.21 Get Robot Real-time Status Structure . . . 1514
2.1.8.22 Robot Status Query Code Example . . . 1514
2.1.8.23 Inverse Kinematics Calculation . . . 1516
2.1.8.24 Inverse Kinematics Calculation (Reference Position) . . . 1516
2.1.8.25 Inverse Kinematics Solution, Cartesian Space Including Extended Axis Position . . . 1516
2.1.8.26 Example Code for Inverse Kinematics Solution Including Extended Axis Position . . . 1517
2.1.8.27 Check If Inverse Kinematics Has Solution . . . 1517
2.1.8.28 Forward Kinematics Calculation . . . 1518
2.1.8.29 Robot Kinematics Calculation Code Example . . . 1518
2.1.8.30 Query Robot Teaching Point Data . . . 1519
2.1.8.31 Get Robot DH Parameter Compensation Values . . . 1519
2.1.8.32 Get Controller SN Code . . . 1519
2.1.8.33 Query Robot Teaching Point Data Code Example . . . 1519
2.1.8.34 Get Tool Coordinate System by ID . . . 1520
2.1.8.35 Get Work Object Coordinate System by ID . . . 1520
2.1.8.36 Get External Tool Coordinate System by ID . . . 1521
2.1.8.37 Get Extended Axis Coordinate System by ID . . . 1521
2.1.8.38 Get Payload Mass and Center of Gravity by ID . . . 1521
2.1.8.39 Get Current Tool Coordinate System . . . 1521
2.1.8.40 Get Current Work Object Coordinate System . . . 1522
2.1.8.41 Get Current External Tool Coordinate System . . . 1522
2.1.8.42 Get Current Extended Axis Coordinate System . . . 1522
2.1.8.43 Get Robot Coordinate Systems and Payload Code Example . . . 1522
2.1.9 Trajectory recurrence . . . 1524
2.1.9.1 Set TPD Trajectory Recording Parameters . . . 1524
2.1.9.2 Start TPD Trajectory Recording . . . 1524
2.1.9.3 Stop TPD Trajectory Recording . . . 1525
2.1.9.4 Delete TPD Trajectory Recording . . . 1525
2.1.9.5 TPD Trajectory Preload . . . 1525
2.1.9.6 TPD Trajectory Playback . . . 1525

2.1.9.7 Get TPD Start Pose . . . 1525
2.1.9.8 Robot TPD Trajectory Recording Code Example . . . 1526
2.1.9.9 Trajectory Preprocessing . . . 1527
2.1.9.10 Trajectory Playback . . . 1527
2.1.9.11 Get Trajectory Start Pose . . . 1527
2.1.9.12 Get Trajectory Point Number . . . 1527
2.1.9.13 Set Trajectory Running Speed . . . 1527
2.1.9.14 Set Trajectory Running Force and Torque . . . 1528
2.1.9.15 Set Trajectory Running Force in X Direction . . . 1528
2.1.9.16 Set Trajectory Running Force in Y Direction . . . 1528
2.1.9.17 Set Trajectory Running Force in Z Direction . . . 1528
2.1.9.18 Set Trajectory Running Torque Around X Axis . . . 1528
2.1.9.19 Set Trajectory Running Torque Around Y Axis . . . 1529
2.1.9.20 Set Trajectory Running Torque Around Z Axis . . . 1529
2.1.9.21 Upload Trajectory J File . . . 1529
2.1.9.22 Delete Trajectory J File . . . 1529
2.1.9.23 Robot Trajectory J File Playback Code Example . . . 1530
2.1.9.24 Trajectory Preprocessing (Trajectory Lookahead) . . . 1531
2.1.9.25 Trajectory Playback (Trajectory Lookahead) . . . 1531
2.1.9.26 Trajectory Playback (Trajectory Lookahead) Code Example . . . 1531
2.1.10 WebAPP program use . . . 1532
2.1.10.1 Set Default Program to Load Automatically on Startup . . . 1532
2.1.10.2 Load Specified Program . . . 1532
2.1.10.3 Get Loaded Program Name . . . 1533
2.1.10.4 Get Current Program Execution Line Number . . . 1533
2.1.10.5 Run Currently Loaded Program . . . 1533
2.1.10.6 Pause Currently Running Program . . . 1533
2.1.10.7 Resume Currently Paused Program . . . 1533
2.1.10.8 Stop Currently Running Program . . . 1534
2.1.10.9 Get Program Execution State . . . 1534
2.1.10.10 Robot LUA Program Operation Code Example . . . 1534
2.1.10.11 Download Lua File . . . 1535
2.1.10.12 Delete Lua File . . . 1535
2.1.10.13 Get All Current Lua File Names . . . 1535
2.1.10.14 Upload Lua File . . . 1536
2.1.10.15 Robot LUA File Upload/Download Code Example . . . 1536
2.1.11 Robot Peripheral Devices . . . 1537
2.1.11.1 Configure Gripper . . . 1537
2.1.11.2 Get Gripper Configuration . . . 1537
2.1.11.3 Activate Gripper . . . 1537
2.1.11.4 Control Gripper . . . 1537
2.1.11.5 Get Gripper Motion Status . . . 1538
2.1.11.6 Get Gripper Activation Status . . . 1538
2.1.11.7 Get Gripper Position . . . 1538
2.1.11.8 Get Gripper Speed . . . 1539
2.1.11.9 Get Gripper Current . . . 1539
2.1.11.10 Get Gripper Voltage . . . 1539
2.1.11.11 Get Gripper Temperature . . . 1539
2.1.11.12 Calculate Pre-Pick Point - Vision . . . 1540
2.1.11.13 Calculate Retreat Point - Vision . . . 1540
2.1.11.14 Robot Gripper Operation Code Example . . . 1540
2.1.11.15 Get Rotation Count of Rotary Gripper . . . 1542
2.1.11.16 Get Rotation Speed of Rotary Gripper . . . 1542
2.1.11.17 Get Rotation Torque of Rotary Gripper . . . 1542

2.1.11.18 Rotary Gripper Status Code Example . . . 1542
2.1.11.19 Conveyor Start/Stop . . . 1543
2.1.11.20 Record IO Detection Point . . . 1543
2.1.11.21 Record Point A . . . 1543
2.1.11.22 Record Reference Point . . . 1544
2.1.11.23 Record Point B . . . 1544
2.1.11.24 Conveyor Workpiece IO Detection . . . 1544
2.1.11.25 Get Current Object Position . . . 1544
2.1.11.26 Conveyor Tracking Start . . . 1544
2.1.11.27 Conveyor Tracking Stop . . . 1545
2.1.11.28 Conveyor Parameter Configuration . . . 1545
2.1.11.29 Conveyor Capture Point Compensation . . . 1545
2.1.11.30 Conveyor Linear Motion . . . 1546
2.1.11.31 Conveyor Communication Input Detection . . . 1546
2.1.11.32 Conveyor Communication Input Detection Trigger . . . 1546
2.1.11.33 Robot Conveyor Operation Example Program . . . 1546
2.1.11.34 End Sensor Configuration . . . 1548
2.1.11.35 Get End Sensor Configuration . . . 1548
2.1.11.36 End Sensor Activation . . . 1548
2.1.11.37 End Sensor Register Write . . . 1548
2.1.11.38 End Sensor Code Example . . . 1549
2.1.11.39 Get Robot Peripheral Protocol . . . 1549
2.1.11.40 Set Robot Peripheral Protocol . . . 1550
2.1.11.41 Set Robot Peripheral Protocol Example . . . 1550
2.1.11.42 Get End Communication Parameters . . . 1550
2.1.11.43 Set End Communication Parameters . . . 1551
2.1.11.44 Set End File Transfer Type . . . 1551
2.1.11.45 Set Enable End LUA Execution . . . 1551
2.1.11.46 End LUA File Error Recovery . . . 1551
2.1.11.47 Get End LUA Execution Enable Status . . . 1552
2.1.11.48 Set End LUA End Device Enable Type . . . 1552
2.1.11.49 Get End LUA End Device Enable Type . . . 1552
2.1.11.50 Get Currently Configured End Devices . . . 1552
2.1.11.51 Set Enable Gripper Action Control Function . . . 1553
2.1.11.52 Get Enable Gripper Action Control Function . . . 1553
2.1.11.53 Robot Ethercat Slave File Write . . . 1553
2.1.11.54 Upload End Lua Open Protocol File . . . 1554
2.1.11.55 Robot Ethercat Slave Enter Boot Mode . . . 1554
2.1.11.56 Robot End LUA File Operation Code Example . . . 1554
2.1.11.57 Get SmartTool Button Status . . . 1556
2.1.11.58 SmartTool Button Code Example . . . 1556
2.1.11.59 Control Array Suction Cup . . . 1556
2.1.11.60 Get Array Suction Cup Status . . . 1557
2.1.11.61 Wait for Suction Cup Status . . . 1557
2.1.11.62 Array Suction Cup Control Command Code Example . . . 1557
2.1.11.63 Upload Peripheral Open Protocol LUA File . . . 1558
2.1.11.64 Get Slave Board Parameters . . . 1559
2.1.11.65 Write Slave DO . . . 1559
2.1.11.66 Write Slave AO . . . 1559
2.1.11.67 Read Slave DI . . . 1559
2.1.11.68 Read Slave AI . . . 1560
2.1.11.69 Wait for Extended DI Input . . . 1560
2.1.11.70 Wait for Extended AI Input . . . 1560
2.1.11.71 Slave Mode Related Interface Command Code Example . . . 1560

2.1.11.72 Laser Peripheral On/Off . . . 1562
2.1.11.73 Laser Tracking Start/Stop . . . 1562
2.1.11.74 Laser Seam Search Start - Fixed Direction . . . 1562
2.1.11.75 Laser Seam Search Start - Arbitrary Point Direction . . . 1562
2.1.11.76 Laser Seam Search Stop . . . 1563
2.1.11.77 Laser Network Parameter Configuration . . . 1563
2.1.11.78 Laser Peripheral Sampling Period Configuration . . . 1563
2.1.11.79 Laser Peripheral Driver Load . . . 1563
2.1.11.80 Laser Peripheral Driver Unload . . . 1564
2.1.11.81 Laser Seam Trajectory Recording . . . 1564
2.1.11.82 Laser Seam Trajectory Replay . . . 1564
2.1.11.83 Laser Tracking Replay . . . 1564
2.1.11.84 Laser Seam Trajectory Recording and Replay . . . 1565
2.1.11.85 Move to Laser Recorded Start Point . . . 1565
2.1.11.86 Move to Laser Recorded End Point . . . 1565
2.1.11.87 Move to Laser Sensor Seam Search Point . . . 1566
2.1.11.88 Get Laser Sensor Seam Search Point Coordinates . . . 1566
2.1.11.89 Laser Peripheral Sensor Parameter Configuration and Debugging Code Example . . . 1566
2.1.11.90 Laser Trajectory Scanning and Trajectory Replay Code Example . . . 1567
2.1.11.91 Laser Seam Search and Real-time Tracking Code Example . . . 1568
2.1.11.92 Extended Axis and Robot Synchronized Laser Tracking Code Example . . . 1570
2.1.12 Force control . . . 1571
2.1.12.1 Force sensor configuration . . . 1571
2.1.12.2 Get the force sensor configuration . . . 1571
2.1.12.3 Force Sensor Activation . . . 1572
2.1.12.4 Force sensor calibration . . . 1572
2.1.12.5 Set Force Sensor Reference Coordinate System . . . 1572
2.1.12.6 Set Payload Weight Under Force Sensor . . . 1572
2.1.12.7 Set Payload Center of Gravity Under Force Sensor . . . 1573
2.1.12.8 Get Payload Weight Under Force Sensor . . . 1573
2.1.12.9 Get Payload Center of Gravity Under Force Sensor . . . 1573
2.1.12.10 Force Sensor Auto Zero Calibration . . . 1573
2.1.12.11 Get Force/Torque Data in Reference Coordinate System . . . 1574
2.1.12.12 Get Raw Force/Torque Data from Force Sensor . . . 1574
2.1.12.13 Force Sensor Configuration and Auto Zero Calibration Code Example . . . 1574
2.1.12.14 Payload Weight Identification Record . . . 1575
2.1.12.15 Payload Weight Identification Calculation . . . 1575
2.1.12.16 Payload COG Identification Record . . . 1576
2.1.12.17 Payload COG Identification Calculation . . . 1576
2.1.12.18 Force Sensor Payload Identification Code Example . . . 1576
2.1.12.19 Collision Guard . . . 1577
2.1.12.20 Collision Guard Code Example . . . 1578
2.1.12.21 Constant force control . . . 1579
2.1.12.22 Example of constant force control code with damping . . . 1579
2.1.12.23 Spiral Search . . . 1580
2.1.12.24 Rotary Insertion . . . 1581
2.1.12.25 Example code for force sensor rotation insertion . . . 1581
2.1.12.26 Linear Insertion . . . 1582
2.1.12.27 Spiral Search, Linear Insertion and Other Instruction Code Examples . . . 1583
2.1.12.28 Surface Localization . . . 1585
2.1.12.29 Calculate Middle Plane Position - Start . . . 1585
2.1.12.30 Calculate Middle Plane Position - End . . . 1585
2.1.12.31 Surface Localization Code Example . . . 1586
2.1.12.32 Compliance Control Activation . . . 1587

2.1.12.33 Compliance Control Deactivation . . . 1587
2.1.12.34 Compliance Control Code Example . . . 1587
2.1.12.35 Payload Identification Initialization . . . 1589
2.1.12.36 Payload Identification Initialization . . . 1589
2.1.12.37 Payload Identification Main Program . . . 1589
2.1.12.38 Get Payload Identification Result . . . 1590
2.1.12.39 Robot Payload Identification Code Example . . . 1590
2.1.12.40 Force Sensor Assisted Drag . . . 1591
2.1.12.41 Get Force Sensor Drag Switch Status . . . 1591
2.1.12.42 Force Sensor Auto Activation After Error Clear . . . 1592
2.1.12.43 Force Sensor Assisted Drag Code Example . . . 1592
2.1.12.44 Set Six-Dimensional Force and Joint Impedance Hybrid Drag Switch and Parameters 1592
2.1.12.45 Six-Dimensional Force and Joint Impedance Hybrid Drag Code Example . . . 1593
2.1.12.46 Impedance Start/Stop Control . . . 1593
2.1.12.47 Robot Impedance Start/Stop Control Code Example . . . 1594
2.1.12.48 Enable torque compensation function and compensation coefficient . . . 1595
2.1.13 Extended Axis . . . 1596
2.1.13.1 Set 485 Extended Axis Parameters . . . 1596
2.1.13.2 Get 485 Extended Axis Configuration Parameters . . . 1596
2.1.13.3 Set 485 Extended Axis Enable/Disable . . . 1596
2.1.13.4 Set 485 Extended Axis Control Mode . . . 1597
2.1.13.5 Set 485 Extended Axis Target Position (Position Mode) . . . 1597
2.1.13.6 Set 485 Extended Axis Target Torque (Torque Mode) - Temporarily Unavailable . . . 1597
2.1.13.7 Set 485 Extended Axis Homing . . . 1597
2.1.13.8 Clear 485 Extended Axis Error Information . . . 1598
2.1.13.9 Get 485 Extended Axis Servo Status . . . 1598
2.1.13.10 Set 485 Extended Axis Target Speed (Velocity Mode) . . . 1598
2.1.13.11 Set Status Feedback 485 Extended Axis Data Axis Number . . . 1599
2.1.13.12 Set 485 Extended Axis Motion Acceleration/Deceleration . . . 1599
2.1.13.13 Set 485 Extended Axis Emergency Stop Acceleration/Deceleration . . . 1599
2.1.13.14 Get 485 Extended Axis Motion Acceleration/Deceleration . . . 1599
2.1.13.15 Get 485 Extended Axis Emergency Stop Acceleration/Deceleration . . . 1600
2.1.13.16 Extended Axis Control Code Example . . . 1600
2.1.13.17 UDP Extended Axis Communication Parameter Configuration . . . 1602
2.1.13.18 Get UDP Extended Axis Communication Parameter Configuration . . . 1602
2.1.13.19 Load UDP Communication . . . 1603
2.1.13.20 Unload UDP Communication . . . 1603
2.1.13.21 UDP Extended Axis Communication Exception Disconnection Recovery . . . 1603
2.1.13.22 UDP Extended Axis Communication Exception Disconnection Close Communication 1603
2.1.13.23 UDP Extended Axis Parameter Configuration . . . 1604
2.1.13.24 Set Extended Axis Installation Position . . . 1604
2.1.13.25 Set Extended Axis System DH Parameter Configuration . . . 1604
2.1.13.26 UDP Extended Axis Enable . . . 1605
2.1.13.27 UDP Extended Axis Homing . . . 1605
2.1.13.28 UDP Extended Axis Jog Start . . . 1605
2.1.13.29 UDP Extended Axis Jog Stop . . . 1606
2.1.13.30 UDP Extended Axis Configuration and Jog Code Example . . . 1606
2.1.13.31 Set Extended Axis Coordinate System Reference Point - Four-Point Method . . . 1607
2.1.13.32 Calculate Extended Axis Coordinate System - Four-Point Method . . . 1608
2.1.13.33 Positioner Coordinate System Reference Point Setting . . . 1608
2.1.13.34 Positioner Coordinate System Calculation - Four-Point Method . . . 1608
2.1.13.35 Set Calibration Reference Point Pose in Extended Axis End Coordinate System . . . 1608
2.1.13.36 Apply Extended Axis Coordinate System . . . 1609
2.1.13.37 Get Extended Axis Coordinate System . . . 1609

2.1.13.38 Extended Axis Coordinate System Calibration Code Example . . . 1609
2.1.13.39 UDP Extended Axis Motion . . . 1612
2.1.13.40 UDP Extended Axis Motion Code Example . . . 1612
2.1.13.41 UDP Extended Axis and Robot Joint Motion Synchronous Motion . . . 1612
2.1.13.42 UDP Extended Axis and Robot Joint Motion Synchronization (Automatic Forward Kinematics Calculation) . . . 1613
2.1.13.43 UDP Extended Axis and Robot Joint Motion Synchronous Motion Code Example . 1613
2.1.13.44 UDP Extended Axis and Robot Linear Motion Synchronous Motion . . . 1615
2.1.13.45 UDP Extended Axis and Robot Linear Motion Synchronization (Automatic Inverse Kinematics Calculation) . . . 1615
2.1.13.46 UDP Extended Axis and Robot Linear Motion Synchronous Motion Code Example 1616
2.1.13.47 UDP Extended Axis and Robot Arc Motion Synchronous Motion . . . 1618
2.1.13.48 UDP Extended Axis and Robot Circular Motion Synchronization (Automatic Inverse Kinematics Calculation) . . . 1618
2.1.13.49 UDP Extended Axis and Robot Arc Motion Synchronous Motion Code Example . . 1619
2.1.13.50 Set Extended DO . . . 1621
2.1.13.51 Set Extended AO . . . 1621
2.1.13.52 Set Extended DI Input Filter Time . . . 1621
2.1.13.53 Set Extended AI Input Filter Time . . . 1622
2.1.13.54 Wait for Extended DI Input . . . 1622
2.1.13.55 Wait for Extended AI Input . . . 1622
2.1.13.56 Get Extended DI Value . . . 1622
2.1.13.57 Get Extended AI Value . . . 1623
2.1.13.58 Extended IO Code Example . . . 1623
2.1.13.59 Mobile Device Enable . . . 1624
2.1.13.60 Mobile Device Homing . . . 1624
2.1.13.61 Mobile Device Linear Motion . . . 1624
2.1.13.62 Mobile Device Arc Motion . . . 1625
2.1.13.63 Mobile Device Stop Motion . . . 1625
2.1.13.64 Mobile Device Code Example . . . 1625
2.1.14 Robot Welding . . . 1626
2.1.14.1 Set Welding Process Curve Parameters . . . 1626
2.1.14.2 Get Welding Process Curve Parameters . . . 1627
2.1.14.3 Set Welding Current to Analog Output Relationship . . . 1627
2.1.14.4 Set Welding Voltage to Analog Output Relationship . . . 1627
2.1.14.5 Get Welding Current to Analog Output Relationship . . . 1628
2.1.14.6 Get Welding Voltage to Analog Output Relationship . . . 1628
2.1.14.7 Set Welding Current . . . 1628
2.1.14.8 Set Welding Voltage . . . 1629
2.1.14.9 Set Weaving Parameters . . . 1629
2.1.14.10 Welding Parameter Setup Code Example . . . 1630
2.1.14.11 Instantaneously Set Weaving Parameters . . . 1631
2.1.14.12 Set Robot Welding Arc Unexpected Interruption Detection Parameters . . . 1632
2.1.14.13 Get Robot Welding Arc Unexpected Interruption Detection Parameters . . . 1632
2.1.14.14 Set Robot Welding Interruption Recovery Parameters . . . 1633
2.1.14.15 Get Robot Welding Interruption Recovery Parameters . . . 1633
2.1.14.16 Set Welder Control Mode Extended DO Port . . . 1633
2.1.14.17 Set Welder Control Mode . . . 1633
2.1.14.18 Welding Start . . . 1634
2.1.14.19 Welding End . . . 1634
2.1.14.20 Weaving Start . . . 1634
2.1.14.21 Weaving End . . . 1634
2.1.14.22 Forward Wire Feeding . . . 1635
2.1.14.23 Reverse Wire Feeding . . . 1635

2.1.14.24 Gas Feeding . . . 1635
2.1.14.25 Set Robot to Resume Welding After Interruption . . . 1635
2.1.14.26 Set Robot to Abort Welding After Interruption . . . 1636
2.1.14.27 Robot Welding Control Code Example . . . 1636
2.1.14.28 Segment Welding Start . . . 1637
2.1.14.29 Robot Segment Welding Code Example . . . 1637
2.1.14.30 Simulation Weaving Start . . . 1638
2.1.14.31 Simulation Weaving End . . . 1638
2.1.14.32 Start Trajectory Detection Warning (No Movement) . . . 1638
2.1.14.33 End Trajectory Detection Warning (No Movement) . . . 1639
2.1.14.34 Weaving Gradual Change Start . . . 1639
2.1.14.35 Robot Weaving Gradual Change Welding Code Example . . . 1639
2.1.14.36 Weaving Gradual Change End . . . 1640
2.1.14.37 Extended IO-Configure Welder Gas Detection Signal . . . 1640
2.1.14.38 Extended IO-Configure Welder Arc Start Signal . . . 1641
2.1.14.39 Extended IO-Configure Welder Reverse Wire Feed Signal . . . 1641
2.1.14.40 Extended IO-Configure Welder Forward Wire Feed Signal . . . 1641
2.1.14.41 Extended IO-Configure Welder Arc Success Signal . . . 1641
2.1.14.42 Extended IO-Configure Welder Ready Signal . . . 1642
2.1.14.43 Extended IO-Configure Welding Interruption Recovery Signal . . . 1642
2.1.14.44 Set Extended IO Welding Signal Code Example . . . 1642
2.1.14.45 Arc Tracking Control . . . 1643
2.1.14.46 Set Arc Tracking Input Signal Port . . . 1643
2.1.14.47 Arc Tracking + Multi-layer Multi-pass Compensation Start . . . 1644
2.1.14.48 Arc Tracking + Multi-layer Multi-pass Compensation End . . . 1644
2.1.14.49 Offset Coordinate Transformation - Multi-layer Multi-pass Welding . . . 1644
2.1.14.50 Multi-layer Multi-pass Welding Arc Tracking Code Example . . . 1644
2.1.14.51 Arc Tracking Welder Current Feedback AI Channel Selection . . . 1647
2.1.14.52 Arc Tracking Welder Voltage Feedback AI Channel Selection . . . 1647
2.1.14.53 Arc Tracking Welder Current Feedback Conversion Parameters . . . 1647
2.1.14.54 Arc Tracking Welder Voltage Feedback Conversion Parameters . . . 1648
2.1.14.55 Arc Tracking Code Example . . . 1648
2.1.14.56 Set Wire Search Extended IO Port . . . 1649
2.1.14.57 Example Program . . . 1649
2.1.14.58 Wire Search Start . . . 1651
2.1.14.59 Wire Search End . . . 1651
2.1.14.60 Calculate Wire Search Offset . . . 1651
2.1.14.61 Wait for Wire Search Completion . . . 1652
2.1.14.62 Write Wire Search Contact Point to Database . . . 1652
2.1.14.63 Robot Wire Search Code Example . . . 1652
2.1.14.64 Set Welding Voltage Gradual Change Start . . . 1654
2.1.14.65 Set Welding Voltage Gradual Change End . . . 1654
2.1.14.66 Set Welding Current Gradual Change Start . . . 1654
2.1.14.67 Set Welding Current Gradual Change End . . . 1655
2.1.14.68 Robot Welding Current Voltage Gradual Change Code Example . . . 1655
2.1.14.69 Set Custom Weave Parameters . . . 1656
2.1.14.70 Get Custom Weave Parameters . . . 1656
2.1.14.71 Custom Weave Parameters Code Example . . . 1656
2.1.15 Other Interfaces . . . 1658
2.1.15.1 Get SSH Public Key . . . 1658
2.1.15.2 Send SCP Command . . . 1658
2.1.15.3 Calculate MD5 Value of Specified File . . . 1658
2.1.15.4 Robot SSH/MD5 Command Code Example . . . 1659
2.1.15.5 Set Robot Port 20004 Feedback Cycle . . . 1659

2.1.15.6 Get Robot Port 20004 Feedback Cycle . . . 1660
2.1.15.7 Robot Port 20004 State Feedback Cycle Configuration Example . . . 1660
2.1.15.8 Robot Software Upgrade . . . 1660
2.1.15.9 Get Robot Software Upgrade Status . . . 1661
2.1.15.10 Robot Software Upgrade Code Example . . . 1661
2.1.15.11 Download Point Table Database . . . 1662
2.1.15.12 Upload Point Table Database . . . 1662
2.1.15.13 Update Lua File for Point Table . . . 1662
2.1.15.14 Robot Point Table Operation Code Example . . . 1662
2.1.15.15 Controller Log Download . . . 1663
2.1.15.16 All Data Source Download . . . 1663
2.1.15.17 Data Backup Package Download . . . 1664
2.1.15.18 Download Controller Data Code Example . . . 1664
2.1.15.19 Set Joint Firmware Upgrade . . . 1664
2.1.15.20 Set Controller Firmware Upgrade . . . 1665
2.1.15.21 Set End-Effector Firmware Upgrade . . . 1665
2.1.15.22 Joint Full Parameter Configuration Upgrade . . . 1665
2.1.15.23 Robot Slave Firmware Upgrade Code Example . . . 1665
2.1.15.24 Robot Operating System Upgrade (LA Control Box) . . . 1666
2.1.15.25 Get Robot Operating System Upgrade Result (LA Control Box) . . . 1666
2.1.15.26 Robot MCU log generation . . . 1667
2.1.16 Appendix . . . 1667
2.1.16.1 Source Code Download . . . 1667
2.1.16.2 Windows Platform Compilation . . . 1668
2.1.16.3 Linux Platform Compilation . . . 1671
2.2 C# . . . 1672
2.2.1 Version Update Description . . . 1674
2.2.2 Data Structure Description . . . 1675
2.2.2.1 Joint position data type . . . 1675
2.2.2.2 Cartesian space position data type . . . 1675
2.2.2.3 Euler Angle Attitude data type . . . 1675
2.2.2.4 Cartesian space position data type . . . 1675
2.2.2.5 Extended axis position data type . . . 1676
2.2.2.6 Torque sensor data type . . . 1676
2.2.2.7 Helix parameter data type . . . 1676
2.2.2.8 Extended axis state type . . . 1677
2.2.2.9 Welding interrupt status . . . 1677
2.2.2.10 Robot State Feedback Structures Types . . . 1677
2.2.3 Robot Basics . . . 1680
2.2.3.1 Instantiating the Robot . . . 1680
2.2.3.2 Establishing Communication with the Controller . . . 1680
2.2.3.3 Disconnect from the robot . . . 1681
2.2.3.4 Query the SDK version number . . . 1681
2.2.3.5 Get the controller IP address . . . 1681
2.2.3.6 Control the robot to enter or exit drag teaching mode . . . 1681
2.2.3.7 Check if the robot is in drag mode . . . 1681
2.2.3.8 Control robot enable or disable . . . 1682
2.2.3.9 Control robot manual/automatic mode switching . . . 1682
2.2.3.10 Shut down the robot operating system . . . 1682
2.2.3.11 Code example . . . 1682
2.2.3.12 Set robot communication reconnection parameters . . . 1682
2.2.3.13 Code example . . . 1683
2.2.3.14 Initialize log parameters . . . 1683
2.2.3.15 Set log filtering level . . . 1684

2.2.3.16 Get robot software version . . . 1684
2.2.3.17 Get robot hardware version . . . 1684
2.2.3.18 Get robot firmware version . . . 1685
2.2.3.19 Code example . . . 1685
2.2.4 Robot Motion . . . 1686
2.2.4.1 jog point motion . . . 1686
2.2.4.2 jog nudging deceleration stop . . . 1686
2.2.4.3 jog pointing stops immediately . . . 1686
2.2.4.4 Sample Robot Tap Control Code . . . 1686
2.2.4.5 Joint space motion . . . 1688
2.2.4.6 Joint space motion (automatic forward kinematics calculation) . . . 1689
2.2.4.7 Linear motion in Cartesian space . . . 1689
2.2.4.8 Cartesian space linear motion (automatic inverse kinematics calculation) . . . 1690
2.2.4.9 Cartesian Space Linear Motion (Added velAccParamMode parameter for velocity and acceleration modes) . . . 1690
2.2.4.10 Cartesian Space Linear Motion (Overload Function 1, Added blendMode) . . . 1691
2.2.4.11 Cartesian Space Linear Motion (Overload Function 2, No Joint Position Input Required) . . . 1692
2.2.4.12 Circular motion in Cartesian space . . . 1692
2.2.4.13 Cartesian space circular motion (automatic inverse kinematics calculation) . . . 1693
2.2.4.14 Cartesian Space Arc Motion (Added velAccParamMode parameter for velocity and acceleration modes) . . . 1694
2.2.4.15 Cartesian Space Arc Motion (Overload Function 1, No Joint Position Input Required) 1695
2.2.4.16 Point-to-point motion in Cartesian space . . . 1696
2.2.4.17 Whole circle motion in Cartesian space . . . 1696
2.2.4.18 Cartesian space full circle motion (automatic inverse kinematics calculation) . . . 1697
2.2.4.19 Cartesian Space Full Circle Motion (Added velAccParamMode parameter for velocity and acceleration modes) . . . 1697
2.2.4.20 Cartesian Space Full Circle Motion (Overload Function 1, No Joint Position Input Required) . . . 1698
2.2.4.21 Sample Code for Whole Circle Motion in Cartesian Space . . . 1699
2.2.4.22 Sample code for basic robot motion instructions . . . 1701
2.2.4.23 Spiral motion in Cartesian space . . . 1702
2.2.4.24 Cartesian space spiral motion (automatic inverse kinematics calculation) . . . 1702
2.2.4.25 Sample code for spiral motion . . . 1703
2.2.4.26 Starting servo motion . . . 1704
2.2.4.27 ServoMoveStart . . . 1704
2.2.4.28 Joint space servo mode movement . . . 1704
2.2.4.29 Joint space servo mode motion code example . . . 1704
2.2.4.30 Start of joint torque control . . . 1705
2.2.4.31 Joint torque control . . . 1706
2.2.4.32 End of joint torque control . . . 1706
2.2.4.33 Joint Torque Control Code Example . . . 1706
2.2.4.34 Joint Torque Control Code Example with Overspeed Protection . . . 1707
2.2.4.35 Servo mode motion in Cartesian space . . . 1708
2.2.4.36 Cartesian Space Servo Mode Motion . . . 1708
2.2.4.37 Example of servo mode motion code in Cartesian space . . . 1708
2.2.4.38 Spline motion starts . . . 1709
2.2.4.39 Spline motion PTP . . . 1709
2.2.4.40 Joint space spline motion (automatic forward kinematics calculation) . . . 1709
2.2.4.41 End of spline motion . . . 1710
2.2.4.42 Sample code for spline motion . . . 1710
2.2.4.43 Starting new spline motion . . . 1711
2.2.4.44 New spline instruction point . . . 1711

2.2.4.45 New spline command point (automatic inverse kinematics calculation) . . . 1711
2.2.4.46 End of new spline motion . . . 1712
2.2.4.47 Sample code for new spline motion . . . 1712
2.2.4.48 Terminate the motion . . . 1713
2.2.4.49 Pause the motion. . . . 1713
2.2.4.50 Resumes motion. . . . 1713
2.2.4.51 Motion Pause, Resume, Stop Code Example . . . 1714
2.2.4.52 Start the overall point offset . . . 1714
2.2.4.53 Points overall offset end . . . 1715
2.2.4.54 Points Offset Code Example . . . 1715
2.2.4.55 Control box AO flytap start . . . 1716
2.2.4.56 Control box AO flytap stop . . . 1716
2.2.4.57 End AO flyswatter start . . . 1716
2.2.4.58 End AO flytap stop . . . 1717
2.2.4.59 AO flyswatter code example . . . 1717
2.2.4.60 Start Ptp motion FIR filtering . . . 1718
2.2.4.61 Turn off Ptp motion FIR filtering . . . 1718
2.2.4.62 Start LIN, ARC motion FIR filtering. . . . 1718
2.2.4.63 Turn off LIN, ARC motion FIR filtering . . . 1718
2.2.4.64 FIR filtering code example . . . 1719
2.2.4.65 Acceleration smoothing on . . . 1719
2.2.4.66 Acceleration SmoothStart(bool saveFlag); int AccSmoothStart . . . 1720
2.2.4.67 Code example . . . 1720
2.2.4.68 Specify the attitude speed to turn on . . . 1720
2.2.4.69 Specifies that the attitude speed is off . . . 1721
2.2.4.70 Robot Specify Angular Speed Code Example . . . 1721
2.2.4.71 Starting singular position protection . . . 1721
2.2.4.72 Stop singular attitude protection . . . 1722
2.2.4.73 Code example . . . 1722
2.2.4.74 Safety Stop Trigger . . . 1722
2.2.4.75 Clear the motion command queue . . . 1723
2.2.4.76 Move to Intersecting Line Start Point . . . 1723
2.2.4.77 Intersecting Line Motion . . . 1723
2.2.4.78 Robot Intersecting Line Motion Code Example . . . 1724
2.2.4.79 Stationary Air Motion . . . 1725
2.2.4.80 Stationary Air Motion Code Example . . . 1725
2.2.5 Robot IO . . . 1726
2.2.5.1 Setting the control box digital outputs . . . 1726
2.2.5.2 Set the tool digital output . . . 1726
2.2.5.3 Set the control box analog output . . . 1726
2.2.5.4 Set the tool analog output . . . 1727
2.2.5.5 Set digital, analog output code example . . . 1727
2.2.5.6 etting control box digital inputs . . . 1728
2.2.5.7 Get tool digital input . . . 1728
2.2.5.8 Get control box analog input . . . 1729
2.2.5.9 Get tool analog input . . . 1729
2.2.5.10 Get the robot end record button status . . . 1729
2.2.5.11 Get the robot end DO output state . . . 1729
2.2.5.12 Get the DO output state of the machine controller . . . 1730
2.2.5.13 Get robot DI, DO state code example . . . 1730
2.2.5.14 Wait for control box digital input . . . 1731
2.2.5.15 Wait for multiple digital inputs to the control box . . . 1731
2.2.5.16 Wait for tool digital input . . . 1731
2.2.5.17 Wait for control box analog input . . . 1732

2.2.5.18 Wait for tool analog input . . . 1732
2.2.5.19 Wait for the control box digital, analog input signal code example . . . 1732
2.2.5.20 Set Whether Control Box DO Output Resets After Stop/Pause . . . 1733
2.2.5.21 Set Whether Control Box AO Output Resets After Stop/Pause . . . 1733
2.2.5.22 Set Whether End Tool DO Output Resets After Stop/Pause . . . 1733
2.2.5.23 Set Whether End Tool AO Output Resets After Stop/Pause . . . 1733
2.2.5.24 Set Whether Extended DO Output Resets After Stop/Pause . . . 1734
2.2.5.25 Set Whether Extended AO Output Resets After Stop/Pause . . . 1734
2.2.5.26 Set Whether SmartTool Output Resets After Stop/Pause . . . 1734
2.2.5.27 Code Example for Setting Output Reset After Lua Program Stop/Pause . . . 1734
2.2.6 Common Robot Settings . . . 1735
2.2.6.1 Setting the tool reference point - six-point method . . . 1735
2.2.6.2 Calculating the Tool Coordinate System - Six Point Method . . . 1735
2.2.6.3 Setting the tool reference point - four-point method . . . 1735
2.2.6.4 Calculate the tool coordinate system - four-point method . . . 1736
2.2.6.5 Set the tool coordinate system . . . 1736
2.2.6.6 Calculate the tool coordinate system from the point information . . . 1736
2.2.6.7 Set the list of tool coordinate systems . . . 1736
2.2.6.8 Get the current tool coordinate system . . . 1737
2.2.6.9 Robot tool coordinate system operation code example . . . 1737
2.2.6.10 Setting the external tool coordinate reference point - three-point method . . . 1739
2.2.6.11 Calculating an external tool coordinate system - three-point method . . . 1739
2.2.6.12 Set the external tool coordinate system . . . 1739
2.2.6.13 Setting up a list of external tool coordinate systems . . . 1739
2.2.6.14 Calculate the workpiece coordinate system from the point information . . . 1740
2.2.6.15 Sample code for manipulating the robot's external tool coordinate system . . . 1740
2.2.6.16 Setting the reference point of the workpiece coordinate system - three-point method 1741
2.2.6.17 Calculate the work coordinate system . . . 1741
2.2.6.18 Set the workpiece coordinate system . . . 1741
2.2.6.19 Set the list of workpiece coordinate systems . . . 1741
2.2.6.20 Get the current workpiece coordinate system . . . 1742
2.2.6.21 Robot workpiece coordinate system operation code example . . . 1742
2.2.6.22 Setting the global speed . . . 1743
2.2.6.23 Set the robot acceleration . . . 1743
2.2.6.24 Get the default robot speed . . . 1743
2.2.6.25 Set the end load weight . . . 1743
2.2.6.26 Set the center of mass of the end load . . . 1744
2.2.6.27 Get the weight of the current load . . . 1744
2.2.6.28 Get the center of mass of the current load . . . 1744
2.2.6.29 Set the robot mounting method . . . 1744
2.2.6.30 Set the robot installation angle . . . 1744
2.2.6.31 Get the robot installation angle . . . 1745
2.2.6.32 Sets the value of the system variable . . . 1745
2.2.6.33 Get the value of a system variable . . . 1745
2.2.6.34 Sample code for common robot settings . . . 1745
2.2.6.35 Joint friction compensation switch . . . 1746
2.2.6.36 Sets the joint friction compensation factor - proper . . . 1747
2.2.6.37 Set joint friction compensation coefficients - side mount . . . 1747
2.2.6.38 Set joint friction compensation coefficients - inverted . . . 1747
2.2.6.39 Set joint friction compensation coefficients - free mounting . . . 1747
2.2.6.40 Robot set joint friction compensation code example . . . 1747
2.2.6.41 Querying the robot error code . . . 1748
2.2.6.42 Error status clearing . . . 1748
2.2.6.43 Example of getting robot fault status and clearing error code . . . 1748

2.2.6.44 Setting the parameters for monitoring the temperature and fan speed of the wide voltage control box . . . 1749
2.2.6.45 Get the parameters for monitoring the temperature and fan speed of the wide-voltage control box. . . . 1749
2.2.6.46 Code example . . . 1749
2.2.6.47 Set the focus calibration point . . . 1750
2.2.6.48 Set the focus coordinates . . . 1750
2.2.6.49 Enable focus following . . . 1750
2.2.6.50 Stop focus following . . . 1751
2.2.6.51 Focus Follow Code Example . . . 1751
2.2.6.52 Enable joint torque sensor sensitivity calibration function . . . 1752
2.2.6.53 Sensitivity data acquisition of joint torque sensors . . . 1753
2.2.6.54 Get the sensitivity calibration results of the joint torque sensor . . . 1753
2.2.6.55 Get Joint Torque Sensor Hysteresis Error . . . 1753
2.2.6.56 Get Joint Torque Sensor Repeatability . . . 1753
2.2.6.57 Set Joint Force Sensor Parameters . . . 1754
2.2.6.58 Joint torque sensor sensitivity automatic calibration Code Example . . . 1754
2.2.6.59 Get the number of 8 slave port error frames of the robot . . . 1756
2.2.6.60 Clear the slave port error num . . . 1756
2.2.6.61 Gets the slave port error frame code example . . . 1756
2.2.6.62 Set the feedforward coefficients of the velocities of each axis . . . 1758
2.2.6.63 Get the feedforward coefficients of the velocities of each axis . . . 1758
2.2.6.64 Robot velocity feedforward coefficient code example . . . 1758
2.2.6.65 Photoelectric Sensor TCP Calibration - Compute Tool RPY . . . 1758
2.2.6.66 Photoelectric Sensor TCP Calibration - Compute Tool XYZ . . . 1759
2.2.6.67 Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position . . . 1759
2.2.6.68 Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position . . . 1759
2.2.6.69 Photoelectric Sensor TCP Calibration - Get Tool Center Point Position . . . 1759
2.2.6.70 Photoelectric Sensor TCP Calibration . . . 1760
2.2.6.71 Photoelectric Sensor TCP Calibration Code Example . . . 1760
2.2.7 Robot Safety Settings . . . 1760
2.2.7.1 Set collision level . . . 1760
2.2.7.2 Set collision post-strategy . . . 1761
2.2.7.3 Custom collision detection threshold function start . . . 1761
2.2.7.4 Custom collision detection threshold function disabled . . . 1761
2.2.7.5 Robot collision level settings code example . . . 1761
2.2.7.6 Set positive limit . . . 1762
2.2.7.7 Set negative limit . . . 1763
2.2.7.8 Get joint soft limit angles . . . 1763
2.2.7.9 Robot limit setting code example . . . 1763
2.2.7.10 Set robot collision detection method . . . 1763
2.2.7.11 Set collision detection start/stop in static mode . . . 1764
2.2.7.12 Code example for setting the robot collision detection method . . . 1764
2.2.7.13 Joint Torque Power Detection . . . 1764
2.2.7.14 Joint torque power detection code example . . . 1764
2.2.8 Robot Status Check . . . 1765
2.2.8.1 Get current joint position (angle). . . . 1765
2.2.8.2 Get current joint position in degrees of arc. . . . 1765
2.2.8.3 Get the joint feedback velocity . . . 1765
2.2.8.4 Get joint feedback acceleration . . . 1766
2.2.8.5 Get TCP command velocity - joint velocity . . . 1766
2.2.8.6 Get TCP feedback speed-composite speed . . . 1766
2.2.8.7 Get TCP command speed-composite speed . . . 1766
2.2.8.8 Get TCP feedback speed-split speed . . . 1767

2.2.8.9 Get current tool position . . . 1767
2.2.8.10 Get the current tool coordinate system number . . . 1767
2.2.8.11 Get the current tool coordinate system number . . . 1767
2.2.8.12 Get the current end flange position . . . 1768
2.2.8.13 Get the current joint torque . . . 1768
2.2.8.14 Get system time . . . 1768
2.2.8.15 Queries if the robot movement is complete . . . 1768
2.2.8.16 Queries the length of the robot motion queue cache . . . 1768
2.2.8.17 Get the robot emergency stop state . . . 1769
2.2.8.18 Get the state of the communication between the SDK and the robot . . . 1769
2.2.8.19 Get safety stop signal . . . 1769
2.2.8.20 Get robot joint actuator temperature (℃) . . . 1769
2.2.8.21 Get robot joint driver torque(Nm). . . . 1769
2.2.8.22 Get robot real-time status structure . . . 1770
2.2.8.23 Sample Robot Status Query Code . . . 1770
2.2.8.24 Inverse kinematics solution . . . 1772
2.2.8.25 Inverse kinematics solution (reference position) . . . 1772
2.2.8.26 Inverse Kinematics Solution, Cartesian Space Includes Extended Axis Position . . . 1773
2.2.8.27 Example Code for Inverse Kinematics Solution Including Extended Axis Position . 1773
2.2.8.28 Example Code for Inverse Kinematics Solution Including Extended Axis Position . 1774
2.2.8.29 Positive kinematics solution . . . 1774
2.2.8.30 Robot Forward and Reverse Kinematics Calculation Code Example . . . 1774
2.2.8.31 Querying Robot Teaching Management Point Data . . . 1775
2.2.8.32 Get the robot DH parameter compensation value . . . 1775
2.2.8.33 Get control box SN code . . . 1775
2.2.8.34 Query the robot teaching management point data code example . . . 1775
2.2.8.35 Get Tool Coordinate System by ID . . . 1776
2.2.8.36 Get Work Object Coordinate System by ID . . . 1776
2.2.8.37 Get External Tool Coordinate System by ID . . . 1776
2.2.8.38 Get Extended Axis Coordinate System by ID . . . 1777
2.2.8.39 Get Current Tool Coordinate System . . . 1777
2.2.8.40 Get Current Work Object Coordinate System . . . 1777
2.2.8.41 Get Current External Tool Coordinate System . . . 1777
2.2.8.42 Get Current Extended Axis Coordinate System . . . 1778
2.2.8.43 Get Current Extended Axis Coordinate System . . . 1778
2.2.9 Robot trajectory reproduction . . . 1779
2.2.9.1 Set the parameters for TPD trajectory recording . . . 1779
2.2.9.2 Start TPD trajectory recording . . . 1780
2.2.9.3 Stop TPD trajectory recording . . . 1780
2.2.9.4 Delete the TPD trajectory record . . . 1780
2.2.9.5 TPD trajectory preloading . . . 1780
2.2.9.6 Obtain the starting pose of the TPD trajectory . . . 1781
2.2.9.7 TPD trajectory reproduction . . . 1781
2.2.9.8 A sample code for robot TPD trajectory recording . . . 1781
2.2.9.9 Preprocessing of external trajectory files . . . 1782
2.2.9.10 External trajectory file trajectory reproduction . . . 1782
2.2.9.11 Obtain the starting position of the trajectory file's trajectory . . . 1782
2.2.9.12 Obtain the trajectory point number in the trajectory file . . . 1783
2.2.9.13 Set the running speed of the trajectory file trajectory . . . 1783
2.2.9.14 Set the force and torque during the trajectory file's trajectory operation . . . 1783
2.2.9.15 Set the force along the x direction during the trajectory's operation . . . 1783
2.2.9.16 Set the force along the y direction during the trajectory's operation . . . 1783
2.2.9.17 Set the force along the z direction during the trajectory's operation . . . 1784
2.2.9.18 Set the torque around the X-axis during the trajectory's operation . . . 1784

2.2.9.19 Set the torque around the Y-axis during the trajectory's operation . . . 1784
2.2.9.20 Set the torque around the Z-axis during the trajectory's operation . . . 1784
2.2.9.21 Upload the trajectory J file . . . 1784
2.2.9.22 Delete the trajectory J file . . . 1785
2.2.9.23 Sample code for reproducing robot trajectory J files . . . 1785
2.2.9.24 Trajectory preprocessing (trajectory forward-looking) . . . 1786
2.2.9.25 Trajectory reproduction (Trajectory Foresight) . . . 1786
2.2.9.26 Trajectory reproduction (trajectory forward-looking) code example . . . 1787
2.2.10 The use of robot WebAPP programs . . . 1787
2.2.10.1 Set the default job program to be automatically loaded upon startup . . . 1787
2.2.10.2 Load the specified job program . . . 1787
2.2.10.3 Get the name of the loaded job program . . . 1788
2.2.10.4 Obtain the execution line number of the current robot operation program . . . 1788
2.2.10.5 Run the currently loaded job program . . . 1788
2.2.10.6 Pause the currently running job program . . . 1788
2.2.10.7 Resume the currently suspended job procedure . . . 1788
2.2.10.8 Terminate the currently running job program . . . 1789
2.2.10.9 Obtain the execution status of the robot's operation program . . . 1789
2.2.10.10 Example of robot LUA program operation code . . . 1789
2.2.10.11 Download the Lua file . . . 1790
2.2.10.12 Upload the Lua file . . . 1790
2.2.10.13 Delete the Lua file . . . 1790
2.2.10.14 Get the names of all current lua files . . . 1790
2.2.10.15 Code example for uploading and downloading robot LUA files . . . 1791
2.2.11 Robot Peripherals . . . 1791
2.2.11.1 Configure Gripper . . . 1791
2.2.11.2 Get gripper configuration . . . 1791
2.2.11.3 Activate Gripper . . . 1792
2.2.11.4 Control gripper . . . 1792
2.2.11.5 Get gripper motion status . . . 1792
2.2.11.6 Get gripper activation status . . . 1793
2.2.11.7 Get Gripper Position . . . 1793
2.2.11.8 Get gripper speed . . . 1793
2.2.11.9 Get gripper current . . . 1793
2.2.11.10 Get gripper voltage . . . 1794
2.2.11.11 Get gripper temperature . . . 1794
2.2.11.12 Calculate pre-gripping point - vision . . . 1794
2.2.11.13 Calculate retreat point - vision . . . 1794
2.2.11.14 Robot Gripper Operation Code Example . . . 1795
2.2.11.15 Get the number of rotations of the rotating gripper . . . 1796
2.2.11.16 Get the rotation speed percentage of the rotating gripper . . . 1796
2.2.11.17 Get the rotation torque percentage of the rotating gripper . . . 1797
2.2.11.18 Example of retrieving the rotational gripper status code . . . 1797
2.2.11.19 Drive belt start/stop . . . 1797
2.2.11.20 Record IO detection points . . . 1798
2.2.11.21 Record point A . . . 1798
2.2.11.22 Record reference point . . . 1798
2.2.11.23 Record Point B . . . 1798
2.2.11.24 Conveyor belt workpiece IO detection . . . 1798
2.2.11.25 Get Object Current Position . . . 1799
2.2.11.26 Start conveyor tracking . . . 1799
2.2.11.27 Conveyor tracking stop . . . 1799
2.2.11.28 Drive Belt Parameter Configuration . . . 1799
2.2.11.29 Set conveyor belt pickup point compensation . . . 1800

2.2.11.30 Conveyor belt tracking linear motion . . . 1800
2.2.11.31 Conveyor communication input detection . . . 1800
2.2.11.32 Conveyor Communication Input Detection Trigger . . . 1800
2.2.11.33 Conveyor Communication Input Detection Trigger Example Program . . . 1801
2.2.11.34 Robot Conveyor Belt Operation Example Program . . . 1802
2.2.11.35 End Sensor Configuration . . . 1804
2.2.11.36 Get endpoint sensor configuration . . . 1804
2.2.11.37 End-of-line sensor activation . . . 1804
2.2.11.38 End Sensor Register Write . . . 1804
2.2.11.39 End Sensor Code Example . . . 1805
2.2.11.40 Obtain robot peripheral protocol . . . 1805
2.2.11.41 Set robot peripheral protocol . . . 1805
2.2.11.42 Example program for setting robot peripheral protocol . . . 1806
2.2.11.43 Get end-point communication parameters . . . 1806
2.2.11.44 Set terminal communication parameters . . . 1806
2.2.11.45 Set terminal file transfer type . . . 1806
2.2.11.46 Set enable terminal LUA execution . . . 1807
2.2.11.47 End LUA file exception error recovery . . . 1807
2.2.11.48 Get the enable status of the terminal LUA execution . . . 1807
2.2.11.49 Set the enable type of the terminal LUA terminal device . . . 1807
2.2.11.50 Get the enable type of the end-of-line LUA device . . . 1807
2.2.11.51 Get the currently configured end device . . . 1808
2.2.11.52 Set enable gripper action control functionality . . . 1808
2.2.11.53 Get enable claw action control function . . . 1808
2.2.11.54 Writing robot Ethercat slave file . . . 1809
2.2.11.55 Upload terminal Lua open protocol file . . . 1809
2.2.11.56 Robot Ethercat slave enters boot mode . . . 1809
2.2.11.57 Robot End-of-Arm LUA File Operation Code Example . . . 1809
2.2.11.58 Get SmartTool button status . . . 1811
2.2.11.59 Code example . . . 1811
2.2.11.60 Upload Open Protocol Lua File . . . 1811
2.2.11.61 Get Slave Board Parameters . . . 1812
2.2.11.62 Write Slave DO . . . 1812
2.2.11.63 Write Slave AO . . . 1812
2.2.11.64 Read Slave DI . . . 1812
2.2.11.65 Read Slave AI . . . 1813
2.2.11.66 Wait for Extended DI Input . . . 1813
2.2.11.67 Wait for Extended AI Input . . . 1813
2.2.11.68 Slave Mode Related Interface Command Code Example . . . 1813
2.2.11.69 Control Array Sucker . . . 1815
2.2.11.70 Get Array Sucker Status . . . 1815
2.2.11.71 Wait for Sucker Status . . . 1815
2.2.11.72 Array Sucker Control Command Code Example . . . 1816
2.2.11.73 Laser peripheral on/off function . . . 1817
2.2.11.74 Laser tracking start/stop function . . . 1817
2.2.11.75 Laser positioning - fixed direction . . . 1817
2.2.11.76 Laser positioning - arbitrary direction . . . 1817
2.2.11.77 Laser positioning stop . . . 1818
2.2.11.78 Laser IP configuration . . . 1818
2.2.11.79 Laser peripheral sampling period configuration . . . 1818
2.2.11.80 Laser peripheral driver loading . . . 1818
2.2.11.81 Laser Peripheral Driver Unloading . . . 1819
2.2.11.82 Laser Weld Seam Trajectory Recording . . . 1819
2.2.11.83 Laser Weld Seam Trajectory Replay . . . 1819

2.2.11.84 Laser Tracking Replay . . . 1819
2.2.11.85 Laser Weld Seam Trajectory Recording and Replay . . . 1820
2.2.11.86 Move to Laser Record Start Point . . . 1820
2.2.11.87 Move to Laser Record End Point . . . 1820
2.2.11.88 Move to Laser Sensor Positioning Point . . . 1821
2.2.11.89 Get laser sensor positioning point coordinate information . . . 1821
2.2.11.90 Laser Peripheral Sensor Parameter Configuration and Debugging Code Example . . . 1821
2.2.11.91 Laser Trajectory Scanning and Trajectory Replay Code Example . . . 1822
2.2.11.92 Laser Positioning and Real-time Tracking Code Example . . . 1823
2.2.11.93 Extended Axis and Robot Synchronized Laser Tracking Code Example . . . 1824
2.2.12 Machine Manpower Control . . . 1825
2.2.12.1 Force Sensor Configuration . . . 1825
2.2.12.2 Get the force transducer configuration . . . 1825
2.2.12.3 Force sensor activation . . . 1825
2.2.12.4 Force Transducer Zeroing . . . 1825
2.2.12.5 Set the force transducer reference coordinate system . . . 1826
2.2.12.6 Set the force transducer lower load weight . . . 1826
2.2.12.7 Set the force sensor payload center of mass . . . 1826
2.2.12.8 Get the force sensor pay load weight . . . 1826
2.2.12.9 Get force sensor payload center of mass . . . 1826
2.2.12.10 Automatic zeroing of the force sensor. . . . 1827
2.2.12.11 Get force/torque data in reference coordinate system. . . . 1827
2.2.12.12 Get force sensor raw force/torque data . . . 1827
2.2.12.13 Force Transducer Configuration and Auto-Zero Code Example . . . 1827
2.2.12.14 Load Weight Recognition Record . . . 1828
2.2.12.15 Load weight recognition calculation . . . 1829
2.2.12.16 Load center of mass identification record . . . 1829
2.2.12.17 Load center of mass identification calculation . . . 1829
2.2.12.18 Force Transducer Load Recognition Code Example . . . 1829
2.2.12.19 Collision Guard . . . 1830
2.2.12.20 Collision Guard Code Example . . . 1831
2.2.12.21 Constant force control . . . 1832
2.2.12.22 Constant force control with damping code example . . . 1832
2.2.12.23 Rotational Insertion . . . 1833
2.2.12.24 Robot Force Sensor Rotational Insertion Code Example . . . 1834
2.2.12.25 Flex control on . . . 1835
2.2.12.26 Flex control off . . . 1835
2.2.12.27 Sample Flex Control Code . . . 1835
2.2.12.28 Load recognition initialization . . . 1837
2.2.12.29 Load identification variable initialization . . . 1837
2.2.12.30 Load Identification Main Program . . . 1837
2.2.12.31 Get the load identification result . . . 1837
2.2.12.32 Robot Load Identification Code Example . . . 1838
2.2.12.33 Force sensor assisted drag . . . 1838
2.2.12.34 Get the state of the force sensor drag switch . . . 1839
2.2.12.35 The force sensor is automatically turned on after the error is cleared . . . 1839
2.2.12.36 Force Sensor Assisted Drag Code Example . . . 1839
2.2.12.37 Setting up the six-dimensional force and joint impedance hybrid drag switch and parameters . . . 1840
2.2.12.38 Force Sensor Assisted Drag Code Example . . . 1840
2.2.12.39 Setting up the Wire Seek Expansion IO Port . . . 1841
2.2.12.40 Impedance Control Start/Stop . . . 1841
2.2.12.41 Robot Impedance Control Start/Stop Code Example . . . 1841
2.2.12.42 Enable torque compensation function and compensation coefficient . . . 1843

2.2.13 Extended axis . . . 1843
2.2.13.1 Set 485 extended axis parameters . . . 1843
2.2.13.2 Get 485 extended axis configuration parameters . . . 1843
2.2.13.3 Set 485 extended axis enable/disable . . . 1844
2.2.13.4 Set 485 extension axis control mode . . . 1844
2.2.13.5 Set the target position of the 485 extended axis (position mode) . . . 1844
2.2.13.6 Set the target speed of the 485 extended axis (speed mode) . . . 1844
2.2.13.7 Set the target torque of the 485 extended axis (torque mode) – Not available yet . . . 1845
2.2.13.8 Set 485 extended axis zero . . . 1845
2.2.13.9 Clear 485 extended axis error information . . . 1845
2.2.13.10 Get 485 extended axis servo status . . . 1846
2.2.13.11 Set the 485 extended axis data axis number in the status feedback . . . 1846
2.2.13.12 Set the acceleration and deceleration of the 485 extended axis . . . 1846
2.2.13.13 Set 485 extended axis emergency stop acceleration and deceleration . . . 1847
2.2.13.14 Get 485 extended axis movement acceleration and deceleration . . . 1847
2.2.13.15 Get 485 extended axis emergency stop acceleration and deceleration . . . 1847
2.2.13.16 Extended axis control code example . . . 1847
2.2.13.17 UDP extended axis communication parameter configuration . . . 1849
2.2.13.18 Get UDP extended axis communication parameter configuration . . . 1850
2.2.13.19 Load UDP communication . . . 1850
2.2.13.20 Unload UDP communication . . . 1850
2.2.13.21 Restore connection after abnormal disconnection of UDP extended axis communication . . . 1851
2.2.13.22 Close communication after abnormal disconnection of UDP extended axis communication . . . 1851
2.2.13.23 UDP extended axis parameter configuration . . . 1851
2.2.13.24 Set the installation position of the extended axis . . . 1852
2.2.13.25 Set extended axis system DH parameter configuration . . . 1852
2.2.13.26 UDP extended axis enable . . . 1852
2.2.13.27 UDP extended axis reset . . . 1853
2.2.13.28 UDP extended axis jog start . . . 1853
2.2.13.29 UDP extended axis jog stop . . . 1853
2.2.13.30 UDP extended axis configuration and jog code example . . . 1854
2.2.13.31 Set the reference point of the extended axis coordinate system - four-point method . . . 1855
2.2.13.32 Calculate extended axis coordinate system - four-point method . . . 1855
2.2.13.33 Apply extended axis coordinate system . . . 1855
2.2.13.34 Set the calibration reference point in the end coordinate system of the positioner . . . 1856
2.2.13.35 Set the reference point of the positioner coordinate system . . . 1856
2.2.13.36 Positioner coordinate system calculation - four-point method . . . 1856
2.2.13.37 Get extended axis coordinate system . . . 1856
2.2.13.38 Extended axis coordinate system calibration code example . . . 1857
2.2.13.39 Extended axis motion via UDP . . . 1859
2.2.13.40 UDP extended axis movement code example . . . 1859
2.2.13.41 UDP extended axis and robot joint motion synchronization . . . 1860
2.2.13.42 Code example . . . 1860
2.2.13.43 UDP extended axis and robot linear motion synchronization . . . 1862
2.2.13.44 Code example . . . 1862
2.2.13.45 UDP extended axis and robot arc motion synchronization . . . 1864
2.2.13.46 Code example . . . 1865
2.2.13.47 Set extended DO . . . 1866
2.2.13.48 Set extended AO . . . 1867
2.2.13.49 Set extended DI input filter time . . . 1867
2.2.13.50 Set extended AI input filter time . . . 1867
2.2.13.51 Wait for extended DI input . . . 1867

2.2.13.52 Wait for extended AI input . . . 1868
2.2.13.53 Get extended DI value . . . 1868
2.2.13.54 Get extended AI value . . . 1868
2.2.13.55 Extended IO code example . . . 1869
2.2.13.56 Enable movable device . . . 1870
2.2.13.57 Stop movable device movement . . . 1870
2.2.13.58 Reset the movable device . . . 1870
2.2.13.59 Movable device moves in a straight line . . . 1870
2.2.13.60 Circular motion of a movable device . . . 1871
2.2.13.61 Code example . . . 1871
2.2.13.62 Set the synchronous motion strategy of the extension axis and the robot . . . 1872
2.2.13.63 Code example for setting up the extended axis to move synchronously with the robot 1872
2.2.14 Robotic Welding . . . 1873
2.2.14.1 Setting the welding process parameters . . . 1873
2.2.14.2 Get weld process curve parameters . . . 1873
2.2.14.3 Setting the weld current in relation to the output analog . . . 1873
2.2.14.4 Set Welding Voltage and Output Analog Relation . . . 1874
2.2.14.5 Get the relationship between the welding current and the output analog . . . 1874
2.2.14.6 Get Welding Voltage Relation to Output Analog . . . 1875
2.2.14.7 Set weld current . . . 1875
2.2.14.8 Set Welding Voltage . . . 1875
2.2.14.9 Set the swing parameter . . . 1876
2.2.14.10 Sample code for setting welding parameters . . . 1876
2.2.14.11 Setting the swing parameters on the fly . . . 1878
2.2.14.12 Set parameters for detecting unexpected interruptions of the robot's welding arc . . . 1879
2.2.14.13 Get parameters for detecting accidental interruptions of the robot's welding arc . . . 1879
2.2.14.14 Set the robot welding interruption recovery parameter . . . 1879
2.2.14.15 Get the robot welding break recovery parameter . . . 1879
2.2.14.16 Set welder control mode extension DO port . . . 1880
2.2.14.17 Set weld machine control mode . . . 1880
2.2.14.18 Weld start . . . 1880
2.2.14.19 Weld End . . . 1880
2.2.14.20 Oscillation start . . . 1881
2.2.14.21 WeaveStart(int weaveNum); int WeaveStart(int weaveNum). . . . 1881
2.2.14.22 Forward Wire Feed . . . 1881
2.2.14.23 Reverse wire feed . . . 1881
2.2.14.24 Wire Feed . . . 1882
2.2.14.25 Setting the robot to resume welding after an interruption . . . 1882
2.2.14.26 Sets the robot to exit welding after a weld break . . . 1882
2.2.14.27 Code example . . . 1882
2.2.14.28 Start of segment welding . . . 1883
2.2.14.29 Robot segment welding code example . . . 1883
2.2.14.30 Simulate the start of the swing . . . 1884
2.2.14.31 End of swing simulation . . . 1884
2.2.14.32 Start trajectory detection warning (no motion) . . . 1884
2.2.14.33 End trajectory detection warning (no motion) . . . 1885
2.2.14.34 Weave fading start . . . 1885
2.2.14.35 Weave fade end . . . 1885
2.2.14.36 Sample Welding Code for Robot Weave Change . . . 1885
2.2.14.37 Extended IO-Configuration of Welder Gas Detection Signal . . . 1886
2.2.14.38 Extended IO-Configuration of the welder arc start signal . . . 1886
2.2.14.39 Extended IO-Configuring the Welder Reverse Wire Feed Signal . . . 1887
2.2.14.40 Extended IO-Configuration Welder Forward Wire Feed Signal . . . 1887
2.2.14.41 Extended IO-Configuration Welder Arc Start Success Signal . . . 1887

2.2.14.42 Extended IO-Configuration Welder Ready Signal . . . 1887
2.2.14.43 Extended IO-Configure weld interrupt recovery signal . . . 1887
2.2.14.44 Set Extended IO Weld Signal Code Example . . . 1888
2.2.14.45 Arc tracking control . . . 1888
2.2.14.46 Arc tracking AI passband selection . . . 1889
2.2.14.47 Arc Trace + Multi-Layer Multi-Channel Compensation on . . . 1889
2.2.14.48 ArcWeldTrace + MultiLayerMultiChannelCompensation OFF . . . 1889
2.2.14.49 Offset Coordinate Change - Multi-Layer Multi-Pass Welding . . . 1889
2.2.14.50 Multi-layer multi-pass welding arc tracking code example . . . 1890
2.2.14.51 Arc Tracking Welder Current Feedback AI Channel Selection . . . 1893
2.2.14.52 Arc tracking welder voltage feedback AI channel selection . . . 1893
2.2.14.53 Arc tracking welder current feedback conversion parameters . . . 1893
2.2.14.54 Arc Trace Welder Voltage Feedback Conversion Parameters . . . 1894
2.2.14.55 Arc Trace Code Example . . . 1894
2.2.14.56 Start of wire position finding . . . 1895
2.2.14.57 End of wire search . . . 1895
2.2.14.58 Calculate the wire seek offset . . . 1896
2.2.14.59 Wait for the wire search to complete . . . 1896
2.2.14.60 Write wire search contact to database. . . . 1896
2.2.14.61 Robot Welding Wire Position Finding Code Example . . . 1897
2.2.14.62 Setting the end of the welding voltage gradient . . . 1898
2.2.14.63 Set weld voltage gradual change end . . . 1899
2.2.14.64 Sets the weld current gradual change end . . . 1899
2.2.14.65 Set weld current gradual change end . . . 1899
2.2.14.66 Robot Welding Current Voltage Gradual Change Code Example . . . 1899
2.2.14.67 Set custom weaving parameters . . . 1901
2.2.14.68 Get custom weaving parameters . . . 1901
2.2.14.69 Custom Weaving Parameters Code Example . . . 1901
2.2.15 Other interfaces . . . 1903
2.2.15.1 Get SSH public key . . . 1903
2.2.15.2 Send SCP command . . . 1903
2.2.15.3 Calculate the MD5 value of a file in a specified path . . . 1903
2.2.15.4 Robot SSH and MD5 command code example . . . 1904
2.2.15.5 Set robot 20004 port feedback cycle . . . 1904
2.2.15.6 Get the robot's 20004 port feedback period . . . 1904
2.2.15.7 Robot 20004 port status feedback period configuration code example . . . 1905
2.2.15.8 Robot Software Upgrade . . . 1905
2.2.15.9 Get robot software upgrade status . . . 1905
2.2.15.10 Robot software upgrade code example . . . 1905
2.2.15.11 Download point table . . . 1906
2.2.15.12 Upload point table . . . 1906
2.2.15.13 Point Table Update Lua Program . . . 1906
2.2.15.14 Switch point tables and apply . . . 1906
2.2.15.15 Robot point table operation code example . . . 1907
2.2.15.16 Controller log download . . . 1907
2.2.15.17 All Data Source Download . . . 1907
2.2.15.18 Data backup package download . . . 1907
2.2.15.19 Download controller data code example . . . 1908
2.2.15.20 Robot Operating System Upgrade (LA Control Box) . . . 1908
2.2.15.21 Get Robot Operating System Upgrade Result (LA Control Box) . . . 1908
2.2.15.22 Set encoder upgrade . . . 1908
2.2.15.23 Set joint firmware upgrade . . . 1909
2.2.15.24 Set firmware upgrade for control box . . . 1909
2.2.15.25 Set end firmware upgrade . . . 1909

2.2.15.26 Upgrade of the joint full parameter configuration file . . . 1909
2.2.15.27 Example of upgrading code for robot from firmware . . . 1910
2.2.15.28 Robot MCU log generation . . . 1910
2.2.16 Appendix . . . 1910
2.2.16.1 Source Code Download . . . 1910
2.2.16.2 Windows Platform Source Code Compilation . . . 1913
2.2.16.2.1 C# SDK Compilation . . . 1913
2.2.16.2.2 C# SDK Usage . . . 1919
2.2.16.3 Notes . . . 1921
2.2.16.3.1 Potential Issues . . . 1921
2.2.16.3.1.1 Handling No Effect After Code Updates . . . 1921
2.2.16.3.1.2 Error Codes . . . 1921
2.3 Java . . . 1921
2.3.1 Version Update Description . . . 1924
2.3.2 Data Structure Description . . . 1925
2.3.2.1 Joint Position Data Type . . . 1925
2.3.2.2 Cartesian Space Position Data Type . . . 1925
2.3.2.3 Euler Angle Attitude Data Type . . . 1926
2.3.2.4 Cartesian Space Pose Data Type . . . 1926
2.3.2.5 Extended Axis Position Data Type . . . 1927
2.3.2.6 Force Torque Sensor Data Type . . . 1928
2.3.2.7 Spiral Parameter Data Type . . . 1928
2.3.2.8 Extended Axis Status Type . . . 1929
2.3.2.9 Sensor Type . . . 1929
2.3.2.10 485 Extended Axis Configuration . . . 1930
2.3.2.11 Servo Controller Status . . . 1930
2.3.2.12 Welding Breakoff Status . . . 1931
2.3.2.13 UDP Extended Axis Communication Parameters . . . 1931
2.3.2.14 Robot State Feedback Structure Type . . . 1931
2.3.3 Robot basics . . . 1934
2.3.3.1 Instantiate robot . . . 1934
2.3.3.2 Establish communication with controller . . . 1934
2.3.3.3 Close communication with robot . . . 1934
2.3.3.4 Query sdk version . . . 1934
2.3.3.5 Get controller ip . . . 1935
2.3.3.6 Control robot to enter or exit drag teaching mode . . . 1935
2.3.3.7 Query whether robot is in drag teaching mode . . . 1935
2.3.3.8 Control robot enable or disable . . . 1935
2.3.3.9 Control robot manual/auto mode switch . . . 1935
2.3.3.10 Shut down robot os . . . 1936
2.3.3.11 Set robot communication reconnect parameters . . . 1936
2.3.3.12 Initialize log parameters . . . 1936
2.3.3.13 Set log filter level . . . 1936
2.3.3.14 Robot basic control code example . . . 1937
2.3.3.15 Get robot software version . . . 1937
2.3.3.16 Get robot hardware version . . . 1938
2.3.3.17 Get robot firmware version . . . 1938
2.3.3.18 Get robot software/firmware version code example . . . 1938
2.3.4 Robot movement . . . 1939
2.3.4.1 Jog movement . . . 1939
2.3.4.2 Jog deceleration stop . . . 1940
2.3.4.3 Jog immediate stop . . . 1940
2.3.4.4 Robot jog control code example . . . 1940
2.3.4.5 Joint space movement . . . 1941

2.3.4.6 Joint space motion (automatic forward kinematics calculation) . . . 1941
2.3.4.7 Cartesian space linear movement . . . 1942
2.3.4.8 Cartesian space linear motion (automatic inverse kinematics calculation) . . . 1942
2.3.4.9 Cartesian Space Linear Motion (Added velAccParamMode parameter for velocity and acceleration modes) . . . 1943
2.3.4.10 Cartesian Space Linear Motion (Overload Function 1, Added blendMode) . . . 1944
2.3.4.11 Cartesian Space Linear Motion (Overload Function 2, No Joint Position Input Required) . . . 1944
2.3.4.12 Cartesian space circular movement . . . 1945
2.3.4.13 Cartesian Space Arc Motion (Added velAccParamMode parameter for velocity and acceleration modes) . . . 1946
2.3.4.14 Cartesian Space Arc Motion (Overload Function 1, No Joint Position Input Required) 1947
2.3.4.15 Cartesian space full circle movement . . . 1947
2.3.4.16 Cartesian space full circle motion (automatic inverse kinematics calculation) . . . 1948
2.3.4.17 Cartesian Space Full Circle Motion (Added velAccParamMode parameter for velocity and acceleration modes) . . . 1949
2.3.4.18 Cartesian Space Full Circle Motion (Overload Function 1, No Joint Position Input Required) . . . 1950
2.3.4.19 Cartesian space point-to-point movement . . . 1950
2.3.4.20 Basic robot movement command code example . . . 1951
2.3.4.21 Cartesian space spiral movement . . . 1952
2.3.4.22 Cartesian space spiral motion (automatic inverse kinematics calculation) . . . 1952
2.3.4.23 Spiral movement code example . . . 1953
2.3.4.24 Servo movement start . . . 1954
2.3.4.25 Servo movement end . . . 1954
2.3.4.26 Joint space servo mode movement . . . 1954
2.3.4.27 Joint space servo mode movement example program . . . 1954
2.3.4.28 Joint torque control start . . . 1955
2.3.4.29 Joint torque control . . . 1955
2.3.4.30 Joint torque control end . . . 1956
2.3.4.31 Joint space servo mode movement example program . . . 1956
2.3.4.32 Joint Torque Control Code Example with Overspeed Protection . . . 1956
2.3.4.33 Cartesian Space Servo Mode Motion . . . 1957
2.3.4.34 Cartesian Space Servo Mode Motion Code Example . . . 1958
2.3.4.35 Spline movement start . . . 1958
2.3.4.36 Joint movement PTP . . . 1958
2.3.4.37 Joint space spline motion (automatic forward kinematics calculation) . . . 1959
2.3.4.38 Spline movement end . . . 1959
2.3.4.39 Spline movement code example . . . 1959
2.3.4.40 New spline movement start . . . 1960
2.3.4.41 New spline command point . . . 1960
2.3.4.42 New spline command point (automatic inverse kinematics calculation) . . . 1961
2.3.4.43 New spline movement end . . . 1961
2.3.4.44 New spline movement code example . . . 1961
2.3.4.45 Stop movement . . . 1962
2.3.4.46 Pause movement . . . 1962
2.3.4.47 Resume movement . . . 1962
2.3.4.48 Movement pause, resume, stop code example . . . 1963
2.3.4.49 Point global offset start . . . 1963
2.3.4.50 Point global offset end . . . 1964
2.3.4.51 Point offset code example . . . 1964
2.3.4.52 Controller AO flying start . . . 1965
2.3.4.53 Controller AO flying stop . . . 1965
2.3.4.54 End effector AO flying start . . . 1965

2.3.4.55 End effector AO flying stop . . . 1965
2.3.4.56 AO flying code example . . . 1966
2.3.4.57 Start Ptp movement FIR filtering . . . 1967
2.3.4.58 Close Ptp movement FIR filtering . . . 1967
2.3.4.59 Start LIN, ARC movement FIR filtering . . . 1967
2.3.4.60 Close LIN, ARC movement FIR filtering . . . 1967
2.3.4.61 FIR filtering code example . . . 1968
2.3.4.62 Acceleration smoothing enable . . . 1968
2.3.4.63 Acceleration smoothing disable . . . 1969
2.3.4.64 Acceleration smoothing code example . . . 1969
2.3.4.65 Specified pose speed enable . . . 1969
2.3.4.66 Specified pose speed disable . . . 1970
2.3.4.67 Robot specified pose speed code example . . . 1970
2.3.4.68 Start singular pose protection . . . 1970
2.3.4.69 Stop singular pose protection . . . 1971
2.3.4.70 Robot singular pose protection code example . . . 1971
2.3.4.71 Clear Motion Command Queue . . . 1971
2.3.4.72 Move to Intersecting Line Start Point . . . 1972
2.3.4.73 Intersecting Line Motion . . . 1972
2.3.4.74 Robot Intersecting Line Motion Code Example . . . 1973
2.3.4.75 Stationary Air Motion . . . 1974
2.3.4.76 Stationary Air Motion Code Example . . . 1974
2.3.5 Robot IO . . . 1974
2.3.5.1 Set control box digital output . . . 1974
2.3.5.2 Set tool digital output . . . 1975
2.3.5.3 Set control box analog output . . . 1975
2.3.5.4 Set tool analog output . . . 1975
2.3.5.5 Digital and analog output setting example . . . 1975
2.3.5.6 Get control box digital input . . . 1977
2.3.5.7 Get tool digital input . . . 1977
2.3.5.8 Get control box analog input . . . 1977
2.3.5.9 Get tool analog input . . . 1977
2.3.5.10 Get robot end-point record button state . . . 1978
2.3.5.11 Get robot end DO output state . . . 1978
2.3.5.12 Get robot controller DO output state . . . 1978
2.3.5.13 Get robot DI/DO state example . . . 1978
2.3.5.14 Wait for control box digital input . . . 1979
2.3.5.15 Wait for multiple control box digital inputs . . . 1979
2.3.5.16 Wait for tool digital input . . . 1980
2.3.5.17 Wait for control box analog input . . . 1980
2.3.5.18 Wait for tool analog input . . . 1980
2.3.5.19 Wait for digital/analog input signal example . . . 1981
2.3.5.20 Set Whether Control Box DO Output Resets After Stop/Pause . . . 1981
2.3.5.21 Set Whether Control Box AO Output Resets After Stop/Pause . . . 1981
2.3.5.22 Set Whether End Tool DO Output Resets After Stop/Pause . . . 1982
2.3.5.23 Set Whether End Tool AO Output Resets After Stop/Pause . . . 1982
2.3.5.24 Set Whether Extended DO Output Resets After Stop/Pause . . . 1982
2.3.5.25 Set Whether Extended AO Output Resets After Stop/Pause . . . 1982
2.3.5.26 Set Whether SmartTool Output Resets After Stop/Pause . . . 1983
2.3.5.27 Code Example for Setting Output Reset After Lua Program Stop/Pause . . . 1983
2.3.6 Robot common settings . . . 1983
2.3.6.1 Set tool reference point - six point method . . . 1983
2.3.6.2 Calculate tool coordinate system - six point method . . . 1984
2.3.6.3 Set tool reference point - four point method . . . 1984

2.3.6.4 Calculate tool coordinate system - four point method . . . 1984
2.3.6.5 Calculate tool coordinate system based on point information . . . 1984
2.3.6.6 Set tool coordinate system . . . 1985
2.3.6.7 Set tool coordinate system list . . . 1985
2.3.6.8 Get current tool coordinate system . . . 1985
2.3.6.9 Robot tool coordinate system operation code example . . . 1985
2.3.6.10 Set external tool reference point - six point method . . . 1987
2.3.6.11 Calculate external tool coordinate system - six point method . . . 1987
2.3.6.12 Set external tool coordinate system . . . 1987
2.3.6.13 Set external tool coordinate system list . . . 1987
2.3.6.14 Robot external tool coordinate system operation code example . . . 1988
2.3.6.15 Set workpiece reference point - three point method . . . 1988
2.3.6.16 Calculate workpiece coordinate system . . . 1989
2.3.6.17 Set workpiece coordinate system . . . 1989
2.3.6.18 Set workpiece coordinate system list . . . 1989
2.3.6.19 Calculate workpiece coordinate system based on point information . . . 1989
2.3.6.20 Get current workpiece coordinate system . . . 1990
2.3.6.21 Workpiece coordinate system operation code example . . . 1990
2.3.6.22 Set global speed . . . 1991
2.3.6.23 Set robot acceleration . . . 1991
2.3.6.24 Get robot default speed . . . 1991
2.3.6.25 Set end load weight . . . 1991
2.3.6.26 Set end load center of mass coordinates . . . 1991
2.3.6.27 Set End Effector Load Center of Mass Coordinates . . . 1992
2.3.6.28 Get current load weight . . . 1992
2.3.6.29 Get current load center of mass . . . 1992
2.3.6.30 Set robot installation method . . . 1992
2.3.6.31 Set robot installation angle . . . 1992
2.3.6.32 Get robot installation angle . . . 1993
2.3.6.33 Set system variable value . . . 1993
2.3.6.34 Get system variable value . . . 1993
2.3.6.35 Robot common settings code example . . . 1993
2.3.6.36 Joint friction compensation switch . . . 1994
2.3.6.37 Set joint friction compensation coefficient - standard installation . . . 1995
2.3.6.38 Set joint friction compensation coefficient - side installation . . . 1995
2.3.6.39 Set joint friction compensation coefficient - inverted installation . . . 1995
2.3.6.40 Set joint friction compensation coefficient - free installation . . . 1995
2.3.6.41 Robot set joint friction compensation code example . . . 1995
2.3.6.42 Query robot error code . . . 1996
2.3.6.43 Error state clear . . . 1996
2.3.6.44 Robot fault state get and clear error code example . . . 1996
2.3.6.45 Set wide voltage control box temperature and fan speed monitoring parameters . . . 1997
2.3.6.46 Get wide voltage control box temperature and fan speed monitoring parameters . . . 1997
2.3.6.47 Wide voltage control box temperature and fan current state get code example . . . 1997
2.3.6.48 Set Focus Calibration Point . . . 1998
2.3.6.49 Compute Focus Calibration Result . . . 1998
2.3.6.50 Start focus following . . . 1998
2.3.6.51 Stop Focus Following . . . 1999
2.3.6.52 Set Focus Position . . . 1999
2.3.6.53 Focus Following Code Example . . . 1999
2.3.6.54 Joint Torque Sensor Sensitivity Calibration Function Enabled . . . 2001
2.3.6.55 Joint Torque Sensor Sensitivity Data Acquisition . . . 2001
2.3.6.56 Get Joint Torque Sensor Sensitivity Calibration Results . . . 2001
2.3.6.57 Get Joint Torque Sensor Hysteresis Error . . . 2001

2.3.6.58 Get Joint Torque Sensor Repeatability . . . 2002
2.3.6.59 Set Joint Force Sensor Parameters . . . 2002
2.3.6.60 Joint Torque Sensor Sensitivity Auto-Calibration Code Example . . . 2002
2.3.6.61 Retrieve the number of error frames for the robot's 8 slave ports . . . 2005
2.3.6.62 Reset slave port error frames . . . 2005
2.3.6.63 Example for retrieving slave port error frame codes . . . 2005
2.3.6.64 Set speed feedforward coefficients for each axis . . . 2006
2.3.6.65 Retrieve speed feedforward ratios for each axis . . . 2007
2.3.6.66 Robot Speed Feedforward Coefficient Code Example . . . 2007
2.3.6.67 Photoelectric Sensor TCP Calibration - Compute Tool RPY . . . 2007
2.3.6.68 Photoelectric Sensor TCP Calibration - Compute Tool XYZ . . . 2007
2.3.6.69 Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position . . . 2008
2.3.6.70 Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position . . . 2008
2.3.6.71 Photoelectric Sensor TCP Calibration - Get Tool Center Point Position . . . 2008
2.3.6.72 Photoelectric Sensor TCP Calibration . . . 2008
2.3.6.73 Photoelectric Sensor TCP Calibration Code Example . . . 2009
2.3.7 Robot safety settings . . . 2009
2.3.7.1 Set collision level . . . 2009
2.3.7.2 Set post-collision strategy . . . 2009
2.3.7.3 Custom collision detection threshold function start . . . 2010
2.3.7.4 Custom collision detection threshold function end . . . 2010
2.3.7.5 Robot collision level setting code example . . . 2010
2.3.7.6 Set positive limit . . . 2011
2.3.7.7 Set negative limit . . . 2011
2.3.7.8 Get joint soft limit angles . . . 2012
2.3.7.9 Robot limit setting code example . . . 2012
2.3.7.10 Set robot collision detection method . . . 2012
2.3.7.11 Set static collision detection on/off . . . 2013
2.3.7.12 Robot collision detection method code example . . . 2013
2.3.7.13 Joint torque power detection . . . 2013
2.3.7.14 Joint torque power detection code example . . . 2013
2.3.8 Robot status inquiry . . . 2014
2.3.8.1 Get current joint positions (degrees) . . . 2014
2.3.8.2 Get joint feedback speed (deg/s) . . . 2014
2.3.8.3 Get joint feedback acceleration . . . 2014
2.3.8.4 Get TCP command composite speed . . . 2015
2.3.8.5 Get TCP feedback composite speed . . . 2015
2.3.8.6 Get TCP command speed . . . 2015
2.3.8.7 Get TCP feedback speed . . . 2015
2.3.8.8 Get current tool pose . . . 2016
2.3.8.9 Get current tool coordinate system number . . . 2016
2.3.8.10 Get current workpiece coordinate system number . . . 2016
2.3.8.11 Get current end flange pose . . . 2016
2.3.8.12 Get current joint torque . . . 2017
2.3.8.13 Get system time . . . 2017
2.3.8.14 Check if robot motion is completed . . . 2017
2.3.8.15 Query robot motion queue buffer length . . . 2017
2.3.8.16 Get robot emergency stop state . . . 2017
2.3.8.17 Get SDK-robot communication state . . . 2018
2.3.8.18 Get safety stop signal . . . 2018
2.3.8.19 Get robot joint driver temperature (°C) . . . 2018
2.3.8.20 Get robot joint driver torque (Nm) . . . 2018
2.3.8.21 Get robot real-time state structure . . . 2018
2.3.8.22 Robot status inquiry code example . . . 2019

2.3.8.23 Inverse kinematics calculation . . . 2020
2.3.8.24 Inverse kinematics calculation (reference position) . . . 2020
2.3.8.25 Inverse Kinematics Solution, Cartesian Space Includes Extended Axis Position . . . 2021
2.3.8.26 Inverse Kinematics Solution Including Extended Axis Position Code Example . . . 2021
2.3.8.27 Check if inverse kinematics has solution . . . 2022
2.3.8.28 Forward kinematics calculation . . . 2022
2.3.8.29 Robot forward/inverse kinematics calculation code example . . . 2022
2.3.8.30 Query robot teaching management point data . . . 2023
2.3.8.31 Get robot DH parameter compensation values . . . 2023
2.3.8.32 Get controller SN code . . . 2023
2.3.8.33 Query robot teaching management point data code example . . . 2023
2.3.8.34 Get Tool Coordinate System by ID . . . 2024
2.3.8.35 Get Work Object Coordinate System by ID . . . 2024
2.3.8.36 Get External Tool Coordinate System by ID . . . 2024
2.3.8.37 Get Extended Axis Coordinate System by ID . . . 2025
2.3.8.38 Get Current Tool Coordinate System . . . 2025
2.3.8.39 Get Current Work Object Coordinate System . . . 2025
2.3.8.40 Get Current External Tool Coordinate System . . . 2025
2.3.8.41 Get Current Extended Axis Coordinate System . . . 2026
2.3.8.42 Get Robot Coordinate System and Payload Code Example . . . 2026
2.3.9 Robot trajectory playback . . . 2028
2.3.9.1 Set TPD trajectory recording parameters . . . 2028
2.3.9.2 Start TPD trajectory recording . . . 2028
2.3.9.3 Stop TPD trajectory recording . . . 2028
2.3.9.4 Delete TPD trajectory recording . . . 2029
2.3.9.5 TPD trajectory preloading . . . 2029
2.3.9.6 TPD trajectory playback . . . 2029
2.3.9.7 Get TPD starting pose . . . 2029
2.3.9.8 Robot TPD trajectory recording code example . . . 2030
2.3.9.9 Trajectory preprocessing . . . 2030
2.3.9.10 Trajectory playback . . . 2031
2.3.9.11 Get trajectory starting pose . . . 2031
2.3.9.12 Get trajectory point number . . . 2031
2.3.9.13 Set trajectory playback speed . . . 2031
2.3.9.14 Set force/torque during trajectory playback . . . 2031
2.3.9.15 Set x-direction force during trajectory playback . . . 2032
2.3.9.16 Set y-direction force during trajectory playback . . . 2032
2.3.9.17 Set z-direction force during trajectory playback . . . 2032
2.3.9.18 Set x-axis torque during trajectory playback . . . 2032
2.3.9.19 Set y-axis torque during trajectory playback . . . 2032
2.3.9.20 Set z-axis torque during trajectory playback . . . 2033
2.3.9.21 Upload trajectory J file . . . 2033
2.3.9.22 Delete trajectory J file . . . 2033
2.3.9.23 Robot trajectory J file playback code example . . . 2033
2.3.9.24 Trajectory preprocessing (lookahead) . . . 2034
2.3.9.25 Trajectory playback (lookahead) . . . 2035
2.3.9.26 Trajectory playback (lookahead) code example . . . 2035
2.3.10 Robot WebAPP program usage . . . 2035
2.3.10.1 Set default program to load automatically on startup . . . 2035
2.3.10.2 Load specified program . . . 2036
2.3.10.3 Get loaded program name . . . 2036
2.3.10.4 Get current program execution line number . . . 2036
2.3.10.5 Run currently loaded program . . . 2036
2.3.10.6 Pause current running program . . . 2037

2.3.10.7 Resume paused program . . . 2037
2.3.10.8 Stop current running program . . . 2037
2.3.10.9 Get robot program execution state . . . 2037
2.3.10.10 Robot LUA program operation code example . . . 2037
2.3.10.11 Download Lua program . . . 2038
2.3.10.12 Delete Lua program . . . 2038
2.3.10.13 Get all current Lua file names . . . 2038
2.3.10.14 Upload Lua program . . . 2039
2.3.10.15 Robot LUA file upload/download code example . . . 2039
2.3.11 Robot peripherals . . . 2039
2.3.11.1 Configure gripper . . . 2039
2.3.11.2 Get gripper configuration . . . 2040
2.3.11.3 Activate gripper . . . 2040
2.3.11.4 Control gripper . . . 2040
2.3.11.5 Get gripper motion status . . . 2041
2.3.11.6 Get gripper activation status . . . 2041
2.3.11.7 Get gripper position . . . 2041
2.3.11.8 Get gripper speed . . . 2041
2.3.11.9 Get gripper current . . . 2041
2.3.11.10 Get gripper voltage . . . 2042
2.3.11.11 Get gripper temperature . . . 2042
2.3.11.12 Calculate pre-grasp point - vision . . . 2042
2.3.11.13 Calculate retreat point - vision . . . 2042
2.3.11.14 Robot gripper operation code example . . . 2043
2.3.11.15 Get rotary gripper rotation turns . . . 2044
2.3.11.16 Get rotary gripper rotation speed percentage . . . 2044
2.3.11.17 Get rotary gripper rotation torque percentage . . . 2044
2.3.11.18 Code example for getting rotary gripper status . . . 2044
2.3.11.19 Conveyor start/stop . . . 2045
2.3.11.20 Record IO detection point . . . 2045
2.3.11.21 Record point A . . . 2045
2.3.11.22 Record reference point . . . 2045
2.3.11.23 Record point B . . . 2046
2.3.11.24 Conveyor workpiece IO detection . . . 2046
2.3.11.25 Get object current position . . . 2046
2.3.11.26 Conveyor tracking start . . . 2046
2.3.11.27 Conveyor tracking stop . . . 2046
2.3.11.28 Conveyor parameter configuration . . . 2047
2.3.11.29 Set conveyor grasp point compensation . . . 2047
2.3.11.30 Conveyor linear motion . . . 2047
2.3.11.31 Conveyor communication input detection . . . 2048
2.3.11.32 Conveyor communication input detection trigger . . . 2048
2.3.11.33 Robot conveyor operation example program . . . 2048
2.3.11.34 End sensor configuration . . . 2049
2.3.11.35 Get end sensor configuration . . . 2049
2.3.11.36 End sensor activation . . . 2050
2.3.11.37 End sensor register write . . . 2050
2.3.11.38 End sensor code example . . . 2050
2.3.11.39 Get robot peripheral protocol . . . 2051
2.3.11.40 Set robot peripheral protocol . . . 2051
2.3.11.41 Set robot peripheral protocol example program . . . 2051
2.3.11.42 Get end communication parameters . . . 2051
2.3.11.43 Set end communication parameters . . . 2052
2.3.11.44 Set end file transfer type . . . 2052

2.3.11.45 Set enable end LUA execution . . . 2052
2.3.11.46 End LUA file error recovery . . . 2052
2.3.11.47 Get end LUA execution enable status . . . 2052
2.3.11.48 Set end LUA end device enable type . . . 2053
2.3.11.49 Get end LUA end device enable type . . . 2053
2.3.11.50 Get currently configured end devices . . . 2053
2.3.11.51 Set enable gripper action control function . . . 2053
2.3.11.52 Get enable gripper action control function . . . 2054
2.3.11.53 Robot Ethercat slave file write . . . 2054
2.3.11.54 Upload end Lua open protocol file . . . 2054
2.3.11.55 Robot Ethercat slave enter boot mode . . . 2055
2.3.11.56 Robot end LUA file operation code example . . . 2055
2.3.11.57 Get SmartTool button status . . . 2056
2.3.11.58 SmartTool button code example . . . 2056
2.3.11.59 Upload Open Protocol Lua File . . . 2057
2.3.11.60 Get Slave Board Parameters . . . 2057
2.3.11.61 Write Slave DO . . . 2057
2.3.11.62 Write Slave AO . . . 2058
2.3.11.63 Read Slave DI . . . 2058
2.3.11.64 Read Slave AI . . . 2058
2.3.11.65 Wait for Extended DI Input . . . 2058
2.3.11.66 Wait for Extended AI Input . . . 2059
2.3.11.67 Slave Mode Related Interface Command Code Example . . . 2059
2.3.11.68 Control Array Sucker . . . 2060
2.3.11.69 Get Array Sucker Status . . . 2060
2.3.11.70 Wait for Sucker Status . . . 2061
2.3.11.71 Array Sucker Control Command Code Example . . . 2061
2.3.11.72 Laser Peripheral On/Off Function . . . 2062
2.3.11.73 Laser Tracking Start/Stop Function . . . 2062
2.3.11.74 Laser Positioning - Fixed Direction . . . 2062
2.3.11.75 Laser Positioning - Arbitrary Direction . . . 2062
2.3.11.76 Laser Positioning Stop . . . 2063
2.3.11.77 Laser IP Configuration . . . 2063
2.3.11.78 Laser Peripheral Sampling Period Configuration . . . 2063
2.3.11.79 Laser Peripheral Driver Loading . . . 2063
2.3.11.80 Laser Peripheral Driver Unloading . . . 2064
2.3.11.81 Laser Weld Seam Trajectory Recording . . . 2064
2.3.11.82 Laser Weld Seam Trajectory Replay . . . 2064
2.3.11.83 Laser Tracking Replay . . . 2064
2.3.11.84 Laser Weld Seam Trajectory Recording and Replay . . . 2065
2.3.11.85 Move to Laser Record Start Point . . . 2065
2.3.11.86 Move to Laser Record End Point . . . 2065
2.3.11.87 Move to Laser Sensor Positioning Point . . . 2066
2.3.11.88 Get Laser Sensor Positioning Point Coordinate Information . . . 2066
2.3.11.89 Laser Peripheral Sensor Parameter Configuration and Debugging Code Example . . . 2066
2.3.11.90 Laser Trajectory Scanning and Trajectory Replay Code Example . . . 2067
2.3.11.91 Laser Positioning and Real-time Tracking Code Example . . . 2068
2.3.11.92 Extended Axis and Robot Synchronized Laser Tracking Code Example . . . 2069
2.3.12 Robot Force Control . . . 2070
2.3.12.1 Configure Force Sensor . . . 2070
2.3.12.2 Get Force Sensor Configuration . . . 2070
2.3.12.3 Activate Force Sensor . . . 2071
2.3.12.4 Zero Calibration for Force Sensor . . . 2071
2.3.12.5 Set Force Sensor Reference Coordinate System . . . 2071

2.3.12.6 Set Load Weight Under Force Sensor . . . 2071
2.3.12.7 Set Load Center of Gravity Under Force Sensor . . . 2071
2.3.12.8 Get Load Weight Under Force Sensor . . . 2072
2.3.12.9 Get Load Center of Gravity Under Force Sensor . . . 2072
2.3.12.10 Automatic Zero Calibration for Force Sensor . . . 2072
2.3.12.11 Get Force/Torque Data in Reference Coordinate System . . . 2072
2.3.12.12 Get Raw Force/Torque Data from Force Sensor . . . 2072
2.3.12.13 Force Sensor Configuration and Automatic Zero Calibration Example . . . 2073
2.3.12.14 Load Weight Identification Record . . . 2074
2.3.12.15 Load Weight Identification Calculation . . . 2074
2.3.12.16 Load Center of Gravity Identification Record . . . 2074
2.3.12.17 Load Center of Gravity Identification Calculation . . . 2074
2.3.12.18 Force Sensor Load Identification Example . . . 2075
2.3.12.19 Collision Guard . . . 2076
2.3.12.20 Collision Guard Example . . . 2076
2.3.12.21 Constant Force Control . . . 2077
2.3.12.22 Constant Force Control . . . 2077
2.3.12.23 Example Code for Constant Force Control with Damping . . . 2078
2.3.12.24 Rotational Insertion . . . 2079
2.3.12.25 Robot Force Sensor Rotational Insertion Code Example . . . 2079
2.3.12.26 Constant Force Control Example . . . 2081
2.3.12.27 Compliance Control Start . . . 2082
2.3.12.28 Compliance Control Stop . . . 2082
2.3.12.29 Compliance Control Example . . . 2082
2.3.12.30 Load Identification Initialization . . . 2084
2.3.12.31 Load Identification Variable Initialization . . . 2084
2.3.12.32 Load Identification Main Program . . . 2084
2.3.12.33 Get Load Identification Result . . . 2084
2.3.12.34 Robot Load Identification Example . . . 2084
2.3.12.35 Force Sensor Assisted Dragging . . . 2085
2.3.12.36 Get Force Sensor Dragging Switch Status . . . 2086
2.3.12.37 Force Sensor Auto-Enable After Error Clearance . . . 2086
2.3.12.38 Force Sensor Assisted Dragging Example . . . 2086
2.3.12.39 Set Six-Dimensional Force and Joint Impedance Hybrid Dragging Switch and Parameters . . . 2087
2.3.12.40 Six-Dimensional Force and Joint Impedance Hybrid Dragging Example . . . 2087
2.3.12.41 Example Program . . . 2087
2.3.12.42 Impedance Control Start/Stop . . . 2089
2.3.12.43 Robot Impedance Control Start/Stop Code Example . . . 2090
2.3.12.44 Enable Torque Compensation Function and Compensation Coefficients . . . 2091
2.3.13 Extended Axis . . . 2091
2.3.13.1 Set 485 Extended Axis Parameters . . . 2091
2.3.13.2 Get 485 Extended Axis Parameters . . . 2091
2.3.13.3 Set 485 Extended Axis Enable/Disable . . . 2092
2.3.13.4 Set 485 Extended Axis Control Mode . . . 2092
2.3.13.5 Set 485 Extended Axis Target Position (Position Mode) . . . 2092
2.3.13.6 Set 485 Extended Axis Target Torque (Torque Mode) - Not Yet Available . . . 2092
2.3.13.7 Set 485 Extended Axis . . . 2093
2.3.13.8 Clear 485 Extended Axis Error Information . . . 2093
2.3.13.9 Get 485 Extended Axis Servo Status . . . 2093
2.3.13.10 Set 485 Extended Axis Target Speed (Speed Mode) . . . 2094
2.3.13.11 Set Axis Number for Status Feedback of 485 Extended Axis . . . 2094
2.3.13.12 Set 485 Extended Axis Motion Acceleration/Deceleration . . . 2094
2.3.13.13 Set 485 Extended Axis Emergency Stop Acceleration/Deceleration . . . 2094

2.3.13.14 Get 485 Extended Axis Motion Acceleration/Deceleration . . . 2095
2.3.13.15 Get 485 Extended Axis Emergency Stop Acceleration/Deceleration . . . 2095
2.3.13.16 Extended Axis Control Code Example . . . 2095
2.3.13.17 UDP Extended Axis Communication Parameter Configuration . . . 2097
2.3.13.18 Get UDP Extended Axis Communication Parameter Configuration . . . 2097
2.3.13.19 Load UDP Communication . . . 2097
2.3.13.20 Unload UDP Communication . . . 2097
2.3.13.21 Reconnect After UDP Extended Axis Communication Exception Disconnection . . . 2098
2.3.13.22 Close Communication After UDP Extended Axis Communication Exception Disconnection . . . 2098
2.3.13.23 UDP Extended Axis Parameter Configuration . . . 2098
2.3.13.24 Set Extended Axis Installation Position . . . 2099
2.3.13.25 Set Extended Axis System DH Parameter Configuration . . . 2099
2.3.13.26 UDP Extended Axis Enable . . . 2099
2.3.13.27 UDP Extended Axis Homing . . . 2100
2.3.13.28 UDP Extended Axis Jog Start . . . 2100
2.3.13.29 UDP Extended Axis Jog Stop . . . 2100
2.3.13.30 UDP Extended Axis Configuration and Jog Code Example . . . 2100
2.3.13.31 Set Extended Axis Coordinate System Reference Point - Four-Point Method . . . 2101
2.3.13.32 Calculate Extended Axis Coordinate System - Four-Point Method . . . 2102
2.3.13.33 Positioner Coordinate System Reference Point Setting . . . 2102
2.3.13.34 Positioner Coordinate System Calculation - Four-Point Method . . . 2102
2.3.13.35 Set Calibration Reference Point Pose in Extended Axis End Coordinate System . . . 2102
2.3.13.36 Apply Extended Axis Coordinate System . . . 2102
2.3.13.37 Get Extended Axis Coordinate System . . . 2103
2.3.13.38 Extended Axis Coordinate System Calibration Code Example . . . 2103
2.3.13.39 UDP Extended Axis Motion . . . 2105
2.3.13.40 UDP Extended Axis Motion Code Example . . . 2105
2.3.13.41 UDP Extended Axis and Robot Joint Motion Synchronous Motion . . . 2105
2.3.13.42 UDP extended axis and robot joint motion synchronous motion (automatic forward kinematics calculation) . . . 2106
2.3.13.43 UDP Extended Axis and Robot Joint Motion Synchronous Motion Code Example . 2106
2.3.13.44 UDP Extended Axis and Robot Linear Motion Synchronous Motion . . . 2108
2.3.13.45 UDP extended axis and robot linear motion synchronous motion (automatic inverse kinematics calculation) . . . 2108
2.3.13.46 UDP Extended Axis and Robot Linear Motion Synchronous Motion Code Example 2109
2.3.13.47 UDP Extended Axis and Robot Arc Motion Synchronous Motion . . . 2111
2.3.13.48 UDP extended axis and robot circular motion synchronous motion (automatic inverse kinematics calculation) . . . 2111
2.3.13.49 UDP Extended Axis and Robot Arc Motion Synchronous Motion Code Example . . 2112
2.3.13.50 Set Extended DO . . . 2114
2.3.13.51 Set Extended AO . . . 2114
2.3.13.52 Set Extended DI Input Filter Time . . . 2114
2.3.13.53 Set Extended AI Input Filter Time . . . 2114
2.3.13.54 Wait for Extended DI Input . . . 2115
2.3.13.55 Wait for Extended AI Input . . . 2115
2.3.13.56 Get Extended DI Value . . . 2115
2.3.13.57 Get Extended AI Value . . . 2115
2.3.13.58 Extended IO Code Example . . . 2116
2.3.13.59 Movable Device Enable . . . 2116
2.3.13.60 Movable Device Homing . . . 2117
2.3.13.61 Movable Device Linear Motion . . . 2117
2.3.13.62 Movable Device Arc Motion . . . 2117
2.3.13.63 Movable Device Stop Motion . . . 2117

2.3.13.64 Movable Device Code Example . . . 2117
2.3.14 Robot Welding . . . 2118
2.3.14.1 Set welding process curve parameters . . . 2118
2.3.14.2 Get welding process curve parameters . . . 2119
2.3.14.3 Set welding current to analog output relation . . . 2119
2.3.14.4 Set welding voltage to analog output relation . . . 2119
2.3.14.5 Get welding current to analog output relation . . . 2119
2.3.14.6 Get welding voltage to analog output relation . . . 2119
2.3.14.7 Set welding current . . . 2120
2.3.14.8 Set welding voltage . . . 2120
2.3.14.9 Set weaving parameters . . . 2120
2.3.14.10 Welding parameter setting code example . . . 2121
2.3.14.11 Real-time weaving parameter setting . . . 2123
2.3.14.12 Set welding arc interruption detection parameters . . . 2123
2.3.14.13 Get welding arc interruption detection parameters . . . 2123
2.3.14.14 Set welding interruption recovery parameters . . . 2124
2.3.14.15 Get welding interruption recovery parameters . . . 2124
2.3.14.16 Set welder control mode extension DO port . . . 2124
2.3.14.17 Set welder control mode . . . 2124
2.3.14.18 Welding start . . . 2125
2.3.14.19 Welding end . . . 2125
2.3.14.20 Weaving start . . . 2125
2.3.14.21 Weaving end . . . 2125
2.3.14.22 Forward wire feeding . . . 2126
2.3.14.23 Reverse wire feeding . . . 2126
2.3.14.24 Gas feeding . . . 2126
2.3.14.25 Set robot to resume welding after interruption . . . 2126
2.3.14.26 Set robot to abort welding after interruption . . . 2126
2.3.14.27 Robot welding control code example . . . 2127
2.3.14.28 Segment welding start . . . 2127
2.3.14.29 Robot segment welding code example . . . 2128
2.3.14.30 Simulation weaving start . . . 2128
2.3.14.31 Simulation weaving end . . . 2129
2.3.14.32 Start trajectory inspection warning (no movement) . . . 2129
2.3.14.33 End trajectory inspection warning (no movement) . . . 2129
2.3.14.34 Weaving transition start . . . 2129
2.3.14.35 Robot weaving transition welding code example . . . 2130
2.3.14.36 Weaving transition end . . . 2131
2.3.14.37 Extension IO - Configure welder gas detection signal . . . 2131
2.3.14.38 Extension IO - Configure welder arc start signal . . . 2131
2.3.14.39 Extension IO - Configure welder reverse wire feed signal . . . 2131
2.3.14.40 Extension IO - Configure welder forward wire feed signal . . . 2131
2.3.14.41 Extension IO - Configure welder arc success signal . . . 2132
2.3.14.42 Extension IO - Configure welder ready signal . . . 2132
2.3.14.43 Extension IO - Configure welding interruption recovery signal . . . 2132
2.3.14.44 Set extension IO welding signal code example . . . 2132
2.3.14.45 Arc tracking control . . . 2133
2.3.14.46 Arc tracking AI channel selection . . . 2133
2.3.14.47 Arc tracking + multi-layer multi-pass compensation start . . . 2134
2.3.14.48 Arc tracking + multi-layer multi-pass compensation end . . . 2134
2.3.14.49 Offset coordinate transformation - multi-layer multi-pass welding . . . 2134
2.3.14.50 Multi-layer multi-pass welding arc tracking code example . . . 2134
2.3.14.51 Arc tracking welder current feedback AI channel selection . . . 2137
2.3.14.52 Arc tracking welder voltage feedback AI channel selection . . . 2137

2.3.14.53 Arc tracking welder current feedback conversion parameters . . . 2138
2.3.14.54 Arc tracking welder voltage feedback conversion parameters . . . 2138
2.3.14.55 Arc tracking code example . . . 2138
2.3.14.56 Set Wire Search Extension IO Ports . . . 2140
2.3.14.57 Wire search start . . . 2140
2.3.14.58 Wire search end . . . 2140
2.3.14.59 Calculate wire search offset . . . 2141
2.3.14.60 Wait for wire search completion . . . 2141
2.3.14.61 Write wire search contact point to database . . . 2141
2.3.14.62 Robot wire search code example . . . 2141
2.3.14.63 Set welding voltage gradual change start . . . 2143
2.3.14.64 Set welding voltage gradual change end . . . 2144
2.3.14.65 Set welding current gradual change start . . . 2144
2.3.14.66 Set welding current gradual change end . . . 2144
2.3.14.67 Robot welding current/voltage gradual change code example . . . 2144
2.3.14.68 Set Custom Weaving Parameters . . . 2146
2.3.14.69 Get Custom Weaving Parameters . . . 2146
2.3.14.70 Custom Weaving Parameters Code Example . . . 2146
2.3.15 Other interfaces . . . 2148
2.3.15.1 Get SSH public key . . . 2148
2.3.15.2 Issue SCP command . . . 2148
2.3.15.3 Calculate MD5 value of specified file . . . 2149
2.3.15.4 Robot SSH and MD5 command code example . . . 2149
2.3.15.5 Set robot port 20004 feedback period . . . 2149
2.3.15.6 Get robot port 20004 feedback period . . . 2150
2.3.15.7 Robot port 20004 status feedback period configuration example . . . 2150
2.3.15.8 Robot software upgrade . . . 2150
2.3.15.9 Get robot software upgrade status . . . 2150
2.3.15.10 Robot software upgrade code example . . . 2151
2.3.15.11 Download point table database . . . 2151
2.3.15.12 Upload point table database . . . 2151
2.3.15.13 Point table update lua file . . . 2151
2.3.15.14 Robot point table operation code example . . . 2152
2.3.15.15 Controller log download . . . 2152
2.3.15.16 All data source download . . . 2152
2.3.15.17 Data backup package download . . . 2153
2.3.15.18 Download controller data code example . . . 2153
2.3.15.19 Set Encoder Upgrade . . . 2153
2.3.15.20 Set Joint Firmware Upgrade . . . 2153
2.3.15.21 Set Controller Firmware Upgrade . . . 2154
2.3.15.22 Set End-Effector Firmware Upgrade . . . 2154
2.3.15.23 Joint Complete Parameter Configuration Upgrade . . . 2154
2.3.15.24 Robot Slave Firmware Upgrade Code Example . . . 2154
2.3.15.25 Robot Operating System Upgrade (LA Control Box) . . . 2155
2.3.15.26 Get Robot Operating System Upgrade Result (LA Control Box) . . . 2155
2.3.15.27 Robot MCU Log Generation . . . 2156
2.3.16 Appendix . . . 2156
2.3.16.1 Source Code Download . . . 2156
2.3.16.2 Windows Platform Source Code Compilation . . . 2157
2.4 Python . . . 2158
2.4.1 Version Update Description . . . 2161
2.4.2 Data structure description . . . 2162
2.4.2.1 Controller status feedback data packet . . . 2162
2.4.2.2 Status of the servo controller . . . 2164

2.4.2.3 Extended axis status . . . 2164
2.4.2.4 Welding interruption state . . . 2164
2.4.2.5 code example . . . 2165
2.4.3 Robotics Basics . . . 2169
2.4.3.1 Instantiated Robot . . . 2169
2.4.3.2 Close the RPC connection . . . 2169
2.4.3.3 Query SDK version number . . . 2169
2.4.3.4 Get controller IP . . . 2169
2.4.3.5 Controlling the robot into and out of drag-and-drop instructor mode . . . 2170
2.4.3.6 Queries whether the robot is in drag mode . . . 2170
2.4.3.7 Control robot up-enable or down-enable . . . 2170
2.4.3.8 Control of robot hand-automatic mode switching . . . 2170
2.4.3.9 Shut down the robot operating system . . . 2171
2.4.3.10 Initialize the log parameters . . . 2171
2.4.3.11 Setting the log filter level . . . 2171
2.4.3.12 Robot base control code example . . . 2172
2.4.3.13 Get the software version of the robot . . . 2172
2.4.3.14 Getting robot hardware version information . . . 2173
2.4.3.15 Getting robot firmware version information . . . 2173
2.4.3.16 Get the robot software firmware version code sample . . . 2173
2.4.4 Movement . . . 2174
2.4.4.1 jog point and click . . . 2174
2.4.4.2 jog tap to decelerate and stop . . . 2174
2.4.4.3 Immediate stop for jog taps . . . 2175
2.4.4.4 Robot point control code example . . . 2175
2.4.4.5 Joint space motion . . . 2176
2.4.4.6 Cartesian Space Linear Motion . . . 2176
2.4.4.7 Cartesian Space Circular Arc Motion . . . 2178
2.4.4.8 Cartesian Space Full Circle Motion . . . 2180
2.4.4.9 Point-to-point motion in Cartesian space . . . 2181
2.4.4.10 Sample robot basic motion commands code . . . 2181
2.4.4.11 Spiral motion in Cartesian space . . . 2183
2.4.4.12 code example . . . 2183
2.4.4.13 Start of servo motion . . . 2184
2.4.4.14 End of servo motion . . . 2184
2.4.4.15 Joint space servo mode motion . . . 2185
2.4.4.16 Example of joint space servo mode motion code . . . 2185
2.4.4.17 Joint torque control begins . . . 2186
2.4.4.18 Joint torque control . . . 2186
2.4.4.19 Joint torque control is completed . . . 2187
2.4.4.20 Sample code for joint torque control . . . 2187
2.4.4.21 Joint Torque Control Code Example with Overspeed Protection . . . 2187
2.4.4.22 Cartesian Space Servo Mode Motion . . . 2188
2.4.4.23 Cartesian Space Servo Mode Motion Code Example . . . 2188
2.4.4.24 Start of spline motion . . . 2189
2.4.4.25 Sample motion PTP . . . 2189
2.4.4.26 End of spline motion . . . 2189
2.4.4.27 Spline motion code example . . . 2190
2.4.4.28 New spline movement begins . . . 2190
2.4.4.29 New spline command point . . . 2191
2.4.4.30 End of new spline movement . . . 2191
2.4.4.31 Example of new spline motion code . . . 2191
2.4.4.32 Robot termination motion . . . 2192
2.4.4.33 Robot pause . . . 2192

2.4.4.34 Robot resume motion . . . 2193
2.4.4.35 Motion pause, resume, and stop code examples . . . 2193
2.4.4.36 Overall shift in points begins . . . 2194
2.4.4.37 Overall offset of points ends . . . 2194
2.4.4.38 Point offset code example . . . 2194
2.4.4.39 Control box motion AO start . . . 2195
2.4.4.40 End of control box movement AO . . . 2195
2.4.4.41 End Motion AO Start . . . 2195
2.4.4.42 End movement AO end . . . 2196
2.4.4.43 AO flyshot code example . . . 2196
2.4.4.44 Start Ptp motion FIR filtering . . . 2197
2.4.4.45 Disable Ptp motion FIR filtering . . . 2197
2.4.4.46 LIN, ARC motion FIR filtering is started . . . 2197
2.4.4.47 Turn off LIN and ARC motion FIR filtering . . . 2197
2.4.4.48 FIR filtering code example . . . 2198
2.4.4.49 Acceleration smooth on . . . 2198
2.4.4.50 Acceleration smooth closing . . . 2199
2.4.4.51 Acceleration smoothing code example . . . 2199
2.4.4.52 Setting the machine's specified attitude speed on . . . 2199
2.4.4.53 Specify Attitude Velocity Off . . . 2200
2.4.4.54 Robot specified pose velocity code example . . . 2200
2.4.4.55 Odd-position protection on. . . . 2200
2.4.4.56 Odd position protection off . . . 2201
2.4.4.57 Example of robot singular pose protection code . . . 2201
2.4.4.58 Clear the motor command queue . . . 2201
2.4.4.59 Clear Motion Command Queue . . . 2202
2.4.4.60 Intersecting Line Motion . . . 2203
2.4.4.61 Robot Intersecting Line Motion Code Example . . . 2203
2.4.4.62 Stationary Air Motion . . . 2205
2.4.4.63 Stationary Air Motion Code Example . . . 2205
2.4.5 IO . . . 2205
2.4.5.1 Setting the control box digital output . . . 2205
2.4.5.2 Setting Tool Digital Outputs . . . 2206
2.4.5.3 Setting the control box analog output . . . 2206
2.4.5.4 Setting Tool Analog Outputs . . . 2206
2.4.5.5 Set digital, analog output code example . . . 2207
2.4.5.6 Getting control box digital inputs . . . 2207
2.4.5.7 Get Tool Digital Inputs . . . 2208
2.4.5.8 Getting Control Box Analog Inputs . . . 2208
2.4.5.9 Get Tool Analog Inputs . . . 2208
2.4.5.10 Obtain the status of the button for recording the end point of the robot . . . 2209
2.4.5.11 Obtain the DO output status at the end of the robot . . . 2209
2.4.5.12 Obtain the DO output status of the robot controller . . . 2209
2.4.5.13 Get the robot DI, DO status code examples . . . 2209
2.4.5.14 Waiting for control box digital inputs . . . 2210
2.4.5.15 Waiting for control box with multiple digital inputs . . . 2210
2.4.5.16 Waiting for tool digital inputs . . . 2211
2.4.5.17 Waiting for control box analog inputs . . . 2211
2.4.5.18 Waiting for tool analog inputs . . . 2211
2.4.5.19 Waiting control box digital, analog input signal code example . . . 2212
2.4.5.20 Set Whether Control Box DO Output Resets After Stop/Pause . . . 2213
2.4.5.21 Set Whether Control Box AO Output Resets After Stop/Pause . . . 2213
2.4.5.22 Set Whether End Tool DO Output Resets After Stop/Pause . . . 2213
2.4.5.23 Set Whether End Tool AO Output Resets After Stop/Pause . . . 2214

2.4.5.24 Set Whether Extended DO Output Resets After Stop/Pause . . . 2214
2.4.5.25 Set Whether Extended AO Output Resets After Stop/Pause . . . 2214
2.4.5.26 Set Whether SmartTool Output Resets After Stop/Pause . . . 2214
2.4.5.27 Code Example for Setting Output Reset After Lua Program Stop/Pause . . . 2215
2.4.6 Common Robot Settings . . . 2215
2.4.6.1 Setting Tool Reference Points - Six-Point Method . . . 2215
2.4.6.2 Calculation tool coordinate system - six-point method . . . 2216
2.4.6.3 Setting Tool Reference Points - Four Point Method . . . 2216
2.4.6.4 Calculation Tool Coordinate System - Four Point Method . . . 2216
2.4.6.5 Calculate the tool coordinate system based on the point information . . . 2217
2.4.6.6 Setting the tool coordinate system . . . 2217
2.4.6.7 Setting the tool coordinate system list . . . 2217
2.4.6.8 Get the current tool coordinate system . . . 2218
2.4.6.9 Robot tool coordinate system manipulation code example . . . 2218
2.4.6.10 Setting External Tool Reference Points-Six-Point Method . . . 2219
2.4.6.11 Calculation of the external tool coordinate system - Six-point method . . . 2219
2.4.6.12 Setting the external tool coordinate system . . . 2220
2.4.6.13 Setting up a list of external tool coordinate systems . . . 2220
2.4.6.14 Example code for robot external tool coordinate system operation . . . 2220
2.4.6.15 Setting the workpiece reference point - three-point method . . . 2221
2.4.6.16 Calculation of the workpiece coordinate system - three-point method . . . 2221
2.4.6.17 Setting the workpiece coordinate system . . . 2221
2.4.6.18 Setting the list of workpiece coordinate systems . . . 2222
2.4.6.19 Calculate the workpiece coordinate system based on the point information . . . 2222
2.4.6.20 Get the current workpiece coordinate system . . . 2222
2.4.6.21 Example of robot workpiece coordinate system manipulation code . . . 2223
2.4.6.22 Setting the global speed . . . 2223
2.4.6.23 Setting robot acceleration . . . 2224
2.4.6.24 Getting the default speed . . . 2224
2.4.6.25 Setting the end load weight . . . 2224
2.4.6.26 Setting the end load center of mass coordinates . . . 2224
2.4.6.27 Get the weight of the current load . . . 2225
2.4.6.28 Get the center of mass of the current load . . . 2225
2.4.6.29 Setting the robot mounting method - fixed mounting . . . 2225
2.4.6.30 Setting the robot mounting angle - free mounting . . . 2225
2.4.6.31 Getting the robot mounting angle . . . 2226
2.4.6.32 Setting system variable values . . . 2226
2.4.6.33 Getting system variable values . . . 2226
2.4.6.34 Robot common setup code examples . . . 2226
2.4.6.35 Joint Friction Compensation Switch . . . 2227
2.4.6.36 Setting the joint friction compensation coefficients - positive loading . . . 2227
2.4.6.37 Setting the joint friction compensation coefficient - side mounting . . . 2228
2.4.6.38 Setting the Joint Friction Compensation Factor - Inverted . . . 2228
2.4.6.39 Setting the joint friction compensation factor - free mounting . . . 2228
2.4.6.40 Robot setup joint friction compensation code example . . . 2228
2.4.6.41 Query Robot Error Code . . . 2229
2.4.6.42 Error state clearing . . . 2229
2.4.6.43 Robot fault state acquisition and clearing error code examples . . . 2229
2.4.6.44 Set the monitoring parameters for the temperature and fan speed of the wide-voltage control box . . . 2230
2.4.6.45 Obtain the monitoring parameters of the temperature and fan speed of the wide-voltage control box . . . 2230
2.4.6.46 Sample code for obtaining wide voltage control box temperature and fan current status 2230
2.4.6.47 Sets the focus point . . . 2231

2.4.6.48 Calculate the focus calibration result . . . 2231
2.4.6.49 Enable focus following . . . 2231
2.4.6.50 Stop focusing following . . . 2232
2.4.6.51 Set the focus coordinates . . . 2232
2.4.6.52 Robot focus following code example . . . 2232
2.4.6.53 Joint torque sensor sensitivity calibration function is enabled . . . 2234
2.4.6.54 Joint torque sensor sensitivity data acquisition . . . 2234
2.4.6.55 The sensitivity calibration results of the joint torque sensor were obtained . . . 2234
2.4.6.56 Get Joint Torque Sensor Hysteresis Error . . . 2234
2.4.6.57 Get Joint Torque Sensor Repeatability . . . 2235
2.4.6.58 Set Joint Force Sensor Parameters . . . 2235
2.4.6.59 Sample code for automatic calibration of joint torque sensor sensitivity . . . 2235
2.4.6.60 The number of error frames at eight slave ports of the robot is obtained . . . 2238
2.4.6.61 Slave port error frame reset . . . 2238
2.4.6.62 Get sample slave port error frame code . . . 2239
2.4.6.63 Set the feed-forward coefficient of each axis speed . . . 2239
2.4.6.64 The velocity feedforward coefficients of each axis are obtained . . . 2240
2.4.6.65 Robot velocity feedforward coefficient code example . . . 2240
2.4.6.66 Photoelectric Sensor TCP Calibration - Compute Tool RPY . . . 2240
2.4.6.67 Photoelectric Sensor TCP Calibration - Compute Tool XYZ . . . 2241
2.4.6.68 Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position . . . 2241
2.4.6.69 Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position . . . 2241
2.4.6.70 Photoelectric Sensor TCP Calibration - Get Tool Center Point Position . . . 2242
2.4.6.71 Photoelectric Sensor TCP Calibration . . . 2242
2.4.6.72 Photoelectric Sensor TCP Calibration Code Example . . . 2242
2.4.7 Security Settings . . . 2243
2.4.7.1 Setting the collision level . . . 2243
2.4.7.2 Setting the post-collision strategy . . . 2243
2.4.7.3 The Custom collision detection threshold function starts to set the collision detection thresholds of the joint end and TCP end . . . 2243
2.4.7.4 The custom collision detection threshold function is disabled . . . 2244
2.4.7.5 Robot collision level setting code example . . . 2244
2.4.7.6 Setting the positive limit . . . 2245
2.4.7.7 Setting the negative limit . . . 2245
2.4.7.8 Obtaining the soft limiting angle of a joint . . . 2246
2.4.7.9 Robot limit setting code example . . . 2246
2.4.7.10 Setting up a robot collision detection method . . . 2246
2.4.7.11 Set static undercollision detection to start off . . . 2247
2.4.7.12 Set up the robot collision detection method code example . . . 2247
2.4.7.13 Joint torque and power detection . . . 2247
2.4.7.14 Sample code for joint torque power detection . . . 2248
2.4.8 Status query . . . 2248
2.4.8.1 Get the current joint position (angle). . . . 2248
2.4.8.2 Get the current joint position in radians. . . . 2248
2.4.8.3 Get joint feedback speed -deg/s . . . 2249
2.4.8.4 Obtain joint feedback acceleration-deg/s^2 . . . 2249
2.4.8.5 Get TCP command synthesis speed . . . 2249
2.4.8.6 Getting TCP Feedback Hopping Speed . . . 2250
2.4.8.7 Get TCP command speed . . . 2250
2.4.8.8 Getting TCP feedback speed . . . 2250
2.4.8.9 Get current tool position . . . 2251
2.4.8.10 Get the current tool coordinate system number . . . 2251
2.4.8.11 Get the current workpiece coordinate system number . . . 2251
2.4.8.12 Get the current end flange position . . . 2252

2.4.8.13 Get current joint torque . . . 2252
2.4.8.14 Get system time . . . 2252
2.4.8.15 Queries whether robot motion is complete . . . 2252
2.4.8.16 Query the cache length of the robot motion queue . . . 2253
2.4.8.17 Obtain the emergency stop status of the robot . . . 2253
2.4.8.18 Obtain the communication status between the SDK and the robot . . . 2253
2.4.8.19 Obtain the safety stop signal . . . 2253
2.4.8.20 Obtain the current temperature of the joint drive(℃) . . . 2254
2.4.8.21 Obtain the current torque of the joint drive(Nm) . . . 2254
2.4.8.22 Obtain the status of the robot . . . 2254
2.4.8.23 Robot status query code example . . . 2254
2.4.8.24 Inverse kinematics solution . . . 2256
2.4.8.25 Inverse Kinematics Solution - Specifying Reference Positions . . . 2256
2.4.8.26 Inverse Kinematics Solution, Cartesian Space Includes Extended Axis Position . . . 2257
2.4.8.27 Inverse Kinematics Solution Including Extended Axis Position Code Example . . . 2257
2.4.8.28 Inverse kinematics solving-whether there is a solution . . . 2258
2.4.8.29 Positive kinematics solving . . . 2258
2.4.8.30 Example code for robot forward and inverse kinematics calculation . . . 2258
2.4.8.31 Query Robot Teaching Management Points Data . . . 2259
2.4.8.32 Get DH compensation parameters . . . 2259
2.4.8.33 Obtain the SN code of the control box . . . 2259
2.4.8.34 Query robot teaching management point data code example . . . 2260
2.4.8.35 Get the tool coordinate system according to the number . . . 2260
2.4.8.36 The workpiece coordinate system is obtained according to the No . . . 2260
2.4.8.37 The external tool coordinate system is obtained according to the number . . . 2261
2.4.8.38 The extended axis coordinate system is obtained according to the No . . . 2261
2.4.8.39 Get the load mass and centroid according to the number . . . 2261
2.4.8.40 Gets the current tool coordinate system . . . 2262
2.4.8.41 Gets the current workpiece coordinate system . . . 2262
2.4.8.42 Gets the current external tool coordinate system . . . 2262
2.4.8.43 Gets the current extended axis coordinate system . . . 2262
2.4.8.44 Get robot coordinate system and load code sample . . . 2263
2.4.9 Trajectory recurrence . . . 2265
2.4.9.1 Setting Track Recording Parameters . . . 2265
2.4.9.2 Start Track Recording . . . 2265
2.4.9.3 Stop Track Recording . . . 2265
2.4.9.4 Deleting track records . . . 2266
2.4.9.5 code example . . . 2266
2.4.9.6 Trajectory preloading . . . 2266
2.4.9.7 Trajectory Reproduction . . . 2267
2.4.9.8 Get the starting position of the trajectory . . . 2267
2.4.9.9 Example of robot TPD trajectory recording code . . . 2267
2.4.9.10 Trajectory preprocessing . . . 2268
2.4.9.11 Trajectory Reproduction . . . 2268
2.4.9.12 Getting the starting position of the trajectory . . . 2268
2.4.9.13 Get track point number . . . 2268
2.4.9.14 Setting the speed of the trajectory in operation . . . 2269
2.4.9.15 Setting the force and torque during trajectory operation . . . 2269
2.4.9.16 Setting the force along the x-direction in the trajectory run . . . 2269
2.4.9.17 Setting the force along the y-direction in the trajectory run . . . 2269
2.4.9.18 Setting the force along the z-direction in a trajectory run . . . 2269
2.4.9.19 Setting the torque around the x-axis in a trajectory run . . . 2270
2.4.9.20 Setting the torque around the y-axis in trajectory operation . . . 2270
2.4.9.21 Setting the torque around the z-axis in trajectory operation . . . 2270

2.4.9.22 Upload trace J file . . . 2270
2.4.9.23 Delete the track J file . . . 2270
2.4.9.24 Robot trajectory J file reproduction code example . . . 2271
2.4.9.25 Trajectory preprocessing(Trajectory foresight) . . . 2272
2.4.9.26 Trajectory reproduction(Trajectory foresight) . . . 2272
2.4.9.27 Code example for trajectory reproduction . . . 2272
2.4.10 WebAPP program use . . . 2273
2.4.10.1 Setting the default job program to load automatically on boot . . . 2273
2.4.10.2 Load the specified job program . . . 2273
2.4.10.3 Get the name of the loaded job program . . . 2273
2.4.10.4 Get the line number of the current robot job program . . . 2274
2.4.10.5 Run the currently loaded job program . . . 2274
2.4.10.6 Suspend the currently running job program . . . 2274
2.4.10.7 Resuming a currently suspended program . . . 2274
2.4.10.8 Terminate the currently running job program . . . 2274
2.4.10.9 Obtaining robot job program execution status . . . 2275
2.4.10.10 Robot LUA program operation code example . . . 2275
2.4.10.11 Download Lua files . . . 2276
2.4.10.12 Deleting Lua files . . . 2276
2.4.10.13 Get the names of all current lua files . . . 2276
2.4.10.14 Uploading Lua files . . . 2277
2.4.10.15 Robot LUA file upload and download code examples . . . 2277
2.4.11 Peripherals . . . 2277
2.4.11.1 Configuration of jaws . . . 2277
2.4.11.2 Get Jaw Configuration . . . 2278
2.4.11.3 Activate jaws . . . 2278
2.4.11.4 Control jaws . . . 2278
2.4.11.5 Getting the jaw movement status . . . 2279
2.4.11.6 Obtain the activated status of the gripper . . . 2279
2.4.11.7 Obtain the position of the gripper . . . 2279
2.4.11.8 Obtain the gripper speed . . . 2280
2.4.11.9 Obtain the gripper current . . . 2280
2.4.11.10 Obtain the gripper voltage . . . 2280
2.4.11.11 Obtain the temperature of the gripper . . . 2281
2.4.11.12 Calculate pre-capture point-visual . . . 2281
2.4.11.13 Calculate retreat point-visual . . . 2281
2.4.11.14 Robot claw operation code example . . . 2282
2.4.11.15 Get the number of rotation turns of the rotary gripper . . . 2283
2.4.11.16 Gets the percentage of rotation speed of the rotating gripper . . . 2283
2.4.11.17 Obtains the percentage of rotating torque of the rotating gripper . . . 2284
2.4.11.18 Get the rotary gripper status code example . . . 2284
2.4.11.19 Drive belt start and stop . . . 2284
2.4.11.20 Record IO detection points . . . 2285
2.4.11.21 Record point A . . . 2285
2.4.11.22 Recording reference points . . . 2285
2.4.11.23 Record point B . . . 2285
2.4.11.24 Conveyorized workpiece IO inspection . . . 2285
2.4.11.25 Get the current position of the object . . . 2286
2.4.11.26 Drive belt tracking started . . . 2286
2.4.11.27 Belt tracking stop . . . 2286
2.4.11.28 Drive Belt Parameter Configuration . . . 2287
2.4.11.29 Belt Grip Point Compensation . . . 2287
2.4.11.30 linear motion . . . 2288
2.4.11.31 Conveyor communication input detection . . . 2288

2.4.11.32 Conveyor communication input detection triggered . . . 2288
2.4.11.33 Robot conveyor operation code example . . . 2289
2.4.11.34 End Sensor Configuration . . . 2290
2.4.11.35 Get End Sensor Configuration . . . 2290
2.4.11.36 End sensor activation . . . 2290
2.4.11.37 End Sensor Register Write . . . 2291
2.4.11.38 End sensor code example . . . 2291
2.4.11.39 Obtaining Robot Peripheral Protocols . . . 2291
2.4.11.40 Setting up robot peripheral protocols . . . 2292
2.4.11.41 Example of setup robot peripheral protocol code . . . 2292
2.4.11.42 Getting end communication parameters . . . 2292
2.4.11.43 Setting the end communication parameters . . . 2293
2.4.11.44 Setting the end file transfer type . . . 2293
2.4.11.45 Setting Enable End LUA Execution . . . 2293
2.4.11.46 End LUA file exception error recovery . . . 2294
2.4.11.47 Get end LUA execution enable status . . . 2294
2.4.11.48 Setting the end LUA end device enable type . . . 2294
2.4.11.49 Get End LUA End Device Enablement Type . . . 2294
2.4.11.50 Get the currently configured end device . . . 2295
2.4.11.51 Setting to enable the jaw movement control function . . . 2295
2.4.11.52 Getting to Enable Jaw Motion Control . . . 2296
2.4.11.53 The Ethercat slave file is written by the robot . . . 2296
2.4.11.54 Upload the end Lua open protocol file . . . 2296
2.4.11.55 The robot Ethercat enters boot mode from the station . . . 2297
2.4.11.56 Example of LUA file manipulation code at the end of the robot . . . 2297
2.4.11.57 Obtain the status of the SmartTool button . . . 2298
2.4.11.58 SmartTool button code example . . . 2298
2.4.11.59 Set the load detection before drag is started . . . 2298
2.4.11.60 Laser peripheral open and close function . . . 2299
2.4.11.61 Laser tracking start-end function . . . 2299
2.4.11.62 Laser positioning - Fixed direction . . . 2299
2.4.11.63 Laser positioning - in any direction . . . 2300
2.4.11.64 Laser IP configuration . . . 2300
2.4.11.65 Configuration of sampling period for laser peripherals . . . 2300
2.4.11.66 Laser peripheral driver loading . . . 2301
2.4.11.67 Laser peripheral driver unloading . . . 2301
2.4.11.68 Laser weld seam trajectory recording . . . 2301
2.4.11.69 Laser weld seam trajectory reproduction . . . 2301
2.4.11.70 Laser tracking reproduction . . . 2302
2.4.11.71 Laser weld seam trajectory reproduction . . . 2302
2.4.11.72 Movement to the starting point of weld record . . . 2302
2.4.11.73 Movement to the end of the weld record . . . 2303
2.4.11.74 Move to the laser sensor to find the site . . . 2303
2.4.11.75 Obtain the coordinate information of the laser sensor location . . . 2303
2.4.11.76 Example of laser peripheral sensor parameter configuration and debugging code . . . 2304
2.4.11.77 Code example of laser trajectory scanning and trajectory reproduction . . . 2304
2.4.11.78 Code examples for laser locating and real-time tracking . . . 2305
2.4.11.79 Code example of the extended axis synchronized with the robot for laser tracking . . . 2306
2.4.12 Force Control . . . 2307
2.4.12.1 Force Sensor Configuration . . . 2307
2.4.12.2 Get Force Sensor Configuration . . . 2307
2.4.12.3 Force sensor activation . . . 2307
2.4.12.4 Force Sensor Zeroing . . . 2308
2.4.12.5 Setting the force transducer reference coordinate system . . . 2308

2.4.12.6 Setting the load weight under the force transducer . . . 2308
2.4.12.7 Setting the load center of mass under the force transducer . . . 2308
2.4.12.8 Getting the load weight under the force transducer . . . 2309
2.4.12.9 Obtaining the center of mass of the load under the force transducer . . . 2309
2.4.12.10 Automatic zeroing of force sensors . . . 2309
2.4.12.11 Obtaining force/torque data in the reference coordinate system . . . 2310
2.4.12.12 Obtaining Force Sensor Raw Force/Torque Data . . . 2310
2.4.12.13 Force sensor configuration and automatic zero correction code example . . . 2310
2.4.12.14 Load weight identification records . . . 2311
2.4.12.15 Load weight identification calculation . . . 2311
2.4.12.16 Load center of mass identification records . . . 2311
2.4.12.17 Load center of mass identification calculation . . . 2312
2.4.12.18 Force sensor load identification code example . . . 2312
2.4.12.19 Collision Guard . . . 2313
2.4.12.20 Collision guard code example . . . 2313
2.4.12.21 constant force control . . . 2314
2.4.12.22 A code example of constant force control with damping . . . 2315
2.4.12.23 Helix Exploration . . . 2316
2.4.12.24 Rotational Insertion . . . 2316
2.4.12.25 Code Example for Spiral Search, Linear Insertion, and Other Commands . . . 2316
2.4.12.26 Linear insertion . . . 2318
2.4.12.27 Examples of instruction code for spiral exploration, straight line insertion, etc . . . 2318
2.4.12.28 Surface positioning . . . 2320
2.4.12.29 Calculation of the center plane position begins . . . 2320
2.4.12.30 Calculate end of mid-plane position . . . 2320
2.4.12.31 Sample code for surface localization . . . 2321
2.4.12.32 Soft control on . . . 2322
2.4.12.33 Soft control off . . . 2322
2.4.12.34 Soft control code example . . . 2322
2.4.12.35 Load recognition filter initialization . . . 2323
2.4.12.36 Initialization of load recognition variables . . . 2324
2.4.12.37 Load Recognition Main Program . . . 2324
2.4.12.38 Getting Load Recognition Results . . . 2324
2.4.12.39 Robot load identification code example . . . 2324
2.4.12.40 Force Sensor Assisted Drag . . . 2325
2.4.12.41 Get force sensor drag switch status . . . 2326
2.4.12.42 The force sensor turns on automatically after the error is cleared. . . . 2326
2.4.12.43 Force sensor assisted drag code example . . . 2326
2.4.12.44 Setting up hybrid drag switches and parameters for six-dimensional force and joint impedance . . . 2327
2.4.12.45 Six dimensional force and joint impedance mixed drag code example . . . 2327
2.4.12.46 Impedance start and stop control . . . 2328
2.4.12.47 Example code for impedance start and stop control . . . 2328
2.4.12.48 Enable Torque Compensation Function and Compensation Coefficients . . . 2329
2.4.13 Extended Axis . . . 2330
2.4.13.1 Setting the 485 Extended Axis Parameters . . . 2330
2.4.13.2 Getting 485 Expansion Axis Configuration Parameters . . . 2330
2.4.13.3 Setting the 485 expansion axis enable/disable . . . 2330
2.4.13.4 Setting the 485 Extended Axis Control Mode . . . 2331
2.4.13.5 Setting the 485 extended axis target position (position mode) . . . 2331
2.4.13.6 Setting the 485 extended axis target torque (torque mode)-not yet available . . . 2331
2.4.13.7 Setting the 485 extended axis back to zero . . . 2331
2.4.13.8 Clearing 485 Expansion Axis Error Messages . . . 2332
2.4.13.9 Get 485 extended axis servo status . . . 2332

2.4.13.10 Setting the 485 extended axis target speed (velocity mode) . . . 2333
2.4.13.11 Setting the 485 extended axis data axis number in the status feedback . . . 2333
2.4.13.12 Setting the 485 Extended Axis Motion Acceleration and Deceleration Speed . . . 2333
2.4.13.13 Setting the 485 extended axis emergency stop acceleration and deceleration speeds . 2333
2.4.13.14 Get 485 Extended Axis Motion Acceleration and Deceleration . . . 2334
2.4.13.15 Get 485 extended axis emergency stop acceleration and deceleration speeds . . . 2334
2.4.13.16 Extended axis control code example . . . 2334
2.4.13.17 Parameter configuration for UDP extended axis communication . . . 2336
2.4.13.18 Get UDP extended axis communication parameters . . . 2337
2.4.13.19 Load UDP communication . . . 2337
2.4.13.20 Offloading UDP communication . . . 2337
2.4.13.21 UDP Extended Axis Communication Recovery after Abnormal Disconnection . . . 2338
2.4.13.22 UDP extension axis communication is closed after abnormal disconnection. . . . 2338
2.4.13.23 UDP Extended Axis Parameter Configuration . . . 2338
2.4.13.24 Setting the extended robot position relative to the extended axis . . . 2339
2.4.13.25 Setting the extended axis system DH parameter configuration . . . 2339
2.4.13.26 UDP Extended Axis Enable . . . 2339
2.4.13.27 UDP Extended Axis Zero Return . . . 2340
2.4.13.28 UDP Extended Axis Tap Start . . . 2340
2.4.13.29 UDP Extended Axis Tap Stop . . . 2340
2.4.13.30 Example of UDP extension axis configuration and tapping code . . . 2341
2.4.13.31 Setting the reference point of the extended axis coordinate system - four-point method 2342
2.4.13.32 Calculating the Extended Axis Coordinate System - Four Point Method . . . 2342
2.4.13.33 Reference Point Setting for the Shifter Coordinate System - Four-Point Method . . . 2342
2.4.13.34 Shifter Coordinate System Calculation - Four Point Method . . . 2343
2.4.13.35 Setting of the calibration reference point in the position in the coordinate system of the end of the translator . . . 2343
2.4.13.36 Applying the Extended Axis Coordinate System . . . 2343
2.4.13.37 Obtain the extended axis coordinate system . . . 2344
2.4.13.38 Extended axis coordinate system calibration code example . . . 2344
2.4.13.39 UDP Extended Axis Motion . . . 2346
2.4.13.40 UDP Extended axis motion code example . . . 2346
2.4.13.41 UDP extension axes synchronized with robot joint motion . . . 2347
2.4.13.42 UDP extension axes synchronized with robot joint motion code example . . . 2347
2.4.13.43 UDP extension axes synchronized with robot linear motion . . . 2348
2.4.13.44 UDP extension axes synchronized with robot linear motion code example . . . 2349
2.4.13.45 UDP extension axes synchronized with robot circular motion . . . 2349
2.4.13.46 UDP extension axes synchronized with robot circular motion code example . . . 2351
2.4.13.47 Setting the Extended DO . . . 2352
2.4.13.48 Setting up Extended AO . . . 2352
2.4.13.49 Setting the Extended DI Input Filter Time . . . 2352
2.4.13.50 Setting the Extended AI Input Filter Time . . . 2353
2.4.13.51 Waiting for extended DI input . . . 2353
2.4.13.52 Waiting for extended AI input . . . 2353
2.4.13.53 Get Extended DI Value . . . 2354
2.4.13.54 Get Extended AI Value . . . 2354
2.4.13.55 Extended IO code examples . . . 2354
2.4.13.56 Removable Device Enable . . . 2355
2.4.13.57 Zeroing of removable units . . . 2355
2.4.13.58 Movable unit linear motion . . . 2355
2.4.13.59 Movable unit circular motion . . . 2356
2.4.13.60 Stopping motion of movable devices . . . 2356
2.4.13.61 Portable device code example . . . 2356
2.4.13.62 Laser sensor recording points . . . 2357

2.4.13.63 Sample code for laser sensor recording points . . . 2357
2.4.13.64 Set the synchronous movement strategy of the extended axis and the robot . . . 2358
2.4.13.65 Code example for setting the synchronous motion strategy of the extended axis and the robot . . . 2358
2.4.13.66 Control array type sucker . . . 2359
2.4.13.67 Get the state of the array suction cup . . . 2359
2.4.13.68 Wait for the sucker state . . . 2360
2.4.13.69 Array type sucker control command code example . . . 2360
2.4.13.70 Upload an open protocol Lua file . . . 2361
2.4.13.71 Get the slave board parameters . . . 2361
2.4.13.72 Write the slave DO . . . 2361
2.4.13.73 Write the slave AO . . . 2362
2.4.13.74 Read the slave DI . . . 2362
2.4.13.75 Read the slave AI . . . 2362
2.4.13.76 Wait for the extension DI input . . . 2363
2.4.13.77 Wait for extended AI input . . . 2363
2.4.13.78 Example of slave mode related interface instruction code . . . 2363
2.4.14 Weld . . . 2364
2.4.14.1 Setting Welding Process Curve Parameters . . . 2364
2.4.14.2 Obtaining Welding Process Curve Parameters . . . 2365
2.4.14.3 Setting of welding current and output analog correspondences . . . 2365
2.4.14.4 Setting the welding voltage and output analog correspondence . . . 2366
2.4.14.5 Acquiring the correspondence between welding current and output analog quantity . 2366
2.4.14.6 Getting welding voltage and output analog correspondence . . . 2367
2.4.14.7 Setting the welding current . . . 2367
2.4.14.8 Setting the welding voltage . . . 2367
2.4.14.9 Setting Oscillation Parameters . . . 2368
2.4.14.10 Example code for setting welding parameters . . . 2368
2.4.14.11 Instant setup of swing parameters . . . 2370
2.4.14.12 The detection parameters of unexpected interruption of robot welding arc were obtained 2371
2.4.14.13 Set the detection parameters of robot welding arc unexpected interruption . . . 2371
2.4.14.14 Obtain the robot welding interrupt recovery parameters . . . 2371
2.4.14.15 Set the robot welding interrupt recovery parameters . . . 2372
2.4.14.16 Set welder control mode to expand DO port . . . 2372
2.4.14.17 Setting the welder control mode . . . 2372
2.4.14.18 Welding Start . . . 2372
2.4.14.19 End of welding . . . 2373
2.4.14.20 swing start . . . 2373
2.4.14.21 end of swing (math.) . . . 2373
2.4.14.22 Positive wire feed . . . 2374
2.4.14.23 Reverse wire feed . . . 2374
2.4.14.24 aspiration . . . 2374
2.4.14.25 Set the robot to resume welding after welding interruption . . . 2374
2.4.14.26 Set the robot to exit welding after welding interruption . . . 2375
2.4.14.27 Sample code for robot welding control . . . 2375
2.4.14.28 Segmented welding startup . . . 2376
2.4.14.29 Sample robot segment welding code . . . 2376
2.4.14.30 Simulated swing start . . . 2377
2.4.14.31 End of simulation swing . . . 2377
2.4.14.32 Start trajectory detection warning (no movement) . . . 2377
2.4.14.33 End trajectory detection warning (no movement) . . . 2378
2.4.14.34 Wobble start . . . 2378
2.4.14.35 Robot swing gradient welding code example . . . 2378
2.4.14.36 Wobble end . . . 2379

2.4.14.37 Extended IO-Configuration Welder Gas Detection Signal . . . 2379
2.4.14.38 Extended IO-Configuration of welder arc start signal . . . 2379
2.4.14.39 Extended IO-Configuration of the welder's reverse wire feed signal . . . 2380
2.4.14.40 Extended IO-Configuration of the welder's forward wire feed signal . . . 2380
2.4.14.41 Extended IO-Configuration of the welder's arc start success signal . . . 2380
2.4.14.42 Extended IO-Configuration Welder Ready Signal . . . 2380
2.4.14.43 Extended IO-Configuration Weld Interrupt Recovery Signal . . . 2381
2.4.14.44 Example code for setting up extended IO solder signals . . . 2381
2.4.14.45 Arc tracking control . . . 2382
2.4.14.46 Arc tracking AI passband selection . . . 2382
2.4.14.47 Arc tracking + multi-layer multi-channel compensation on . . . 2383
2.4.14.48 Arc Tracking + Multi-Layer Multi-Channel Compensation Off . . . 2383
2.4.14.49 Offset Coordinate Change - Multi-layer Multi-pass Welding . . . 2383
2.4.14.50 Example code for multi-layer multi-pass welding arc tracking . . . 2384
2.4.14.51 Selection of AI channels for current feedback in arc tracking welding machines . . . 2386
2.4.14.52 Selection of AI channel for voltage feedback of arc tracking welding machine . . . 2386
2.4.14.53 Current feedback conversion parameters of arc tracking welding machine . . . 2386
2.4.14.54 Voltage feedback conversion parameters of arc tracking welding machine . . . 2387
2.4.14.55 Example arc tracking code . . . 2387
2.4.14.56 Setting Up the Weld Wire Seek Expansion IO Port . . . 2388
2.4.14.57 code example . . . 2388
2.4.14.58 Welding wire position finding start . . . 2390
2.4.14.59 End of wire position finding . . . 2390
2.4.14.60 Calculate the wire finding offset . . . 2391
2.4.14.61 Waiting for wire seek to complete . . . 2391
2.4.14.62 Wire seek contact points written to database . . . 2391
2.4.14.63 Example of robot wire locating code . . . 2392
2.4.14.64 Set the welding voltage to start gradually . . . 2393
2.4.14.65 Set the welding voltage gradient to end . . . 2394
2.4.14.66 Set the welding current to start gradually . . . 2394
2.4.14.67 Set the welding current to gradually end . . . 2394
2.4.14.68 Robot welding current voltage gradient code example . . . 2394
2.4.14.69 Set custom swing parameters . . . 2395
2.4.14.70 Gets custom swing parameters . . . 2396
2.4.14.71 Custom swing parameter code example . . . 2396
2.4.15 Others . . . 2397
2.4.15.1 Get SSH public key . . . 2397
2.4.15.2 Issue the SCP command . . . 2398
2.4.15.3 Calculate the MD5 value of a file in a specified path . . . 2398
2.4.15.4 Robot SSH, MD5 instruction code example . . . 2398
2.4.15.5 Setting the Robot 20004 Port Feedback Cycle . . . 2399
2.4.15.6 Get robot 20004 port feedback cycle . . . 2399
2.4.15.7 Robot 20004 port state feedback cycle configuration code example . . . 2399
2.4.15.8 Robot software upgrade . . . 2400
2.4.15.9 Get robot software upgrade status . . . 2400
2.4.15.10 Robot software upgrade code example . . . 2400
2.4.15.11 Download Point Table Database . . . 2401
2.4.15.12 Upload point table database . . . 2401
2.4.15.13 Point table update lua file . . . 2401
2.4.15.14 Robot point table operation code example . . . 2402
2.4.15.15 Controller log download . . . 2402
2.4.15.16 Download all data sources . . . 2402
2.4.15.17 Data backup package download . . . 2403
2.4.15.18 Download the controller data code example . . . 2403

2.4.15.19 Set up encoder upgrade . . . 2403
2.4.15.20 Set up joint firmware upgrade . . . 2403
2.4.15.21 Set up control box firmware upgrade . . . 2404
2.4.15.22 Set up the end firmware upgrade . . . 2404
2.4.15.23 Joint full parameter profile upgrade . . . 2404
2.4.15.24 Example of robot slave firmware upgrade code . . . 2405
2.4.15.25 Robot Operating System upgrade (LA control box) . . . 2405
2.4.15.26 Obtain upgrade results of robot operating system (LA control box) . . . 2405
2.4.15.27 Robot MCU log generation . . . 2406
2.4.16 Appendix . . . 2406
2.4.16.1 Source Code Download . . . 2406
2.4.16.2 Source Code Compilation . . . 2408
2.4.16.2.1 Windows Platform Python SDK Compilation . . . 2408
2.4.16.2.2 Linux Platform Python SDK Compilation . . . 2411
2.4.16.3 Notes . . . 2415
2.4.16.3.1 Potential Issues . . . 2415
2.4.16.3.1.1 Version Compatibility . . . 2415
2.4.16.3.1.2 Error Codes . . . 2415
2.5 Error Code Comparison Table . . . 2415

3 FAIRINO SimMachine 2421
3.1 Virtual Machine . . . 2421
3.1.1 Overview . . . 2421
3.1.2 Operating Instructions . . . 2421
3.1.2.1 Installing VMware Workstation . . . 2421
3.1.2.2 Opening the Virtual Machine Image . . . 2422
3.1.2.3 Accessing WebApp from Windows . . . 2424
3.2 Virtual Machine - Docker . . . 2424
3.2.1 Linux Docker Image Deployment . . . 2424
3.2.1.1 Operating Environment . . . 2424
3.2.1.2 Install Docker . . . 2425
3.2.1.3 Image Configuration . . . 2425
3.2.1.3.1 Import Docker Image . . . 2425
3.2.1.3.2 Create Custom Bridge Network . . . 2426
3.2.1.3.3 First-time Docker Container Startup . . . 2427
3.2.2 Web Operation of Virtual Robot . . . 2427
3.2.2.1 Container Normal Startup . . . 2427
3.2.2.2 Operating the Virtual Robot . . . 2428
3.2.2.3 User IP Address Modification . . . 2430
3.2.3 Virtual Machine Version Upgrade/Downgrade . . . 2432
3.2.3.1 Overview . . . 2432
3.2.3.2 Upgrade/Downgrade Preparation and Considerations . . . 2432
3.2.3.2.1 Operation Preparation . . . 2432
3.2.3.2.2 Considerations . . . 2432
3.3 Appendix . . . 2433
3.3.1 Appendix 1: Enabling Virtualization in the BIOS . . . 2433
3.3.2 Appendix 2: Adding a Virtual NIC (Loopback Network Adapter) . . . 2438
3.3.3 Appendix 3: Root Permissions . . . 2446
3.3.4 Appendix 4: Docker Basic Commands . . . 2447

4 FRCap Plug-in system 2449
4.1 Introduction . . . 2449
4.2 Quick start . . . 2450
4.2.1 I don't have FRCap . . . 2450

4.2.2 I already have FRCap . . . 2452
4.2.3 Build FRCap . . . 2452
4.2.4 Hello FRCap . . . 2453
4.3 Create wizard . . . 2454
4.3.1 Parameter configuration . . . 2454
4.3.1.1 Basic information . . . 2454
4.3.2 Advanced configuration . . . 2455
4.3.3 Download . . . 2455
4.4 Backstage management . . . 2455
4.4.1 Backend upgrade . . . 2455
4.4.2 Modules version . . . 2456
4.5 Development guidance . . . 2456
4.5.1 Development environment and conditions . . . 2456
4.5.2 Development tools . . . 2457
4.5.3 FRCap project structure . . . 2458
4.5.4 Front-end frcap-ui and frcap-api use . . . 2459
4.5.5 Backend Custom Command Development . . . 2461
4.5.5.1 Database Operation Example (LA) . . . 2461
4.5.5.2 Database Operation Example (QX) . . . 2461
4.5.5.3 Socket Communication Example . . . 2462
4.6 FRCap in WebApp . . . 2463
4.6.1 FRCap management . . . 2463
4.6.2 Configuration class usage . . . 2463
4.6.3 Application class use . . . 2463
4.7 FRCap case . . . 2464
4.7.1 FAIRINO Palletizer . . . 2464
4.7.1.1 Palletizing workpiece configuration . . . 2464
4.7.1.2 Palletizing pallet configuration . . . 2465
4.7.1.3 Advanced palletizing configuration . . . 2465
4.7.1.4 Palletizing equipment size configuration . . . 2466
4.7.1.5 Palletizing mode configuration . . . 2467
4.7.1.6 Palletizing program generation . . . 2468
4.7.1.7 Get palletizing recipe . . . 2468
4.7.1.8 Get the list of existing formula names for palletizing . . . 2471
4.8 API description . . . 2471
4.8.1 act command . . . 2471
4.8.1.1 Save teaching points . . . 2471
4.8.2 sta command . . . 2472
4.8.2.1 Get robot status data . . . 2472
4.8.3 get command . . . 2473
4.8.3.1 Get teaching points . . . 2473
4.8.3.2 Get system configuration . . . 2474
4.8.4 set command . . . 2475
4.8.4.1 Issue system variable instructions . . . 2475
4.8.4.2 Get system variable instructions . . . 2475
4.8.5 better-sqlite3 directive . . . 2476
4.8.5.1 Query the first row of records in the database . . . 2476
4.8.5.2 Query all records in the database . . . 2476
4.8.5.3 Execute database statements . . . 2477
4.8.6 socket command . . . 2477
4.8.6.1 socket send . . . 2477
4.8.6.2 socket recv . . . 2477
4.8.7 File Operation Commands . . . 2478
4.8.7.1 Write File Content . . . 2478

4.8.7.2 Read File Content . . . 2478
4.8.7.3 Modify File Permissions . . . 2478
4.8.7.4 Read Directory Contents (Including Subdirectories) . . . 2478
4.8.8 Compression/Decompression Commands . . . 2479
4.8.8.1 Create tar.gz Archive . . . 2479
4.8.8.2 Extract tar.gz File . . . 2479
4.9 Version update instructions . . . 2480
4.10 Palletizing FRCap . . . 2480
4.10.1 Palletizing FRCap plug-in package management . . . 2480
4.10.2 Recipe management . . . 2482
4.10.2.1 Obtain . . . 2483
4.10.2.2 New . . . 2483
4.10.2.3 Rename . . . 2486
4.10.2.4 Export . . . 2487
4.10.2.5 Copy . . . 2488
4.10.2.6 Delete . . . 2489
4.10.2.7 Edit . . . 2491
4.10.2.8 Import . . . 2492
4.10.3 Recipe configuration . . . 2493
4.10.3.1 Workstation Settings . . . 2495
4.10.3.2 Palletizing Function I/O Wiring Configuration . . . 2495
4.10.3.3 Palletizing Function I/O Communication Test . . . 2498
4.10.3.4 Box operation . . . 2499
4.10.3.5 Box parameters . . . 2510
4.10.3.6 Pallet configuration . . . 2512
4.10.3.6.1 Pattern configuration . . . 2514
4.10.3.7 Pattern operation . . . 2514
4.10.3.8 Pattern parameters . . . 2520
4.10.3.8.1 Advanced configuration . . . 2523
4.10.4 Program Generation . . . 2532
4.10.4.1 Single Pick Point Program . . . 2535
4.10.4.2 Dual Pick Point Program . . . 2535

5 FR LUA Programming Script 2537
5.1 PDF Download . . . 2537

6 Robot Communication 2539
6.1 Introduction to CNDE . . . 2539
6.2 CNDE data frame protocol format . . . 2543
6.3 CNDE function operation . . . 2543
6.3.1 Input configuration and input data . . . 2543
6.3.2 Output configuration and output data . . . 2545
6.3.2.1 Output configuration . . . 2545
6.3.2.2 Output start and stop . . . 2547
6.3.2.3 The client receives the output data . . . 2547
6.3.3 CNDE auxiliary function . . . 2548
6.3.3.1 Messages . . . 2548
6.3.3.2 Switch the version number of CNDE protocol of robot . . . 2548
6.3.3.3 Acquire that version information of the robot soft firmware . . . 2548
6.4 Robot input and output register . . . 2548
6.4.1 Read input register . . . 2549
6.4.2 Write output register . . . 2551
6.4.3 CNDE input,output register interactive application . . . 2552

7 frcobot_ros **2555**
7.1 Overview . . . 2555
7.2 Install . . . 2556
7.2.1 Environmental requirements . . . 2556
7.2.2 ROS installation & requirements . . . 2556
7.2.3 Compile . . . 2556
7.3 Quick start . . . 2557
7.3.1 frcobot_hw . . . 2557

8 frcobot_ros2 **2559**
8.1 Introduction . . . 2559
8.2 fairino_hardware . . . 2559
8.2.1 Basic Environment Installation . . . 2559
8.2.2 Compile and build fairino_hardware . . . 2560
8.3 Quick start . . . 2561
8.3.1 Starting the process . . . 2561
8.3.2 View the manipulator state feedback process . . . 2562
8.3.3 Process for issuing instructions . . . 2562
8.3.4 Modifying the parameter flow . . . 2564
8.4 API Description . . . 2565

9 moveIt2 **2577**
9.1 Plug-in Introduction . . . 2577
9.2 Rapid Access . . . 2577
9.2.1 MoveIt2 Installation . . . 2577
9.2.1.1 Clone Fairino MoveIt2 Plugin . . . 2577
9.2.1.2 Compile Feature Packs . . . 2578
9.2.2 Configuring the Moveit2 model of the Fairino robotic arm . . . 2578
9.2.2.1 Creating A Workspace . . . 2579
9.2.2.2 Configuration Robot . . . 2579
9.2.2.2.1 Launching the configuration interface . . . 2579
9.2.2.2.2 Configuring Self-Collisions . . . 2580
9.2.2.2.3 Configuring Virtual Joints . . . 2581
9.2.2.2.4 Configuring Planning Groups . . . 2582
9.2.2.2.5 Configuring Robot Poses . . . 2584
9.2.2.2.6 Configuring End Effectors . . . 2585
9.2.2.2.7 ros2_control URDF Modifications . . . 2585
9.2.2.2.8 ROS 2 Controllers . . . 2586
9.2.2.2.9 Moveit Controllers . . . 2587
9.2.2.2.10 Launch Files . . . 2588
9.2.2.2.11 Author Information . . . 2590
9.2.2.2.12 Generate Launch File . . . 2590
9.2.2.3 Launch . . . 2591
9.2.2.4 Moveit2 Usage . . . 2592
9.2.3 Fairino_hardware Plugin (custom robot moveit configuration package) . . . 2594
9.2.3.1 Fairino_hardware Plugin Compilation . . . 2594
9.2.3.2 Fairino_hardware Plugin Usage . . . 2595
9.2.3.3 Running Plug-ins . . . 2595
9.2.3.4 Running Result . . . 2596
9.2.4 Fairino_hardware Plugin (official robot moveit configuration package) . . . 2597
9.3 MTC Sample Code Package . . . 2598
9.3.1 MTC Sample Code Package Introduction . . . 2598
9.3.2 MTC Sample Code Package Compilation . . . 2599
9.3.2.1 MTC Sample Code Package Cloning . . . 2599

9.3.2.2 Robot Model Selection . . . 2599
9.3.2.3 MTC Sample Code Package Compilation . . . 2600
9.3.2.3.1 Compiling the fairino_description function package . . . 2600
9.3.2.3.2 Compile robot function packs . . . 2600
9.3.2.3.3 Compile fairino_mtc_demo function package . . . 2600
9.3.3 The MTC Sample Code Package Runs . . . 2600
9.3.3.1 Rviz2 Interface . . . 2600
9.3.3.2 Robot Motion . . . 2601
9.4 Caveat . . . 2602
9.4.1 Fairino_hardware Plugin Version Synchronization . . . 2602
9.4.2 Problems Likely To Be Encountered . . . 2604
9.4.2.1 The robot model may not be loaded on the right side of the Configure Robot Feature Pack. . . . 2604
9.4.2.2 After generating the package, there is an error running it. . . . 2604
9.5 Summarize . . . 2604

10 Brochure **2605**

11 Qualification Certification **2607**

12 Secondary Development **2609**

13 Ontology & Dimensional Drawings **2611**

14 2D CAD Files **2613**

15 3D Models **2615**

16 FAIRINO SimMachine **2617**

17 Palletizer frcap **2619**

18 CPP SDK **2621**

19 C# SDK **2623**

20 Python SDK **2625**

21 Java SDK **2627**

22 Status Feedback Protocol **2629**

23 Communication Command Protocol **2631**

24 Robot software **2633**

FAIRINO ROBOT



Language Switching

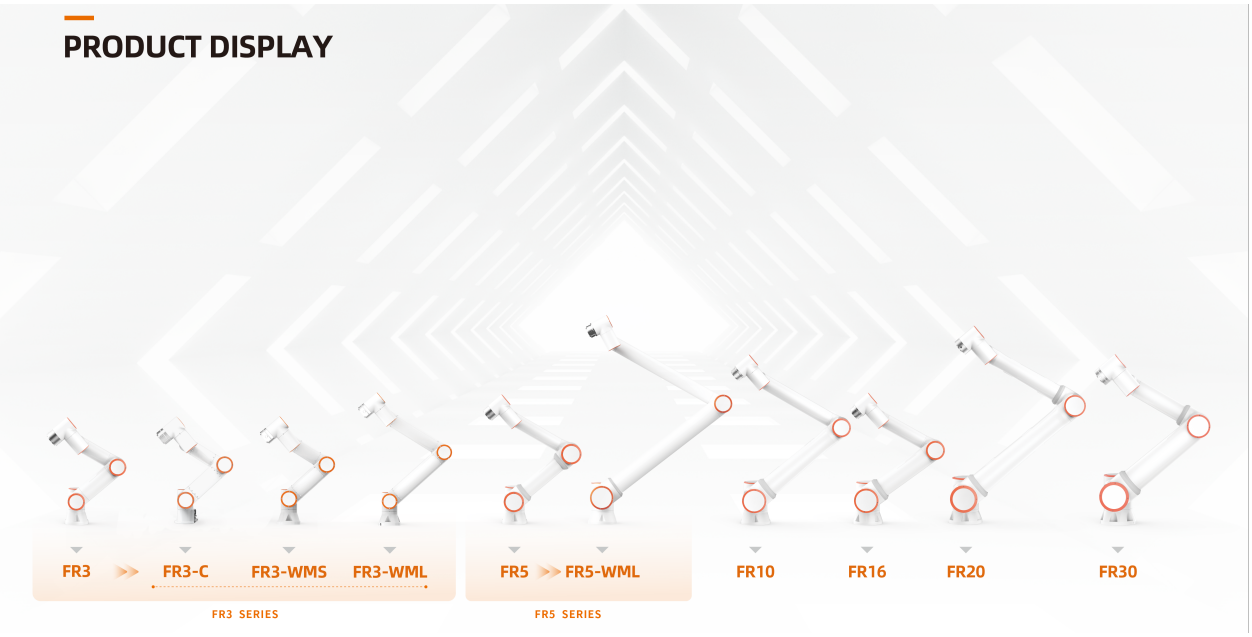
Italiano

CHAPTER

ONE

COLLABORATIVE ROBOT

1.1 Product matrix



1.2 Quick start

1.2.1 Install and power on the robot

1.2.1.1 Install the robot arm

When the collaborative robot is installed on the mounting base, use a compliant number of bolts (strength not less than 8.8) to tighten and fix the robot on the mounting base; it is recommended to use two compliant pin holes and pins on the mounting base for robot positioning to improve the installation accuracy of the robot and prevent the robot from moving due to collisions. When the robot has high running precision requirements, be sure to add pins to position the robot.

Table 1.1-1 Robot mounting parts standard

Collaborative robot model	Bolt	Bolt torque	Pin Hole Specifications
FR3	4 pieces of M6	10Nm	5mm
FR3-WMS	4 pieces of M6	10Nm	5mm
FR3-WML	4 pieces of M6	10Nm	5mm
FR3-C	4 pieces of M6	10Nm	5mm
FR5	4 pieces of M8	20Nm	8mm
FR10	4 pieces of M8	25Nm	8mm
FR16	4 pieces of M8	25Nm	8mm
FR20	6 pieces of M10	45Nm	8mm
FR30	6 pieces of M10	45Nm	8mm
FR30L	6 pieces of M10	45Nm	8mm

Important: It is recommended that the robot mounting base meet the following requirements to ensure that the robot is installed firmly and stably:

(1)The robot mount needs to be strong enough and have sufficient load-bearing capacity. It should be able to carry at least 5 times the weight of the robot and at least 10 times the 1-axis torque.

(2)The surface of the robot mounting seat should be flat to ensure close contact with the contact surface of the robot;

(3)The robot mounting base should be strong enough, fixed firmly, and will not resonate with the robot;

(4)When the robot and other parts move at the same time, the mounting base should be isolated from other moving parts, and should not be fixed together to avoid vibration interference during the movement;

(5)If the robot is installed on a mobile platform or an external axis, the acceleration of the mobile platform or external axis should be as low as possible;

1.2.1.2 Connect the control box

This series of robots can be configured with three control boxes with different power inputs. For details on the control box power input, see the control box nameplate information. The robot needs to be electrically grounded. The external wiring of the manipulator control system is connected using pluggable and quickly installed plugs.

A. 30-60VDC

B. 176-264VAC~50-60Hz

C. 100-240VAC~50-60HZ

Note: The AC input control box has two versions: narrow voltage and wide voltage. The control box wiring terminals and appearance are consistent, and cannot be distinguished by appearance alone. Please confirm by the control box nameplate and power on after confirmation.

The wiring panel of the collaborative robot is shown in the following figure:



Figure 1.2-1 Control box wiring panel

The button box interface is the control port of the teach pendant by default, and the IP address is 192.168.58.2. Use a network cable to connect the button box interface and the computer. The computer IP address is set to 192.168.58.10 or the same network segment. Open the Google browser and enter 192.168.58.2 You can access the teach pendant page.

1.2.1.3 Know the button box and end LED

1.2.1.3.1 The button box

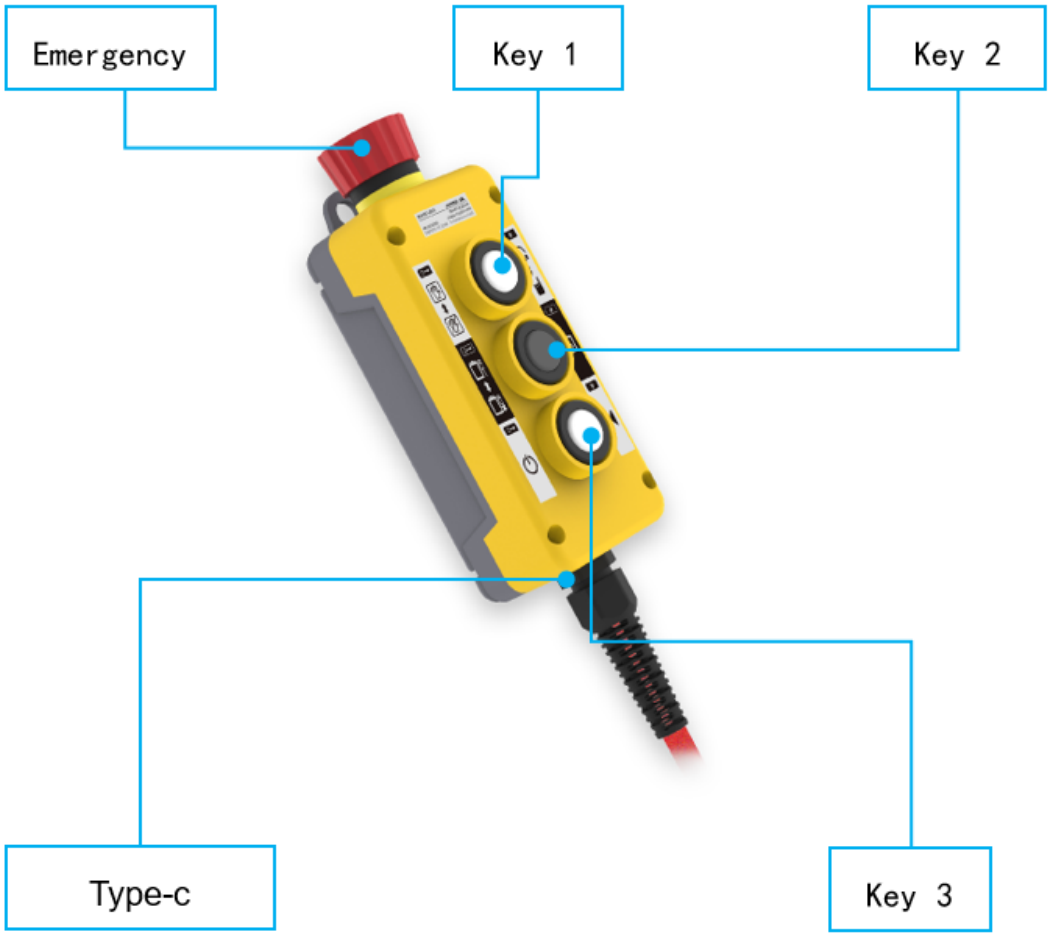
1.2.1.3.1.1 60 Button Box (POE) (BX01)

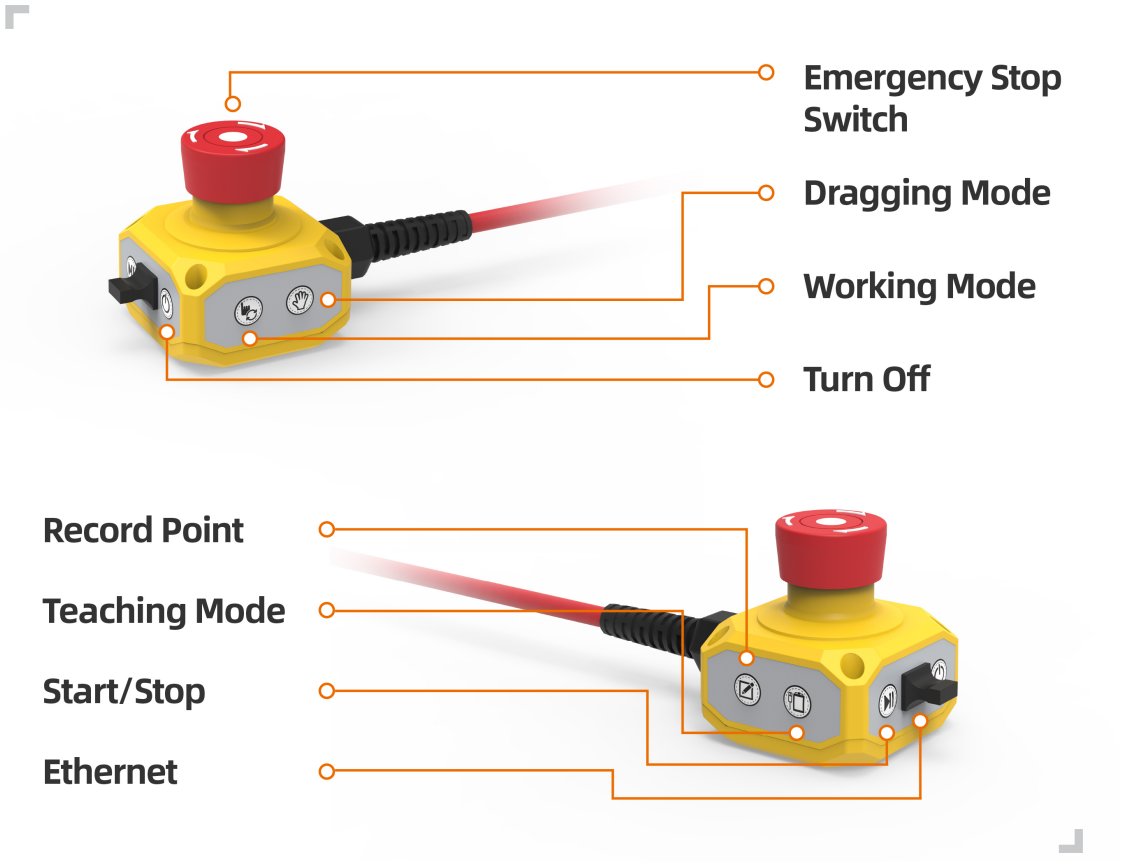
Figure 1.3-1 60 Button Box (POE) (BX01)

1.2.1.3.1.2 60 Button Box (POE) (BX02)-V1.0

Figure 1.3-2 60 Button Box (POE) (BX02)-V1.0

Table 1.3-1 Control box wiring panel button description


Emergency
Key 1
Key 2
Type-c
Key 3


Emergency Stop Switch
Dragging Mode
Working Mode
Turn Off
Record Point
Teaching Mode
Start/Stop
Ethernet

Name	Function
Emergency stop switch	When pressing the emergency stop switch, the robot enters the state of emergency stop.
Start/Stop	Start/stop running program.
Ethernet	Connect to the teaching pendant.
Turn off	No enabled.
Record point	Record the teaching point.
Teaching mode	Enter/exit with the teaching pendant state.
Working mode	Automatic/manual mode switch.
Dragging mode	Enter/exit drag mode.

1.2.1.3.1.3 60 Button Box (POE) (BX02)-V2.0

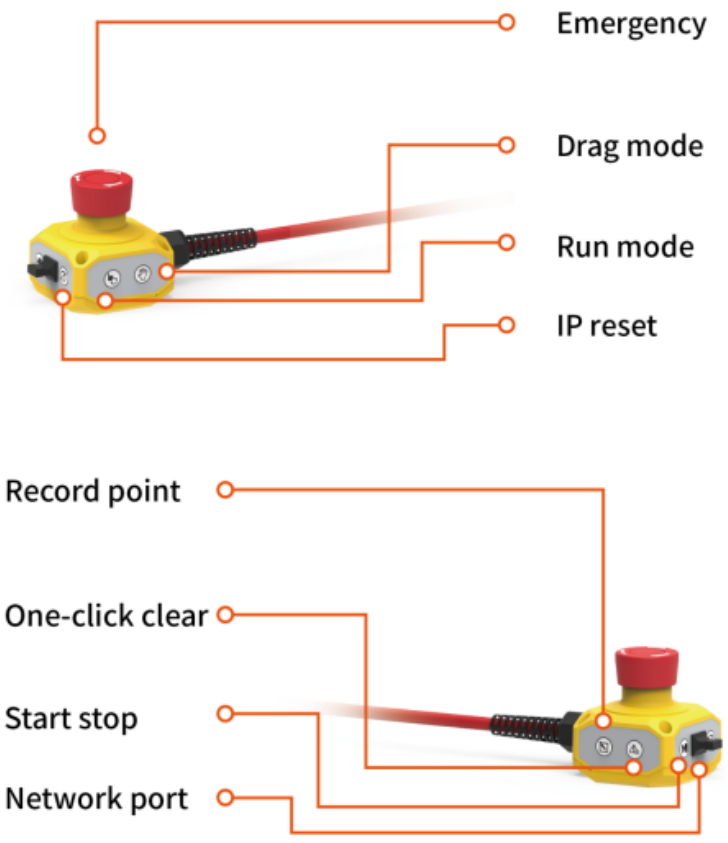


Figure 1.3-3 60 Button Box (POE) (BX02)-V2.0

Table 1.3-2 Control box wiring panel button description

Button name	Function
Emergency stop switch	When the emergency stop switch is pressed, the robot enters the emergency stop state
Start stop	Start/stop running the program
Network port	Connect to web teach pendant
IP reset	Reset network port IP
Record point	Record teaching point
One-click clear	Clear all recoverable errors
Running mode	Automatic/manual mode switching
Drag mode	Enter/exit drag mode

1.2.1.3.2 The end LED

Table 1.3-4 The end LED definition table

Function	LED color
When communication is not established	“Off”, “Red”, “Green” and “Blue” alternately
Automatic mode	Blue long bright
Manual mode	Green long bright
Drag Mode	White cyan long bright
Button box record point (only when using button box)	Purple blinks twice
Start running (only when using the button box)	Cyan blue flashes twice
Enter the state of unmatched button box (only when using the button box)	Blue flashes twice
Stop operation (only when using the button box)	Red flashes twice
Error reporting (only when using the button box)	Red long bright
Zero calibration completed	White cyan flashes three times
Enable	Yellow flashes twice

1.2.1.4 Power on enable

Before powering on, please confirm that the emergency stop button of the button box is released, press the red switch button of the control box to power on, and the LED light at the end will be in a steady green state after enabling successfully.

1.2.1.5 Power Off

Important: When using this device, please make sure to stop all running programs, disable the status query function, and confirm that the operating status is “Stopped” before turning off the power. This operation is to protect the safety of the device and stored data and avoid data loss or system damage caused by sudden power outages.



Figure 1.5-1 Turn off the power button

1.2.2 WebApp access login

1.2.2.1 Access and log in to the WebApp interface

1. Turn on the control box and connect the network cable to the PC;
2. Open the chrome browser on the PC and access the target URL 192.168.58.2;
3. Enter the user name and password and click Login to log in to the WebApp.

The initial user name is admin and the password is 123.

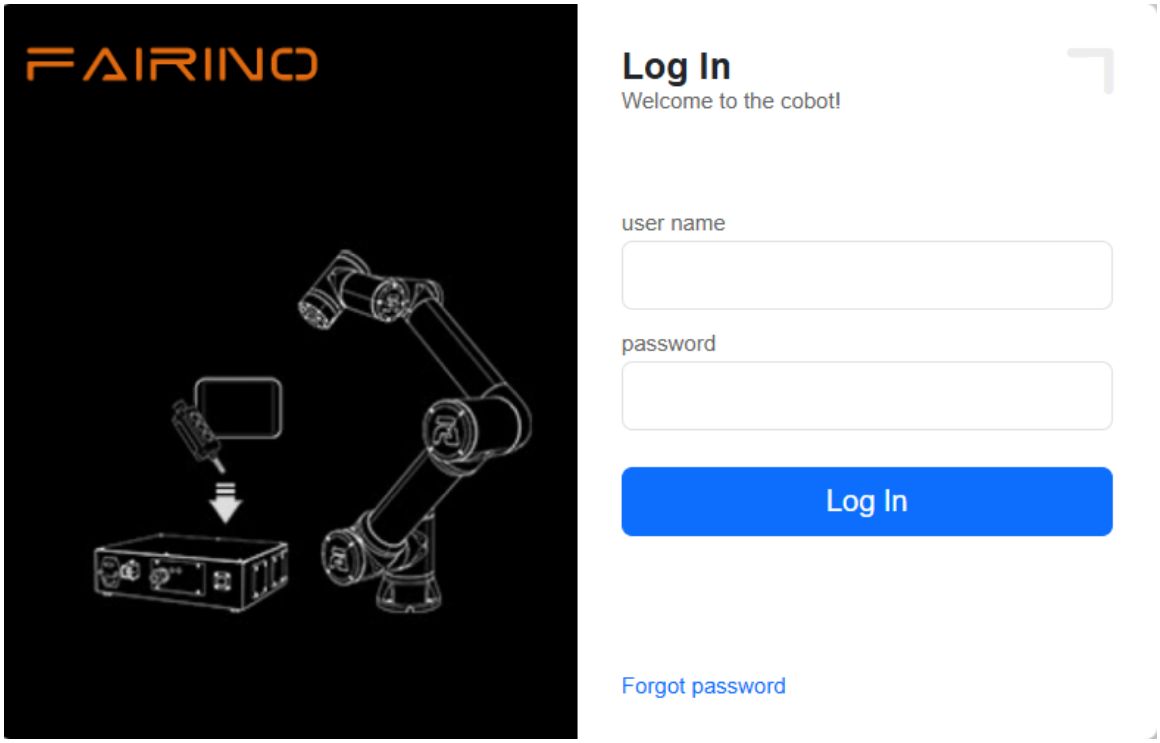


Figure 2.1-1 Login interface

1.2.2.2 Simple understanding of WebApp interface

After the login is successful, the system enters the “Initial”. Mainly include:

1. FAIRINO LOGO;
2. Menu bar zoom button;
3. Menu bar;
4. Robot control area;
5. Robot status area;
6. 3D simulation robot - 3D scene operation;
7. 3D simulation robot - robot body operation;
8. Robot supporting functions;
9. Robot and supporting function status.

The initial interface of the system is shown in the figure below:

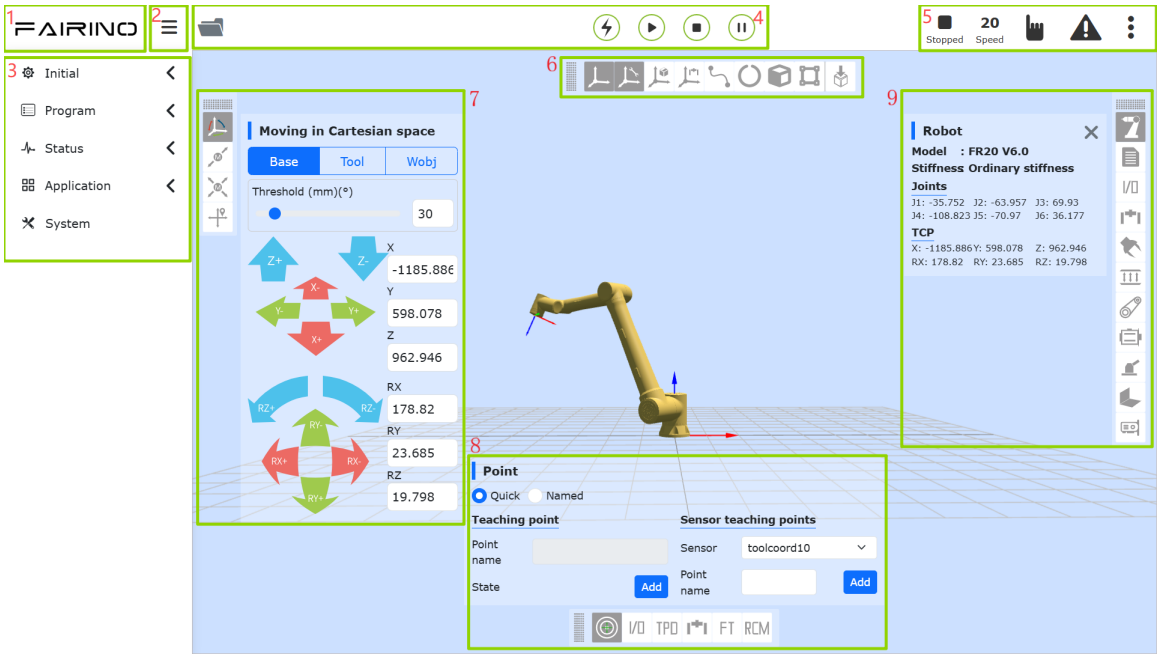


Figure 2.2-1 Schematic diagram of the initial interface of the system

1.2.2.2.1 Control area



Note:

name: **“Press the teach program button”**

effect: Open the teaching programs for coding programming, graphical programming, and Node Graph programming.



Note:

name: **Enable button**

effect: Enable the robot



Note:

name: **Start button**

effect: Upload and start running the teaching program



Note:

name: **Stop button**

effect: Stop the current teaching program running



Note:

name: **Pause/Resume button**

effect: Pause and resume the current teaching program

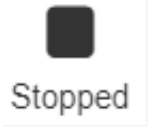
Important: The pause instruction is at the end of the program and cannot be judged.

1.2.2.2.2 Status Bar

Note: 

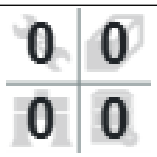
name: **Error state**

effect: There is an error in the current robot operation, hide when no error.

Note: 

name: **robot status**

effect: Stopped-stopRunning-runPause-pauseDrag-drag

Note: 

name: **Robot Tool Coordinate System, Workpiece Coordinate System, Extended Axis Coordinate System, and Load Number**

effect: Top Left – Current Tool Coordinate System Number, Top Right – Current Workpiece Coordinate System Number, Bottom Left – Current Extended Axis Coordinate System Number, Bottom Right – Current Load Number

---

Note: 

name: **Running speed percentage**

effect: The speed of the robot when it is running in the current mode

---

Note: 

name: **Automatic mode**

effect: Robot automatic operation mode.When the global speed adjustment in manual mode and automatic mode is turned on and the speed is specified, the global speed will be automatically adjusted to the specified speed

Note: 

name: **Manual mode**

effect: Robot manual mode for robot teaching operations.

Note: 

name: **Robot State Fold/Unfold Button**

effect: Fold/Unfold Tool Coordinate System, Workpiece Coordinate System, Extended Axis Coordinate System, Load, Robot Drag Status, Local/Remote Mode, Robot Connection Status, BOOT Mode, and Account Information Content

Click the collapse/expand button to view the following status information.

Note: 

name: **Tool coordinate system number**

effect: Display the tool coordinate system number of the current application

Note: 

name: **Workpiece coordinate system number**

effect: Display the workpiece coordinate system number currently applied

Note: 

name: **Extended axis coordinate system number**

effect: Displays the currently applied extended axis coordinate system number



Note:

name: **Payload**

effect: Display the load weight and center of gravity coordinates X, Y, Z of the current application.



Note:

name: **Drag state**

effect: The current robot can drag



Note:

name: **Drag state**

effect: The current robot is not draggable



Note:

name: **Robot Local Mode**

effect: The current robot is controlled via the control box.



Note:

name: **Robot Remote Mode**

effect: Currently, robots can only be controlled through PLC.



Note:

name: **Connection status**

effect: Robot connected



Note:

name: **Not connected status**

effect: Robot not connected



Note:

name: **Account information**

effect: Display username and permissions and logout user

1.2.3 Robot parameter setting

1.2.3.1 Set the installation method

The default installation mode of the robot is horizontal installation. When the installation mode of the robot is changed, the actual installation mode of the robot must be set in the "Initial - Base - Mounting" menu in time to ensure the normal operation of the robot.

To accommodate more flexible and diverse robot deployment scenarios, we offer a free installation feature. Users can click the 'Free Installation' tab under the "Initial - Base - Mounting" menu to access the robot free installation settings page. Manually adjust the "Base Tilt" and "Base Rotation" angles, and the 3D model will display the corresponding installation effect. After making changes, click the "Apply" button to complete the robot installation setup.

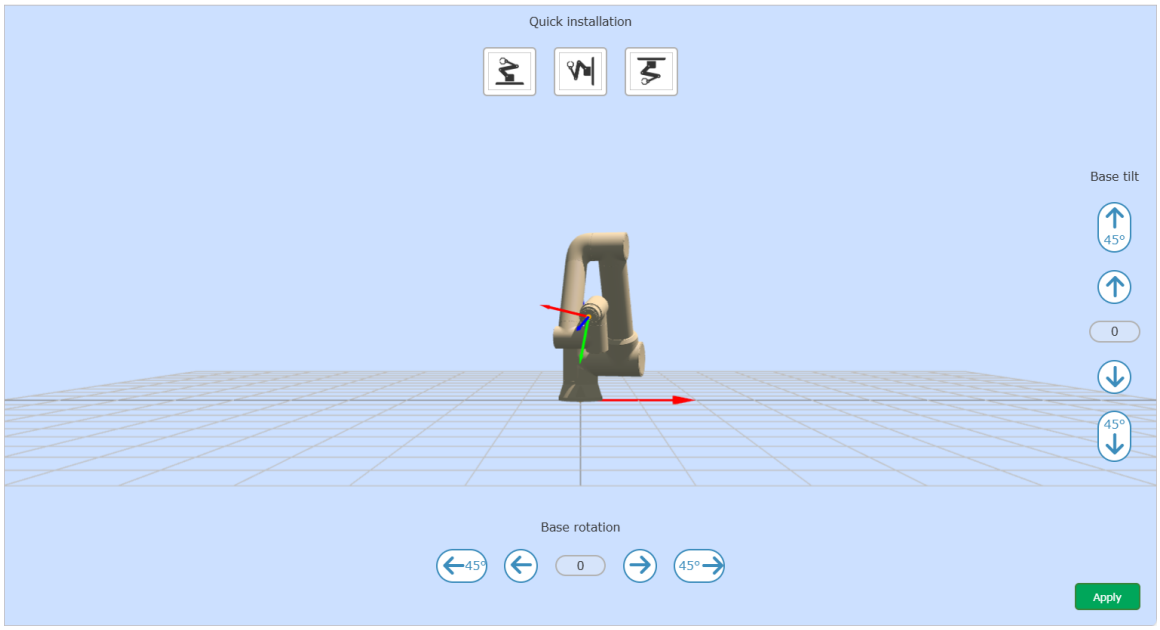


Figure 3.1-1 Robot installation

Important: After the installation of the robot is completed, the installation method of the robot must be set correctly, otherwise it will affect the use of the robot's dragging function and collision detection function.

1.2.3.2 Set end load

In the "Initial - Base - Payload" section, select the identification type for "Trajectory Identification" to enter the end effector load setting interface.

Note:

name: **Apply button**

effect: Click the application load number to view the corresponding load weight and center of gravity coordinates.

Note:

name: **Modify button**

effect: Click to open/close the motion recognition interface.

Note: 

name: **Rename button**

effect: Rename load name

Note: 

name: **Clear button**

effect: Clear current load information (set load weight and center of gravity coordinates to 0).

When configuring the end load, you can directly input the mass of the end tool and the corresponding center of mass coordinates X, Y, and Z, then click the “Apply” button to complete the setup.

At the same time, you can click the “Edit” button to open the “Motion Identification” interface for automatic load identification, which will be applied after the identification process is completed.

Important: The load mass cannot exceed the maximum load range of the robot.

- FR3:3kg
- FR5:5kg
- FR10:10kg
- FR16:16kg
- FR20:20kg
- FR30:30kg

The center of mass coordinate setting range is 0-1000, unit mm.

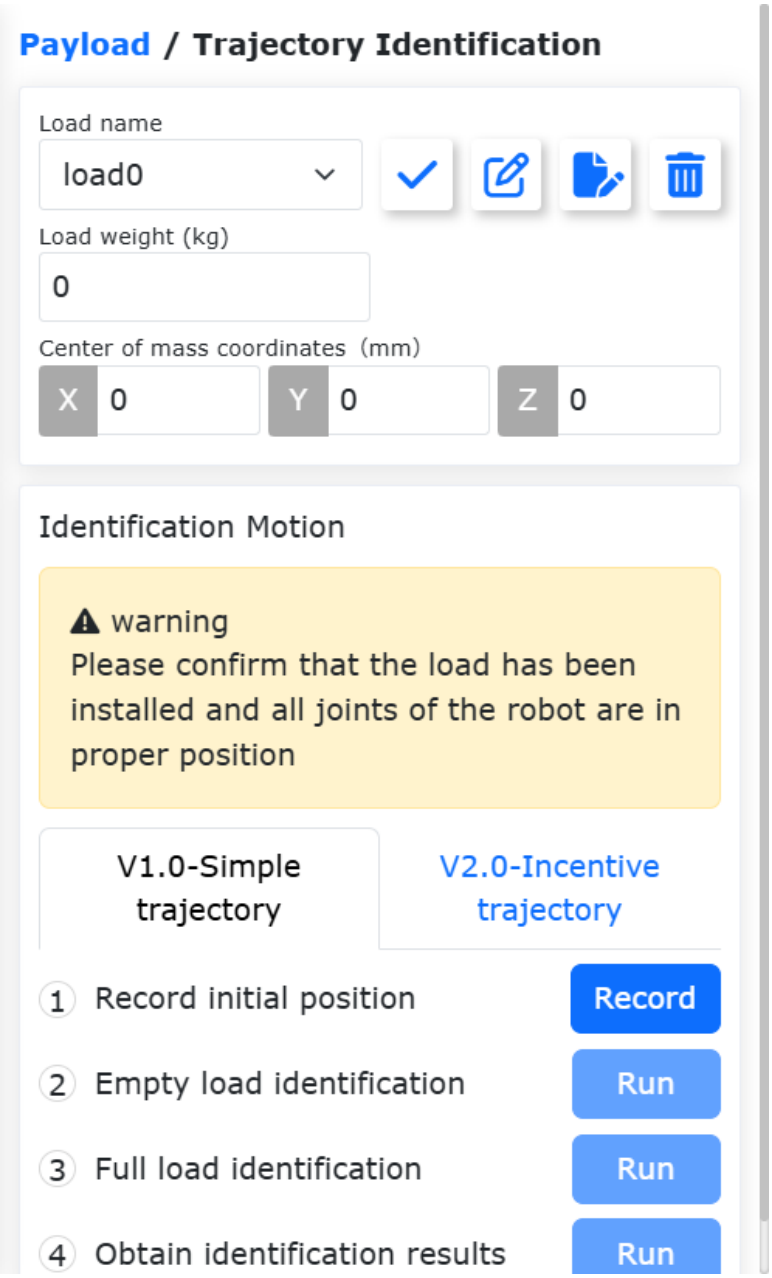


Figure 3.2-1 Schematic diagram of load setting

Important: After the load is installed at the end of the robot, the weight of the end load and the coordinates of the center of mass must be set correctly, otherwise it will affect the drag function of the robot and the use of the collision detection function.

1.2.3.3 Set tool coordinates

Enter the tool coordinates page by going to “Initial Settings” -> “Basic” -> “Tool Coordinates”.



Note:

name: **Apply button**

effect: Click to apply the tool coordinate system



Note:

name: **Modify button**

effect: Click to open/close the coordinate system calibration interface

---



Note:

name: **Rename button**

effect: Rename Tool Coordinate System Name



Note:

name: **Clear button**

effect: Clear current tool coordinate information

Tool coordinates can be modified, cleared and applied. In the drop-down list of tool coordinate systems, after selecting the corresponding coordinate system(the coordinate system name can be customized), the corresponding coordinate value, tool type and installation location (only displayed under sensor type tools) will be displayed below. After selecting a coordinate system, click the “Apply” button, and the currently used tool coordinate system will become the selected coordinate, as shown below.

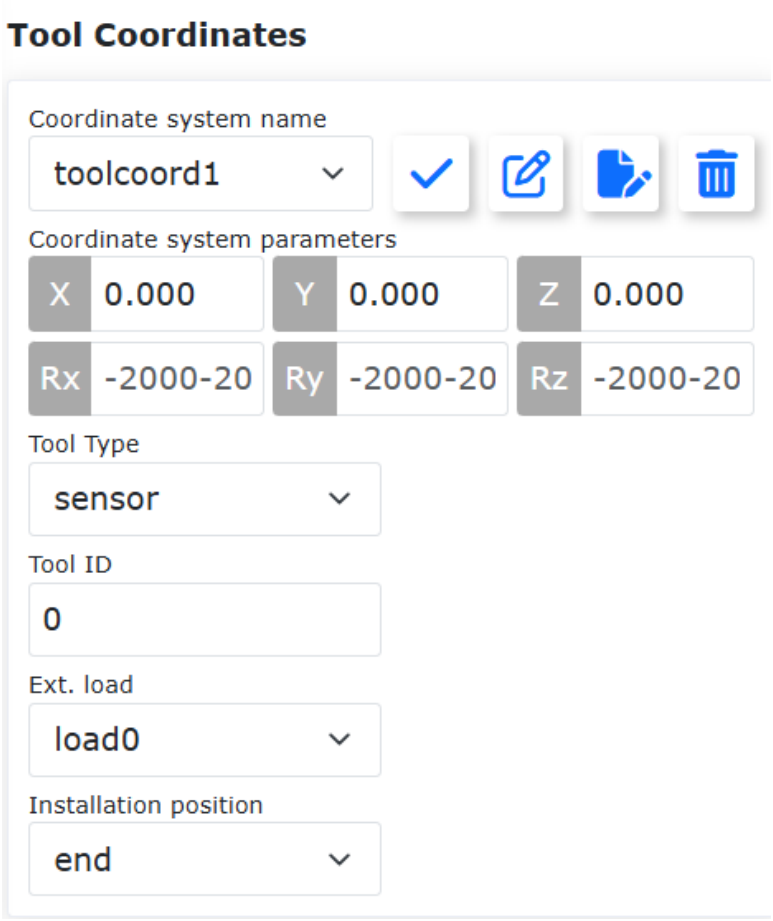


Figure 3.3-1 Set tool coordinates

Click “Modify” to reset the tool coordinate system of the number according to the prompt. The tool calibration method is divided into four-point method and six-point method. The four-point method only calibrates the tool TCP, that is, the position of the tool center point. Its posture is consistent with the end posture by default. The six-point method adds two points to the four-point method. , used to calibrate the pose of the tool.

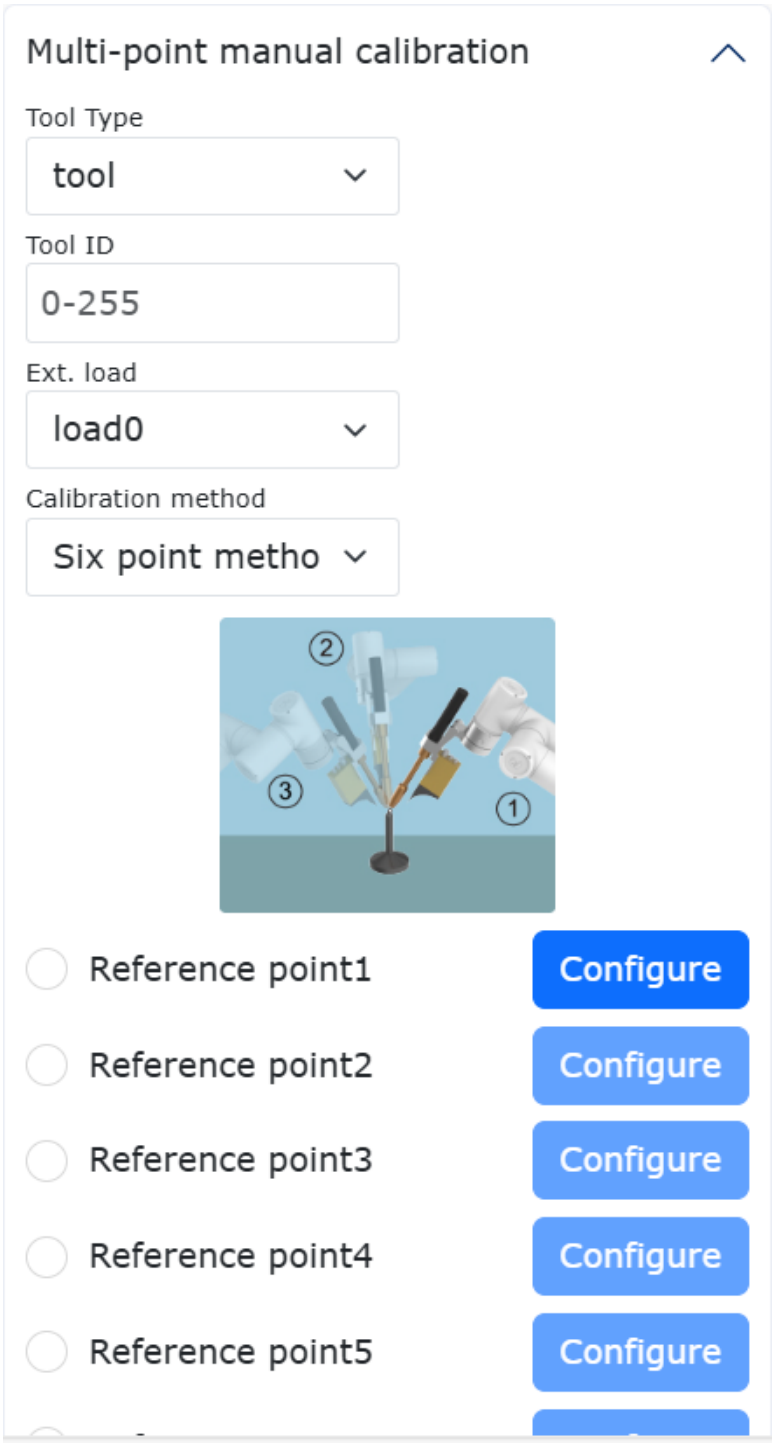


Figure 3.3-2 Tool Frame Calibration

Important:

1. After the tool is installed at the end, the tool coordinate system must be calibrated and applied, otherwise the position and attitude of the tool center point will not meet the expected values when the robot executes the motion command.
2. The tool coordinate system generally uses toolcoord1~toolcoord19, and toolcoord0 is used to indicate that the

position center of the tool TCP is at the center of the end flange. When calibrating the tool coordinate system, it is first necessary to apply the tool coordinate system to toolcoord0, and then select other tool coordinate systems for calibration. Calibration and application.

1.2.4 Robot manual teaching

1.2.4.1 Manual teaching and recording of teaching points

Manual teaching includes two methods:

- 1. Press and hold the end drag button to perform drag teaching;
- 2. Perform jog in 3D simulated robot - 3D object operation.

After teaching to the target position, you can save the teaching point in “Robot supporting functions - Point Name”. When saving the teaching point, the coordinate system of the teaching point is the coordinate system of the current robot application.

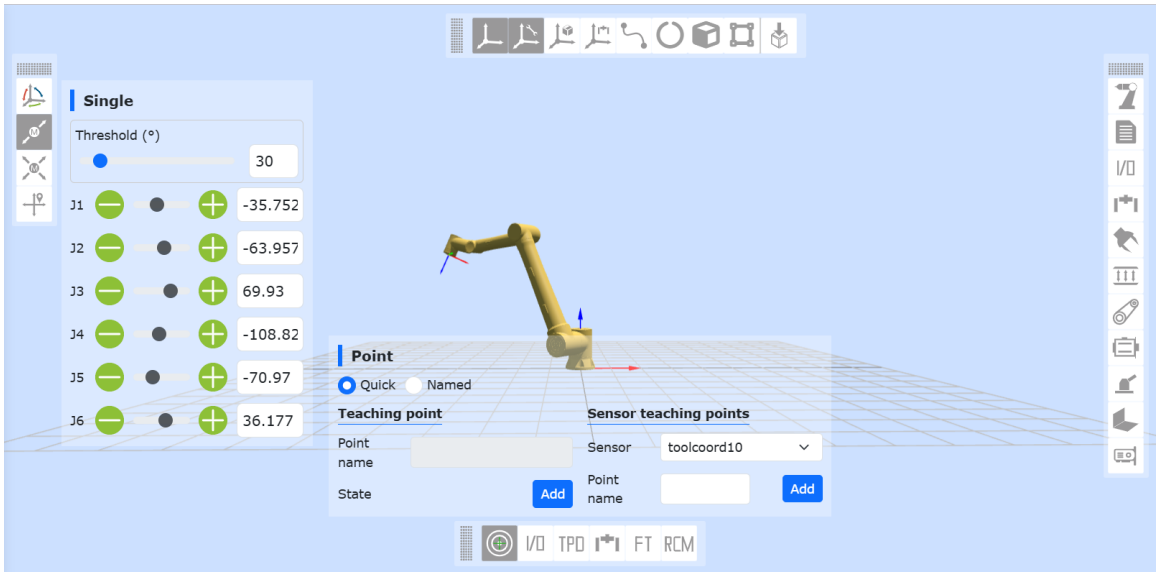


Figure 4.1-1 Manual teaching

1.2.4.2 View teaching point information

Click “Program - Points” to display all saved teaching point information. The current point mode is divided into “System mode” and “Point table mode”.

In this interface, you can import and export teaching point files. After selecting a teaching point, click the “Delete” button to delete the point information. The teaching point x, y, z, rx, ry, rz and v values can be modified. Enter the modified value, check the blue checkbox on the left, and click Modify above to modify the teaching point information.

Click the “Start Run” button to perform a single-point operation of the local teaching point and move the robot to the position of the point. In addition, users can search for teaching points by name.

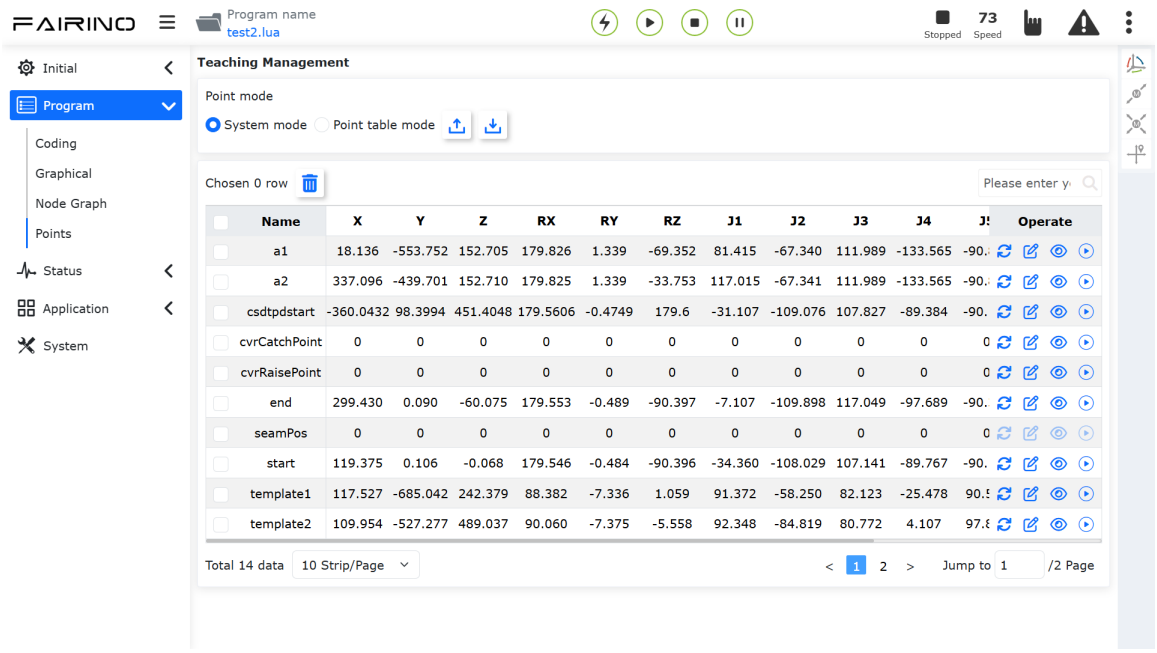


Figure 4.2-1 Teaching management interface

Important: The modified values of the teaching points x, y, z, rx, ry, rz should not exceed the working range of the robot.

1.2.5 Robot Quick Programming

1.2.5.1 Introduction to Simple Motion Instructions

PTP command: Click the “PTP” icon to enter the PTP command editing interface.

You can choose the point to be reached, and the smooth transition time setting can realize that the movement from this point to the next point is continuous. Whether to set the offset, you can choose to offset based on the base coordinate system and based on the tool coordinates, and pop up x, y, z, rx, ry, rz offset settings, PTP specific path is the optimal path automatically planned by the motion controller, click “Add” and “Apply” to save this command.

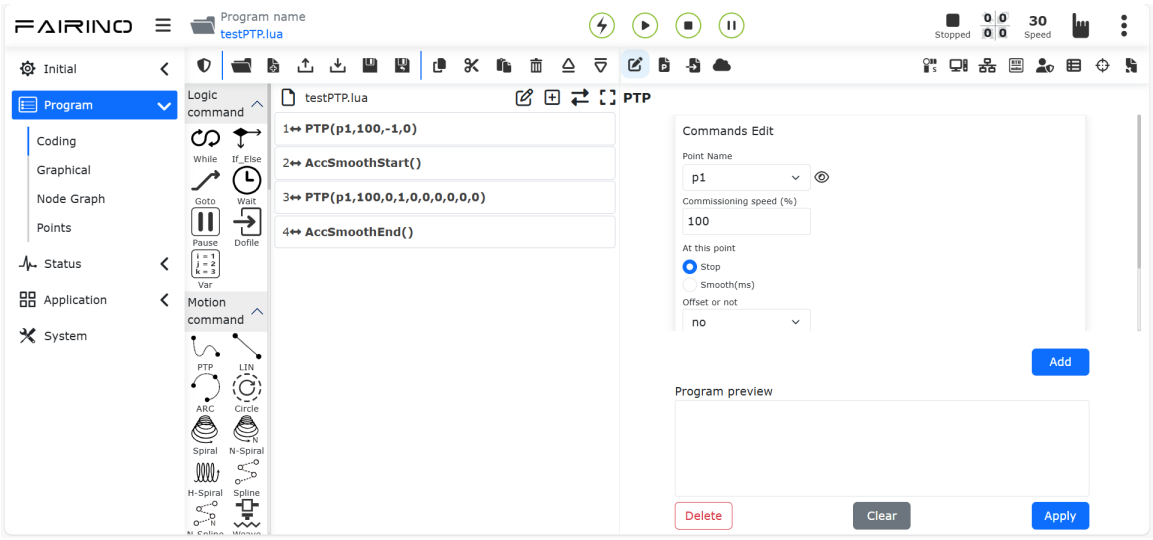


Figure 5.1-1 PTP command interface

Lin command: Click the “Lin” icon to enter the Lin command editing interface.

The function of this command is similar to the “PTP” command, but the path of the point reached by this command is a straight line.

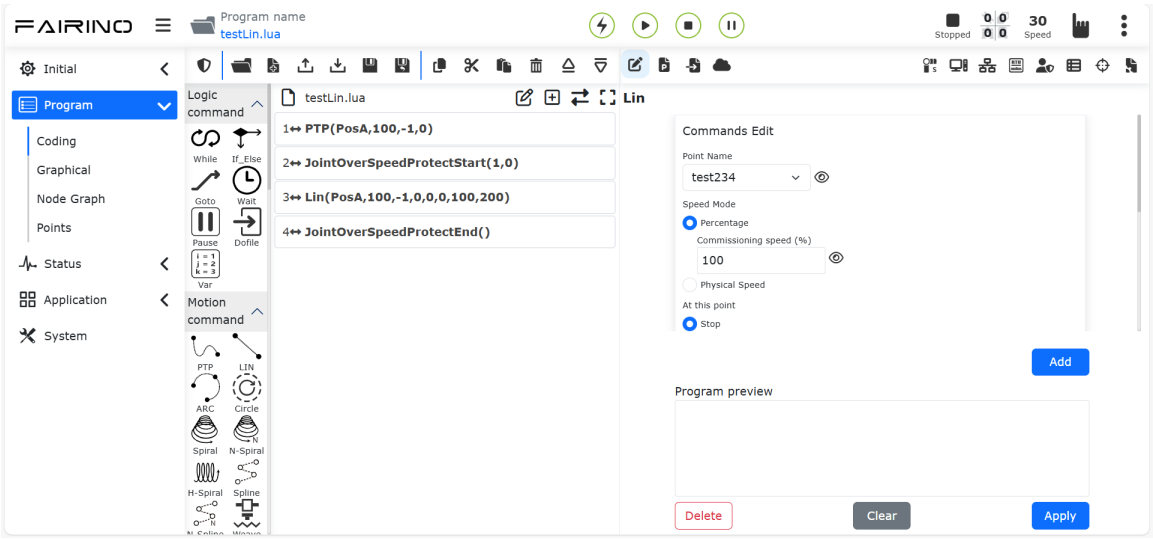


Figure 5.1-2 Lin command interface

1.2.5.2 Operate on program files

Use the toolbar at the top of the program tree to modify the program tree.



Note:

name: **Open**

effect: Open user program file



Note:

name: **New build**

effect: Select a template to create a new program file



Note:

name: **Import**

effect: Import the file into the user program folder



Note:

name: **Export**

effect: Export user program files to a local point.



Note:

name: **Save**

effect: Save file edits



Note:

name: **Save as**

effect: Rename the file and store it in the user program or template program folder.



Note:

name: **Copy**

effect: Duplicates a node and allows it to be used for other operations (eg: paste it elsewhere in the program tree).



Note:

name: **Paste**

effect: Allows you to paste previously cut or copied nodes.



Note:

name: **To cut**

effect: Cuts a node and allows it to be used for other operations (eg: paste it elsewhere in the program tree).



Note:

name: **Delete**

effect: Deletes a node from the program tree.



Note:

name: **Move up**

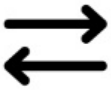
effect: Move the node up.



Note:

name: **Move down**

effect: Move the node down.



Note:

name: **Toggle edit mode**

effect: The program tree mode and lua editing mode switch each other.

1.2.5.3 Write and run a program

The left side is mainly for adding program commands. Click the icon above each keyword to enter the detailed interface. There are two main operations for adding program commands to files:

- 1. Open the relevant command and click the Apply button to add the command to the program;
- 2. Click the "Add" button first. At this time, the command is not saved in the program file. You need to click "Apply" to save the command to the file.

The second method often appears when multiple commands of the same type are issued. We add the Add button and display the added command content function for this type of command. Click the Add button to add an instruction. The Added Instructions display all added instructions. Click "Apply" to save the added instructions to the opened file on the right.

Click the Start button to run the program; click the Stop button to stop the program; click the Pause/Resume button to pause/resume the program; when the program is running, the currently executed program node is highlighted in green.

In manual mode, click the first icon on the right side of the node to make the robot execute the instruction alone, and the second icon is to edit the node content.

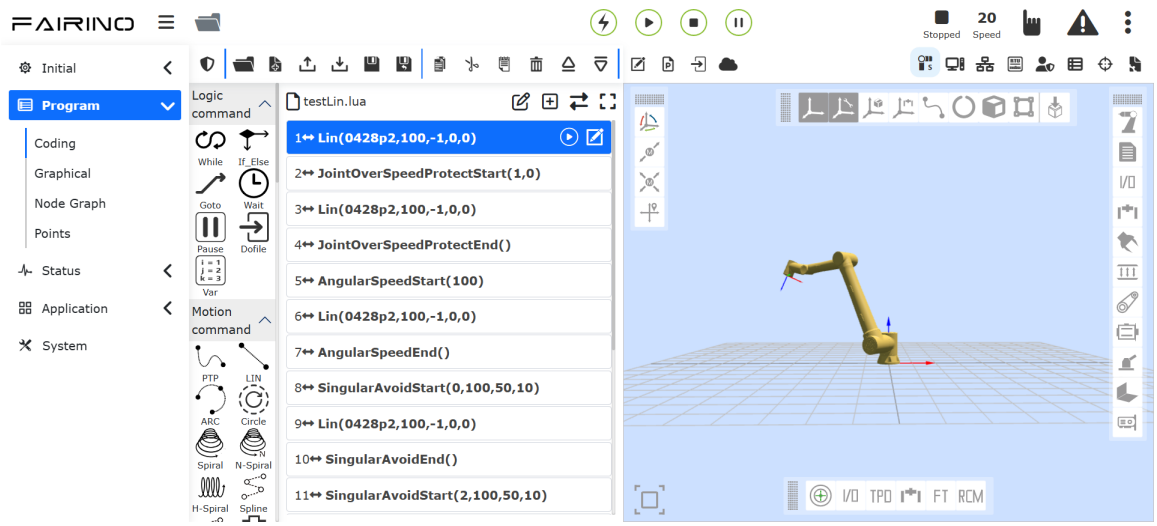


Figure 5.3-1 Program tree interface

1.3 Manual

1.3.1 Foreword

First of all, thank you very much for buying our company's FR series collaborative robot products. Our products have been carefully designed and tested many times to ensure that they can meet your needs in all aspects.

Please read this product user manual carefully to ensure that you can use our products correctly and get the best experience. If you encounter any problems during use, please refer to the fault exclusion part in the manual, or contact our after-salesperson. We thank you very much for your support and trust, and look forward to providing you with better services and products.

1.3.1.1 What is installed in the box

When you order a FR series collaborative robot, you will receive a box. Contains:

- **One collaborative robot**
- **One button box**
- **One control box (including one control box cable)**

Important: The power cable provided with FARINO is for use with FARINO robots only. It cannot be used for electrical equipment other than the robot.

1.3.1.2 Important security description

Robots are a device involving personal safety, so a safety assessment must be performed after each installation of the robot. You must follow all the security descriptions in Chapter 1.

1.3.1.3 How to use this manual

This manual contains the guidance information of robot installation programming. Manual includes:

- **Hardware installation part**:Machinery installation and electrical installation of robots
- **Analysis of the teachor software**:Robot teaching and programming

This manual is aimed at robot integrators, and integrators should receive basic mechanical and electrical training and be familiar with primary programming concepts.

1.3.1.4 Follow the relevant standards

Standard	Definition
2006/42/EC:2006	Machinery Directive: Directive 2006/42/EC of the European Parliament and of the Council of 17 May 2006 on machinery, and amending Directive 95/16/EC(recast)
2004/108/EC:2004	EMC Directive: Directive 2004/108/EC of the European Parliament and of the Council of 15 December 2004 on the approximation of the laws of the Member States relating to electromagnetic compatibility and repealing Directive 89/336/EEC
EN ISO 13850:2008	Safety of machinery: Emergency stop - Principles for design
EN ISO 13849-1:2008	Safety of machinery: Safety-related parts of control systems - Part 1: General principles of design
EN ISO 13849-2:2012	Safety of machinery: Safety-related parts of control systems - Part 2: Validation
EN ISO 12100:2010	Safety of machinery: General principles of design, risk assessment and risk reduction
EN ISO 10218-1:2011	Industrial robots: Safety Note: Content equivalent to ANSI/RIA R.15.06-2012, Part 1
ISO/TS 15066: 2016	Safety requirements for collaborative industrial robot Robots and robotic devices —Collaborative robots

1.3.2 Robot brief introduction

1.3.2.1 Basic parameters

Table 2.1-1 Basic parameters of robotics

Important: When the FR series robot performs attitude or coordinate system transformation, the angle rotation sequence calculated by the homogeneous transformation matrix is “ZYX” of the floating coordinate system.

Model / Parameter	FR3	FR5	FR10	FR16	FR20	FR30
Payload	3kg	5kg	10kg	16kg	20kg	30kg
Reach	622mm	922mm	1400mm	1034mm	1854mm	1403mm
Range	6 rotation angles					
Repeatability	±0.02mm	±0.03mm	±0.05mm	±0.03mm	±0.1mm	±0.1mm
Working range (soft limit)	Axis 1： +175° , -175°; Axis 2： +85° , -265°; Axis 3： +150° , -150°; Axis 4： +85° , -265°; Axis 5： +175° , -175°; Axis 6： +175° , -175°;			Axis 1： +175° , -175°; Axis 2： +85° , -265°; Axis 3： +160° , -160°; Axis 4： +85° , -265°; Axis 5： +175° , -175°; Axis 6： +175° , -175°;		
Working range (hard limit)	Axis 1： +179° , -179°; Axis 2： +89° , -269°; Axis 3： +152° , -152°; Axis 4： +89° , -269°; Axis 5： +179° , -179°; Axis 6： +179° , -179°;			Axis 1： +179° , -179°; Axis 2： +89° , -269°; Axis 3： +162° , -162°; Axis 4： +89° , -269°; Axis 5： +179° , -179°; Axis 6： +179° , -179°;		
Maximum speed	180°/s	180°/s	Axis1,2: 120°/s Other Axis: 180°/s		Axis1,2,3: 120°/s Other Axis: 180°/s	
Typical TCP speed	1m/s	1m/s	1.5m/s	1m/s	2m/s	2m/s
IP classification	IP54 (Optional IP65)					
Noise	<65dB				<70dB	
Robot mounting	Any orientation					
Input & Output	Power supply (24V, 1.5A), digital IO, analog IO, 485 communication					
Communication	I/O、TCP/IP、Modbus_TCP/RTU、Profinet、CC-Link、Ethernet/IP、EtherCAT					
Operating temperature	0-45℃					
Operating humidity	90%RH(non-condensing)					
Weight	≈15kg	≈22kg	≈40kg	≈40kg	≈85kg	≈85kg
Equipment storage	-25~60°C(no frost)					
Average breakdown repair time	2h					

Model / Parameter	FR3-WMS	FR3-WML	FR3-C	FR30L
Payload	3kg			35kg
Reach	622mm	922mm	622mm	1854mm
Range	6 rotation angles			
Repeatability	±0.02mm	±0.05mm	±0.05mm	±0.01mm
Working range (soft limit)	Axis 1: +175°, -175° Axis 2: +85° , -265° Axis 3: +150°, -150° Axis 4: +85° , -265° Axis 5: +175°, -175° Axis 6: +360°, -360°		Axis 1: +175°, -175° Axis 2: +85° , -265° Axis 3: +150°, -150° Axis 4: +85° , -265° Axis 5: +355°, 0° Axis 6: +175°, -175°	Axis 1: +175°, -175° Axis 2: +60° , -240° Axis 3: +160°, -160° Axis 4: +85° , -265° Axis 5: +175°, -175° Axis 6: +175°, -175°
Working range (hard limit)	Axis 1: +179°, -179° Axis 2: +89° , -269° Axis 3: +152°, -152° Axis 4: +89° , -269° Axis 5: +179°, -179° Axis 6: Unlimited		Axis 1: +179°, -179° Axis 2: +89° , -269° Axis 3: +152°, -152° Axis 4: +89° , -269° Axis 5: +359°, -4° Axis 6: +179°, -179°	Axis 1: +179°, -179° Axis 2: +62° , -242° Axis 3: +162°, -162° Axis 4: +89° , -269° Axis 5: +179°, -179° Axis 6: +179°, -179°
Maximum speed	Axis 1,2,3: 150°/s; Other Axis: 180°/s			Axis 1,2,3: 120°/s; Other Axis: 180°/s
Typical TCP speed	1m/s			2m/s
IP classification	IP54 (Optional IP65)	IP54 (Optional IP65)	IP54	IP54 (Optional IP65)
Noise	<65dB			<70dB
Robot mounting	Any orientation			Standard installation, ceiling installation
Input & Output	Power supply (24V, 1.5A), digital IO, analog IO, 485 communication, ethernet			
Communication	I/O、TCP/IP、Modbus_TCP/RTU、Profinet、CC-Link、Ethernet/IP、EtherCAT		I/O、TCP/IP、Modbus_TCP/RTU	I/O、TCP/IP、Modbus_TCP/RTU、Profinet、CC-Link、Ethernet/IP、EtherCAT
Operating temperature	0—45℃			
Operating humidity	90%RH (non-condensing)			
Weight	10.5kg	≈11kg		≈100kg
Equipment storage	-25~60℃ (no frost)			
Average breakdown repair time	2h			

1.3.2.2 Motion

Robotic arm installation space:

The installation of the robot body requires a space of 3m × 3m × 2m (length × width × height) to meet the motion under the maximum wingspan of the robot; if the user increases the end load by itself, make sure that there is a minimum 500mm gap in the installation space.

Note: The height space is affected by the height of the installation base, where 2m refers to the distance above the installation reference plane

---

Control cabinet installation space:

1. The control box should be placed in easy operation to prevent water flooding, 0.6m-1.5m from the ground.
2. The cabinet must be far away from the heat source.
3. On one side of the control box's heavy load line, there should be no obstruction within 150mm, and the remaining side shall be not covered within 100mm, which is convenient for heat dissipation and extraction.

Figure 2.1-1 FR3 model collaboration robot movement range

Figure 2.2-2 FR3-WML model collaboration robot movement range

Figure 2.2-3 FR3-WMS model collaboration robot movement range

Figure 2.2-4 FR3-C model collaboration robot movement range

Figure 2.1-5 FR5 model collaboration robot movement range

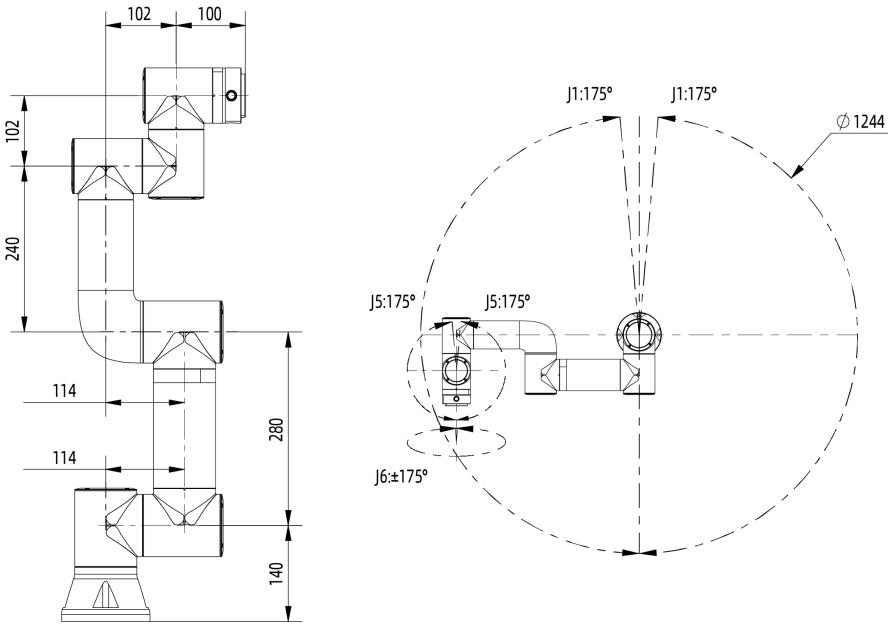
Figure 2.1-6 FR10 model collaboration robot movement range

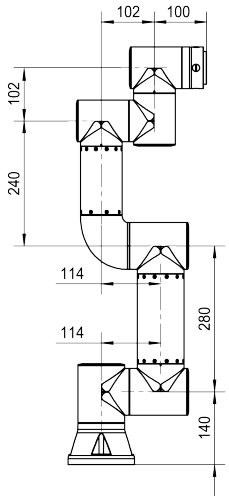
Figure 2.1-7 FR16 model collaboration robot movement range

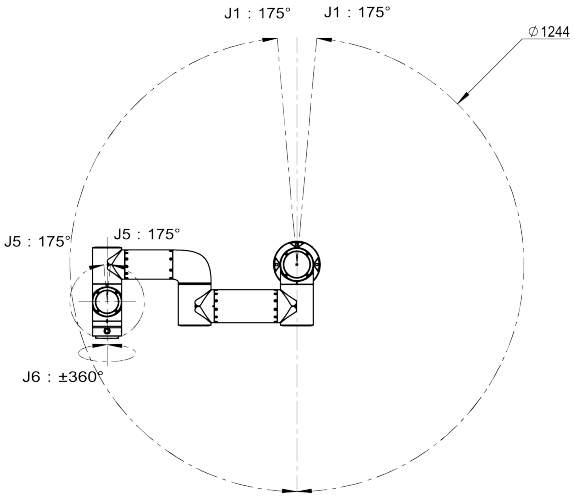
Figure 2.1-8 FR20 model collaboration robot movement range

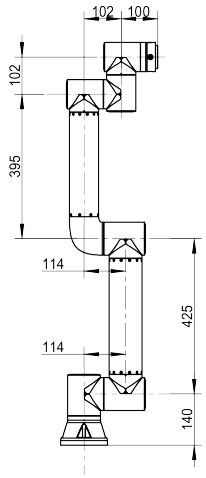
Figure 2.1-9 FR30 model collaboration robot movement range

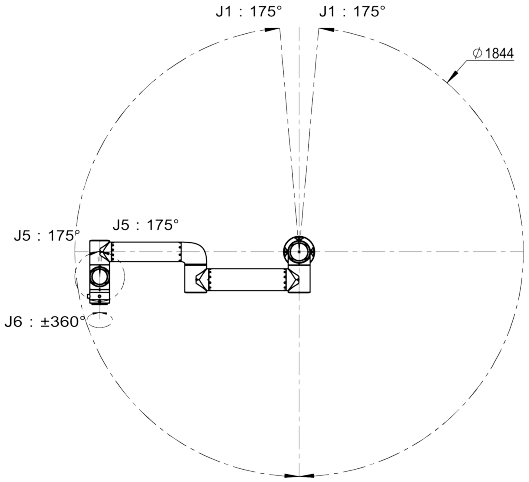
Figure 2.1-10 FR30L model collaboration robot movement range

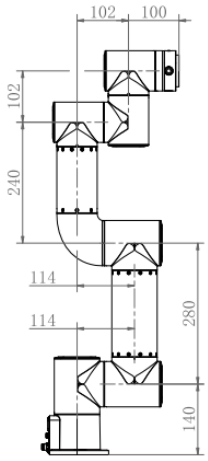

102
100
102
240
114
114
280
140
J1:175°
J1:175°
∅1244
J5:175°
J5:175°
J6:±175°

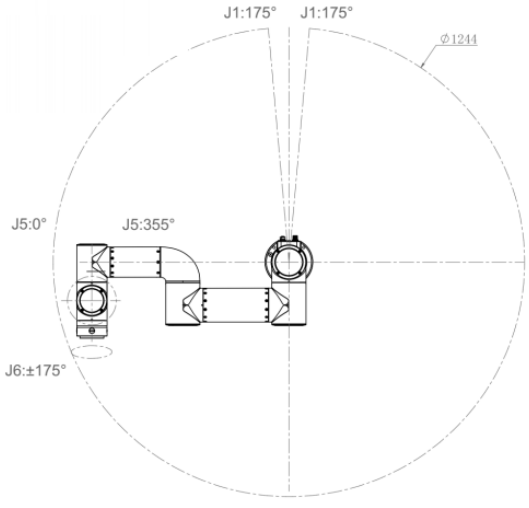

102
100
102
240
114
114
280
140

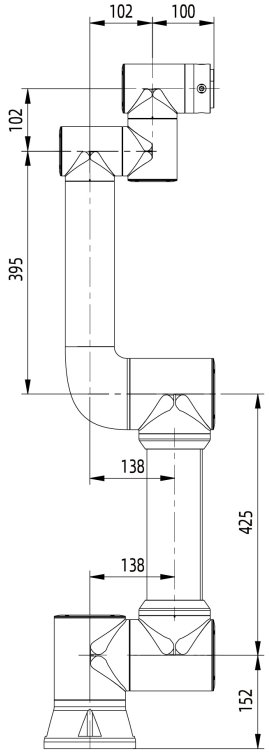

J1 : 175°
J1 : 175°
∅1244
J5 : 175°
J5 : 175°
J6 : ±360°

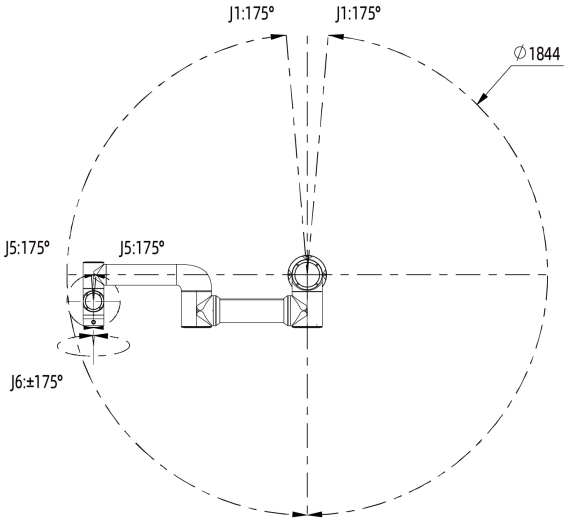

102
100
102
395
114
425
114
140

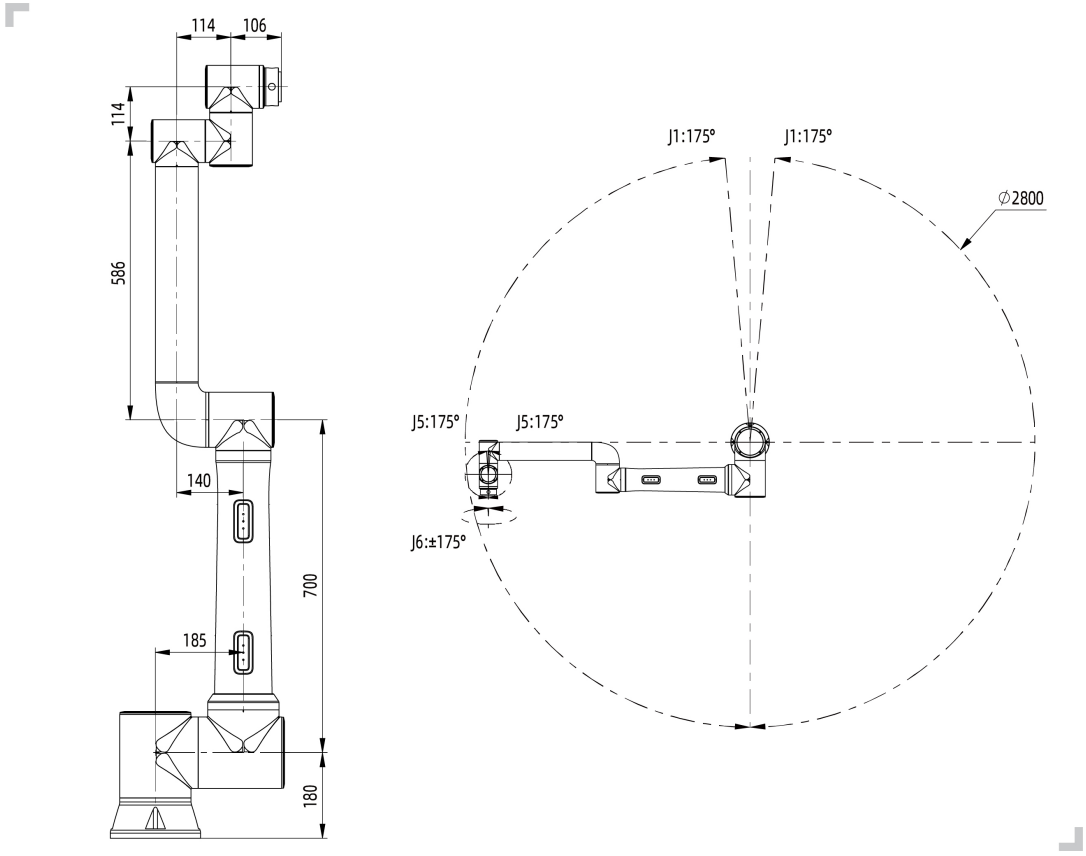

J1 : 175°
J1 : 175°
∅1844
J5 : 175°
J5 : 175°
J6 : ±360°

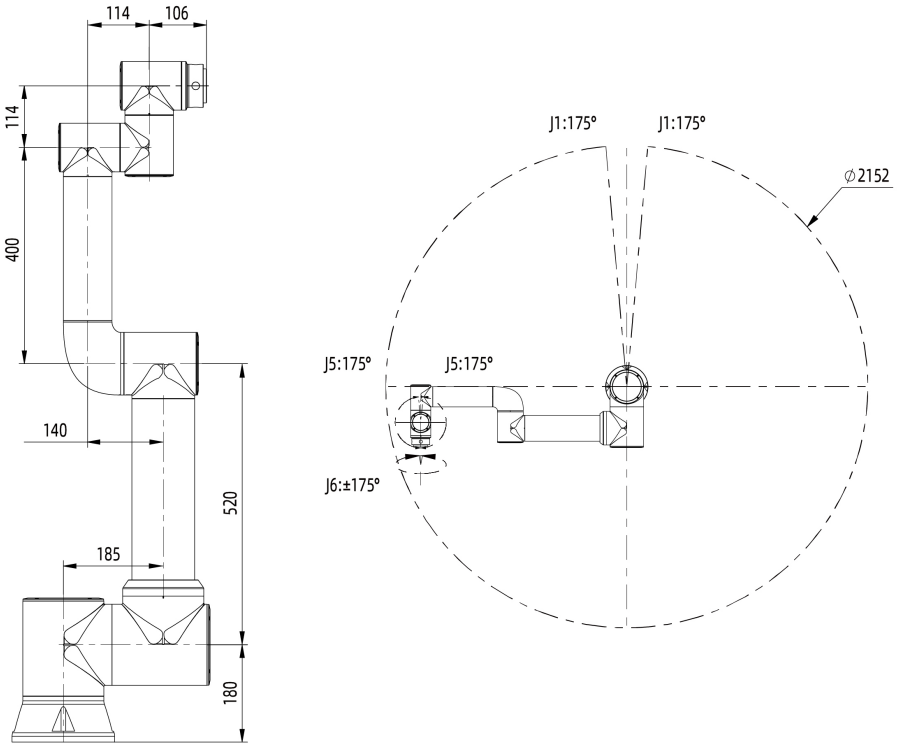

102
100
102
240
114
114
280
140

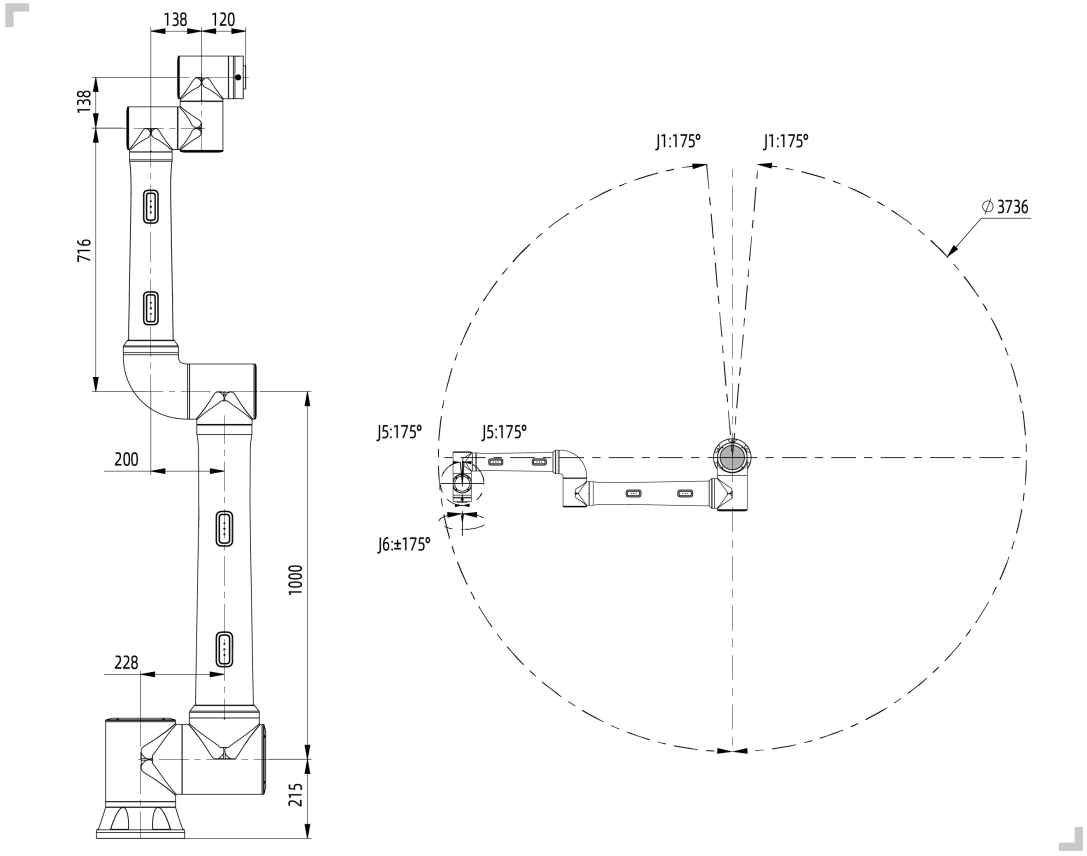

J1:175°
J1:175°
∅1244
J5:0°
J5:355°
J6:±175°

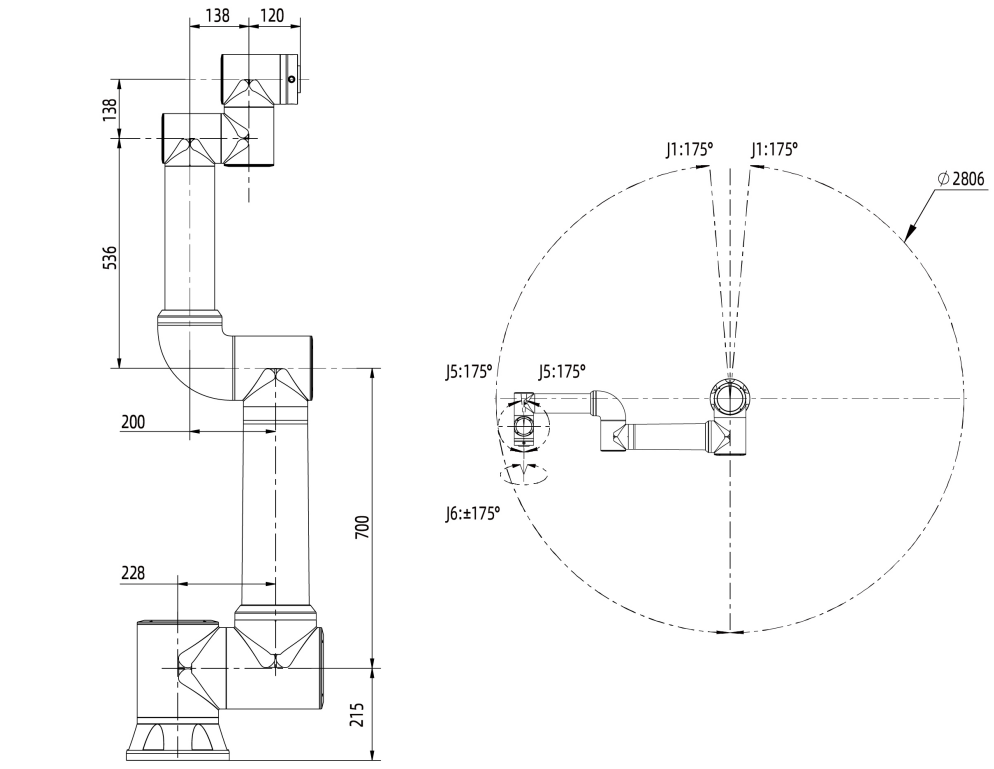

102
100
102
395
138
425
138
152

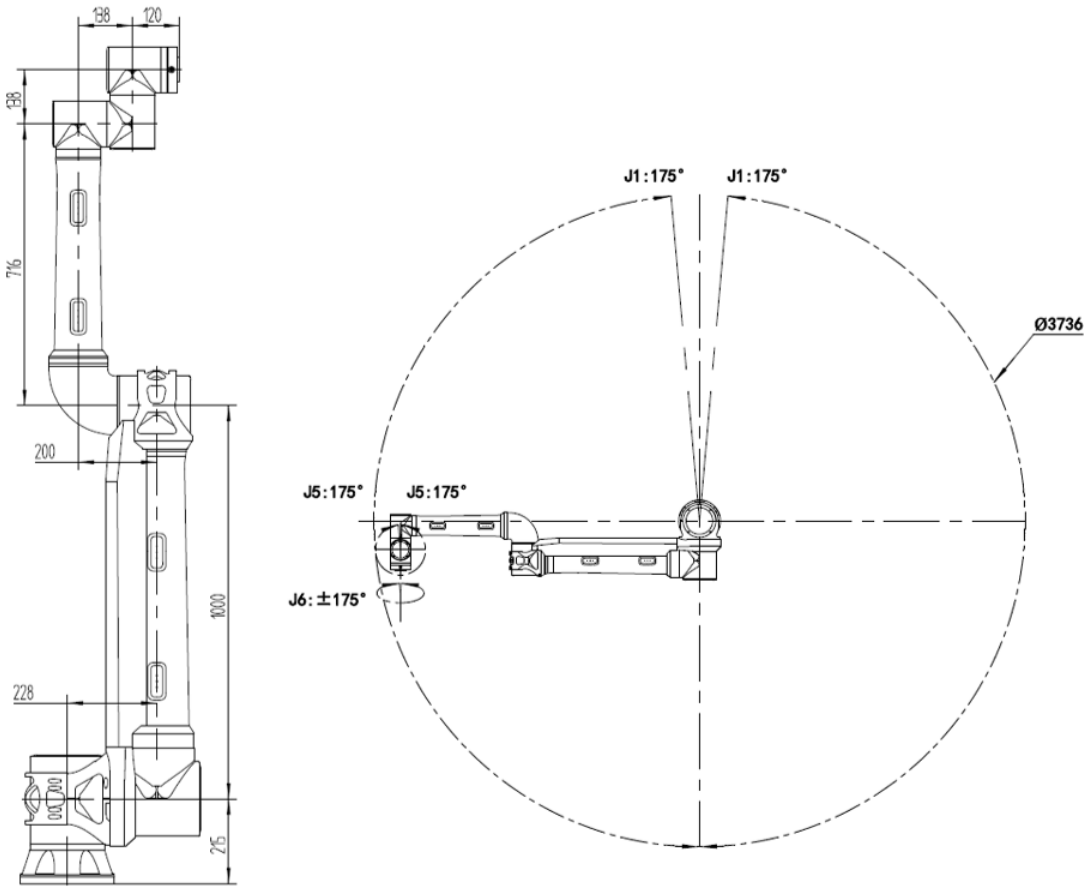

J1:175°
J1:175°
∅1844
J5:175°
J5:175°
J6:±175°


114
106
114
586
140
700
185
180
J1:175°
J1:175°
∅2800
J5:175°
J5:175°
J6:±175°


114
106
114
400
140
520
185
180
J1:175°
J1:175°
∅2152
J5:175°
J5:175°
J6:±175°


138
120
138
716
200
1000
228
215
J1:175°
J1:175°
∅3736
J5:175°
J5:175°
J6:±175°


138
120
138
536
200
700
228
215
J1:175°
J1:175°
∅2806
J5:175°
J5:175°
J6:±175°


138
120
138
716
200
1000
228
215
J1:175°
J1:175°
Ø3736
J5:175°
J5:175°
J6:±175°

1.3.2.3 Robot coordinate system

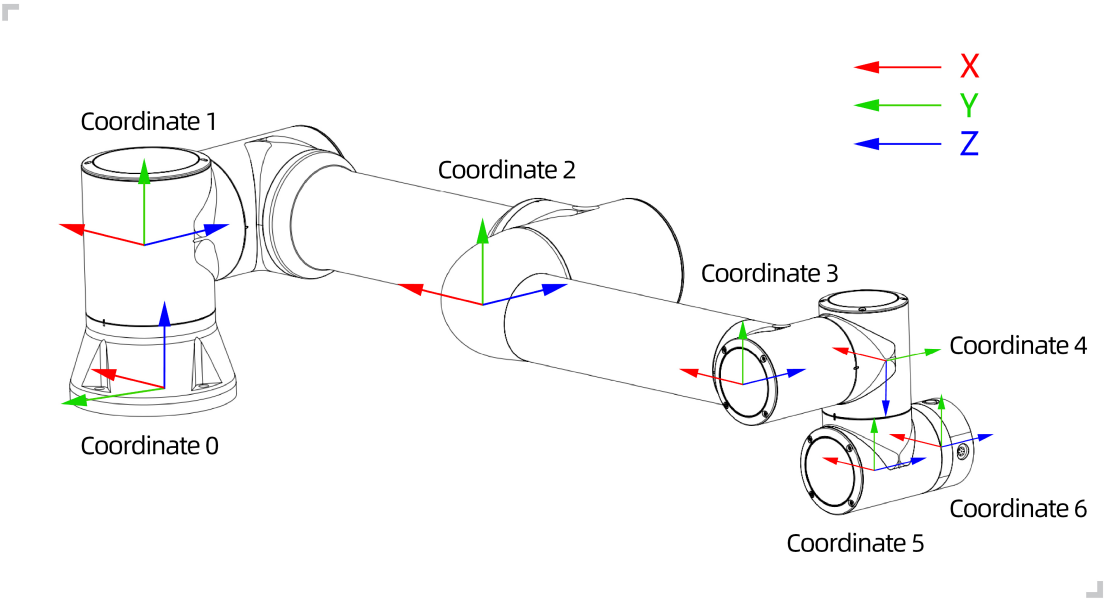


Figure 2.3-1 Robot DH parameter coordinate system

Figure 2.3-2 French -endframe coordinate system

1.3.2.4 Robot Denavit–Hartenberg parameters

Denavit–Hartenberg parameters are used to calculate kinematics and dynamics of FR series collaborative robots.

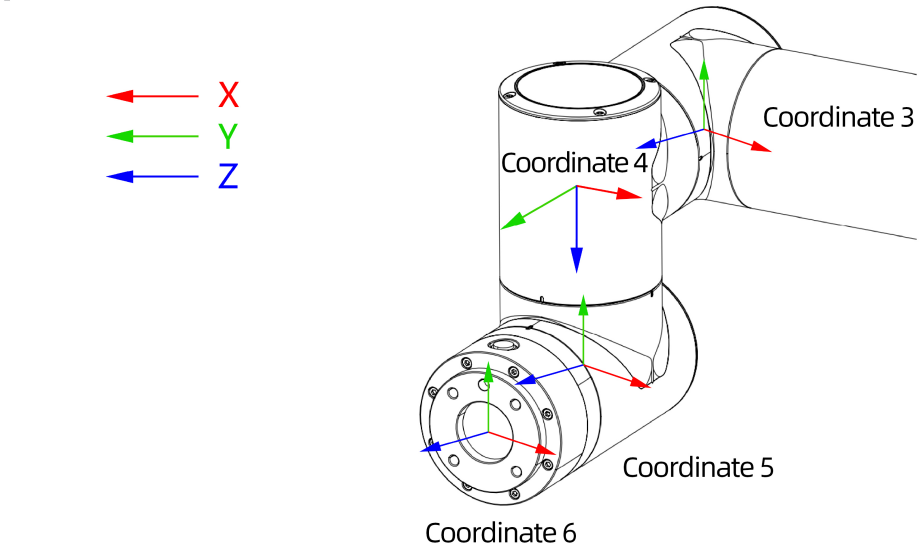
Figure 2.4-1 FR series collaborative robots Denavit–Hartenberg parameters

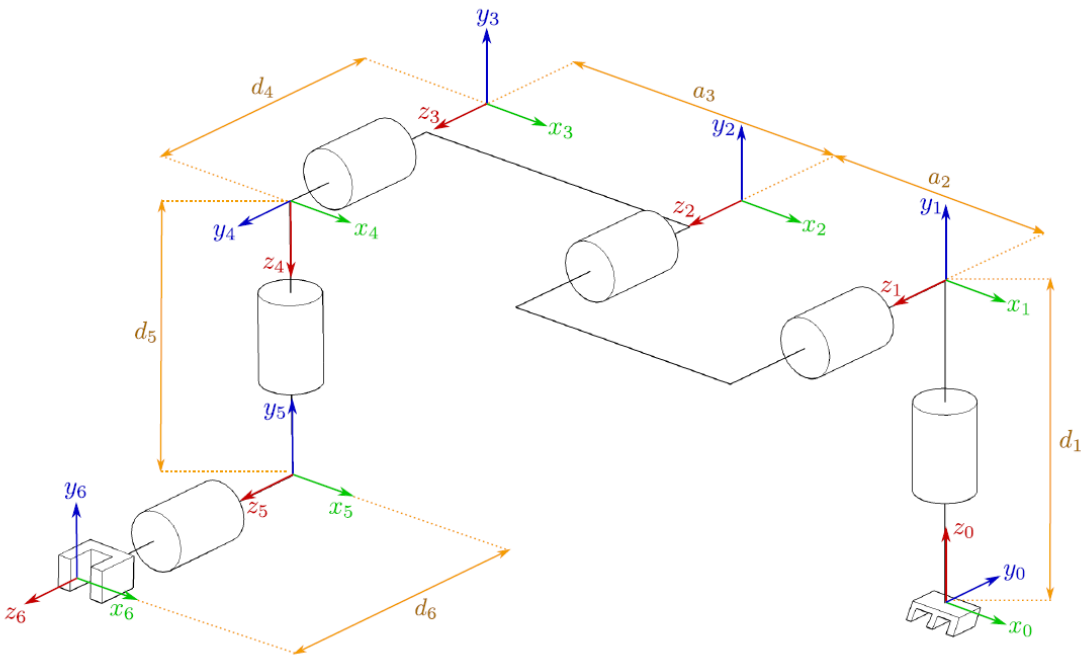
The Denavit–Hartenberg parameters of FR series collaborative robots are shown as below.

Table 2.4-1 FR3 model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta[rad	a[mm]	d[mm]	al-pha[rad]	Dynamics	Mass[kg]	Center of Mass[mm]
Joint1	0	0	140	/2	link1	2.24	[-0.05, -15.92, 2.26]
Joint2	0	-280	0	0	link2	4.94	[139.49, 0, 99.54]
Joint3	0	-240	0	0	link3	2.29	[58.99, 0.08, 12.99]
Joint4	0	0	102	/2	link4	1.56	[0.05, -2.33, 14.67]
Joint5	0	0	102	-/2	link5	1.56	[-0.05, 2.33, 14.67]
Joint6	0	0	100	0	link6	0.36	[-0.55, -1.11, -20.05]

Table 2.4-2 FR3-WMS model collaboration robot Denavit–Hartenberg parameter


X
Y
Z
Coordinate 3
Coordinate 4
Coordinate 5
Coordinate 6


y_3
d_4
z_3
x_3
a_3
y_2
a_2
y_1
z_2
x_2
y_4
x_4
z_4
z_1
x_1
d_5
y_5
d_1
y_6
z_5
x_5
z_0
y_0
z_6
x_6
d_6
x_0

Kinematics	theta [rad]	a[mm]	d[mm]	alpha [rad]	Dynamics	Mass [kg]	Center of Mass[m]
Joint1	0	140	0	/2	Link1	1.66	[-0.06-13.581.68]
Joint2	0	0	-280	0	Link2	3.68	[140.110101.71]
Joint3	0	0	-240	0	Link3	1.81	[63.490.110.94]
Joint4	0	102	0	/2	Link4	1.18	[0.07-2.1812.48]
Joint5	0	102	0	-/2	Link5	1.18	[-0.072.1812.48]
Joint6	0	100	0	0	Link6	0.28	[1.811.33-20.41]

Table 2.4-3 FR3-WML model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta [rad]	a[mm]	d[mm]	alpha [rad]	Dynamics	Mass [kg]	Center of Mass[m]
Joint1	0	140	0	/2	Link1	1.54	[-0.01-14.271.37]
Joint2	0	0	-425	0	Link2	3.49	[212.50101.43]
Joint3	0	0	-395	0	Link3	2	[114.170.089.92]
Joint4	0	102	0	/2	Link4	1.17	[0.07-2.1812.48]
Joint5	0	102	0	-/2	Link5	1.17	[-0.072.1812.48]
Joint6	0	100	0	0	Link6	0.28	[1.91.6-20.08]

Table 2.4-4 FR3-C model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta [rad]	a[mm]	d[mm]	alpha [rad]	Dynamics	Mass [kg]	Center of Mass[m]
Joint1	0	140	0	/2	Link1	1.69	[-0.16-13.991.53]
Joint2	0	0	-280	0	Link2	3.73	[1400101.34]
Joint3	0	0	-240	0	Link3	1.84	[63.240.0811.04]
Joint4	0	102	0	/2	Link4	1.2	[0.1-2.0312.55]
Joint5	0	102	0	-/2	Link5	1.2	[-0.12.0312.55]
Joint6	0	100	0	0	Link6	0.53	[1.481.54-17.9]

Table 2.4-5 FR5 model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta[rad	a[mm]	d[mm]	alpha[rad]	Dynamics	Mass[kg]	Center of Mass[mm]
Joint1	0	0	152	/2	link1	4.64	[-0.19, -18.28, 2.26]
Joint2	0	-425	0	0	link2	10.08	[212.47, 0, 121.2]
Joint3	0	-395	0	0	link3	2.71	[122.62, 0.17, 12.59]
Joint4	0	0	102	/2	link4	1.56	[0.05, -2.33, 14.68]
Joint5	0	0	102	-/2	link5	1.56	[-0.05, 2.33, 14.68]
Joint6	0	0	100	0	link6	0.36	[0.93, 0.81, -20.05]

Table 2.4-6 FR5-WML model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta[rad	a[mm]	d[mm]	alpha[rad]	Dynamics	Mass[kg]	Center of Mass[mm]
Joint1	0	180	0	/2	Link1	11.49	[-0.16, -28.51, 4.16]
Joint2	0	0	-970	0	Link2	21.3	[642.59, 0.04, 165.62]
Joint3	0	0	-8160	0	Link3	4.61	[321.39, 0.16, 52.76]
Joint4	0	159	0	/2	Link4	1.66	[0.21, -3.06, 13.07]
Joint5	0	114	0	-/2	Link5	1.66	[-0.21, 3.06, 13.07]
Joint6	0	160	0	0	Link6	0.36	[1.45, 1.09, -19.98]

Table 2.4-7 FR10 model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta[rad	a[mm]	d[mm]	alpha[rad]	Dynamics	Mass[kg]	Center of Mass[mm]
Joint1	0	0	180	/2	link1	11.97	[-0.10, -26.12, 4.04]
Joint2	0	-700	0	0	link2	19.59	[480.27, 0.01, 164.68]
Joint3	0	-586	0	0	link3	3.7	[211.22, 0.11, 54.21]
Joint4	0	0	159	/2	link4	1.69	[0.12, -3, 12.18]
Joint5	0	0	114	-/2	link5	1.69	[-0.12, 3, 12.18]
Joint6	0	0	106	0	link6	0.35	[1.24, 0.85, -20.34]

Table 2.4-8 FR16 model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta[rad	a[mm]	d[mm]	alpha[rad]	Dynamics	Mass[kg]	Center of Mass[mm]
Joint1	0	0	180	/2	link1	11.97	[-0.10, -26.12, 4.04]
Joint2	0	-520	0	0	link2	18.18	[364.4, 0.01, 163.09]
Joint3	0	-400	0	0	link3	3.22	[135.03, 0.12, 55.58]
Joint4	0	0	159	/2	link4	1.69	[0.12, -3, 12.18]
Joint5	0	0	114	-/2	link5	1.69	[-0.12, 3, 12.18]
Joint6	0	0	106	0	link6	0.35	[1.24, 0.85, -20.34]

Table 2.4-9 FR20 model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta [rad]	a[mm]	d[mm]	alpha [rad]	Dynamics	Mass [kg]	Center of Mass[m]
Joint1	0	0	215	/2	link1	20.79	[-0.19, -36.57, 5.68]
Joint2	0	-1000	0	0	link2	42.84	[605.25, 0.06, 202.94]
Joint3	0	-716	0	0	link3	9.88	[262.84, 0.22, 43.08]
Joint4	0	0	166	/2	link4	4.64	[0.23, -2.28, 18.42]
Joint5	0	0	138	-/2	link5	4.64	[-0.23, 2.28, 18.42]
Joint6	0	0	120	0	link6	0.6	[-2.11, -1.96, -20.38]

Table 2.4-10 FR30 model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta [rad]	a[mm]	d[mm]	alpha [rad]	Dynamics	Mass [kg]	Center of Mass[m]
Joint1	0	0	215	/2	link1	20.64	[-0.22, -37.39, 5.59]
Joint2	0	-700	0	0	link2	36.37	[440.73, 0.05, 198.7]
Joint3	0	-536	0	0	link3	8.41	[185.64, 0.25, 45.82]
Joint4	0	0	166	/2	link4	4.64	[0.23, -2.29, 18.60]
Joint5	0	0	138	-/2	link5	4.64	[-0.23, 2.29, 18.60]
Joint6	0	0	120	0	link6	0.6	[-2.11, -1.96, -20.38]

Table 2.4-11 FR30L model collaboration robot Denavit–Hartenberg parameter

Kinematics	theta [rad]	a[mm]	d[mm]	alpha [rad]	Dynamics	Mass [kg]	Center of Mass[m]
Joint1	0	0	215	/2	Link1	27.85	[-0.18, -26.29, 14.57]
Joint2	0	-1000	0	0	Link2	50.43	[580.62, 0.04, 223.34]
Joint3	0	-716	0	0	Link3	9.88	[262.84, 0.22, 43.08]
Joint4	0	0	166	/2	Link4	4.64	[0.23, -2.28, 18.42]
Joint5	0	0	138	-/2	Link5	4.64	[-0.23, 2.28, 18.42]
Joint6	0	0	120	0	Link6	0.6	[-2.11, -1.96, -20.38]

1.3.2.5 DH parameter download

```
FR Robots DH Transformation
```

1.3.3 Installation

1.3.3.1 Instructions for security

1.3.3.1.1 Brief introduction

This manual will use the following warnings. The role of these warnings is to ensure the safety of personal and equipment. When you read this manual, you must comply with and execute all the assembly instructions and guidelines in other chapters in this manual. This is very important. Pay special attention to text related to warning signs.

Important:

- If the robot (robot body, control box, teaching pendant, or button box) is damaged, changed or modified due to anthropogenic causes, FAIR INNOVATION refuses to bear all responsibilities;
- FAIR INNOVATION is not responsible for any damage caused by the robot or any other device due to errors written by customers or any other device.

1.3.3.1.2 Personnel safety

When running the robot system, we must first ensure the safety of the operators. The general precautions are listed below. Please take corresponding measures to ensure the safety of the operator.

1. All operators who use the robot system should be trained through training courses sponsored by the FAIR INNOVATION (Suzhou) ROBOTIC SYSTEM COMPANY. Users need to ensure that they fully grasp the safe and standardized operating processes and have the qualifications for robotic operations. For training details, please check with our company, the mailbox is jiling@frtech.fr.
2. All operators who use the robot system, please do not wear loose clothes or jewelry. Make sure the long hair beam is behind the head when operating the robot.
3. During the operation of the device, even if the robot seems to have stopped, it may be because the robot is waiting to start the signal and is in a state of motion. Even in such a state, the robot should be regarded as in action.
4. The lines should be drawn on the floor to mark the scope of the robot, so that the operator understands the action range of the robot includes holding tools (robotics, tools, etc.).
5. Ensure safety measures (such as guardrails, ropes, or protective screens) near the robotic operation area to protect operators and surrounding people. Locks should be set up as needed, so that those who are responsible for operations cannot reach the power supply of the robot.
6. When using the operating panel and teaching pendant, because you may have an operational error in wearing gloves, you must work after taking off the gloves.
7. Under the emergency and abnormalities of people being pinched or wrapped inside by a robot, pushing or pulling the robot arm by force (at least 700 N) to force joint movements. With a power-free driver, the arm of the manual mobile robot is limited to emergency and may damage the joint.

1.3.3.1.3 Danger recognition

Risk assessment should consider all potential contact between operators and robots during normal use and foreseeable misunderstandings. The operator's neck, face, and head should not be exposed to avoid touch. Using robots without using peripheral safety protection devices requires the first risk assessment to determine whether the relevant hazards will constitute an unacceptable risk, such as

- There may be danger of using a sharp end actuator or tool connector;
- There may be danger of treatment of toxicity or other harmful substances;
- The operator's finger has the danger of being clamped by a robot base or joint;
- The danger of collision by robots;
- The danger of robots or tools connected to the end not being fixed in place;
- The risk of the impact between the effective load and the strong surface of the robot.

Integrates must measure such dangers and related risk levels through risk assessment, and determine and implement corresponding measures to reduce the risk to acceptable levels. Please note that there may be other major dangers in specific robots.

By combining the inherent security design measures used by FR robots with the safety specifications or risk assessment implemented by integrated and end users, the risks related to FR collaborative operations will be reduced to a reasonable and feasible level. Through this document, any remaining risk of the robot can be conveyed to the integrated and end users before installation. If the risk assessment of the integrator determines that the existence of a specific application that may have unacceptable risks that may constitute unacceptable risks, integrators must appropriate risk reduction measures must be taken to eliminate or minimize these dangers until the risk is reduced to acceptable levels. It is not safe to use before taking appropriate risk reduction measures (if necessary).

If the robot is installed in non-synergistic installation (for example, when using a hazardous tool), risk assessment may infer the integrated provider to connect additional safety equipment (such as safety startup equipment) to ensure personnel and equipment safety when programming.

1.3.3.1.4 Nameplate information

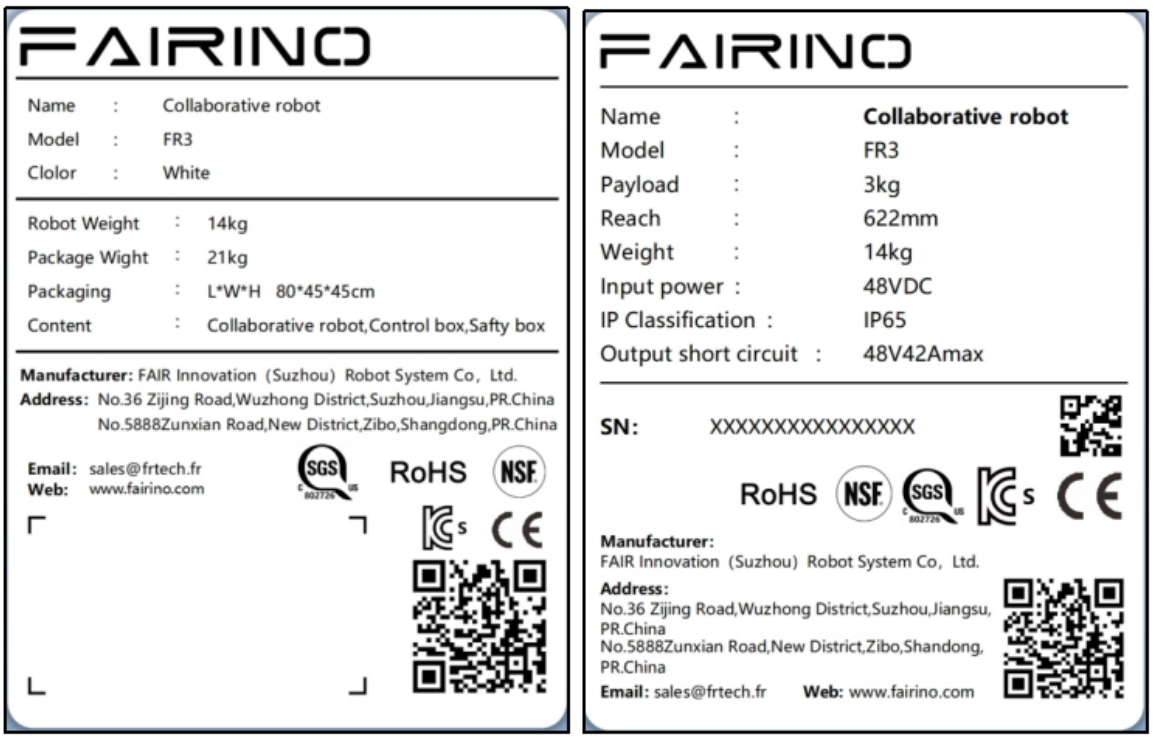


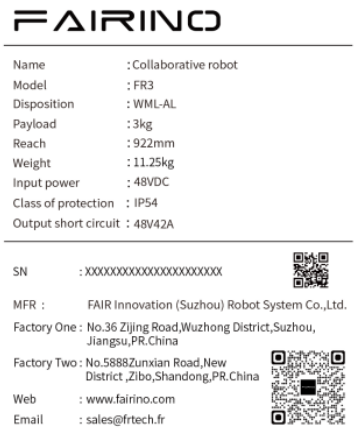
Figure 3.1-1 FR3 model collaborative robot



Figure 3.1-2 FR3-WMS model collaborative robot

Figure 3.1-3 FR3-WML model collaborative robot

Figure 3.1-4 FR3-C model collaborative robot


FAIRINO

Name : Collaborative robot
Model : FR3
Disposition : WML-AL
Payload : 3kg
Reach : 922mm
Weight : 11.25kg
Input power : 48VDC
Class of protection : IP54
Output short circuit : 48V42A

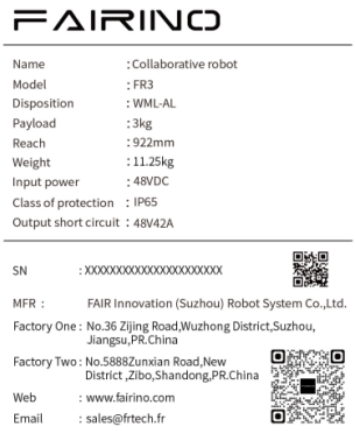
SN : XXXXXXXXXXXXXXXXXXXXX

MFR : FAIR Innovation (Suzhou) Robot System Co.,Ltd.
Factory One : No.36 Zijing Road,Wuzhong District,Suzhou, Jiangsu,PR.China
Factory Two : No.5888Zunxian Road,New District ,Zibo,Shandong,PR.China
Web : www.fairino.com
Email : sales@frtech.fr


FAIRINO

Name : Collaborative robot
Model : FR3
Disposition : WML-AL
Color : White
Robot Weight : 11.25kg
Package Weight : 19kg
Packaging : L*W*H 80*45*46cm
Content : Collaborative robot、Control box、Safty box

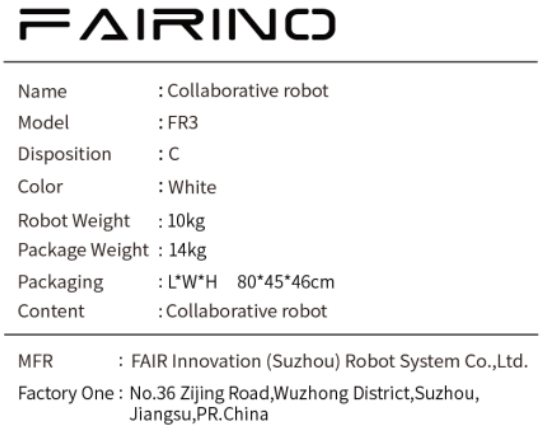
MFR : FAIR Innovation (Suzhou) Robot System Co.,Ltd.
Factory One : No.36 Zijing Road,Wuzhong District,Suzhou, Jiangsu,PR.China
Factory Two : No.5888Zunxian Road,New District ,Zibo, Shandong,PR.China
Email : www.fairino.com
Web : sales@frtech.fr


FAIRINO

Name : Collaborative robot
Model : FR3
Disposition : WML-AL
Payload : 3kg
Reach : 922mm
Weight : 11.25kg
Input power : 48VDC
Class of protection : IP65
Output short circuit : 48V42A

SN : XXXXXXXXXXXXXXXXXXXXX

MFR : FAIR Innovation (Suzhou) Robot System Co.,Ltd.
Factory One : No.36 Zijing Road,Wuzhong District,Suzhou, Jiangsu,PR.China
Factory Two : No.5888Zunxian Road,New District ,Zibo,Shandong,PR.China
Web : www.fairino.com
Email : sales@frtech.fr


FAIRINO

Name : Collaborative robot
Model : FR3
Disposition : C
Color : White
Robot Weight : 10kg
Package Weight : 14kg
Packaging : L*W*H 80*45*46cm
Content : Collaborative robot

MFR : FAIR Innovation (Suzhou) Robot System Co.,Ltd.
Factory One : No.36 Zijing Road,Wuzhong District,Suzhou, Jiangsu,PR.China
Factory Two : No.5888Zunxian Road,New District ,Zibo, Shandong,PR.China
Email : www.fairino.com
Web : sales@frtech.fr


FAIRINO

Name : Collaborative robot
Model : FR3
Disposition : C
Payload : 3kg(Moment 5kg)
Reach : 622mm
Weight : 10kg
Class of protection : IP54
Input power : 48VDC
Output short circuit : 22Amax

SN : XXXXXXXXXXXXXXXXXXXXX

MFR : FAIR Innovation (Suzhou) Robot System Co.,Ltd.
Factory One : No.36 Zijing Road,Wuzhong District,Suzhou, Jiangsu,PR.China
Factory Two : No.5888Zunxian Road,New District ,Zibo,Shandong,PR.China
Web : www.fairino.com
Email : sales@frtech.fr

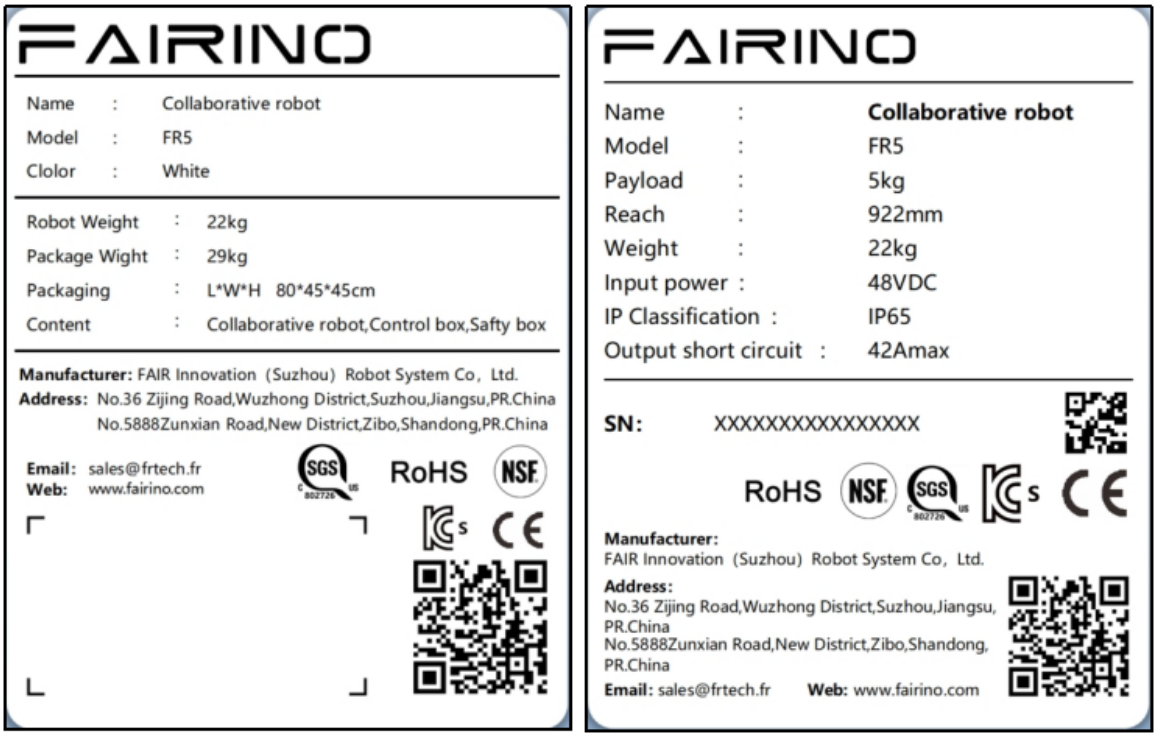


Figure 3.1-5 FR5 model collaborative robot

Figure 3.1-6 FR5-WML model collaborative robot

Figure 3.1-7 FR10 model collaborative robot

Figure 3.1-8 FR16 model collaborative robot

Figure 3.1-9 FR20 model collaborative robot

Figure 3.1-10 FR30 model collaborative robot

Figure 3.1-11 FR30L model collaborative robot

1.3.3.1.5 Effectiveness and responsibility

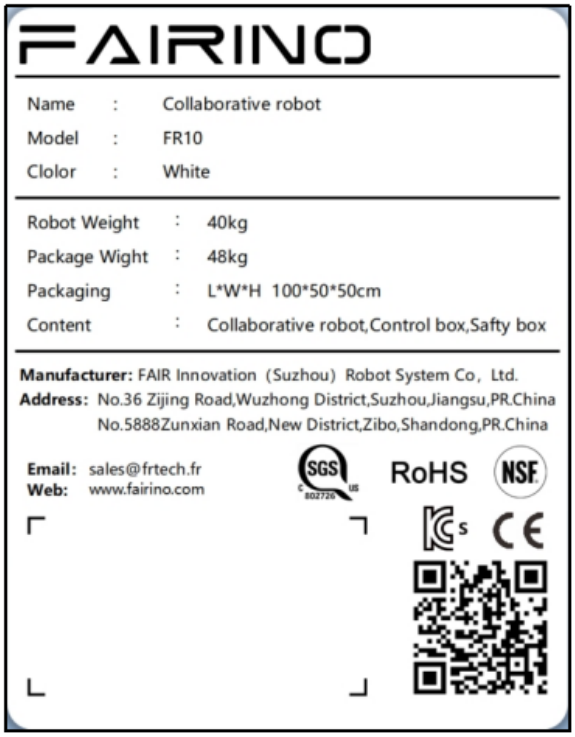
The information in this manual does not include a complete robot application in design, installation and operation, nor does it contain all peripheral equipment that may affect the security of this complete system. The design and installation of this complete system must meet the safety requirements established in the standards and specifications of the country's installation.

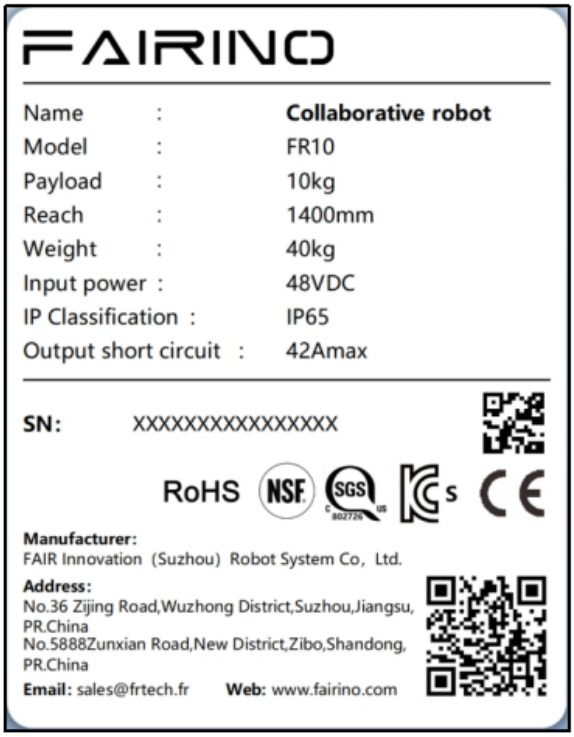
The integrated integrator of FAIR INNOVATION is responsible for ensuring that it follows the laws and regulations of relevant countries, and there is no major danger in the complete robotic application. This includes but not limited to the following:

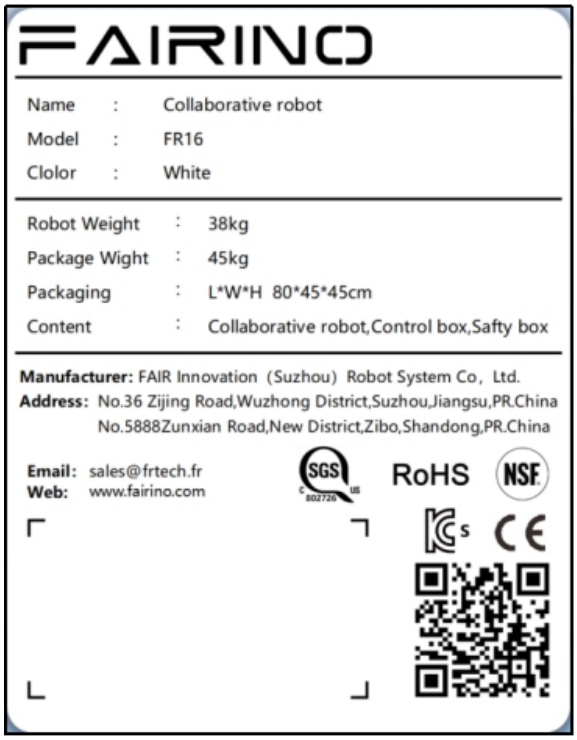
- Do a risk assessment of the complete robot system
- Connect other machinery and additional safety equipment defined by risk assessment and definition
- Establish appropriate security settings in software
- Make sure that users will not modify any security measures
- Confirm that the design and installation of the entire robot system is accurate

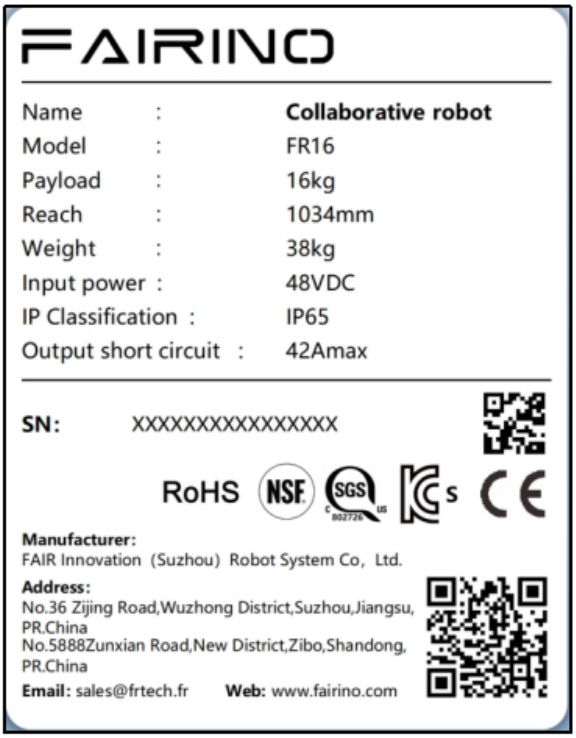

FAIRINO
Name : Collaborative robot
Model : FR5
Disposition : WML-R360
Payload : 5kg
Reach : 1900mm
Weight : 44kg
Class of protecition : IP54
Input power : 48VDC
Output short circuit : 42Amax
SN XXXXXXXXXXXXXXXXXXX
MFR : FAIR Innovation(Suzhou)Robot System Co.,Ltd
Factory One: No.36 Zijing Road,Wuzhong District,Suzhou, Jiangsu,PR.China
Factory Two : No.5888 Zunxian Road,New District, Zibo,Shandong,PR.China
Web : www.fairino.com
Email : sales@fairino.com

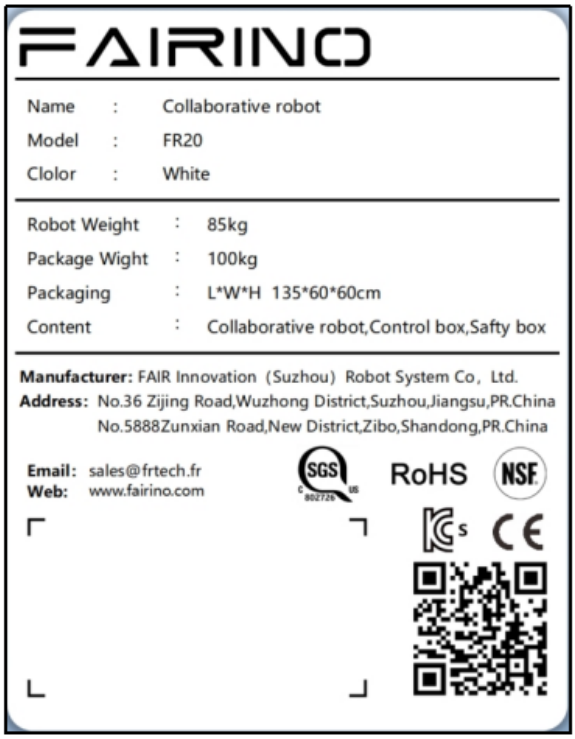

FAIRINO
Name : Collaborative robot
Model : FR5
Disposition : WML-R360
Clolor : White
Class of protecition : IP54
Robot Weight : 44kg
Package Weight : 48kg
Packaging : L*W*H 122*50*50cm
Content : Collaborative robot Control box Safty box
Manufacturer: FAIR Innovation (Suzhou) Robot System Co. Ltd.
Address: No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu,PR.China
No.5888Zunxian Road,New District,Zibo,Shandong,PR.China
Email: sales@frtech.fr
Web: www.fairino.com
Control box SN:
Collaborative robot body SN:

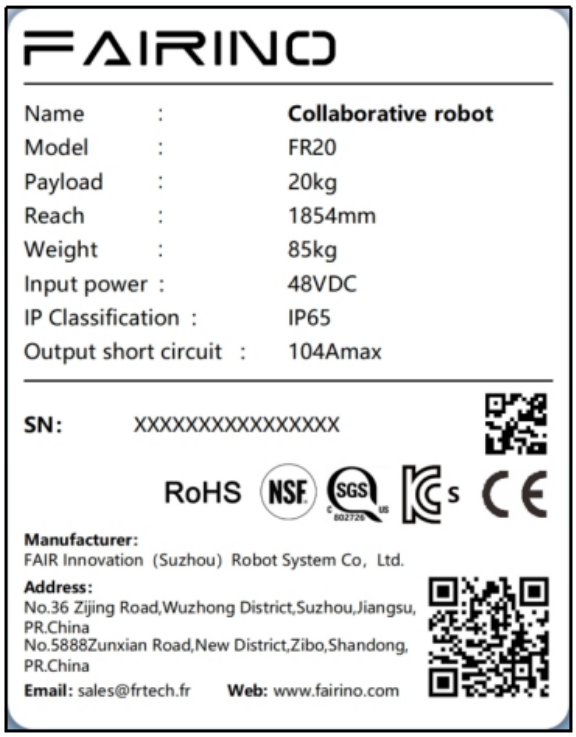

FAIRINO
Name : Collaborative robot
Model : FR10
Clolor : White
Robot Weight : 40kg
Package Wight : 48kg
Packaging : L*W*H 100*50*50cm
Content : Collaborative robot,Control box,Safty box
Manufacturer: FAIR Innovation (Suzhou) Robot System Co, Ltd.
Address: No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu,PR.China
No.5888Zunxian Road,New District,Zibo,Shandong,PR.China
Email: sales@frtech.fr
Web: www.fairino.com
SGS
RoHS
NSF
CE

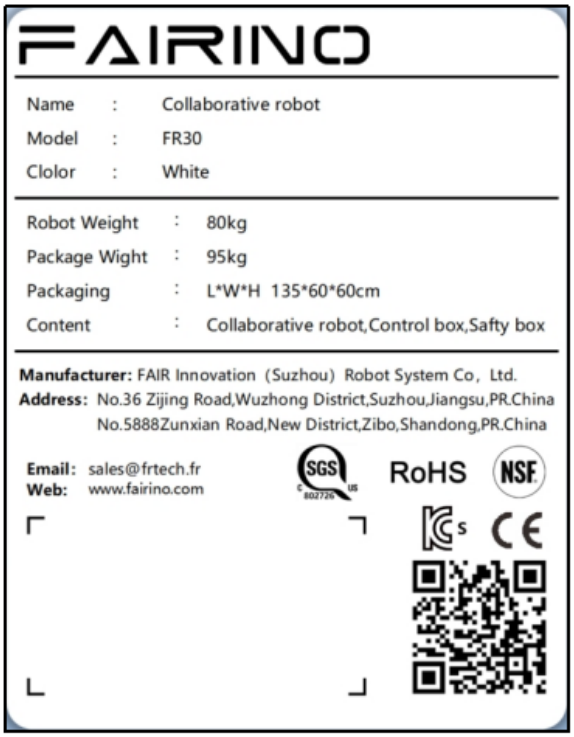

FAIRINO
Name : Collaborative robot
Model : FR10
Payload : 10kg
Reach : 1400mm
Weight : 40kg
Input power : 48VDC
IP Classification : IP65
Output short circuit : 42Amax
SN: XXXXXXXXXXXXXXXX
RoHS
NSF
SGS
CE
Manufacturer:
FAIR Innovation (Suzhou) Robot System Co, Ltd.
Address:
No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu, PR.China
No.5888Zunxian Road,New District,Zibo,Shandong, PR.China
Email: sales@frtech.fr
Web: www.fairino.com

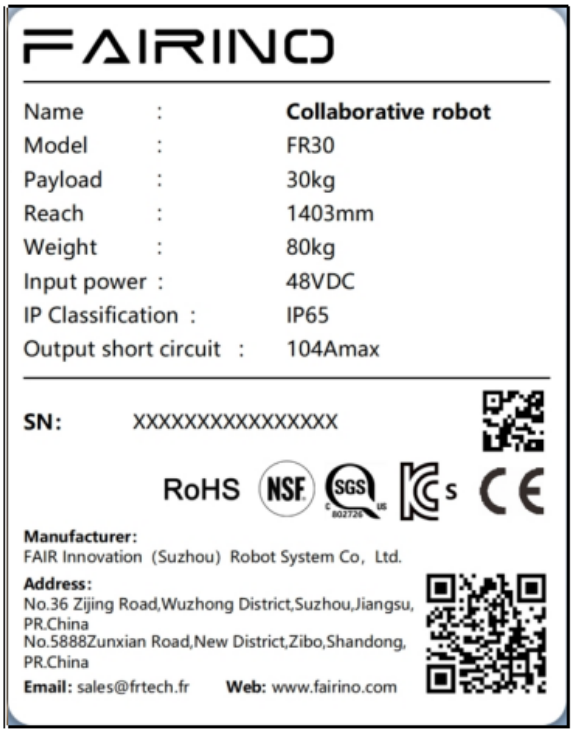

FAIRINO
Name : Collaborative robot
Model : FR16
Clolor : White
Robot Weight : 38kg
Package Wight : 45kg
Packaging : L*W*H 80*45*45cm
Content : Collaborative robot,Control box,Safty box
Manufacturer: FAIR Innovation (Suzhou) Robot System Co, Ltd.
Address: No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu,PR.China
No.5888Zunxian Road,New District,Zibo,Shandong,PR.China
Email: sales@frtech.fr
Web: www.fairino.com
RoHS


FAIRINO
Name : Collaborative robot
Model : FR16
Payload : 16kg
Reach : 1034mm
Weight : 38kg
Input power : 48VDC
IP Classification : IP65
Output short circuit : 42Amax
SN: XXXXXXXXXXXXXXXX
RoHS
Manufacturer:
FAIR Innovation (Suzhou) Robot System Co, Ltd.
Address:
No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu, PR.China
No.5888Zunxian Road,New District,Zibo,Shandong, PR.China
Email: sales@frtech.fr
Web: www.fairino.com


FAIRINO
Name : Collaborative robot
Model : FR20
Clolor : White
Robot Weight : 85kg
Package Wight : 100kg
Packaging : L*W*H 135*60*60cm
Content : Collaborative robot,Control box,Safty box
Manufacturer: FAIR Innovation (Suzhou) Robot System Co, Ltd.
Address: No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu,PR.China
No.5888Zunxian Road,New District,Zibo,Shandong,PR.China
Email: sales@frtech.fr
Web: www.fairino.com
RoHS


FAIRINO
Name : Collaborative robot
Model : FR20
Payload : 20kg
Reach : 1854mm
Weight : 85kg
Input power : 48VDC
IP Classification : IP65
Output short circuit : 104Amax
SN: XXXXXXXXXXXXXXXX
RoHS
Manufacturer:
FAIR Innovation (Suzhou) Robot System Co, Ltd.
Address:
No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu, PR.China
No.5888Zunxian Road,New District,Zibo,Shandong, PR.China
Email: sales@frtech.fr
Web: www.fairino.com


FAIRINO
Name : Collaborative robot
Model : FR30
Clolor : White
Robot Weight : 80kg
Package Wight : 95kg
Packaging : L*W*H 135*60*60cm
Content : Collaborative robot,Control box,Safty box
Manufacturer: FAIR Innovation (Suzhou) Robot System Co, Ltd.
Address: No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu,PR.China
No.5888Zunxian Road,New District,Zibo,Shandong,PR.China
Email: sales@frtech.fr
Web: www.fairino.com
RoHS


FAIRINO
Name : Collaborative robot
Model : FR30
Payload : 30kg
Reach : 1403mm
Weight : 80kg
Input power : 48VDC
IP Classification : IP65
Output short circuit : 104Amax
SN: XXXXXXXXXXXXXXXX
RoHS
Manufacturer:
FAIR Innovation (Suzhou) Robot System Co, Ltd.
Address:
No.36 Zijing Road,Wuzhong District,Suzhou,Jiangsu, PR.China
No.5888Zunxian Road,New District,Zibo,Shandong, PR.China
Email: sales@frtech.fr Web: www.fairino.com


FAIRINO
Name : Collaborative robot
Model : FR30
Disposition : L
Color : White
Robot Weight : 100kg
Package Weight : 104kg
Packaging : L*W*H 1350*600*610cm
Content : Collaborative robot
MFR : FAIR Innovation (Suzhou) Robot System Co.,Ltd.
Factory One : No.36 Zijing Road,Wuzhong District,Suzhou, Jiangsu,PR.China
Factory Two : No.5888Zunxian Road,New District ,Zibo, Shandong,PR.China
Web : www.fairino.com
Email : sales@fairino.com


FAIRINO
Name : Collaborative robot
Model : FR30
Disposition : L
Payload : 35kg
Reach : 1854mm
Weight : 100kg
Class of protection : IP54
Input power : 48VDC
Output short circuit : 104Amax
SN : XXXXXXXXXXXXXXXXXXXXX
MFR : FAIR Innovation (Suzhou) Robot System Co.,Ltd.
Factory One : No.36 Zijing Road,Wuzhong District,Suzhou, Jiangsu,PR.China
Factory Two : No.5888Zunxian Road,New District ,Zibo,Shandong,PR.China
Web : www.fairino.com
Email : sales@frtech.fr

- Clear instructions for use
- Kark the relevant signs and contact information of integrators on the robot
- Collect all the documents in the technical file, including this manual

1.3.3.1.6 Limited responsibility

Any security information contained in this manual shall not be regarded as a general robot safety guarantee. Even if you comply with all security descriptions, it may still cause personnel damage or equipment damage.

1.3.3.1.7 The warning signs in this manual

The following flag defines the explanation of the danger level provisions contained in this manual. The product also uses the same warning signs.

Important:



Danger: This refers to the power consumption that is about to cause danger. If it is not avoided, it can lead to death or severe damage.

Important:



Electric shock danger: This refers to the upcoming electric shock situation that is about to cause danger.

Important:



Dangerous burns: This refers to the hot surface that may cause danger. If you do not avoid contact, it can cause personnel to hurt.

1.3.3.1.8 Pre-use evaluation

After using a robot or any modification for the first time, the robot's default speed is less than 250mm/s. Do not log in to the administrator to modify the speed to enter the high-speed mode. After that, the following test must be performed. It is confirmed that all security input and output are correct and the connection is correct. Test whether all connected security input and output (including multiple machines or robots shared equipments) are normal. So you must:

- Test whether the emergency stop button and input can stop the robot and start the brake.
- Test whether the protection input can stop the robot movement. If the protection reset is configured, check if you need activation before recovery.
- The test operation mode can switch the operation mode, see the icon in the upper right corner of the user interface.
- Test whether the 3rd gear actuation device must be pressed to activate in manual mode, and the robot is under deceleration control (the robot software version V3.0 does not support this function).
- Test whether the system emergency stop output can bring the entire system to a safe state.

1.3.3.1.9 Emergency stop

The emergency stop button is type 0 stop. Press the emergency stop button to stop all the movements of the robot immediately.

The following table shows the stop distance and stop time of the type 0 stop. These measurement results correspond to the following configuration of the robot:

- Extension: 100%(the robotic arm is fully expanded)
- Speed: 100%(Robot general speed is set to 100%, moved at a joint speed of 180 °/s)
- Effective load: Maximum effective load

Joint 1, joint 6 testing robot levels, the rotating shaft is perpendicular to the ground. Joint 2, joint 3, joint 4, joint 5 testing robots follow the vertical trajectory, the rotating shaft is parallel to the ground, and stops when the robot moves down.

Table 3.1-1 Category 0 stop distance(rad)

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
FR3	0.47	0.60	0.56	0.29	0.10	0.06
FR3-WMS	0.47	0.60	0.56	0.29	0.10	0.06
FR3-WML	0.51	0.63	0.60	0.33	0.16	0.10
FR3-C	0.47	0.60	0.56	0.29	0.10	0.06
FR5	0.51	0.63	0.60	0.33	0.16	0.10
FR10	0.64	0.70	0.69	0.42	0.25	0.13
FR16	0.60	0.67	0.65	0.39	0.22	0.12
FR20	0.69	0.75	0.80	0.48	0.31	0.22
FR30L	0.69	0.75	0.80	0.48	0.31	0.22

Table 3.1-2 Category 0 stop time (ms)

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
FR3	400	470	450	280	120	90
FR3-WMS	400	470	450	280	120	90
FR3-WML	400	470	450	280	120	90
FR3-C	400	470	450	280	120	90
FR5	420	500	480	310	150	120
FR10	460	540	510	330	170	140
FR16	440	530	490	320	160	130
FR20	540	600	700	400	260	170
FR30L	540	600	700	400	260	170

After the emergency stop, turn off the power, rotate the emergency stop button, and turn on the power to restart the robot.

At the same time, the stop time and stop distance of the robot safety stop and soft limit stop are shown in the table below. These measurement results correspond to the following configuration of the robot:

- Extension: 100%(the robotic arm is fully expanded)
- Speed: 100%(Robot general speed is set to 100%, moved at a joint speed of 180 °/s)

- Effective load: Maximum effective load

Joint 1, joint 6 testing robot levels, the rotating shaft is perpendicular to the ground. Joint 2, joint 3, joint 4, joint 5 testing robots follow the vertical trajectory, the rotating shaft is parallel to the ground, and stops when the robot moves down.

Table 3.1-3 Safety stop distance(rad)

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
FR3	0.49	0.63	0.58	0.32	0.12	0.09
FR3-WMS	0.49	0.63	0.58	0.32	0.12	0.09
FR3-WML	0.54	0.65	0.63	0.35	0.19	0.12
FR3-C	0.49	0.63	0.58	0.32	0.12	0.09
FR5	0.54	0.65	0.63	0.35	0.19	0.12
FR10	0.66	0.73	0.71	0.45	0.27	0.14
FR16	0.63	0.69	0.68	0.41	0.25	0.14
FR20	0.71	0.78	0.82	0.51	0.33	0.25
FR30L	0.71	0.78	0.82	0.51	0.33	0.25

Table 3.1-4 Safety stop time(ms)

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
FR3	410	490	410	300	130	110
FR3-WMS	410	490	410	300	130	110
FR3-WML	410	490	410	300	130	110
FR3-C	410	490	410	300	130	110
FR5	450	520	510	330	180	140
FR10	480	570	530	360	190	170
FR16	470	550	520	340	190	150
FR20	560	630	720	430	280	200
FR30L	560	630	720	430	280	200

Table 3.1-5 Soft limit stop distance(rad)

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
FR3	0.52	0.65	0.61	0.34	0.15	0.11
FR3-WMS	0.52	0.65	0.61	0.34	0.15	0.11
FR3-WML	0.56	0.68	0.65	0.38	0.21	0.15
FR3-C	0.52	0.65	0.61	0.34	0.15	0.11
FR5	0.56	0.68	0.65	0.38	0.21	0.15
FR10	0.69	0.75	0.74	0.47	0.30	0.18
FR16	0.65	0.72	0.70	0.44	0.27	0.17
FR20	0.74	0.80	0.85	0.53	0.36	0.27
FR30L	0.74	0.80	0.85	0.53	0.36	0.27

Table 3.1-6 Soft limit stop time(ms)

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
FR3	430	500	430	310	150	120
FR3-WMS	430	500	430	310	150	120
FR3-WML	430	500	430	310	150	120
FR3-C	430	500	430	310	150	120
FR5	460	540	520	350	190	160
FR10	500	580	550	370	210	180
FR16	480	570	530	360	200	170
FR20	580	640	740	440	300	210
FR30L	580	640	740	440	300	210

Important: According to IEC 60204-1 and ISO 13850, emergency stop device is not a safe protection device. They are supplementary protection measures and do not need to prevent damage.

1.3.3.1.10 Power-free movement

If you must move the robot joint but cannot power the robot or other emergencies, please contact the robot dealer. If necessary, you can use violent means to force mobile robots to rescue the trapped persons.

1.3.3.2 Equipment transportation

1.3.3.2.1 Transportation

Robot and control boxes have been calibrated as complete equipment. Do not separate them, that would require recalibration.

You can only transport the robot in the original packaging. If you want to move the robot in the future, save the packaging material in a dry place.

When the robot moves from the packaging to the installation space, the two arms of the robot are held at the same time. Hold the robot until all the installation bolts of the robot seat are tight.

1.3.3.2.2 Carry

According to different models, the total quality (including packaging) is 15kg-80 kg depending on the model. When manpower transports or transfers the collaborative robot, multiple people need to help lift it, don't recommend single-person handling, it must be stable during transportation. Avoid equipment tilt or slipping.

Warning:

- If you use professional equipment for handling, be sure to use a crane or forklift to transport or carry the collaborative robot by using a crane or forklift, otherwise it may cause personnel damage or other accidents;
- If you use manual handling, please pay attention to the personal safety on the way to handle;
- The collaborative robot contains precision components, which should avoid severe vibration or shaking during transportation or transportation, otherwise it may reduce the performance of the equipment.

1.3.3.2.3 Storage

The collaborative robot should be stored in -25 ~ 60 ° C, and there is no frost-free environment.

1.3.3.3 Maintenance, inspection, and scrapping

1.3.3.3.1 Maintenance disposal

Please check the emergency stop and protection stop for 1 month. Determine whether the security function is effective.

Please refer to the wiring chapter for emergency stop and protective stop wiring.

1.3.3.3.2 Inspection Plan

1.3.3.3.2.1 Preface

1.3.3.3.2.2 Safety Notice

The following warnings are used in this manual. These warnings are intended to ensure personal and equipment safety. It is important that you observe and follow all assembly instructions and guidelines in other sections of this manual when reading this manual.

Special attention should be paid to the text related to warning signs. Please read the user manual carefully before use. This manual is only used as a customer maintenance instruction manual. Maintenance operators need to have professional competence. Non-professional personnel operation. FAIRINO refuses to assume all responsibilities.

Note: If the robot (robot body, control box, teaching box) is damaged, changed or modified due tohumanreasons,FAIRINO rejects all responsibility; FAIRINO is not responsible for any damage to the robot or any other equipment caused by errors inthe programs written by thecustomer.

1.3.3.3.2.3 Effectiveness and accountability

The information in this manual does not cover the design, installation and operation of a complete robot application, nor does it cover all peripheral equipment that may affect the safety of this complete system. The complete system is designed and installed to meet safety requirements established in the standards and codes of the country where the robot is installed.

It is the responsibility of FAIRINO integrators to ensure compliance with relevant national laws and regulations and to ensure that there are no significant risks in the complete robot application. This includes but is not limited to the following:

- Do a risk assessment of the complete robotic system
- Connect other machinery and additional safety equipment defined by the risk assessment
- Establish appropriate security settings in software
- Ensure that users do not modify any security measures
- Confirm that the design and installation of the entire robot system is correct
- Clear instructions for use

- Mark the relevant logo and contact information of the integrator on the robot
- Collect all documents in technical files, including this manual

1.3.3.3.2.4 Limited Liability

Any safety information contained in this manual should not be considered a general robot safety guarantee, and even if all safety instructions are followed, it may still cause personal injury or equipment damage.

1.3.3.3.2.5 Warning signs in this manual

The following symbols define the hazard classification specifications contained in this manual. The same warning signs are used on the products.

Note: 

Name: **Danger**

Function: This refers to an electrical situation that is about to cause danger and, if not avoided, could result in death or serious injury.

Note: 

Name: **Electric shockhazard**

Function: This refers to an imminent hazardous electric shock situation which, if not avoided, could result in electrocution or serious injury.

Note: 

Name: **Burnhazard**

Function: This refers to hot surfaces that may cause hazards and, if touched, may cause injury to persons.

1.3.3.3.3 Digital Input/Output Description of Control Box

1.3.3.3.3.1 Precautions When Switching Digital Related Functions of Control Box

Important:

(1) When switching digital input/output functions, the safety operation procedures of the robot must be followed to ensure the safety of operators and equipment.

(2) Avoid switching digital input/output functions during robot operation to prevent affecting the normal operation of the robot.

(3) Before switching digital input/output functions, the power supply of the robot must be cut off to prevent electric shock and unexpected machine movement, which may cause personal injury and equipment damage.

(4) Before switching functions, the requirements of the robot control system for digital input/output must be clarified, including signal type, voltage level, load capacity, etc.

(5) Ensure that the connection between digital input/output ports and external devices is correct, including whether the wiring is secure and whether the ports match.

(6) Avoid repeated signal allocation to ensure that each signal is uniquely allocated.

(7) After allocation is completed, the robot control system must be restarted to make the settings take effect.

(8) After completing the configuration, enter the I/O status interface to check whether the status of digital input/output signals is correct.

(9) Verify whether the digital input/output functions are working properly through actual operation or writing test programs.

(10) If digital input/output signals are related to program logic, check whether the processing of these signals in the program is correct.

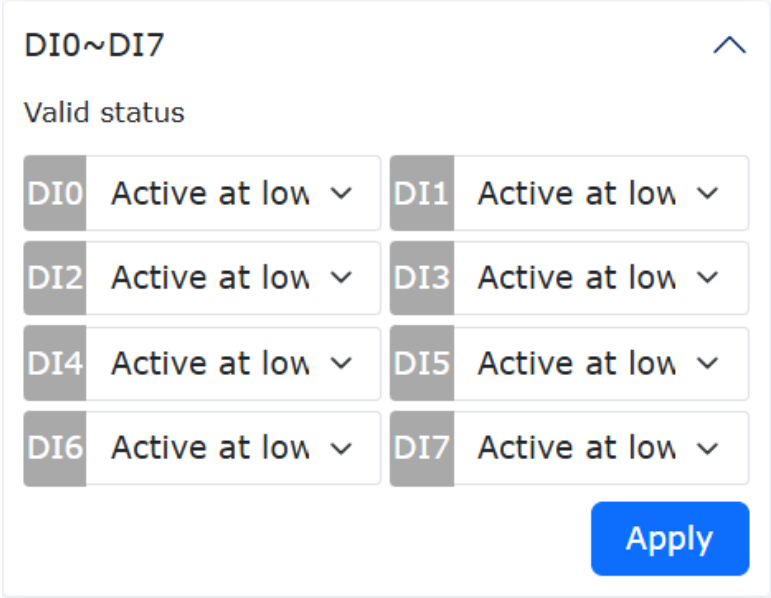
1.3.3.3.3.2 Digital Input Description of Control Box

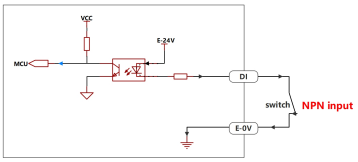
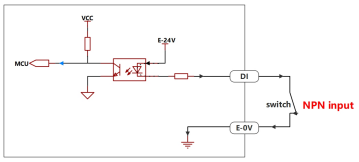
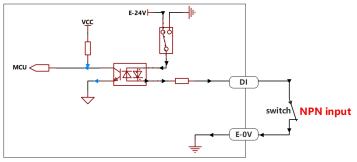
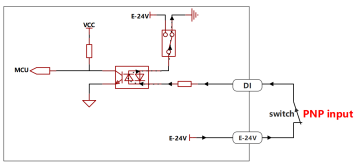
1.3.3.3.3.3 Summary of Digital Input of Control Box

The following lists the input types supported by the digital input of Faro robot integrated mini control box, as well as the corresponding wiring diagrams and configuration comparison tables.

Figure 3.3-1 DI0-DI7 General Input Valid Status

Table 3.3-1 Control Box Digital Input Configuration Comparison Table



Control Box Type	Input Type	Connection Diagram	High Level Valid (Switch Closed)	High Level Valid (Switch Open)	Low Level Valid (Switch Closed)	Low Level Valid (Switch Open)
DC Control Box	NPN Output		Invalid	Valid	Valid	Invalid
AC Narrow Voltage Control Box	NPN Output		Invalid	Valid	Valid	Invalid
AC Wide Voltage Control Box	NPN Output		Invalid	Valid	Valid	Invalid
AC Wide Voltage Control Box	PNP Output		Invalid	Valid	Valid	Invalid

1.3.3.3.3.4 Supported Types of Digital Input of Control Box

The digital input of DC control box and AC narrow voltage control box only supports NPN type input. The digital input of AC wide voltage control box supports optional NPN and PNP types, with NPN type as the default factory setting.

Control Box Type	Input Type
DC Control Box	NPN Input
AC Narrow Voltage Control Box	NPN Input
AC Wide Voltage Control Box	NPN Input/PNP Input

1.3.3.3.3.5 Wiring Diagram of Digital Input of Control Box

The digital input of DC control box and AC narrow voltage control box only supports NPN type input. The wiring diagram is as follows.

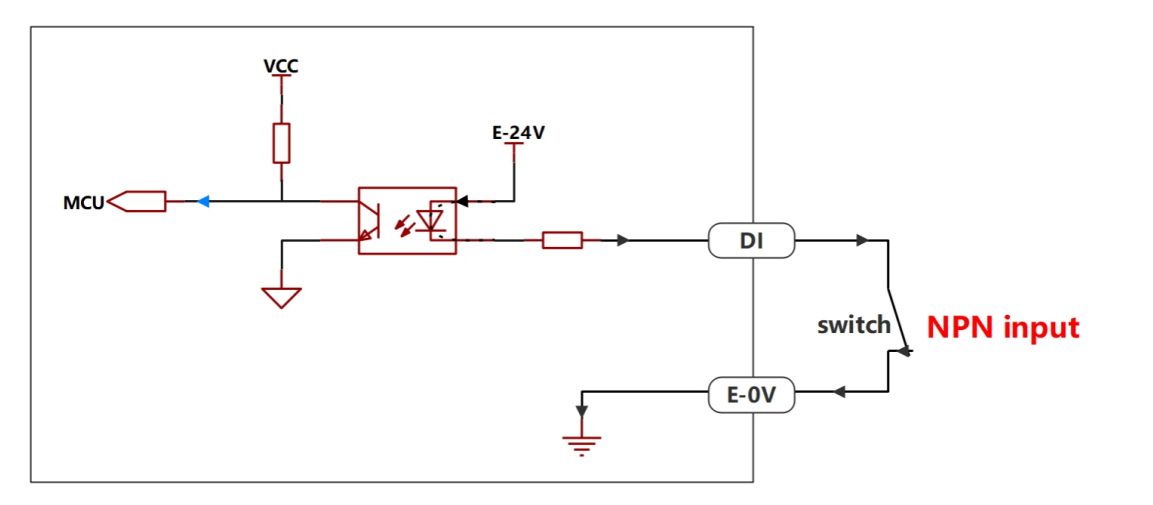
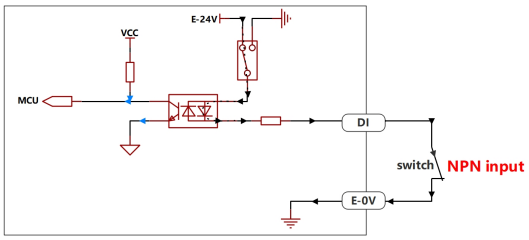
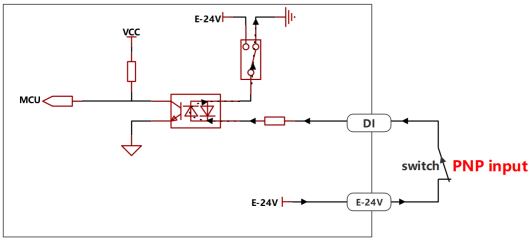
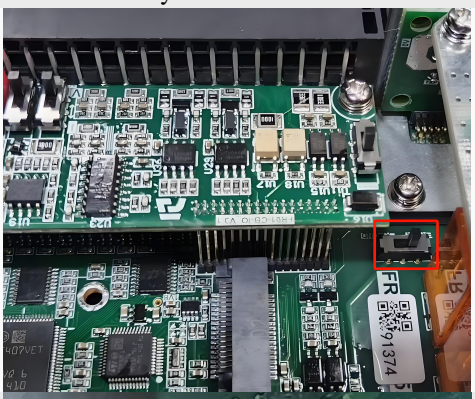
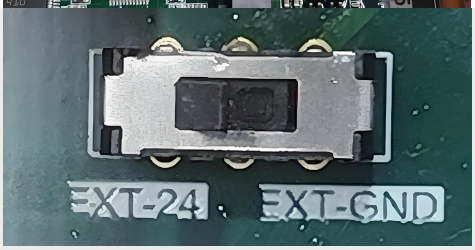


Figure 3.3-2 Wiring Diagram of Digital Input for DC Control Box and AC Narrow Voltage Control Box

The digital input of AC wide voltage control box supports optional NPN and PNP types, with NPN type as the default factory setting. The wiring diagrams are as follows:

Input Type	Connection Diagram
NPN Input	
PNP Input	

The input type of the wide voltage control box digital input is determined by the DIP switch inside the control box. If the user needs to change the input type, the DIP switch needs to be set to the corresponding position.

	DIP Switch Position	DIP Switch Physical Position
NPN Input	EX-24V	
PNP Input	EX-0V	

1.3.3.3.3.6 Software Settings Related to Digital Input of Control Box

The only software setting item for digital input is “DI0-DI7 General Input Valid Status”, which represents the digital voltage level value corresponding to the detected valid input. This setting allows users to use digital input more flexibly.

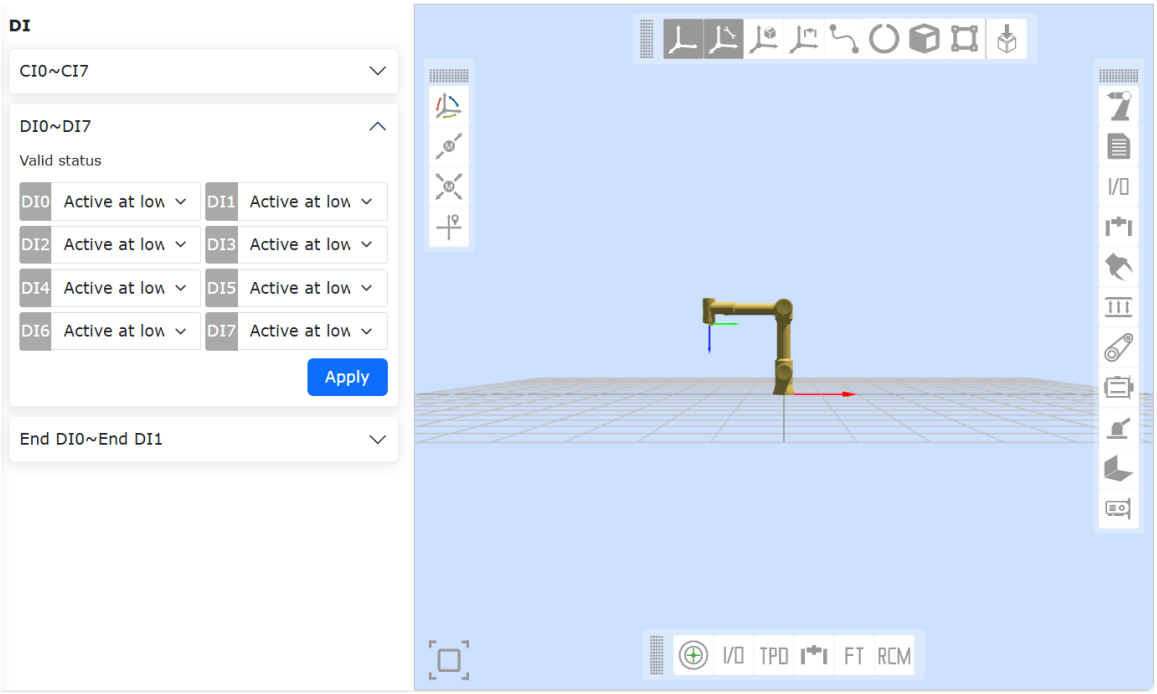


Figure 3.3-3 DI0-DI7 General Input Valid Status

The comparison table of valid status detected by the software under different settings of “DI0-DI7 General Input Valid Status” when the external switch of digital input is in different states is as follows:

Table 3.3-2 Valid Status Comparison Table

Control Box Type	Input Type	High Level Valid (Switch Closed)	High Level Valid (Switch Open)	Low Level Valid (Switch Closed)	Low Level Valid (Switch Open)
DC Control Box	NPN Input	Invalid	Valid	Valid	Invalid
AC Narrow Voltage Control Box	NPN Input	Invalid	Valid	Valid	Invalid
AC Wide Voltage Control Box	NPN Input	Invalid	Valid	Valid	Invalid
AC Wide Voltage Control Box	PNP Input	Invalid	Valid	Valid	Invalid

1.3.3.3.3.7 Digital Output Description of Control Box

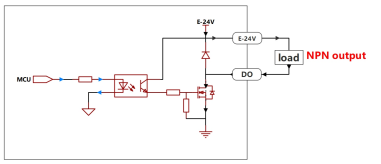
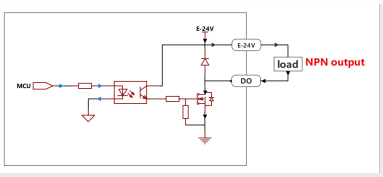
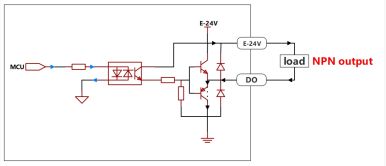
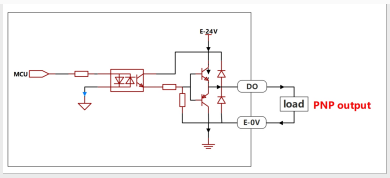
1.3.3.3.3.8 Summary of Digital Output of Control Box

The following lists the output types supported by the digital output of Faro robot integrated mini control box, as well as the corresponding wiring diagrams and configuration comparison tables.



Figure 3.3-4 Control Box DO Output During Power-On

Table 3.3-3 Control Box Digital Output Configuration Comparison Table

Control Box Type	Input Type	Connection Diagram	High Level (Switch Set to ON)	High Level (Switch Set to OFF)	Low Level (Switch Set to ON)	Low Level (Switch Set to OFF)
DC Control Box	NPN Output		Valid	Valid	Invalid	Invalid
AC Narrow Voltage Control Box	NPN Output		Valid	Valid	Invalid	Invalid
AC Wide Voltage Control Box	NPN Output		Valid	Valid	Invalid	Invalid
AC Wide Voltage Control Box	PNP Output		Valid	Valid	Invalid	Invalid

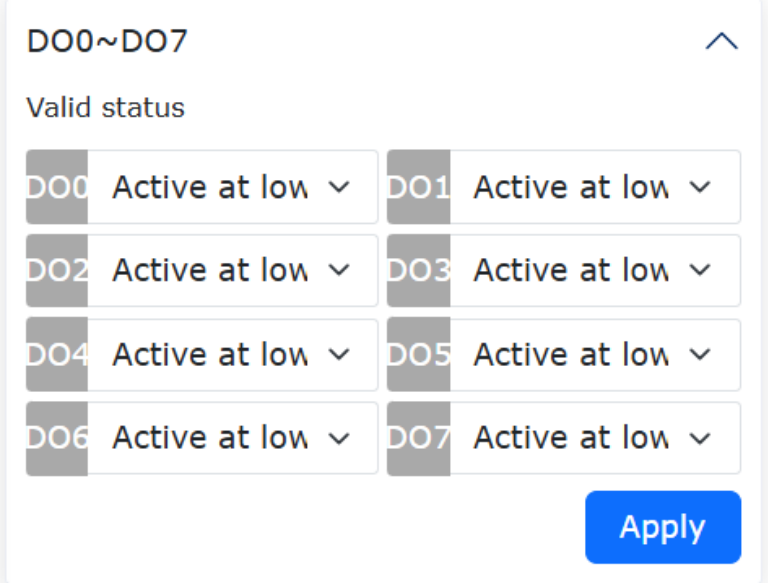
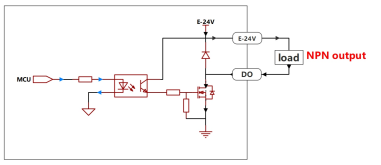
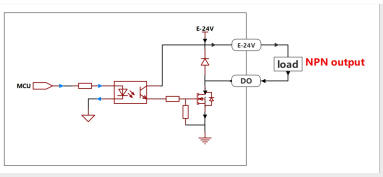
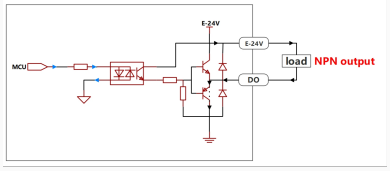
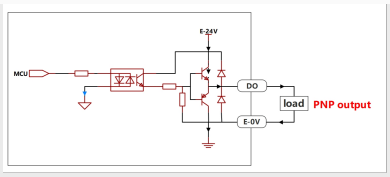


Figure 3.3-5 General Output Valid Status

Table 3.3-4 Control Box Digital Output Configuration Comparison Table

Control Box Type	Input Type	Connection Diagram	High Level Valid (Switch Set to ON)	High Level Valid (Switch Set to OFF)	Low Level Valid (Switch Set to ON)	Low Level Valid (Switch Set to OFF)
DC Control Box	NPN Output		Valid	Invalid	Invalid	Valid
AC Narrow Voltage Control Box	NPN Output		Valid	Invalid	Invalid	Valid
AC Wide Voltage Control Box	NPN Output		Valid	Invalid	Invalid	Valid
AC Wide Voltage Control Box	PNP Output		Valid	Invalid	Invalid	Valid

1.3.3.3.3.9 Supported Types of Digital Output of Control Box

The digital output of DC control box and AC narrow voltage control box only supports NPN type output. The digital output of AC wide voltage control box supports optional NPN and PNP types, with push-pull structure. It only needs to be wired according to the corresponding wiring diagram without special settings.

Control Box Type	Input Type
DC Control Box	NPN Output
AC Narrow Voltage Control Box	NPN Output
AC Wide Voltage Control Box	NPN Output/PNP Output

1.3.3.3.3.10 Wiring Diagram of Digital Output of Control Box

The digital output of DC control box and AC narrow voltage control box only supports NPN type output. The wiring diagram is as follows.

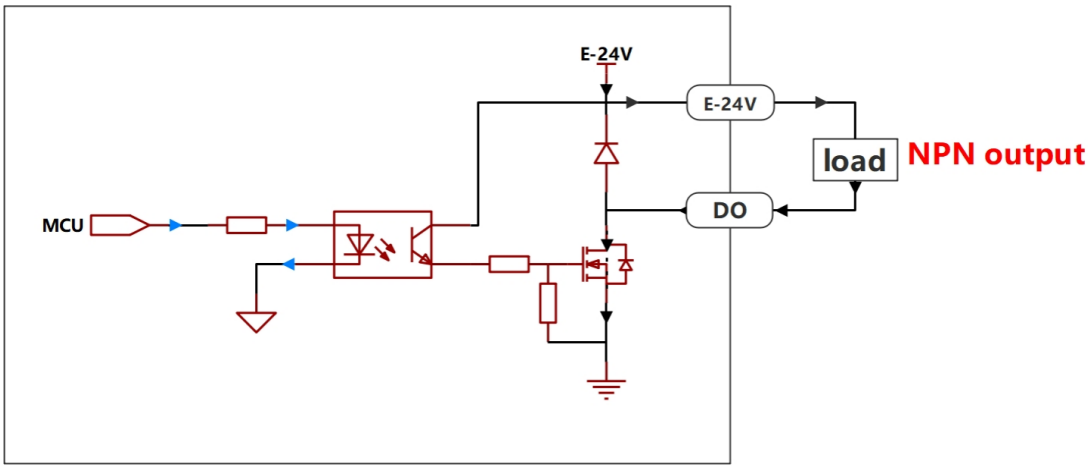
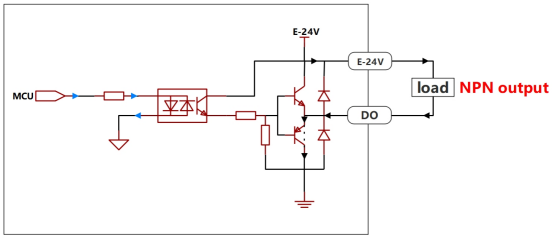
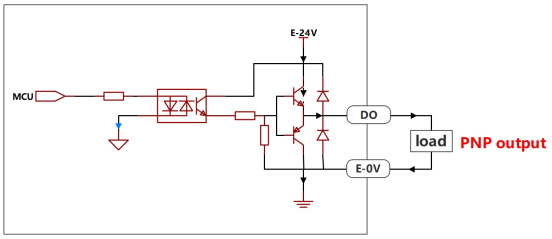


Figure 3.3-6 Wiring Diagram of Digital Output for DC Control Box and AC Narrow Voltage Control Box

The digital output of AC wide voltage control box supports NPN and PNP types. The wiring diagrams are as follows:

Input Type	Connection Diagram
NPN Input	
PNP Input	

1.3.3.3.3.11 Software Settings Related to Digital Output of Control Box

There are two software setting items for digital output: "Control Box DO Output During Power-On" and "DO0-D07 General Output Valid Status". "Control Box DO Output During Power-On" represents the output level during the power-on period of the control box when the control system has not completed initialization. It can correspond to different output valid states, which can flexibly meet the requirements for output status during power-on. "General Output Valid Status" represents the digital output voltage level value that needs to be controlled when the output is valid. This setting allows users to use digital output more flexibly.

(1) The comparison table of valid status under different settings of "Control Box DO Output During Power-On" is as follows:

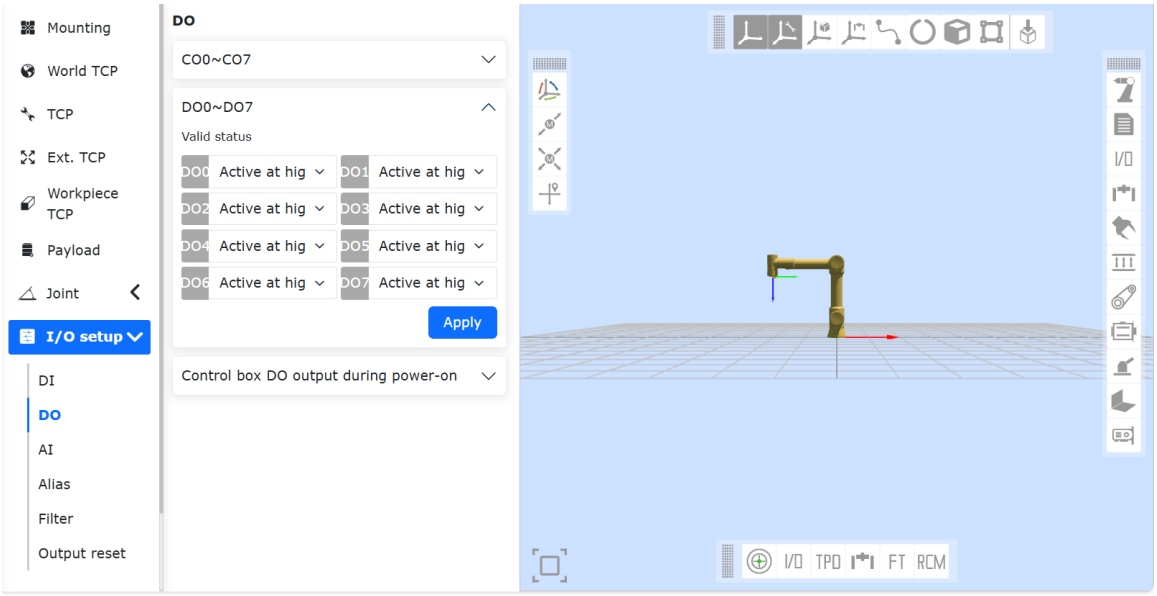


Figure 3.3-7 Control Box DO Output During Power-On

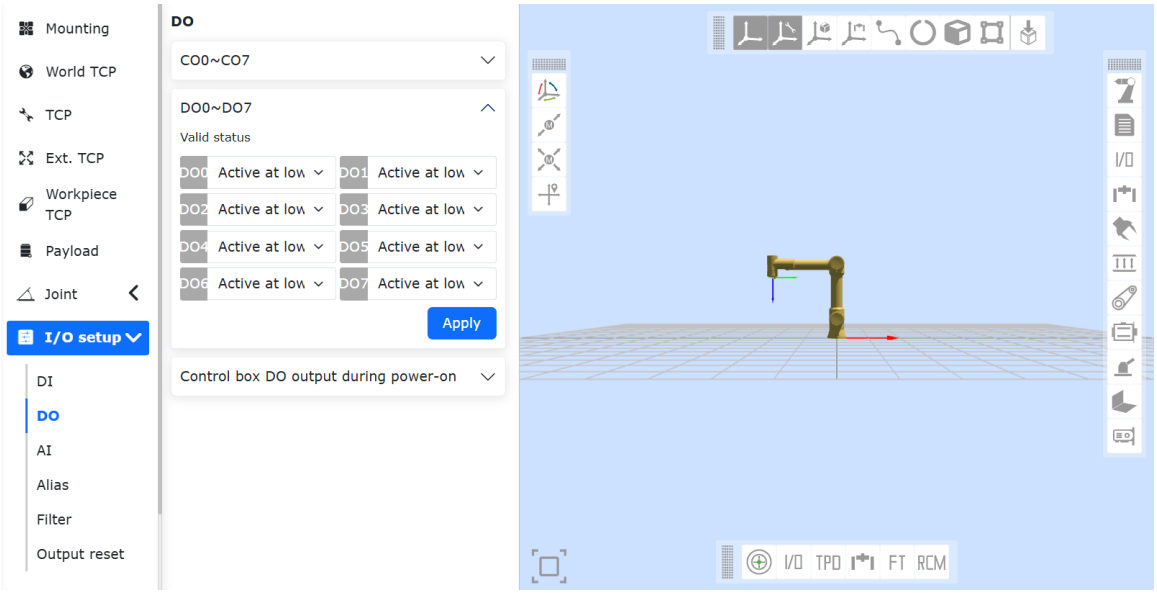
Table 3.3-5 Valid Status Comparison Table

Control Box Type	Input Type	High Level Valid (Switch Set to ON)	High Level Valid (Switch Set to OFF)	Low Level Valid (Switch Set to ON)	Low Level Valid (Switch Set to OFF)
DC Control Box	NPN Output	Valid	Valid	Invalid	Invalid
AC Narrow Voltage Control Box	NPN Output	Valid	Valid	Invalid	Invalid
AC Wide Voltage Control Box	NPN Output	Valid	Valid	Invalid	Invalid
AC Wide Voltage Control Box	PNP Output	Valid	Valid	Invalid	Invalid

(2) The comparison table of valid status under different settings of "DO0-D07 General Output Valid Status" is as follows:

Figure 3.3-8 DO0-D07 General Output Valid Status

Table 3.3-6 Valid Status Comparison Table



Control Box Type	Input Type	High Level Valid (Switch Set to ON)	High Level Valid (Switch Set to OFF)	Low Level Valid (Switch Set to ON)	Low Level Valid (Switch Set to OFF)
DC Control Box	NPN Output	Valid	Invalid	Invalid	Valid
AC Narrow Voltage Control Box	NPN Output	Valid	Invalid	Invalid	Valid
AC Wide Voltage Control Box	NPN Output	Valid	Invalid	Invalid	Valid
AC Wide Voltage Control Box	PNP Output	Valid	Invalid	Invalid	Valid

1.3.3.3.3.12 Inspection maintenance plan

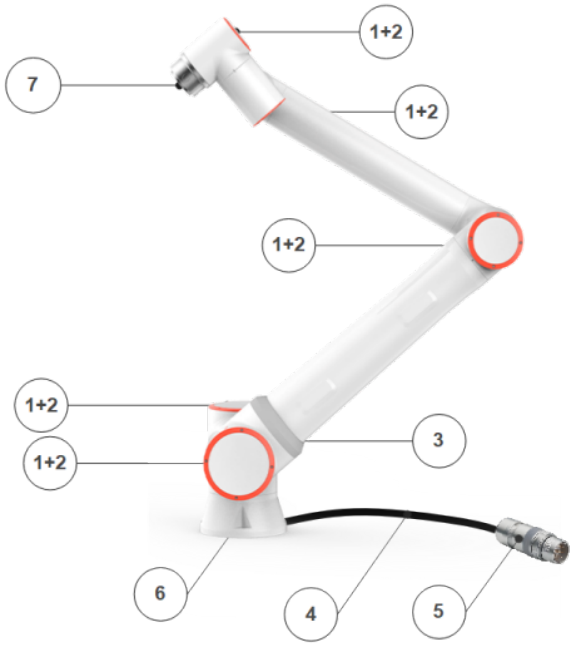
1.3.3.3.3.13 Robotic arm

1. Inspection plan

Below is a checklist of checklists that FAIRINO Robots recommends performing based on the marked time intervals. If the inspection reveals that the condition of the relevant parts is unqualified, please correct it immediately.

Note: F=Functional check,V=visual inspection,*=Must be checked after severe collision.

	Check item		Monthly	Semi-annually	Annually
1	Check joint rear cover *	V		✓	
2	Check joint rear cover screws	F		✓	
3	Check joint rubber ring	V		✓	
4	Check robot cables	V		✓	
5	Check robot cable links	V		✓	
6	Check robot base mounting bolts *	F	✓		
7	Check End Tool Mounting Bolts *	F	✓		



2. Visual inspection

Note: Do not use compressed air to clean robot arms as it may damage components. Do not store the robot for more than 6 months without visual inspection.

- If possible, move the robot arm to the zero position.
- Turn off and disconnect the power cord of the control box.
- Check the cable between the control box and the robot arm for any damage.
- Check whether the base mounting bolts are properly tightened.
- Check whether the tool flange bolts are properly tightened.
- Check whether the flat ring is worn and damaged.
- Check all joint backs for any cracks or damage.
- Check that the screws for the articulated rear cover are seated and tightened correctly.

Note: If the robot shows any damage during the warranty period, please contact the dealer who purchased the robot.

3. Function check

The purpose of the functional inspection is to ensure that screws, bolts, tools and robot arms are not loose. The screws/bolts mentioned in the inspection planshall be checked with torque wrenches and the torque shall comply with the standard specifications, which can be found in chapter of the Installation Specifications of the User Manual for the specifications of the mounting bolts of the robot arm.

4. Cleaning

You can wipe off any dust/dirt/grease observed on the robot arm using a cloth and one of the following cleaners: water, isopropyl alcohol, 10% ethanol or 10% naphtha. If the robot is operating in harsh environments, such as cutting fluids, coolants, etc., it is recommended to clean or replacethe rubber ringregularly.

Do not use bleach. Do not use bleach in any diluted cleaning solution.

In rare cases, a very small amount of grease can be seen from the joint. This does not affect the function, use or longevity of the joint.

1.3.3.3.3.14 Control box, teaching device, button box

1. Inspection plan

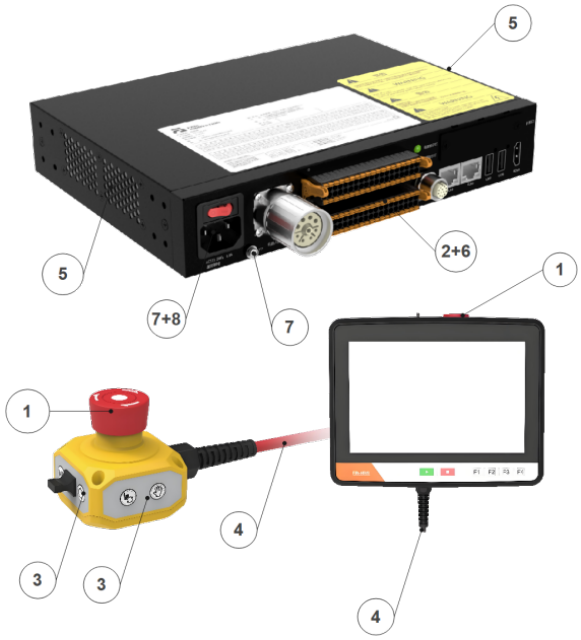
Below is a checklist of checklists that FAIRINO Robots recommends performing based on the marked time intervals. If the inspection reveals that the condition of the relevant parts is unqualified, please correct it immediately.

Note: F=Functional check,V=Visual inspection.

	Check item		Monthly	Semi-annually	Annually
1	Emergency stop button on test button box (teach pendant)	F	✓		
2	Safety input and output functions on the test terminal strip	F	✓		
3	Detection button box start/stop, mode switching function	F	✓		
4	Test button box (teach pendant) cable	V		✓	
5	Check and clean the air filter on the control box	V	✓		
6	Check whether the terminals of the control box are firm	F		✓	
7	Ground resistance of detection control box 1	F			✓
8	Check the main power supply of the control box	F			✓

2. Visual inspection

- Unplug the power cord from the control box.
- Check that the terminals of the control board are inserted correctly and there are no loose wires.



- Check whether there is dirt/dust in the control box. Use an ESD vacuum cleaner for cleaning if necessary.

Note: Do not use compressed air to clean the inside of the control box as this may damage components.

3. Function check

Note: Robot safety features are important and it is recommended to test once a month to ensure proper functionality.

- Emergency stop button onteach pendant/button box:
  A. Press the emergency stop button on the teach pendant/button box.
  B. Observe that the robot stops and turns off the joint power.
  C. Turn on the robot power again.

- Other safety inputs and outputs remain operational

  Check which safety inputs and outputs are active and whether they can be triggered by PolyScope or external devices.

- Date and clock

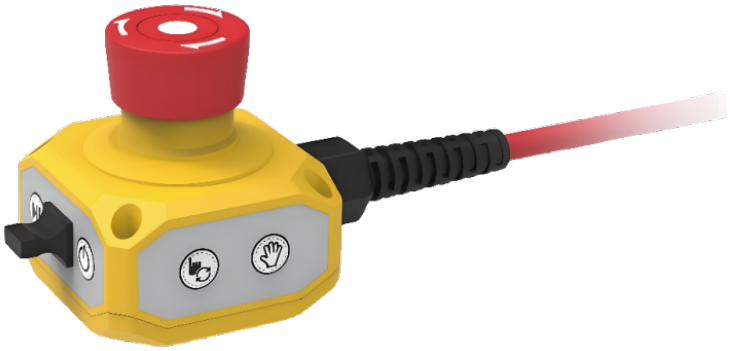
  Check that the date and clock in the Log tab are correct. Incorrect dates and clocks indicate low CMOS battery charge. CMOS batteries have a shelf life of up to5 years.

- Check whether the terminal snaps are in place

4. Cleaning

- Teaching device


STOP
FR-HMI
F1
F2
F3
F4





You may need to clean the teach pendant screen. It is recommended to use a standard mild industrial cleaner that does not contain diluents or any corrosive additives. Do not wipe the screen with abrasive materials. FAIRINO don't market specific cleaning agents.

- The button box

  Do not need to be cleaned regularly when it is not normal. If the key identification is blurred and affects the recognition operation, please clean it with detergent at any time.

- Control box

  The control box contains two filters, one on each side of the control box.

  A. The filter can be observed from the left and right side vents of the control box. Under normal circumstances, you can see the honeycomb structure of the filter.

  B. Remove the filter for cleaning. Clean with low pressure air or change filters as needed. Remember to clean each side. If it is dirty or damaged, replace it (worse, remove the upper cover of the controller and replace the filter from inside the upper cover).

  C. Listen to the sound of the fan when running. If the sound is abnormal, please contact the service provider or replace it.

1.3.3.3.3.15 Check program registration card

1. Robotic arm

InspectionItem	Inspected	Inspector	Data	Remark
Check joint rear cover				
Check joint rear cover screws				
Check joint rubber ring				
Check robot cables				
Check robot cable links				
Check robot base mounting bolts				
Check End Tool Mounting Bolts				

2. Control box, teaching device, button box

InspectionItem	Inspected	Inspector	Data	Remark
Emergency stop button on test button box (teach pendant)				
Safety input and output functions on the test terminal strip				
Detection button box start/stop, mode switching function				
Test button box (teach pendant) cable				
Check and clean the air filter on the control box				
Check whether the terminals of the control box are firm				
Ground resistance of detection control box 1				
Check the main power supply of the control box				

1.3.3.3.4 Robot waste disposal

FR robots need to be disposed of according to the applicable national laws and regulations and national standards. For details, you can contact manufacturers.

1.3.3.4 Installation specifications

1.3.3.4.1 Robot arm installation

Important: The recommended robot installation base meets the following requirements to ensure a secure and stable installation of the robot:

(1)The robot mounting base needs to be strong enough and have sufficient load-bearing capacity, which should be able to bear at least 5 times the weight of the robot and at least 10 times the 1-axis torque.

(2)The surface of the robot mounting base should be flat to ensure close contact with the robot contact surface.

(3)The robot mounting base should have sufficient stiffness, be firmly fixed, and not resonate with the robot.

(4)When the robot and other components are moving simultaneously, the mounting base should be separated from other moving components and not fixed together to avoid vibration interference during the movement process.

(5)If the robot is installed on a mobile platform or external axis, the acceleration of the mobile platform or external axis should be as low as possible.

Warning: The following installation methods should be avoided:

(I)Avoid fixing the robot to other moving devices.

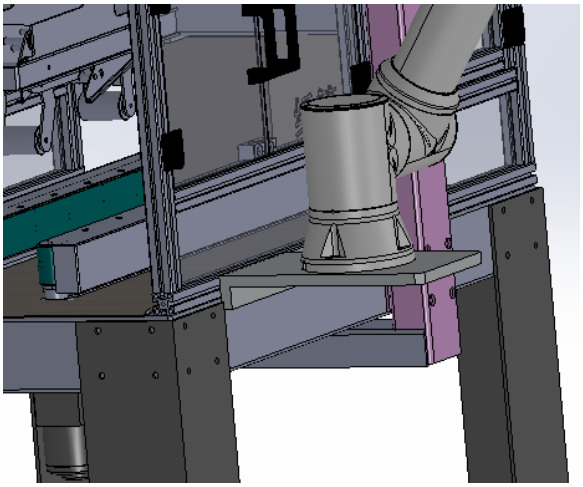


Figure 3.4-1 Avoid installing on other sports equipment

Make sure the robot arm is installed correctly and safely. Unstable installation will cause accidents.

Note: You can purchase accurate bases as attachments. Figure 3.4-23.4-53.4-83.4-11 show the position of the sales hole and the location of the screw.

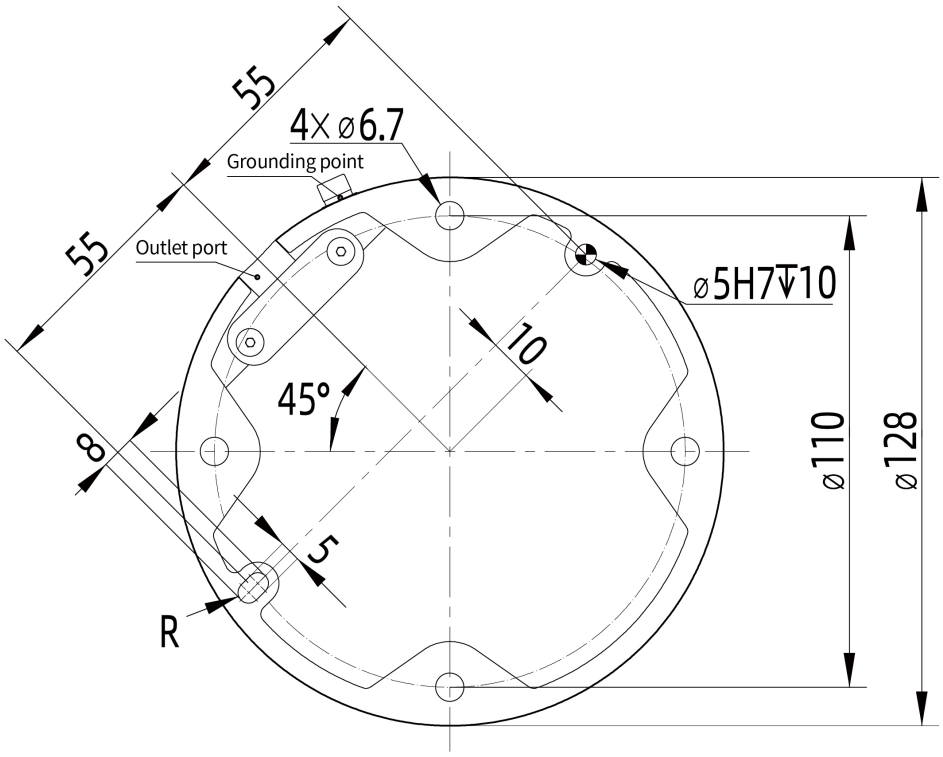
1.3.3.4.1.1 Installation requirements for FR3/FR3-WMS/FR3-WML/FR3-C robot

When installing the robot on the mounting base, use four M6 bolts with a strength of not less than 8.8 to fix the robot on the mounting base. The bolts must be tightened with a torque of not less than 10Nm.Suggest using two on the mounting base 5mm pin hole matched with pins for robot positioning to improve robot installation accuracy and prevent robot movement due to collisions and other factors.When the robot has high operating accuracy requirements, please be sure to add pins to position the robot.

Figure 3.4-2 FR3/FR3-WMS/FR3-WML/FR3-C model collaborative robot installation size

Important: According to different application scenarios, we recommend several robot installation bases as follows

(I)For situations where the motion speed is not too fast, the running speed is not too large, the accuracy requirements are average, and it is not convenient to fix the robot on the ground, the recommended installation base for the robot is as follows.


55
4X ø6.7
Grounding point
55
Outlet port
ø5H7↧10
10
45°
8
5
R
ø110
ø128

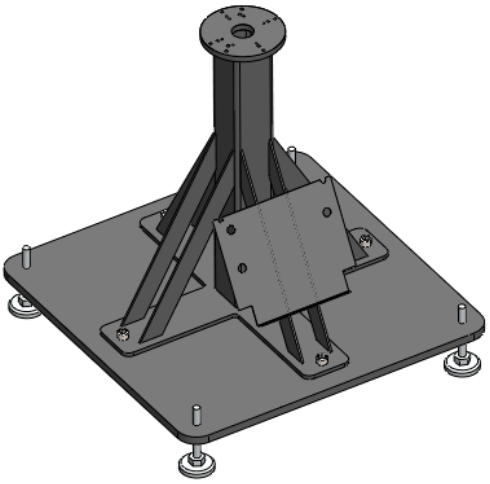


Figure 3.4-3 FR3/FR3-WMS/FR3-WML/FR3-C model collaborative robot low requirement mounting base

(II)For situations where the motion speed is fast, the running speed is high, and the accuracy requirements are high, it is recommended to install the robot on the following base and fix it on a solid ground.

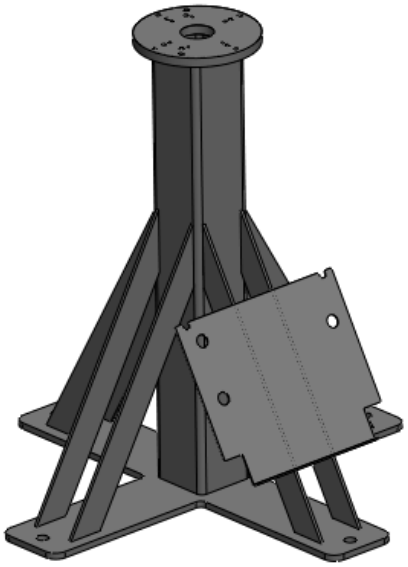


Figure 3.4-4 FR3/FR3-WMS/FR3-WML/FR3-C Model Collaborative Robot High Demand Mounting Base

1.3.3.4.1.2 Installation requirements for FR5 robot

When installing the robot on the mounting base, use four M8 bolts with a strength of not less than 8.8 to fix the robot on the mounting base. The bolts must be tightened with a torque of not less than 20Nm.Suggest using two on the mounting base 8mm pin hole matched with pins for robot positioning to improve robot installation accuracy and prevent robot movement due to collisions and other factors.When the robot has high operating accuracy requirements, please be sure to add pins to position the robot.

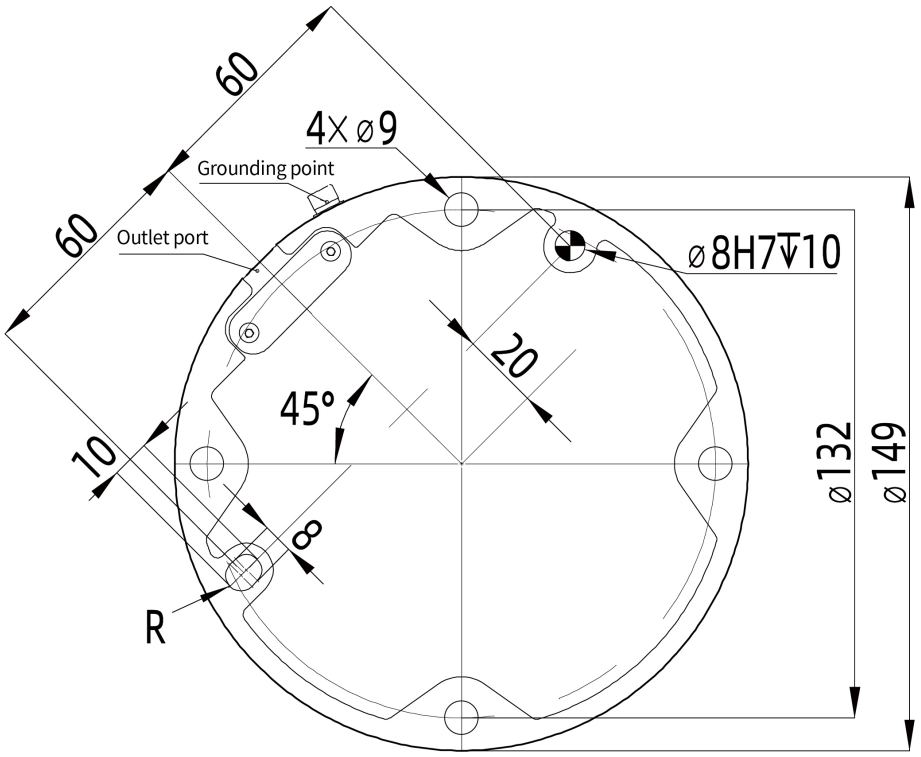


Figure 3.4-5 FR5 model collaborative robot installation size

Important: According to different application scenarios, we recommend several robot installation bases as follows

(I)For situations where the motion speed is not too fast, the running speed is not too large, the accuracy requirements are average, and it is not convenient to fix the robot on the ground, the recommended installation base for the robot is as follows.

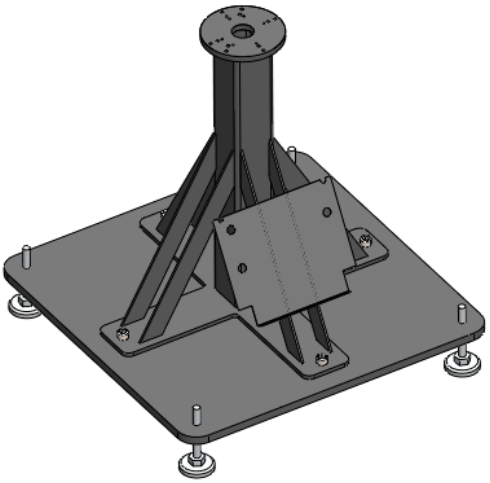


Figure 3.4-6 FR5 Model Collaborative Robot High Demand Mounting Base

(II)For situations where the motion speed is fast, the running speed is high, and the accuracy requirements are high, it is recommended to install the robot on the following base and fix it on a solid ground.

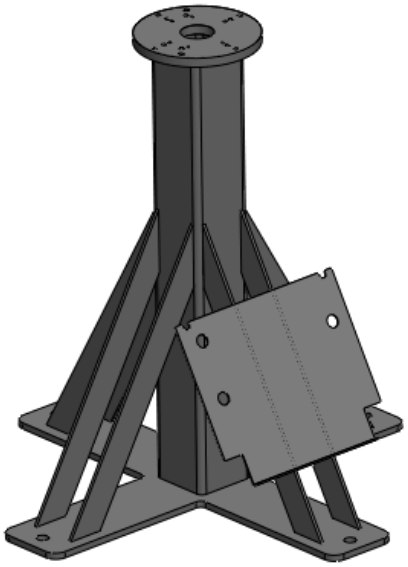


Figure 3.4-7 FR5 model collaborative robot low requirement mounting base

1.3.3.4.1.3 Installation requirements for FR10&FR16 robot

When installing the robot on the mounting base, use four M8 bolts with a strength of not less than 8.8 to fix the robot on the mounting base. The bolts must be tightened with a torque of not less than 25Nm.Suggest using two on the mounting base 8mm pin hole matched with pins for robot positioning to improve robot installation accuracy and prevent robot movement due to collisions and other factors.When the robot has high operating accuracy requirements, please be sure to add pins to position the robot.

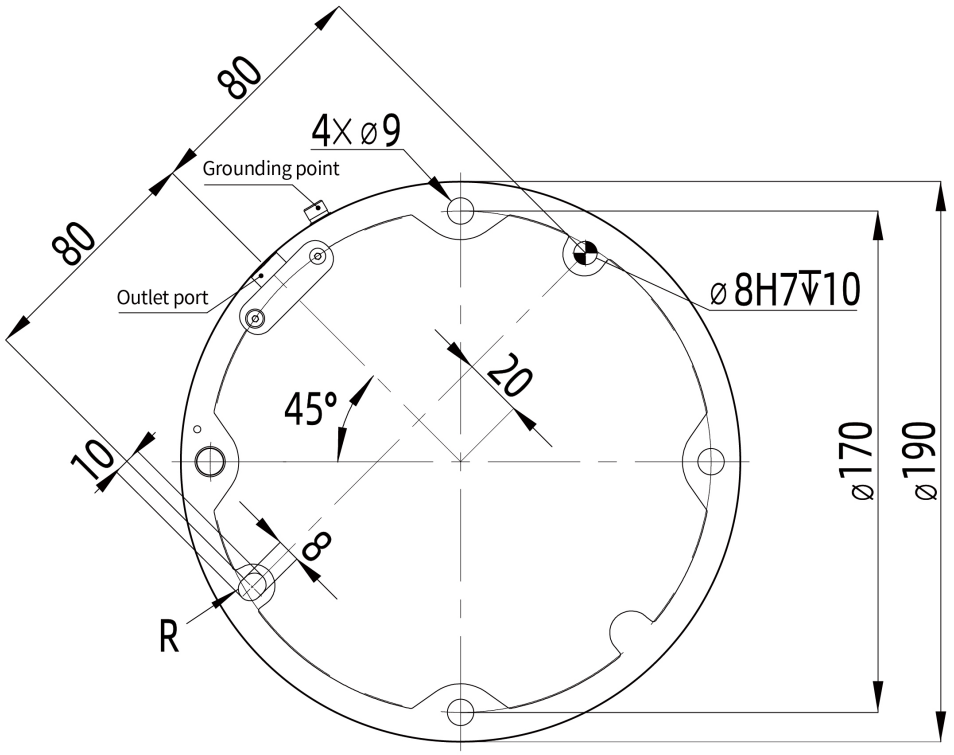


Figure 3.4-8 FR10&FR16 model collaborative robot installation size

Important: According to different application scenarios, we recommend several robot installation bases as follows

(I)For situations where the motion speed is not too fast, the running speed is not too large, the accuracy requirements are average, and it is not convenient to fix the robot on the ground, the recommended installation base for the robot is as follows.

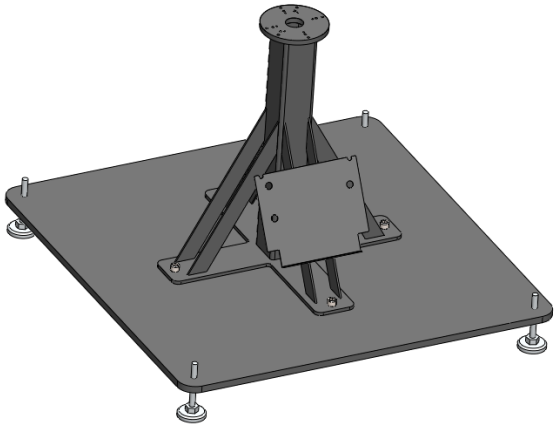


Figure 3.4-9 FR10&FR16 model collaborative robot low requirement mounting base

(II)For situations where the motion speed is fast, the running speed is high, and the accuracy requirements are high, it is recommended to install the robot on the following base and fix it on a solid ground.

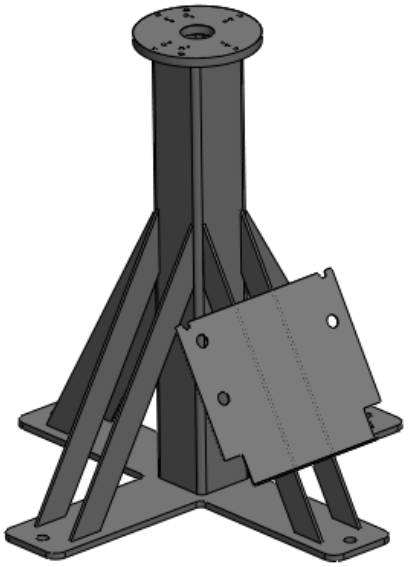


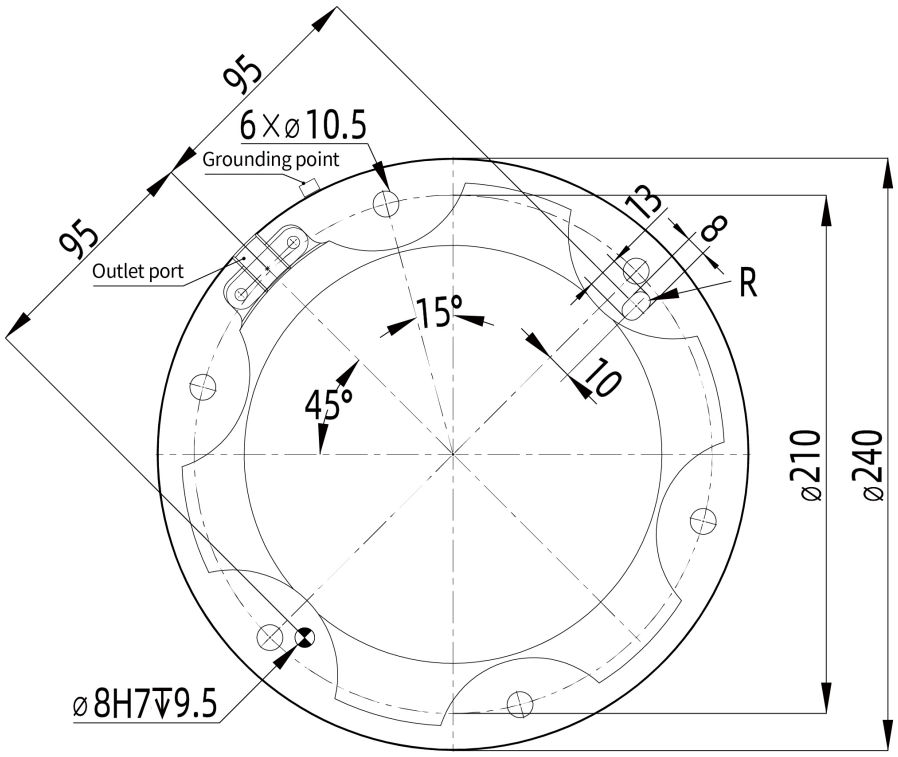
Figure 3.4-10 FR10&FR16 Model Collaborative Robot High Demand Mounting Base

1.3.3.4.1.4 Installation requirements for FR20&FR30&FR30L robot

When installing the robot on the mounting base, use six M10 bolts with a strength of not less than 8.8 to fix the robot on the mounting base. The bolts must be tightened with a torque of not less than 45Nm.Suggest using two on the mounting base 8mm pin hole matched with pins for robot positioning to improve robot installation accuracy and prevent robot movement due to collisions and other factors.When the robot has high operating accuracy requirements, please be sure to add pins to position the robot.

Figure 3.4-11 FR20&FR30&FR30L model collaborative robot installation size

Important: Because the FR20 and FR30 robots have a large weight and running inertia, it is recommended to be


95
95
6×ø10.5
Grounding point
Outlet port
13
8
R
15°
45°
10
ø210
ø240
ø8H7↧9.5

directly fixed on the ground. The recommended base is as follows.

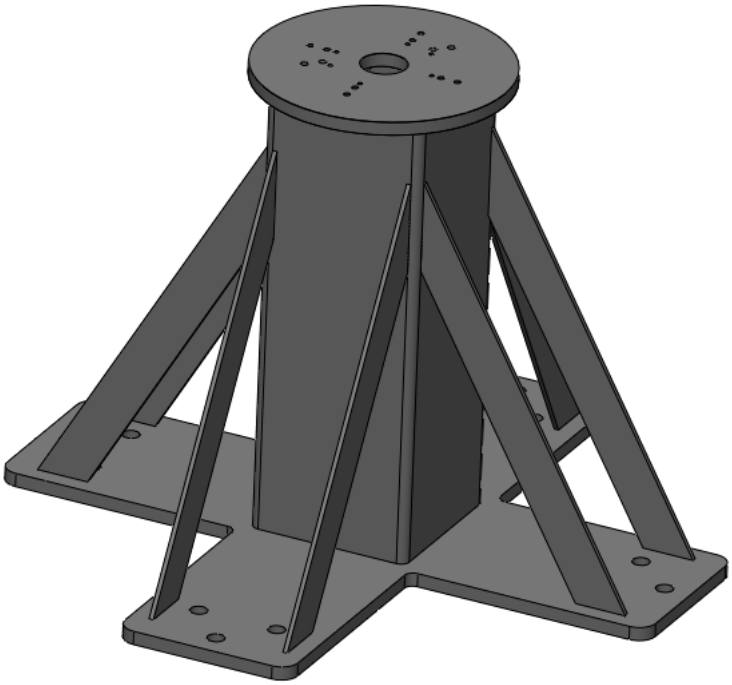


Figure 3.4-12 FR20&FR30 model collaborative robot low requirement mounting base

1.3.3.4.2 Tool end installation

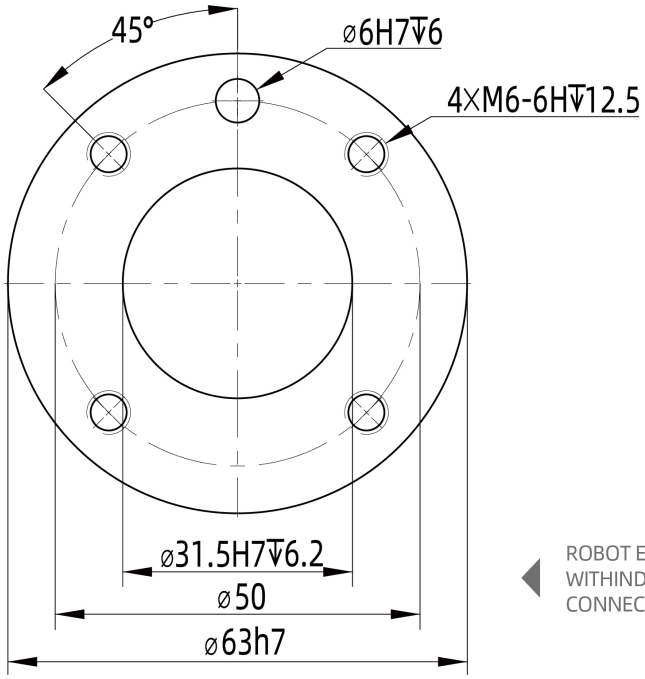
There are four M6 thread holes in the robot tool, which can be used to connect the tool to the robot. The M6 bolt must be tightened with 8nm torque, and its strength level is not less than 8.8. In order to accurately regain the tools, please use the nails in the reserved ø6 sales holes.

Figure 3.4-13 FR3/FR3-WMS/FR3-WML/FR3-C/FR5/FR10/FR16 model robot end flange drawing

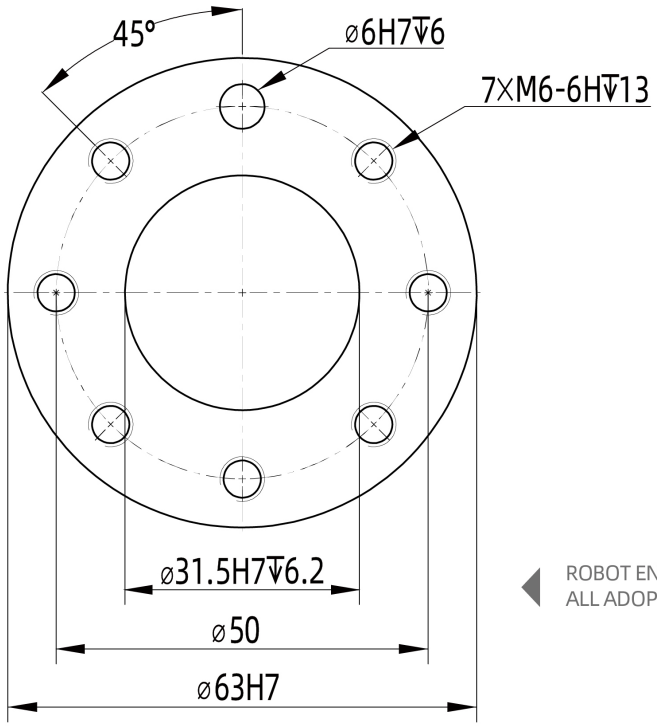
Figure 3.4-14 FR20/FR30/FR30L model robot end flange drawing

Important:

- Make sure the tools are installed correctly and safely.
- Ensure the safety architecture of the tools, and no parts of parts fall into danger.
- Installing M6 bolts with a length of more than 8mm on the robot flange may destroy tool flanges and cause damage that cannot be repaired, causing a tool to change tools.



ROBOT END-EFFECTOR COMPATIBLE WITHINDUSTRIAL ROBOT END-EFFECTOR CONNECTION METHODS



ROBOT END-EFFECTORS
ALL ADOPT INTERNATIONAL STANDARDS

1.3.3.4.3 Installation environment

When installing and using a collaborative robot, make sure to meet the following requirements:

- Environmental temperature 0-45 ℃
- Humidity 0% to 90% RH (no condensation)
- No mechanical impact and shock
- Altitude requires less than 2000M
- No corrosive gases, no liquid, no explosive gases, no oil pollution, no salt fog, no dust or metal powder, no radioactive material, no electromagnetic noise, non-flammable items
- Avoid the device from working under the unstable conditions of the current
- Users need to add an air switch before the robot power supply, and it is recommended to add an EMC filter

Note: If you want to hang or install the collaborative robot, please contact us.

1.3.3.4.4 Floor carrier capacity

Installing the robot on a strong surface, the surface should be sufficient to withstand the weight of the robotic arm at least 5 times, and the surface cannot be vibrated.

1.3.3.4.5 Load curves for all FR series models

1.3.3.4.5.1 Overview

The load curves in this section are based on the tests of each model under specific trajectories. The load curves of each model have two parts: “full performance” and “extended load capacity”, as follows:

(1) The operating environment of “full performance” is: the friction compensation coefficient of each joint is 1; the collision level of each joint is 10; the web interface is set to 100% operating speed and 360deg/s2 acceleration; dynamics 2.0. In this environment, the “full performance” part of the load curve is suitable for most operating trajectories.

(2) If the end load is in the “extended load capacity”, the “time optimal mode” must be turned on and the acceleration limit must be met, or the robot’s working range must be reduced.

1.3.3.4.5.2 Parameter Description

The rated payload of the robot depends on the center of gravity offset of the payload, where the center of gravity offset is defined as the distance between the center of the end flange and the center of gravity of the attached payload.

1.3.3.4.5.3 FR3 Model Collaborative Robot Load Curve

The maximum load that the FR3 collaborative robot can carry is 5kg, and the rated load is 3kg. The load curve is shown in Figure 1. The specific interpretation of the load curve is as follows:

(1) FR3 can carry a load of 3kg or less at full performance, see the "blue envelope";

(2) When the load is 3kg to 5kg, it is the extended load capacity, see the "red envelope", at this time the robot can operate in the following states:

Turn on the "time optimal mode", it is recommended to set the acceleration to less than 360deg/s^2;

Reduce the robot's working range or reduce the operating speed.

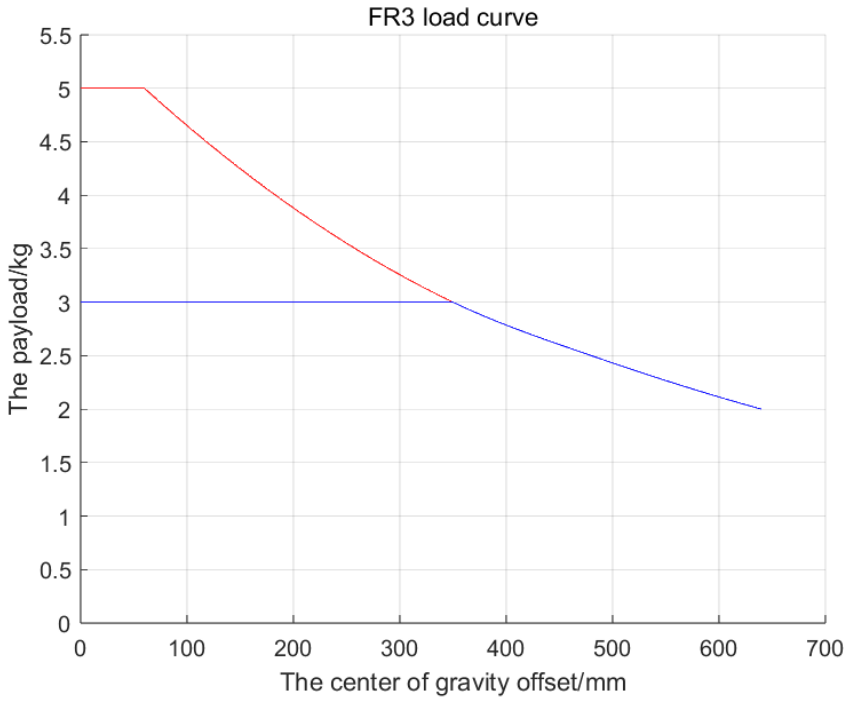


Figure 3.4-15 FR3 Model Collaborative Robot Load Curve

1.3.3.4.5.4 FR3-WMS Model Collaborative Robot Load Curve

The maximum load that the FR3-WMS collaborative robot can carry is 5kg, and the rated load is 3kg. The load curve is shown in Figure 1. The specific interpretation of the load curve is as follows:

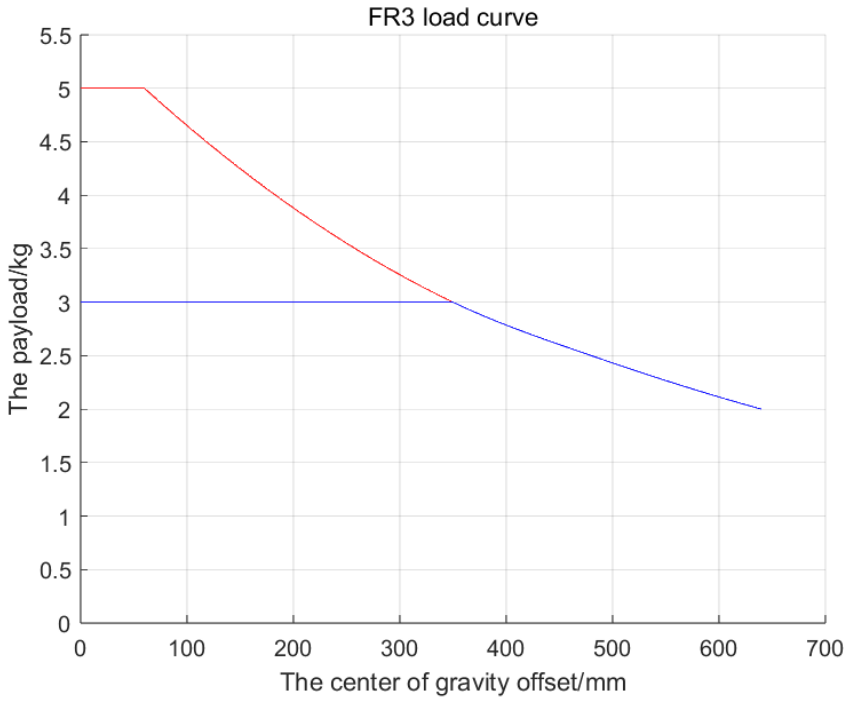
(1) FR3-WMS can carry a load of 3kg or less at full performance, see the "blue envelope";

(2) When the load is 3kg to 5kg, it is the extended load capacity, see the "red envelope", at this time the robot can operate in the following states:

Turn on the "time optimal mode", it is recommended to set the acceleration to less than 360deg/s^2;

Reduce the robot's working range or reduce the operating speed.

Figure 3.4-16 FR3-WMS Model Collaborative Robot Load Curve



1.3.3.4.5.5 FR3-WML Model Collaborative Robot Load Curve

The maximum load that the FR3-WML collaborative robot can carry is 4kg, and the rated load is 3kg. The load curve is shown in Figure 1. The specific interpretation of the load curve is as follows:

(1) FR3-WML can carry a load of 3kg or less at full performance, see the “blue envelope”;

(2) When the load is 3kg to 4kg, it is the extended load capacity, see the “red envelope”, at this time the robot can operate in the following states:

Turn on the “time optimal mode”, it is recommended to set the acceleration to less than 360deg/s^2;

Reduce the robot’s working range or reduce the operating speed.

Figure 3.4-17 FR3-WML Model Collaborative Robot Load Curve

1.3.3.4.5.6 FR3-C Model Collaborative Robot Load Curve

The maximum load that the FR3-C collaborative robot can carry is 5kg, and the rated load is 3kg. The load curve is shown in Figure 1. The specific interpretation of the load curve is as follows:

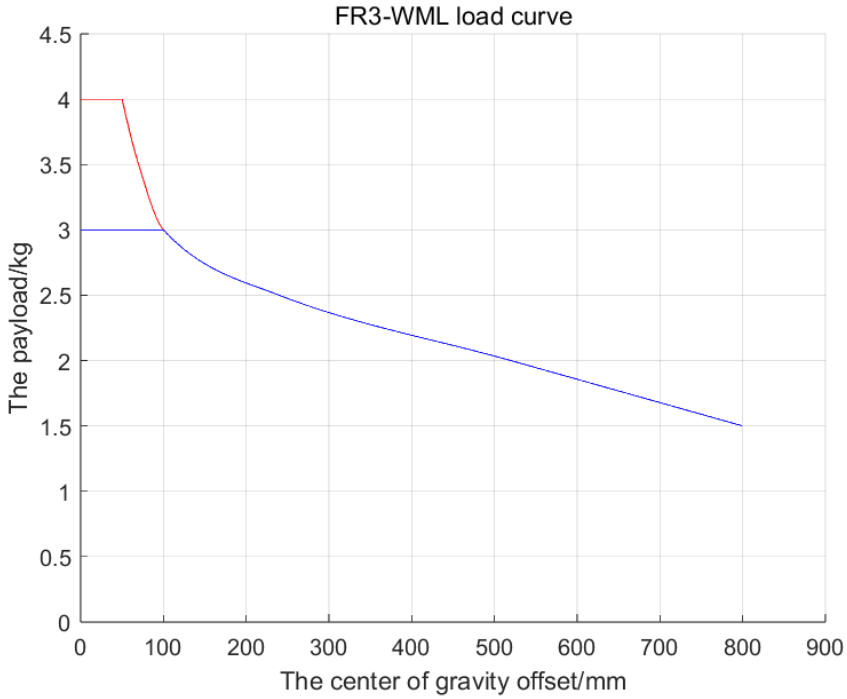
(1) FR3-C can carry a load of 3kg or less at full performance, see the “blue envelope”;

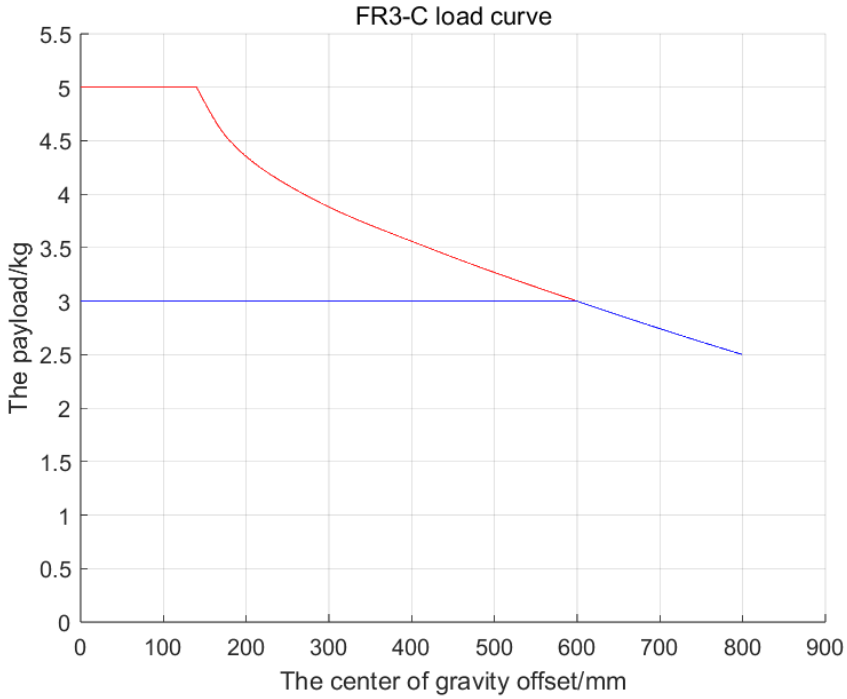
(2) When the load is 3kg to 5kg, it is the extended load capacity, see the “red envelope”, at this time the robot can operate in the following states:

Turn on the “time optimal mode”, it is recommended to set the acceleration to less than 360deg/s^2;

Reduce the robot’s working range or reduce the operating speed.

Figure 3.4-18 FR3-C Model Collaborative Robot Load Curve


FR3-WML load curve
The payload/kg
The center of gravity offset/mm
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
0
100
200
300
400
500
600
700
800
900


FR3-C load curve
The payload/kg
The center of gravity offset/mm
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
0
100
200
300
400
500
600
700
800
900

1.3.3.4.5.7 FR5 Model Collaborative Robot Load Curve

The maximum load that the FR5 collaborative robot can carry is 7kg, and the rated load is 5kg. The load curve is shown in the figure. The specific interpretation of the load curve is as follows:

(1) FR5 can carry a load of 5kg or less at full performance, see the "blue envelope";

(2) When the load is 5kg to 7kg, it is the extended load capacity, see the "red envelope", at this time the robot can operate in the following states:

Turn on the "time optimal mode", it is recommended to set the acceleration to less than 360deg/s^2;

Reduce the robot's working range or reduce the running speed.

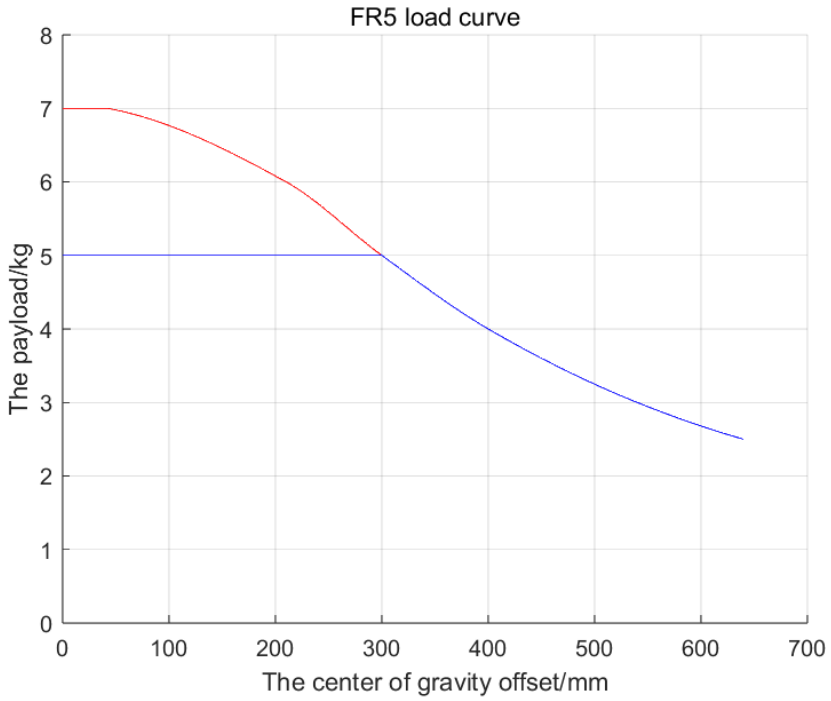


Figure 3.4-19 FR5 Model Collaborative Robot Load Curve

1.3.3.4.5.8 FR5-WML Model Collaborative Robot Payload Curve

The FR5-WML model collaborative robot has a maximum payload capacity of 7kg and a rated payload of 5kg. The payload curve is shown in the figure. The specific interpretation of the payload curve is as follows:

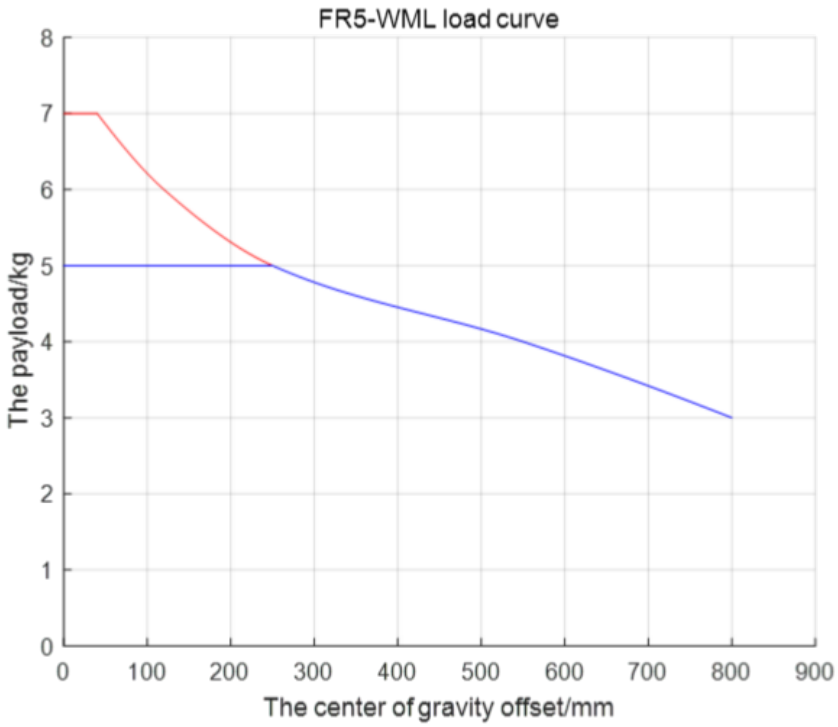
(1) Within the "blue envelope": Full performance - can run most trajectories with friction compensation coefficients all at 1, dynamics 2.0, 100% speed, 360 deg/s^2 acceleration (maintenance mode).

(2) Within the "red envelope": Extended payload capacity - can operate under the following conditions:

Enable "Time-optimal Mode";

Reduce robot working range or lower operating speed.

Figure 3.4-20 FR5-WML Model Collaborative Robot Payload Curve



1.3.3.4.5.9 FR10 Model Collaborative Robot Load Curve

The maximum load that the FR10 collaborative robot can carry is 14kg, and the rated load is 10kg. The load curve is shown in Figure 3. The specific interpretation of the load curve is as follows:

(1) FR10 can carry a load of 10kg or less at full performance, see the “blue envelope”;

(2) When the load is 10kg to 14kg, it is the extended load capacity, see the “red envelope”, at this time the robot can operate in the following states:

Turn on the “time optimal mode”, it is recommended to set the acceleration to less than 180deg/s^2;

Reduce the robot’s working range or reduce the operating speed.

Figure 3.4-21 FR10 Model Collaborative Robot Load Curve

1.3.3.4.5.10 FR16 Model Collaborative Robot Load Curve

The maximum load that the FR16 collaborative robot can carry is 20kg, and the rated load is 16kg. The load curve is shown in the figure. The specific interpretation of the load curve is as follows:

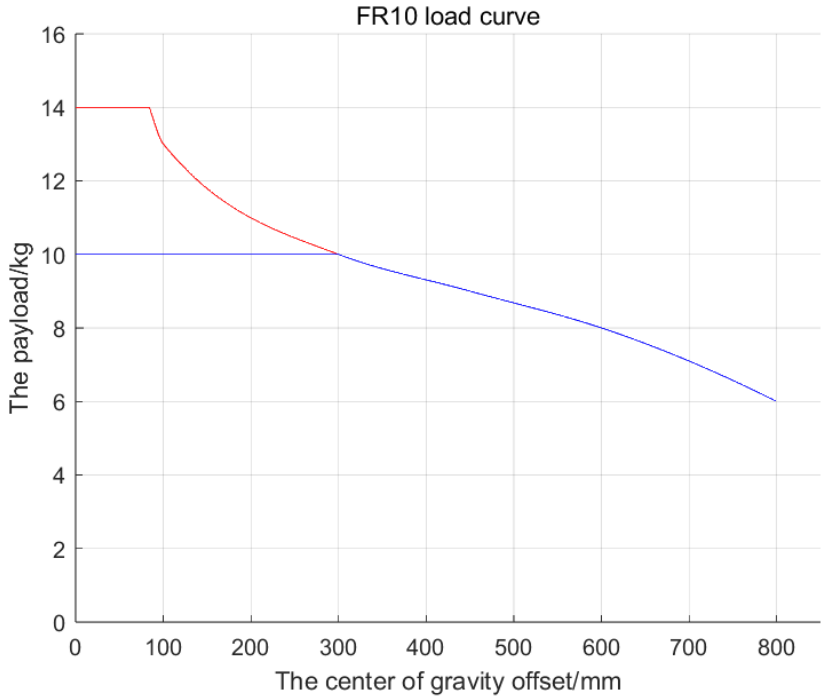
(1) FR16 can carry a load of 16kg or less at full performance, see the “blue envelope”;

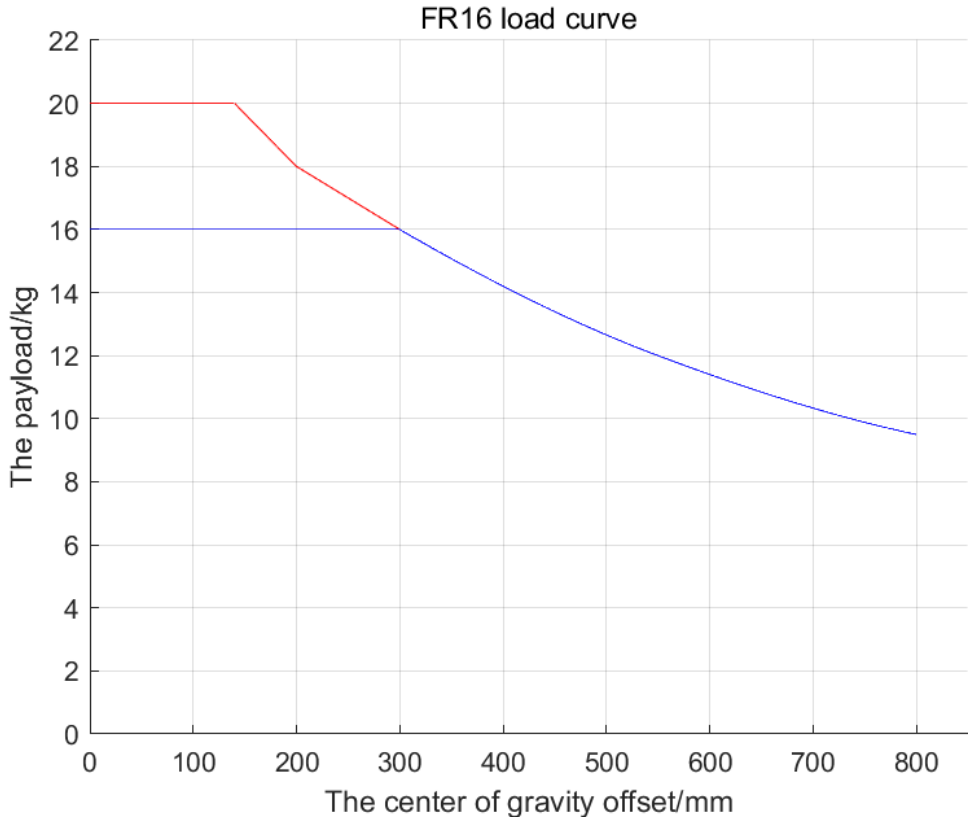
(2) When the load is 16kg to 20kg, it is the extended load capacity, see the “red envelope”, and the robot can operate in the following states:

Turn on the “time optimal mode”, and it is recommended to set the acceleration to less than 180deg/s^2;

Reduce the robot’s working range or reduce the operating speed.

Figure 3.4-22 FR16 Model Collaborative Robot Load Curve


FR10 load curve
The payload/kg
The center of gravity offset/mm
0
2
4
6
8
10
12
14
16
0
100
200
300
400
500
600
700
800


FR16 load curve
The payload/kg
The center of gravity offset/mm
0
2
4
6
8
10
12
14
16
18
20
22
0
100
200
300
400
500
600
700
800

1.3.3.4.5.11 FR20 Model Collaborative Robot Load Curve

The maximum load that the FR20 collaborative robot can carry is 25kg, and the rated load is 20kg. The load curve is shown in the figure. The specific interpretation of the load curve is as follows:

(1) FR20 can carry a load of 20kg or less at full performance, see the "blue envelope";

(2) When the load is 20kg to 25kg, it is the extended load capacity, see the "red envelope", at this time the robot can operate in the following states:

Turn on the "time optimal mode", it is recommended to set the acceleration to less than 150deg/s^2;

Reduce the robot's working range or reduce the operating speed.

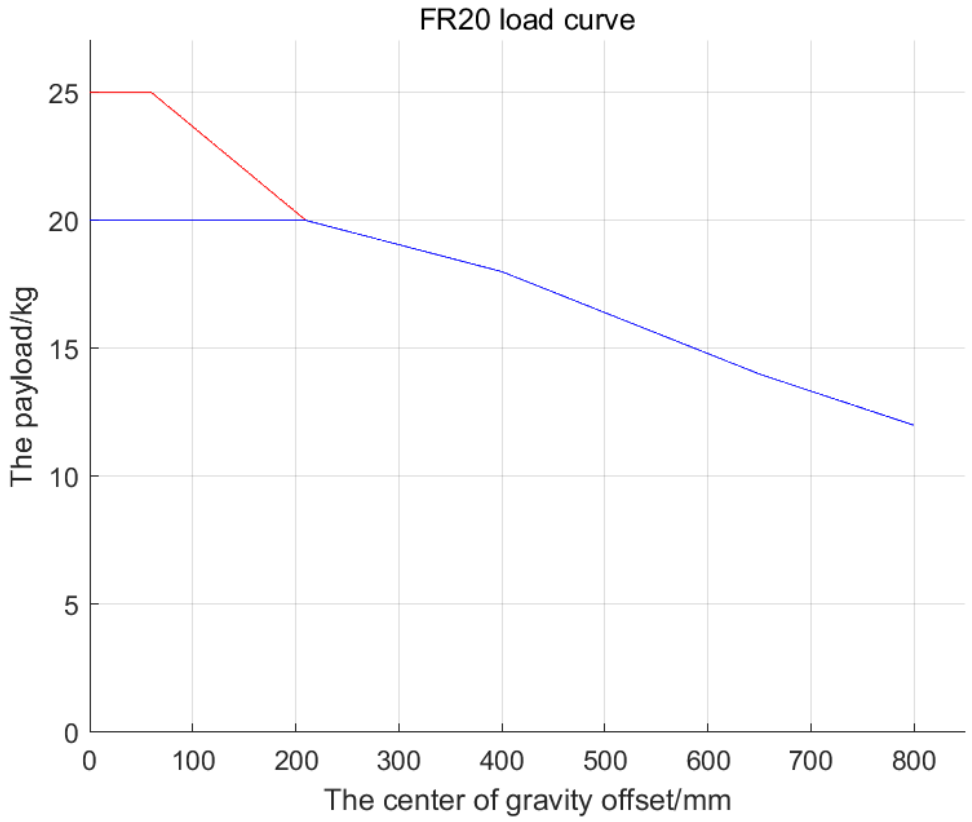


Figure 3.4-23 FR20 Model Collaborative Robot Load Curve

1.3.3.4.5.12 FR30 Model Collaborative Robot Load Curve

The maximum load that the FR30 collaborative robot can carry is 35kg, and the rated load is 30kg. The load curve is shown in the figure.

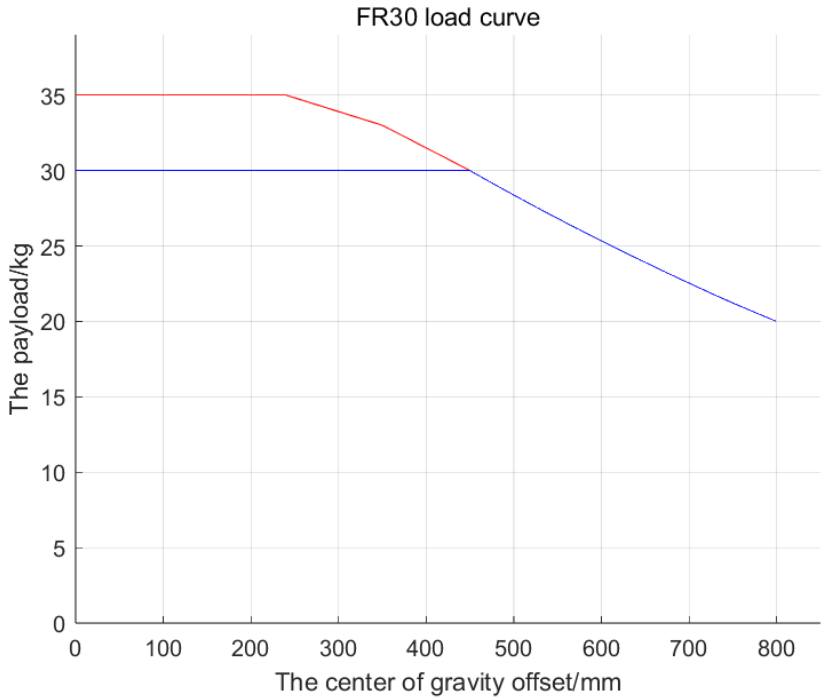
(1) FR30 can carry a load of 30kg or less at full performance, see the "blue envelope";

(2) When the load is 30kg to 35kg, it is the extended load capacity, see the "red envelope", at this time the robot can operate in the following states:

Turn on the "time optimal mode", it is recommended to set the acceleration to less than 150deg/s^2;

Reduce the robot's working range or reduce the operating speed.

Figure 3.4-24 FR30 Model Collaborative Robot Load Curve



1.3.3.5 Control connection

1.3.3.5.1 Controller interface

This series of robots can be equipped with three control boxes with different power inputs. For details on the control box power input, refer to the control box nameplate information. The robot needs to be electrically grounded.

	Maximum Input (for customers to configure the front-stage power supply)	Maximum output Output (maximum output peak)
DC 2kW	30-60VDC/30A	2000W/48VDC/41A
DC 5kW	30-60VDC/40A	5000W/48VDC/104A
AC narrow voltage 2kW	176-264VDC/10A/Single Machine/50Hz	2000W/48VDC/41A
AC wide voltage 2kW	100-240VDC/10A/Single Machine/50-60Hz	2000W/48VDC/41A
AC wide voltage 5kW	100-240VDC/16A/Single Machine/50-60Hz	5000W/48VDC/104A

Warning: Before wiring, please ensure that the power supply is turned off and hang a safety warning sign next to it.

The external wiring of this series of robotic arm control systems is connected using pluggable and quickly installable plugs. The wiring panel of the collaborative robot is shown in Figure 3.5-1.

- Make sure the power button on the control box is off (the button is turned to 0) and connect the power cord to the power socket.
- Connect the robot body overload cable to the control box overload interface.

- Insert the button box aviation plug into the control box teaching device interface.
- The heat dissipation ports on both sides of the control box should be spaced at least 15CM apart.
- At the front of the control box (user Table metal, switch power button, heavy load and teaching pendant wiring harness), the spacing distance should not be less than 25CM.
- The control box is 0.6-1.5m above the ground.
- Do not allow users to replace power cables on their own.



Figure 3.5-1 Robot wiring schematic diagram

1.3.3.5.2 Controller I/O panel

You can use the I/O in the control box to control various devices, including the stop button of pneumatic relay, PLC, and tight limit device. Figure 3.5-2 shows the electrical interface group of the control box. Figure 3.5-3 shows the electrical interface group that is easy to manufacture control box.

Figure 3.5-2 Control box electrical interface schematic diagram

Figure 3.5-3 Easy to manufacture control box electrical interface schematic diagram

Power communication	Universal digital input		Configurable digital inputs		Security	Universal digital output		Configurable digital output		Encoder	Analog
ex24V	GND	GND	GND	GND	EI0+	24V	24V	24V	24V	A1-	GND
exGND	DIO	DI4	CI0	CI4	EI0-	DO0	DO4	CO0	CO4	A1+	AI0
24V	GND	GND	GND	GND	EI1+	24V	24V	24V	24V	B1-	GND
GND	DI1	DI5	CI1	CI5	EI1-	DO1	DO5	CO1	CO5	B1+	AI1
5V	GND	GND	GND	GND	SI0+	24V	24V	24V	24V	A2-	GND
GND	DI2	DI6	CI2	CI6	SI0-	DO2	DO6	CO2	CO6	A2+	AO0
485-B	GND	GND	GND	GND	SI1+	24V	24V	24V	24V	B2-	GND
485-A	DI3	DI7	CI3	CI7	SI1-	DO3	DO7	CO3	CO7	B2+	AO1

User peripheral interface

M8 Waterproof connector



LAN network interface



LAN network interface (spare)



USB-2.0 interface



When the electrical interface of the control cabinet is wired, the control cabinet must be powered off.

Situations that may cause danger and, if not avoided, can lead to personal injury or serious damage to equipment;Matters marked with this symbol may, depending on the circumstances, sometimes have the possibility of significant consequences.



J8: I: Analog current input (0-20mA)
V: Analog voltage input (0-10V)
J12: I: Analog current output (0-20mA)
J14: V: Analog voltage output (0-10V)

FAIR INNOVATION

Product Name: Controller
Product Model: FRC100-AC
Service Temperature: 0-45℃
Protection Class: IP54
Input : 220VAC/10A/Single Phase/50HZ
Output : 6000W/48VDC/41A

Manufacturer: FAIR INNOVATION (Suzhou) Robot System Co.,LTD
Address: 209 Zhuyuan oad,High-tech Zone,Suzou City,Jiangsu Province
5888 Zunxian Road,High-tech Zone,Zibo City,Shandong Province
Telephone: 0512-68562005
Serial Number:
Date Of Production:

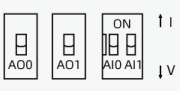
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
E-24V	E-0V	EI0-1	EI1-1	SI0-1	SI1-1	EDM-	EST0-1	EST1-1	485-B0	485-B1	5 V	B1+	B1-	B2+	B2-	GNDA	A00	A01
24V	0V	EI0-2	EI1-2	SI0-2	SI1-2	EDM+	EST0-2	EST1-2	485-A0	485-A1	GND	A1+	A1-	A2+	A2-	GNDA	AI 0	AI 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
D00	D01	D02	D03	E-24V	D04	D05	D06	D07	E-24V	C00	C01	C02	C03	E-24V	C04	C05	C06	C07
DI 0	DI 1	DI 2	DI 3	E- 0V	DI 4	DI 5	DI 6	DI 7	E- 0V	CI 0	CI 1	CI 2	CI 3	E- 0V	CI 4	CI 5	CI 6	CI 7

I/O Layout of user peripheral interfaces

I/O Interface

Instructions for use of dial switch in the chassis:
The switch is divided into AO/AI tow channels, and the analog is set through the dial switch, as shown on the right.



AO: I: Analog current output (0-20mA)
V: Analog voltage output (0-10V)
AI: I: Analog current input (0-20mA)
V: Analog voltage input (0-10V)

1.3.3.5.3 RJ45 network interface group

The network interface group address in the control box is shown in figure below. Note that the graph corresponds to the sequence of the address order of the internal network port of the control box, and the default port of the robot is prohibited from insertion. The user's network port can be used to communicate with the camera and other devices. The IP address is 192.168.57.2. The button box interface is default to the faculty control port, and the IP address is 192.168.58.2. Use the network cable connection button box interface and computer. The computer IP address is set to 192.168.58.10 or the same network segment as it. You can access the teaching pendant page. Easy to manufacture control boxes to access the pages of the oscillator through the network port of the connection button box.

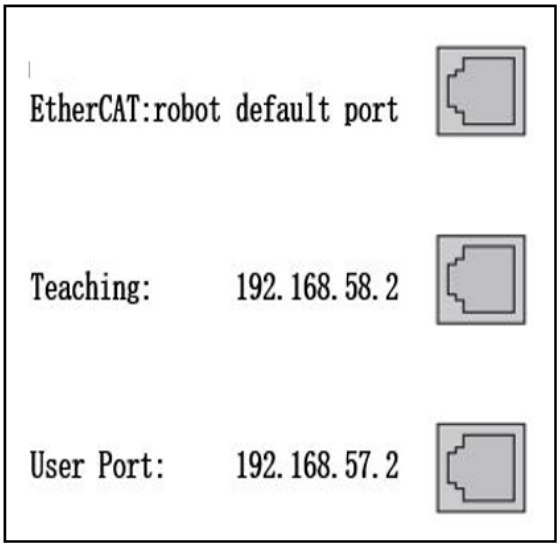


Figure 3.5-4 Significant diagram of network interface group

1.3.3.5.4 End plate

You can use the end -panel's I/O and 485 communication interfaces to control various devices, including pneumatic relay, PLC and emergency stop buttons. The PIN foot distribution and its PIN foot explanation is shown in figure below. The I/O connector model is M12 connector 8 cores.

Note: End board I/O and 485 interfaces are prohibited from hot plugging.

Figure 3.5-5 The schematic diagram of the end version of the electrical interface

1.3.3.5.5 Ground

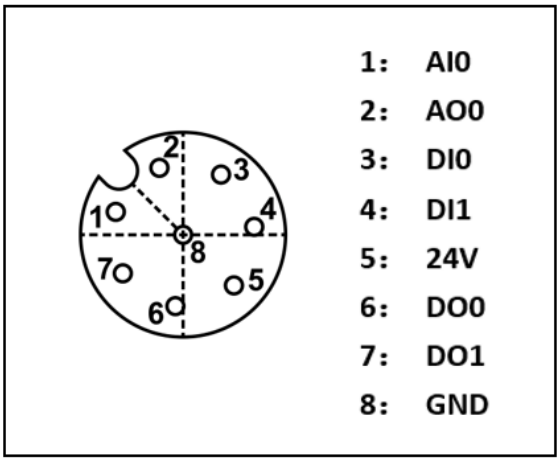
1. The control box is located at the M4 combination screw in the upper left of the power switch, as shown in figure below.

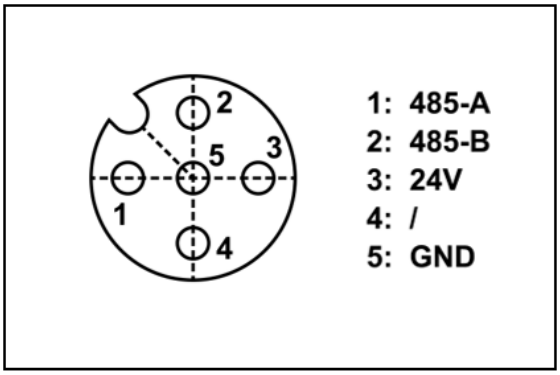
Figure 3.5-6 Demonstration diagram of the control box

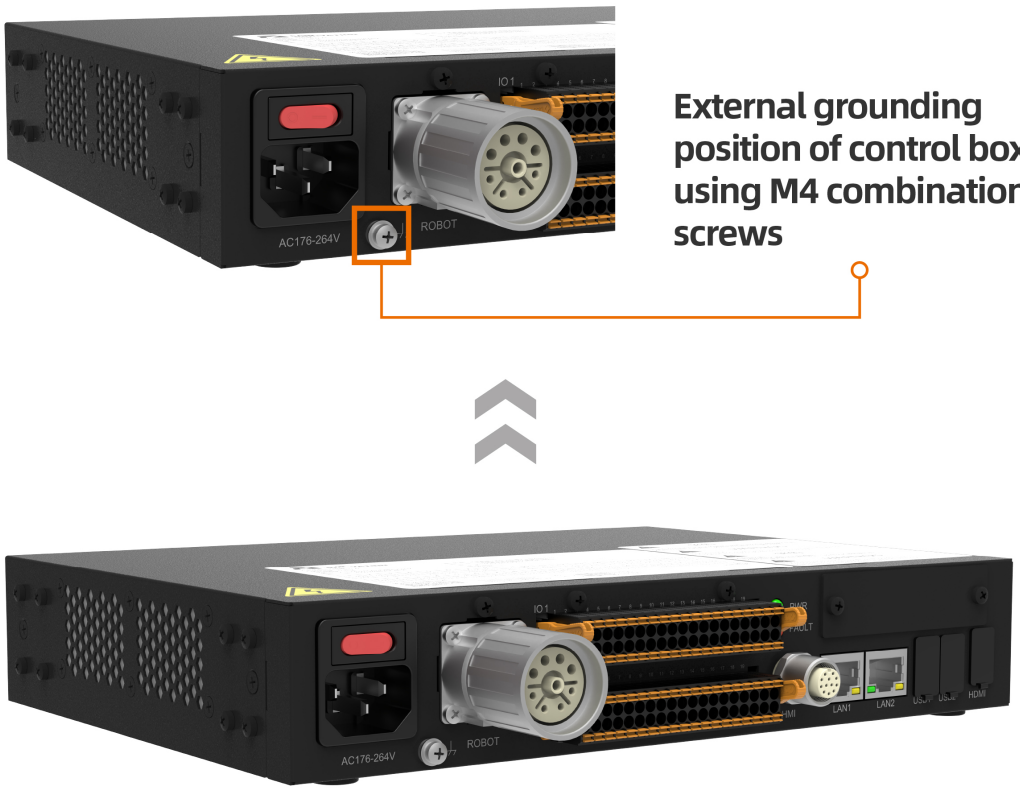
1. The body is located on the right side of the base of the base, as shown in figure below.

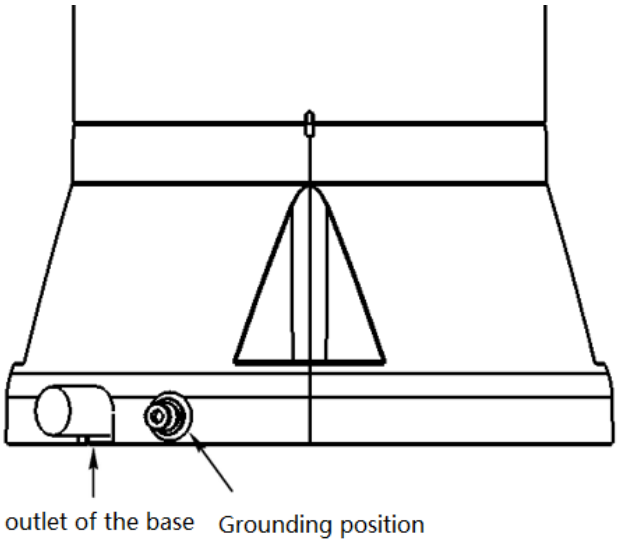
Figure 3.5-7 Dragon schematic diagram of the body

The protective wire used alone, the cross -sectional area should not be less than:


1: AI0
2: AO0
3: DI0
4: DI1
5: 24V
6: DO0
7: DO1
8: GND


1: 485-A
2: 485-B
3: 24V
4: /
5: GND


External grounding position of control box, using M4 combination screws
AC176-264V
ROBOT
IO 1
LAN1
LAN2
HDMI


outlet of the base
Grounding position

- 2.5mm^2 copper or 16mm^2 aluminumif mechanical injury protection is provided (wire pipe, pipeline, etc.)
- 4mm^2 copper or 16mm^2 aluminumif no mechanical damage protection is provided

1.3.3.5.6 The common specifications of all digital I/O

This section stipulates the electrical specifications of the following control box 24 volt digital input/output:

- Safety I/O
- Universal digital amount I/O

Robots must be installed in accordance with electrical specifications.

By configured the "Power Communication" interface, you can use the internal or external 24V power supply to power the digital I/O. The above two terminals (EX24V and EXON) in the interface are 24V and ground with external power supply, and the two terminals (24V and GND) below are 24V and land of internal power supply. The default configuration uses internal power, as shown in figure below.

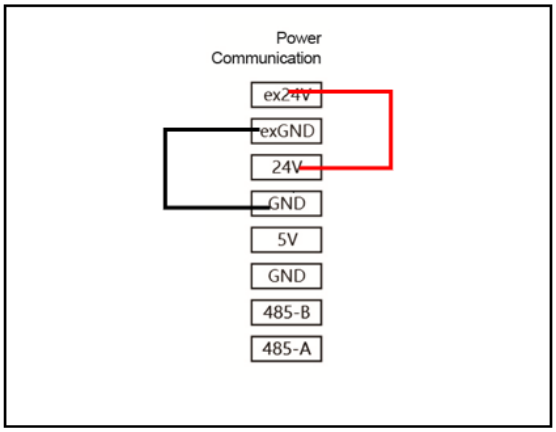


Figure 3.5-8 Power communication schematic diagram 01

If the load power is large, you can connect the external power supply as shown in figure below.

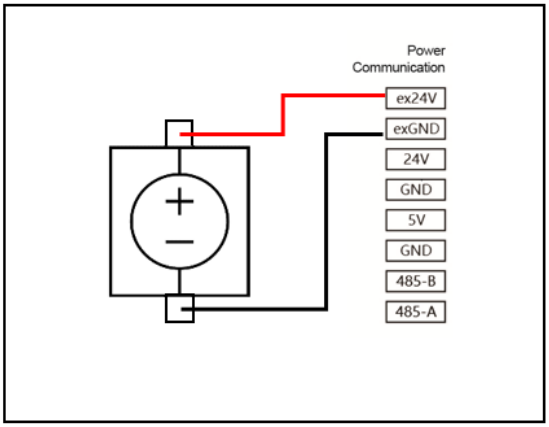


Figure 3.5-9 Power communication schematic diagram 02

The electrical specifications of internal and external power are shown in table below Internal and external electrical specifications:

Table 2.5-1 Internal and external power supply electrical specifications

Terminal	Parameter	Min-inum	Typ i-cal	Max-i-mum	Unit
Internal 24V power supply [ex24V -exGND] [ex24V -exGND]	Voltage Current	23 0	24 -	25 2	V A
Internal 24V power supply [24V- GND] [24V- GND]	Voltage Current	23 0	24 -	25 1.5	V A

The electrical specifications of digital I/O are shown in table below Digital I/O Electric Specifications:

Table 2.5-2 Digital I/O Electric Specification

Terminal	Parameter	Min-inum	Typ i-cal	Max-i-mum	Unit
Digital output [COx/DOx] [COx/DOx] [COx/DOx]	Current Pressure drop Leakage current	0 0 0	- - -	1 0.5 0.1	A V mA
[COx/DOx]	function	-	NPI	-	Type
Digital output [EIx/SIx/CIx/DIx] [EIx/SIx/CIx/DIx] [EIx/SIx/CIx/DIx]	OFF ON Current(11~30)	-3 11 2	- - -	5 30 15	V V mA
[EIx/SIx/CIx/DIx]	function	-	NPI	-	Type

1.3.3.5.7 Safety I/O

This section describes the electrical specifications of security I/O, and must abide by the general electrical specifications in Section 3.5.6.

Safety devices and equipment must be installed in accordance with the safety description and risk assessment, see section 3.1. All security I/O is paired (redundant) and must be stored as two independent branches. Single failures should not cause loss of security function.

Safety I/O includes emergency stop and security stop. Urgent stop input is only used for emergency stop equipment, and safely stops input for various security -related protection equipment. Functional differences are shown in table below:

Table 2.5-3 Functional difference

	Emergency stop	**Safe stop**
Robot stops moving	Yes	Yes
Stop Category	Category 0	Category 1
Program execution	Stop	Pause
Robot power supply	Close	Open
Restart	Manual	Automatic or manual
Frequency of use	Infrequent	Often
Reinitialization required	Need	Needless

Warning:

- Do not connect the security signal to a PLC that does not have the correct and safe level. If this warning does not follow, it may cause serious damage or death because one of the security stop function may be covered. Security interface signals must be separated from normal I/O interface signals.
- All I/O is a redundant -related I/O built (two independent channels). Two channels must be kept separately so that a single failure will not cause security function.
- Before the robot is put into operation, it is necessary to verify the emergency stop safety function (the robot is powered on, press the emergency stop button, the robot is disconnected, the power is turned off, the rotating emergency stop button, the power is turned on, and the robot is re -power to enable it). Safety functions must be tested regularly.
- Robot installation should comply with these specifications. Otherwise, it may lead to severe damage or death, because the safety stop function may be over.

The following sections are given some examples of how to use security I/O.

Default safety configuration When the robot leaves the factory, it has the default configuration. It can be operated without any additional safety devices. Please refer to figure below.

Figure 3.5-10 Safety protection schematic diagram 01

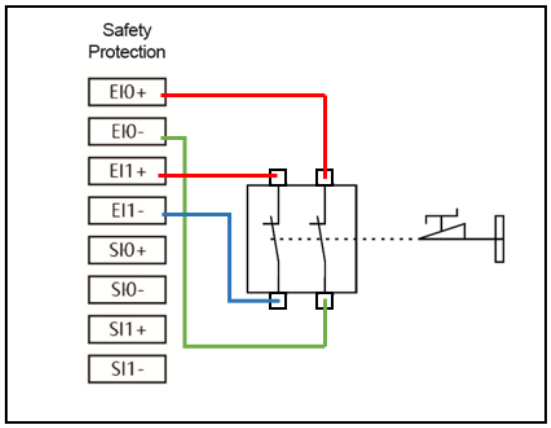
Connect the emergency stop button In most applications, one or more additional emergency stop buttons need to be used. Please refer to figure below.

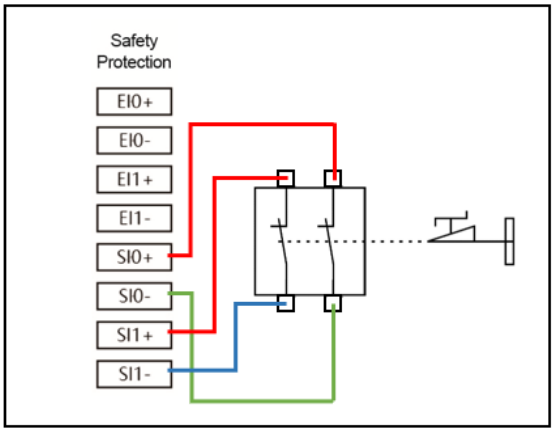
Figure 3.5-11 Safety protection schematic diagram 02

Connect the security stop button An example of a safe stop device is the door switch that the robot stops when the door is turned on. Please refer to figure below.

Figure 3.5-12 Safety protection schematic diagram 03


Safety
Protection
EI0+
EI0-
EI1+
EI1-
SI0+
SI0-
SI1+
SI1-


Safety
Protection
EI0+
EI0-
EI1+
EI1-
SI0+
SI0-
SI1+
SI1-


Safety
Protection
EI0+
EI0-
EI1+
EI1-
SI0+
SI0-
SI1+
SI1-

1.3.3.5.8 Universal digital amount I/O

This section describes the electrical specifications of the general digital I/O, and must abide by the general electrical specifications in Section 3.5.6.

The general digital amount I/O can be used to drive relays, solenoid valves and other devices or interact with other PLCs.

Digital quantity output control load

This example demonstrates how to connect the digital quantity output to control the load, Please refer to figure below.

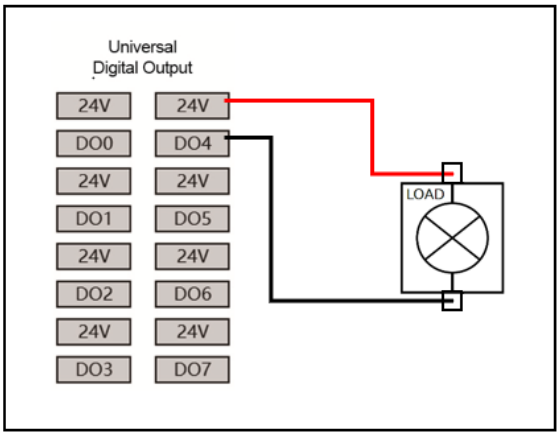


Figure 3.5-13 Great digital quantity output schematic diagram 01

1.3.3.5.9 Digital input from the button

The following example demonstrates how to connect the simple button to the digital quantity input.

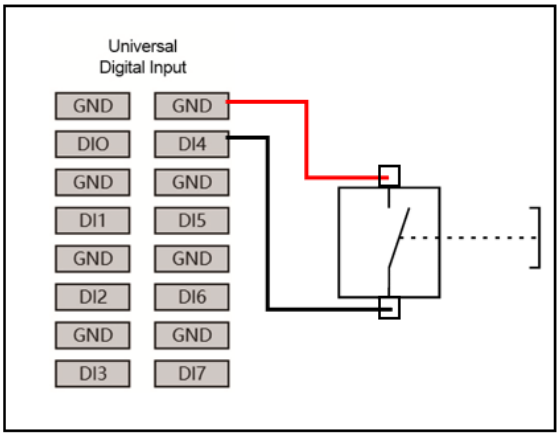


Figure 3.5-14 Great digital quantity output schematic diagram 02

1.3.3.5.10 Interact with other devices or PLC

The following example demonstrates how to interact with other devices or PLC digital input output.

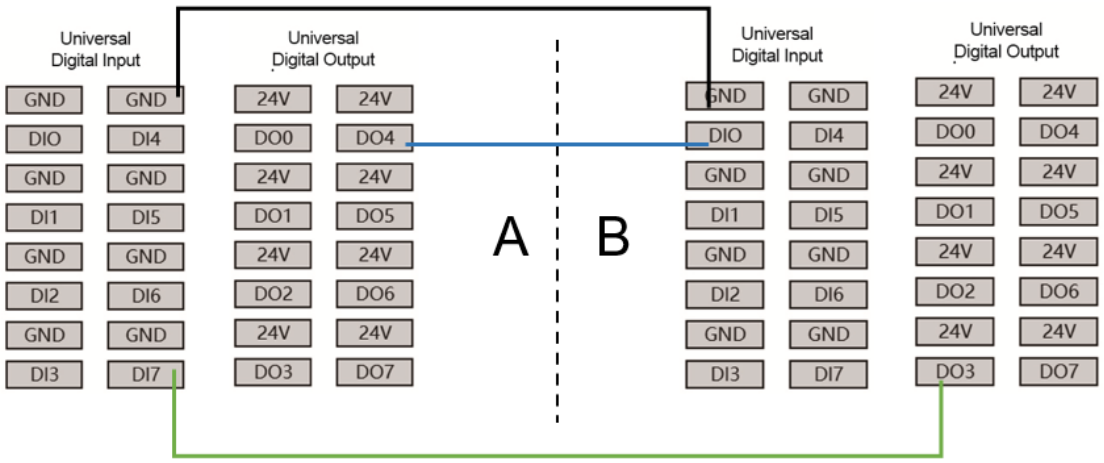


Figure 3.5-15 Interactive diagram with other devices or PLC

1.3.3.5.11 Simulation I/O

Table 2.5-4 Simulation current voltage

Terminal	Parameter	Min-inum	Typ i-cal	Max-i-mum	Unit
Analog current input [AIx-END] [AIx-END] [AIx-END]	Current Impedance Resolution	0 - -	- 500 12	20 - -	mA ohm bit
Analog voltage input [AIx-END] [AIx-END] [AIx-END]	Voltage Impedance Resolution	0 - -	- 510 12	10 - -	V Kohm bit
Analog current input [AOx-END] [AOx-END] [AOx-END]	Current Voltage Resolution	0 0 -	- - 12	20 10 -	mA V bit
Analog voltage input [AOx-END] [AOx-END] [AOx-END] [AOx-END]	Voltage Current Impedance Resolution	0 0 - -	- - 100 12	10 20 - -	V mA ohm bit

The simulation I/O is used to set or measure the voltage (0-10V) or current (0-20mA) of other devices.

In order to achieve high precision, the following methods are recommended.

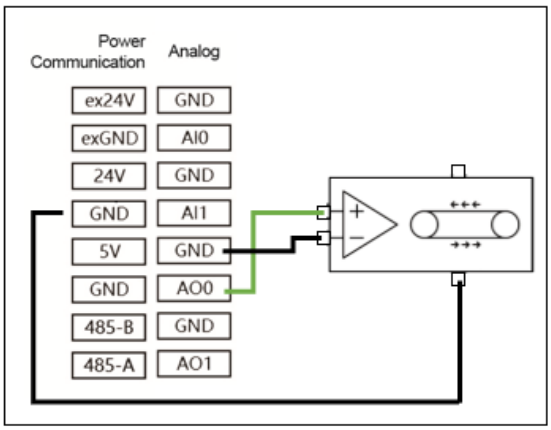
- The equipment and control box use the same ground (GND).
- Use shielding cables or twisted wires.

The following example demonstrates how to use analog I/O.

Use analog output

The following example is to demonstrate the use of analog output control conveyor belt.

Figure 3.5-16 Simulation output schematic diagram



Use analog input

The following example is to demonstrate the simulation input connection simulation sensor.

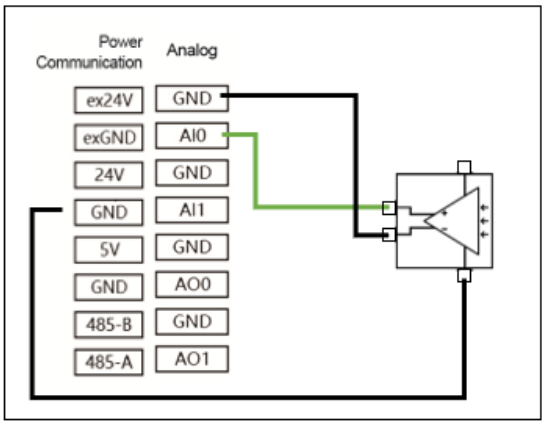


Figure 3.5-17 Simulation input schematic diagram

1.3.3.5.12 FR3MT&3C Optional Modules

1.3.3.5.12.1 Preface

The definition of collaborative robots complies with international ISO standards and national regulations to ensure operator safety. We do not recommend directly applying the robot body in scenarios where the operation target is the human body. However, if the robot user or developer has a specific need to involve human targets, they must conduct thorough evaluations and ensure personnel safety by configuring a reliable, fully tested, and certified safety protection system for the robot body.

1.3.3.5.12.2 Safety Notice

This manual serves only as a safety certification guide for customers. Maintenance operators must possess professional expertise. FaRo () assumes no responsibility for any incidents caused by non-professionals.

Important: If the robot (robot body, power module, or extension module) is damaged, altered, or modified due to human factors, FaRo refuses to bear any responsibility. FaRo is not liable for any damage caused to the robot or other equipment due to programming errors made by the customer.

1.3.3.5.12.3 Validity and Responsibility

The information in this manual does not cover the design, installation, or operation of a complete robot application, nor does it account for all peripheral devices that may affect the safety of the system. The design and installation of the complete system must comply with the safety requirements established by the standards and regulations of the country where the robot is installed.

FaRo's integrators are responsible for ensuring compliance with relevant national laws and regulations, guaranteeing that the complete robot application poses no significant hazards. This includes but is not limited to:

- Conducting a risk assessment for the complete robot system
- Connecting additional mechanical and safety devices as defined in the risk assessment
- Establishing appropriate safety settings in the software
- Ensuring users do not modify any safety measures
- Verifying the correct design and installation of the entire robot system
- Providing clear usage instructions
- Marking the integrator's logo and contact information on the robot
- Collecting all documentation, including this manual, in the technical file

1.3.3.5.12.4 Limited Liability

The safety information in this manual does not constitute a general safety guarantee for robots. Even if all safety instructions are followed, personnel injury or equipment damage may still occur.

1.3.3.5.12.5 Safety Warning Signs

The following safety warning signs are used on the product.

Important:



Name: **DANGER**

Purpose: Indicates an imminent electrical hazard that could result in death or serious injury if not avoided.

Important:



Name: **ELECTRIC SHOCK HAZARD**

Purpose: Indicates an imminent electric shock hazard that could result in death or serious injury if not avoided.

Important:



Name: **BURN HAZARD**

Purpose: Indicates a hot surface that could cause injury if touched.

Important:



Name: **GROUNDING**

Purpose: Indicates that the device must be properly grounded.

1.3.3.5.12.6 FR3MT&3C Base and Module Interface Definitions

1.3.3.5.12.7 Base Interface Definitions

The base of the robot has seven external buttons and interfaces, defined as follows:

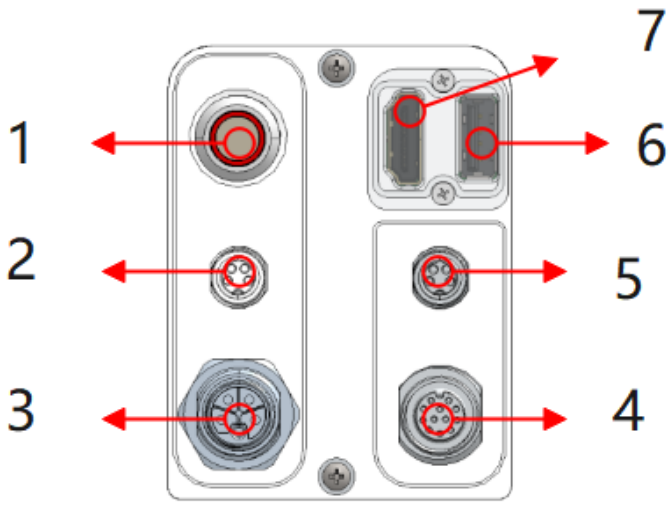
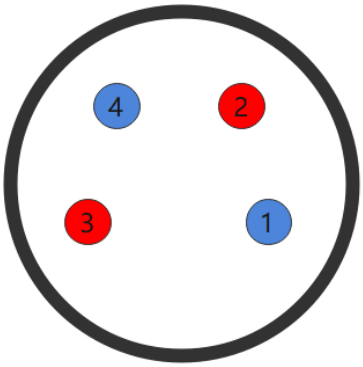


Figure 3.5-18 Base Buttons and Interfaces

Note: The pin definitions for base interfaces are viewed from the installation reference plane.

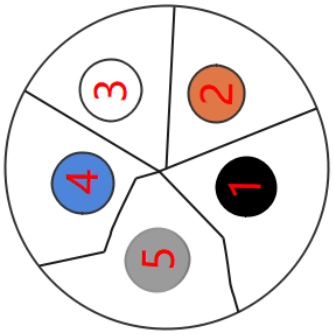
1. Controller Power Button: Powers on automatically by default.

2. M8-A Type-4P-Female Socket Pin Definitions: User Ethernet port (IP: 192.168.57.2). Connector: M8-A Type-4P-Female (with M8-A Type-4P-Male plug). Complies with IEC 61076-2-101.



Pin	Definition	Description
1	TX+	Data Transmit Positive
2	RX+	Data Receive Positive
3	RX-	Data Receive Negative
4	TX-	Data Transmit Negative

3. M12-L Type-5P-Male Socket Pin Definitions: Connector: M12-L Type-5P-Male (with M12-L Type-5P-Female plug). Complies with IEC 61076-2-101.



Pin	Color	Definition	Description	Remarks
1	Black	0V	Control Power Negative	Robot control power negative (backup power, no connection needed)
2	Brown	24V	Control Power Positive	Robot control power positive (backup power, no connection needed)
3	White	48V	Power Supply Positive	Robot power supply positive
4	Blue	0V	Power Supply Negative	Robot power supply negative
5	Gray	PE	Ground	Safety ground

Note: The base includes a 48V-to-24V power converter. The 48V-to-24V converter serves as a backup for the 24V input.

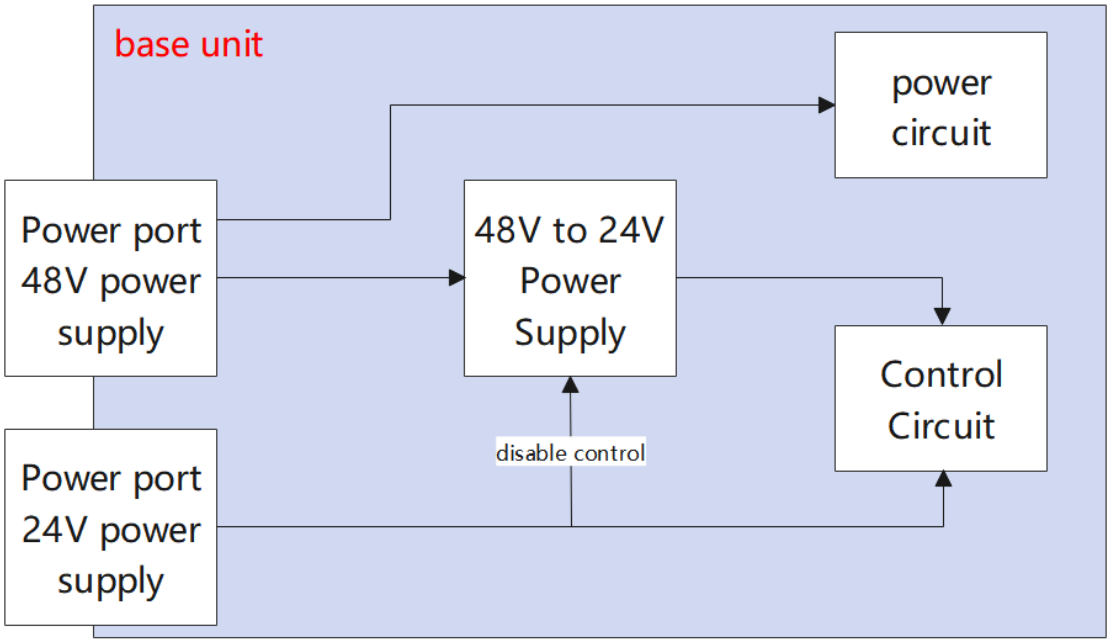
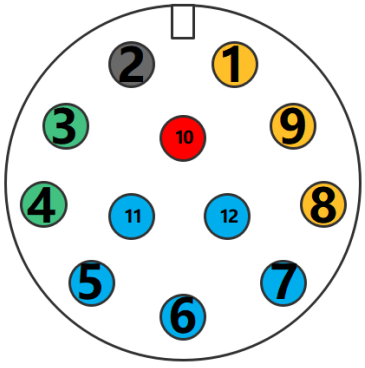


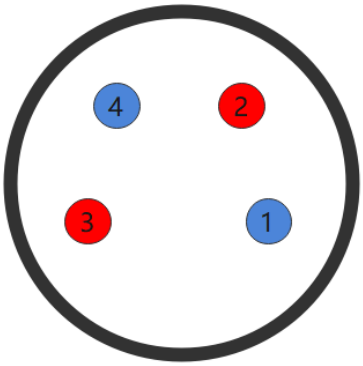
Figure 3.5-19 48V-to-24V Power Conversion Diagram

4. M12-A Type-12P-Female Socket Pin Definitions: Connector: M12-A Type-12P-Female (with M12-A Type-12P-Male plug). Complies with IEC 61076-2-101.



Pin	Color	Definition	Description	Remarks
1	Blue	AGND	Analog Ground	Analog reference ground
2	Brown	0V	24V Power Negative	Control power negative
3	Red	485-A	RS485 Communication A	Reserved for expansion
4	Gray	485-B	RS485 Communication B	Reserved for expansion
5	Black	DI0/DO0	Digital Input/Output 0	Configurable as input or output (mutually exclusive)
6	Yellow	DI1/DO1	Digital Input/Output 1	Configurable as input or output (mutually exclusive)
7	Pink	DI2/DO2	Digital Input/Output 2	Configurable as input or output (mutually exclusive)
8	Dark Green	AI0/AO0	Analog Input/Output 0	Configurable as input or output (mutually exclusive)
9	White	AI1/AO1	Analog Input/Output 1	Configurable as input or output (mutually exclusive)
10	Purple	24V	24V Power Positive	Control power positive
11	Orange	DI3/DO3	Digital Input/Output 3	Configurable as input or output (mutually exclusive)
12	Light Green	DI4/DO4	Digital Input/Output 4	Configurable as input or output (mutually exclusive)

5. M8-A Type-4P-Female Socket Pin Definitions: Debug Ethernet port (IP: 192.168.58.2). Connector: M8-A Type-4P-Female (with M8-A Type-4P-Male plug). Complies with IEC 61076-2-101.



Pin	Definition	Description
1	TX+	Data Transmit Positive
2	RX+	Data Receive Positive
3	RX-	Data Receive Negative
4	TX-	Data Transmit Negative

6. USB-A Port: USB 2.0 for internal debugging. **7. HDMI-A Port**: HDMI display for internal debugging.

1.3.3.5.12.8 Power Module Interface Definitions

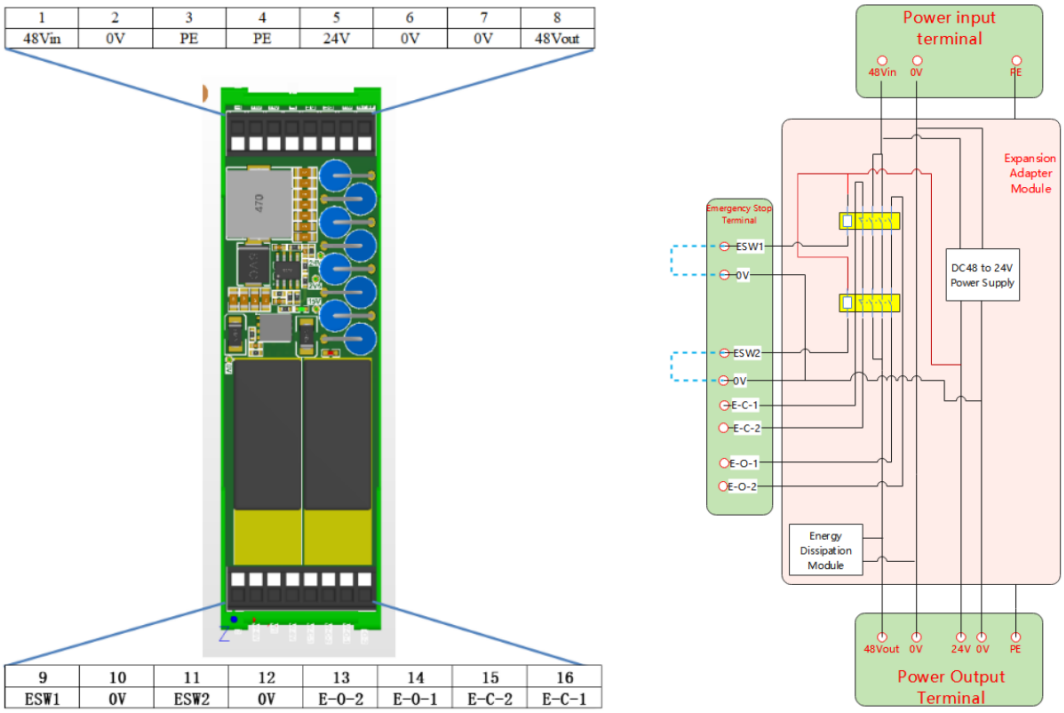
The power module uses a Meanwell NDR-480-48. Pin definitions:



Pin	Definition	Description	Remarks
1	L	Live Wire	Input 100-240V AC
2	N	Neutral Wire	Input 100-240V AC
3	PE	Ground	Earth connection
4	+V	48V	Output 48V/10A
5	+V	48V	Output 48V/10A
6	-V	0V	Output 48V return
7	-V	0V	Output 48V return

1.3.3.5.12.9 Extension Module Interface Definitions

The extension module includes emergency stop and energy discharge functions. External terminals and internal topology:



Pin	Definition	Description
1	48-IN	48V Input Positive
2	0V	48V Input Negative
3	PE	Ground
4	PE	Ground
5	24V	Control Power Positive
6	0V	Control Power Negative
7	0V	Power Supply Negative
8	48-OUT	Power Supply Positive
9	ESW1	Emergency Stop Button 1 Positive
10	0V	Emergency Stop Button 1 Negative
11	ESW2	Emergency Stop Button 2 Positive
12	0V	Emergency Stop Button 2 Negative
13	E-O-2	Passive Normally Open 2
14	E-O-1	Passive Normally Open 1
15	E-C-2	Passive Normally Closed 2
16	E-C-1	Passive Normally Closed 1

1.3.3.5.12.10 FR3MT&3C Application Scenarios

Most scenarios only require the user cable kit. Specific use cases:

No.	Scenario	Power Supply	Functional Requirements	Recommended Configuration
1	Basic Application	48V/10A DC available	No E-stop/energy discharge	User cable kit
2	Safety Extended	48V/10A DC available	E-stop + energy discharge	User cable kit + extension module
3	Independent Power	No 48V/10A DC	No E-stop/energy discharge	User cable kit + power module + power cord
4	Full Function	No 48V/10A DC	E-stop + energy discharge	User cable kit + power module + power cord + extension module

1.3.3.5.12.11 Basic Application

Only user cable kit required. Connection steps:

1. Connect the M12-L-5P-Female power cable to the base (48V/0V/PE to user power supply; insulate 24V/0V).
2. Connect M12-A-12P-Male and M8-A-4P-Male to base.

Figure 3.5-20 Only user cable kit required connection steps

1.3.3.5.12.12 Safety Extended

User cable kit + extension module. Connection steps:

1. Connect the 0.5M extension cable (48V/0V/PE) to user power and module.
2. Connect M12-L-5P-Female to base (all 5 wires to module).
3. Connect M12-A-12P-Male and M8-A-4P-Male to base.

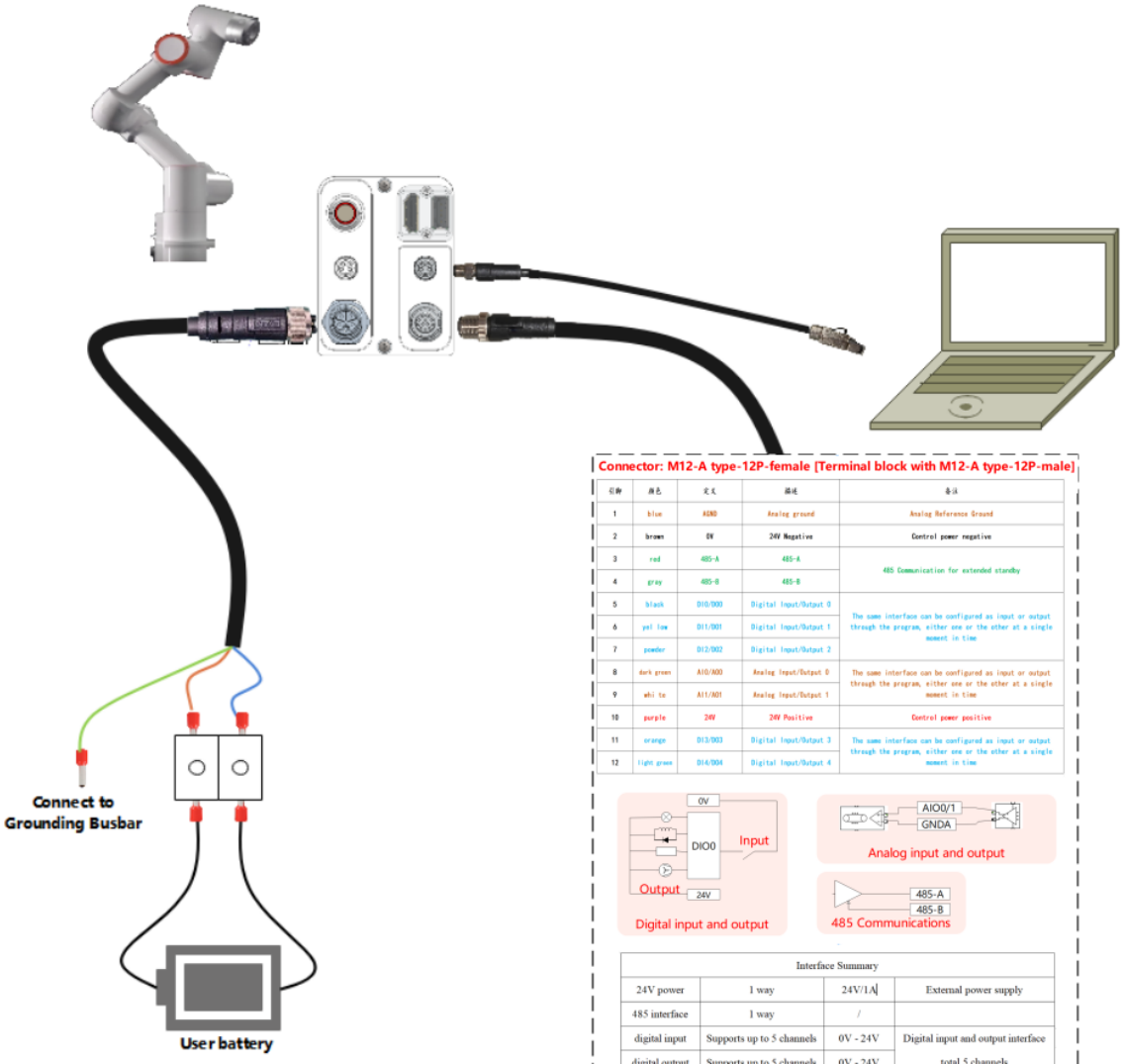
Figure 3.5-21 User cable kit + extension module connection steps

1.3.3.5.12.13 Independent Power

User cable kit + power module + power cord. Connection steps:

1. Connect 1.5M power cord (L/N/PE) to NDR-480-48 input.
2. Connect M12-L-5P-Female to base (48V/0V/PE to power module; insulate 24V/0V).
3. Connect M12-A-12P-Male and M8-A-4P-Male to base.

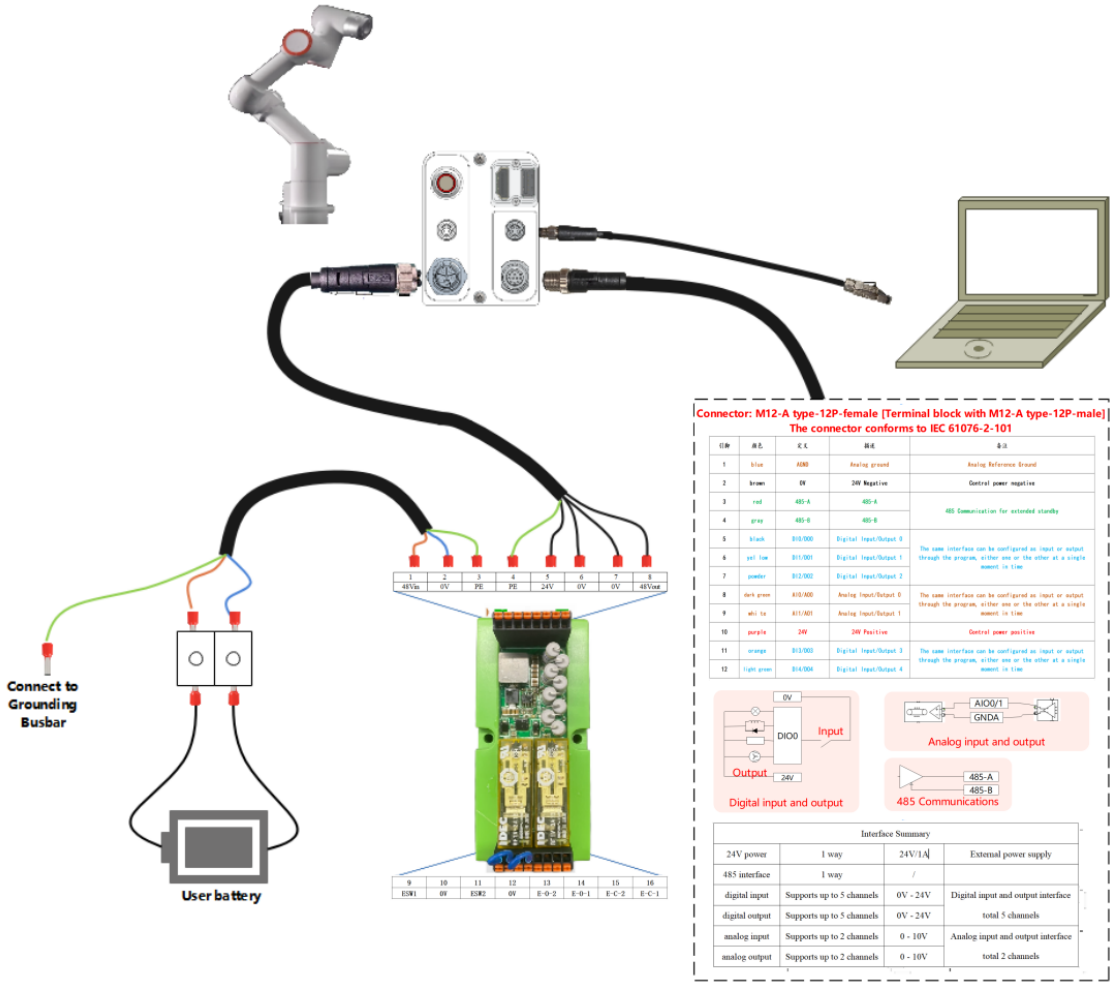
Figure 3.5-22 User cable kit + power module + power cord connection steps



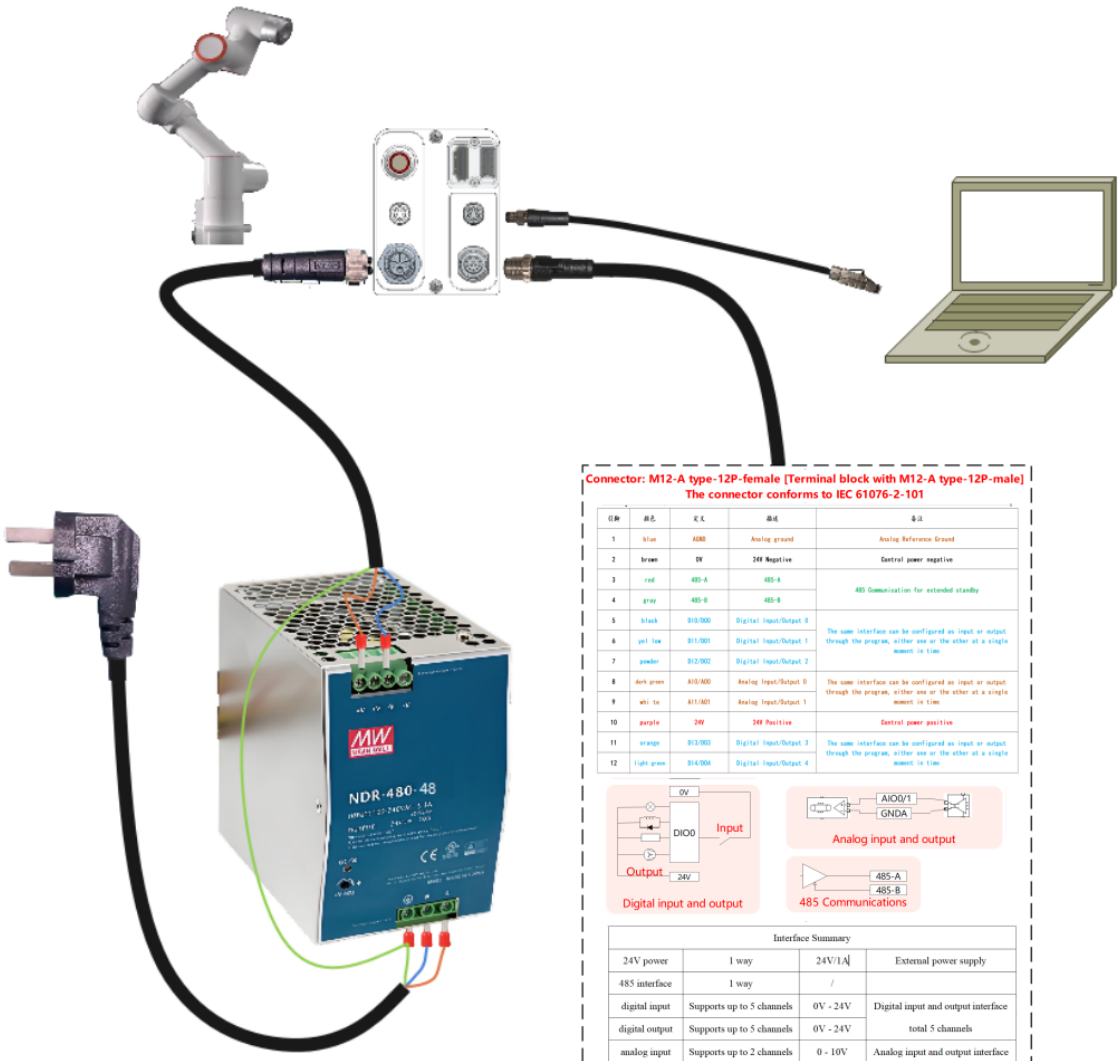
Connector: M12-A type-12P-female [Terminal block with M12-A type-12P-male]

1	blue	AGND	Analog ground	Analog Reference Ground
2	brown	0V	24V Negative	Control power negative
3	red	485-A	485-A	485 Communication for extended standby
4	gray	485-B	485-B	
5	black	DI0/DO0	Digital Input/Output 0	The same interface can be configured as input or output through the program, either one or the other at a single moment in time
6	yellow	DI1/DO1	Digital Input/Output 1	
7	powder	DI2/DO2	Digital Input/Output 2	
8	dark green	AI0/AO0	Analog Input/Output 0	The same interface can be configured as input or output through the program, either one or the other at a single moment in time
9	white	AI1/AO1	Analog Input/Output 1	
10	purple	24V	24V Positive	Control power positive
11	orange	DI3/DO3	Digital Input/Output 3	The same interface can be configured as input or output through the program, either one or the other at a single moment in time
12	light green	DI4/DO4	Digital Input/Output 4	

Interface Summary			
24V power	1 way	24V/1A	External power supply
485 interface	1 way	/	
digital input	Supports up to 5 channels	0V - 24V	Digital input and output interface
digital output	Supports up to 5 channels	0V - 24V	total 5 channels
analog input	Supports up to 2 channels	0 - 10V	Analog input and output interface
analog output	Supports up to 2 channels	0 - 10V	total 2 channels



Interface Summary			
24V power	1 way	24V/1A	External power supply
485 interface	1 way	/	
digital input	Supports up to 5 channels	0V - 24V	Digital input and output interface
digital output	Supports up to 5 channels	0V - 24V	total 5 channels
analog input	Supports up to 2 channels	0 - 10V	Analog input and output interface
analog output	Supports up to 2 channels	0 - 10V	total 2 channels



Connector: M12-A type-12P-female [Terminal block with M12-A type-12P-male]
The connector conforms to IEC 61076-2-101

引脚	颜色	定义	描述	备注
1	blue	AGND	Analog ground	Analog Reference Ground
2	brown	0V	24V Negative	Control power negative
3	red	485-A	485-A	485 Communication for extended standby
4	gray	485-B	485-B	
5	black	DI0/DO0	Digital Input/Output 0	The same interface can be configured as input or output through the program, either one or the other at a single moment in time
6	yel low	DI1/DO1	Digital Input/Output 1	
7	powder	DI2/DO2	Digital Input/Output 2	
8	dark green	AI0/AO0	Analog Input/Output 0	The same interface can be configured as input or output through the program, either one or the other at a single moment in time
9	whi te	AI1/AO1	Analog Input/Output 1	
10	purple	24V	24V Positive	Control power positive
11	orange	DI3/DO3	Digital Input/Output 3	The same interface can be configured as input or output through the program, either one or the other at a single moment in time
12	light green	DI4/DO4	Digital Input/Output 4	

Digital input and output

Analog input and output

485 Communications

Interface Summary			
24V power	1 way	24V/1A	External power supply
485 interface	1 way	/	
digital input	Supports up to 5 channels	0V - 24V	Digital input and output interface total 5 channels
digital output	Supports up to 5 channels	0V - 24V	
analog input	Supports up to 2 channels	0 - 10V	Analog input and output interface total 2 channels
analog output	Supports up to 2 channels	0 - 10V	

1.3.3.5.12.14 Full Function

User cable kit + power module + power cord + extension module. Connection steps:

1. Connect 1.5M power cord (L/N/PE) to NDR-480-48.
2. Connect 0.5M extension cable (48V/0V/PE) between power module and extension.
3. Connect M12-L-5P-Female to base (all 5 wires to module).
4. Connect M12-A-12P-Male and M8-A-4P-Male to base.

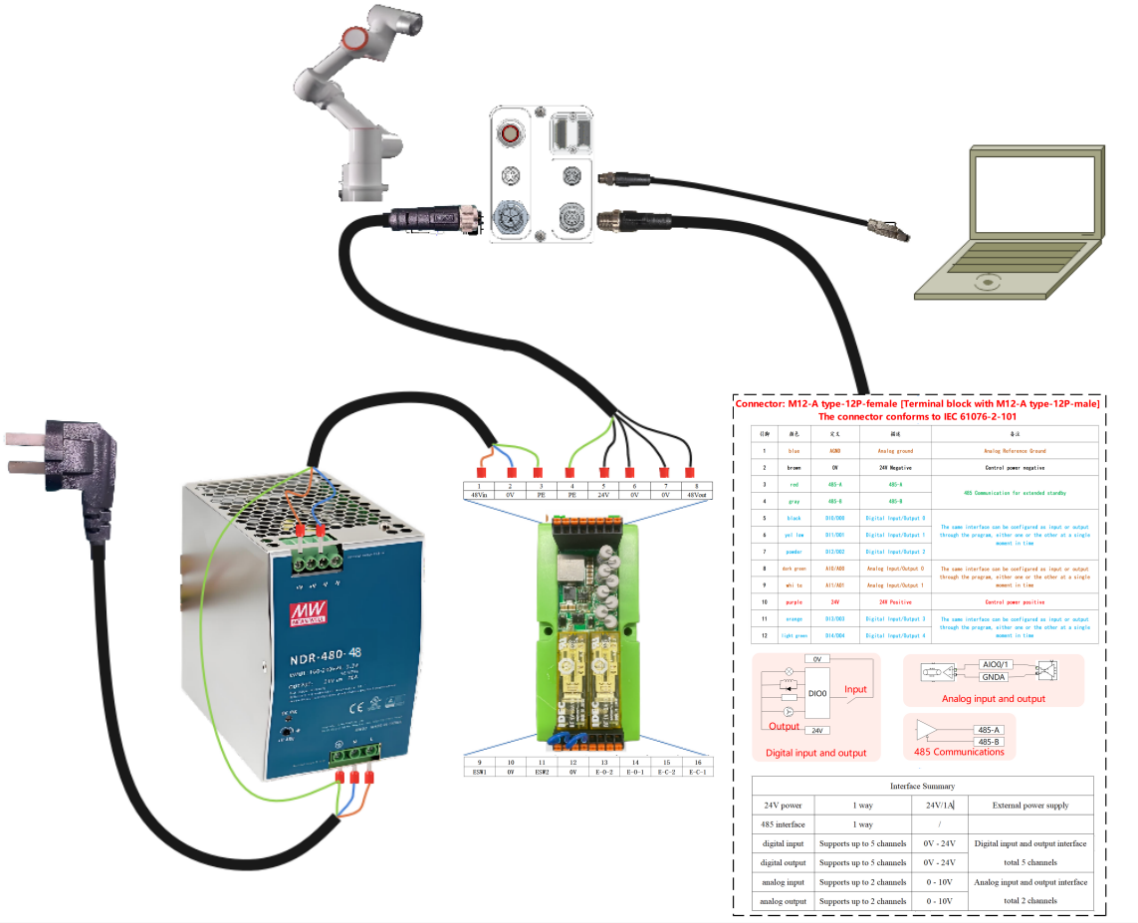


Figure 3.5-23 User cable kit + power module + power cord + extension module connection steps

1.3.3.5.12.15 Optional Parts List

User Cable Kit (5M):

No.	Name	Qty
1	FR3MT&3C-DC Power Cable-5M	1
2	FR3MT&3C-I/O Cable-5M	1
3	FR3MT&3C-Ethernet Cable-5M	1
4	M8 Straight Plug, M8-P4A-PLA05, 4-pin	1

User Cable Kit (1M):

No.	Name	Qty
1	FR3MT&3C-DC Power Cable-1M	1
2	FR3MT&3C-I/O Cable-1M	1
3	FR3MT&3C-Ethernet Cable-1M	1
4	M8 Straight Plug, M8-P4A-PLA05, 4-pin	1

Power Module:

No.	Name	Qty
1	Meanwell Power Supply, NDR-480-48	1

Power Cord:

No.	Name	Qty
1	FR3MT&3C-Power Cord-1.5M	1

Extension Module:

No.	Name	Qty
1	FR3MT&3C Base-Extension Module	1
2	FR3MT&3C Power-Extension Cable-0.5M	1

1.3.3.6 Demonstrate and end LED

The robotic teaching pendant can use a computer or tablet to access and control the robot. The connection method can refer to Section 3.5.3 to explain. In addition, users can also use our FR-HMI osteter.

1.3.3.6.1 Introduction to the button box

1.3.3.6.1.1 60 Button Box (POE) (BX01)

Figure 2.6-1 60 Button Box (POE) (BX01)

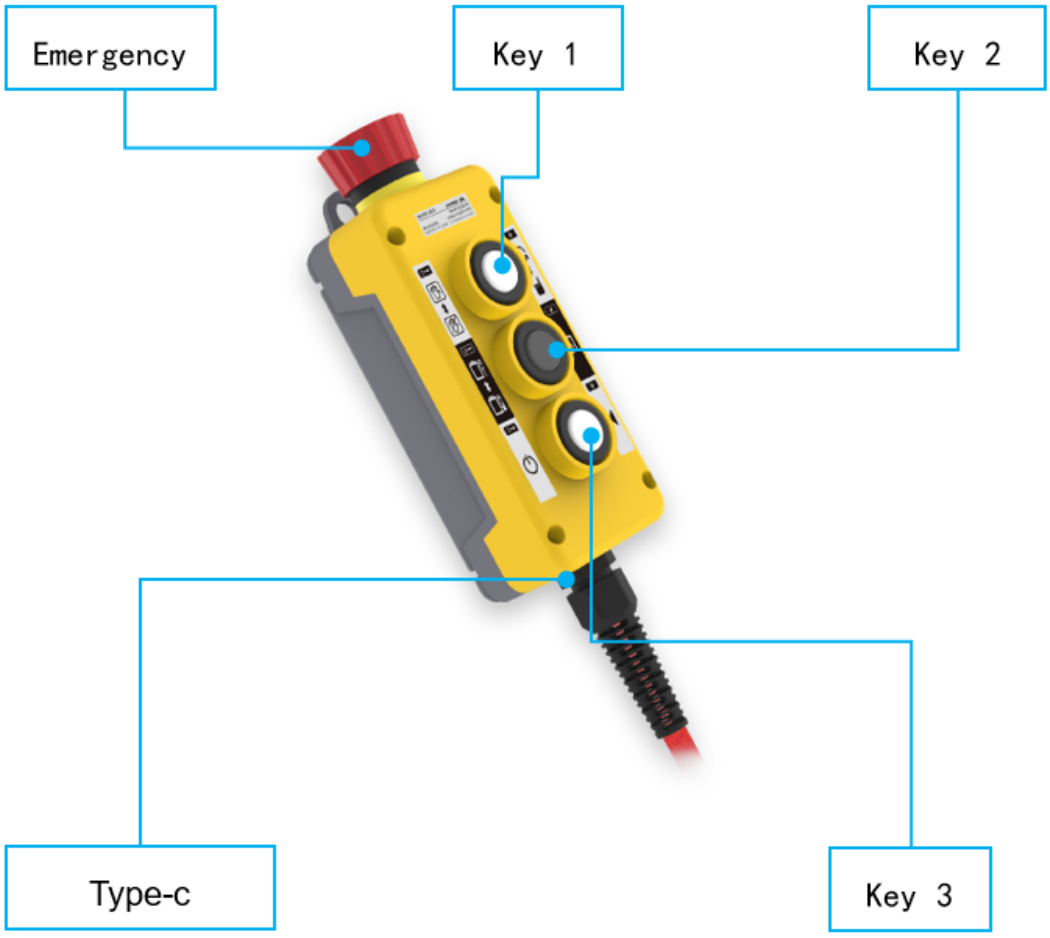
Emergency stop switch: When pressing the emergency stop switch, the robot enters the state of emergency stop.

Type-c: Connect the port of the web teaching pendant.

Button 1: Short press automatic/manual mode switch, long press and enter/exit the drag mode.

Button 2: Short press the record to show the teaching point, long press and enter/exit the state of no demonstrator.

Button 3: Start/stop running program.


Emergency
Key 1
Key 2
Type-c
Key 3

1.3.3.6.1.2 60 Button Box (POE) (BX02)-V1.0

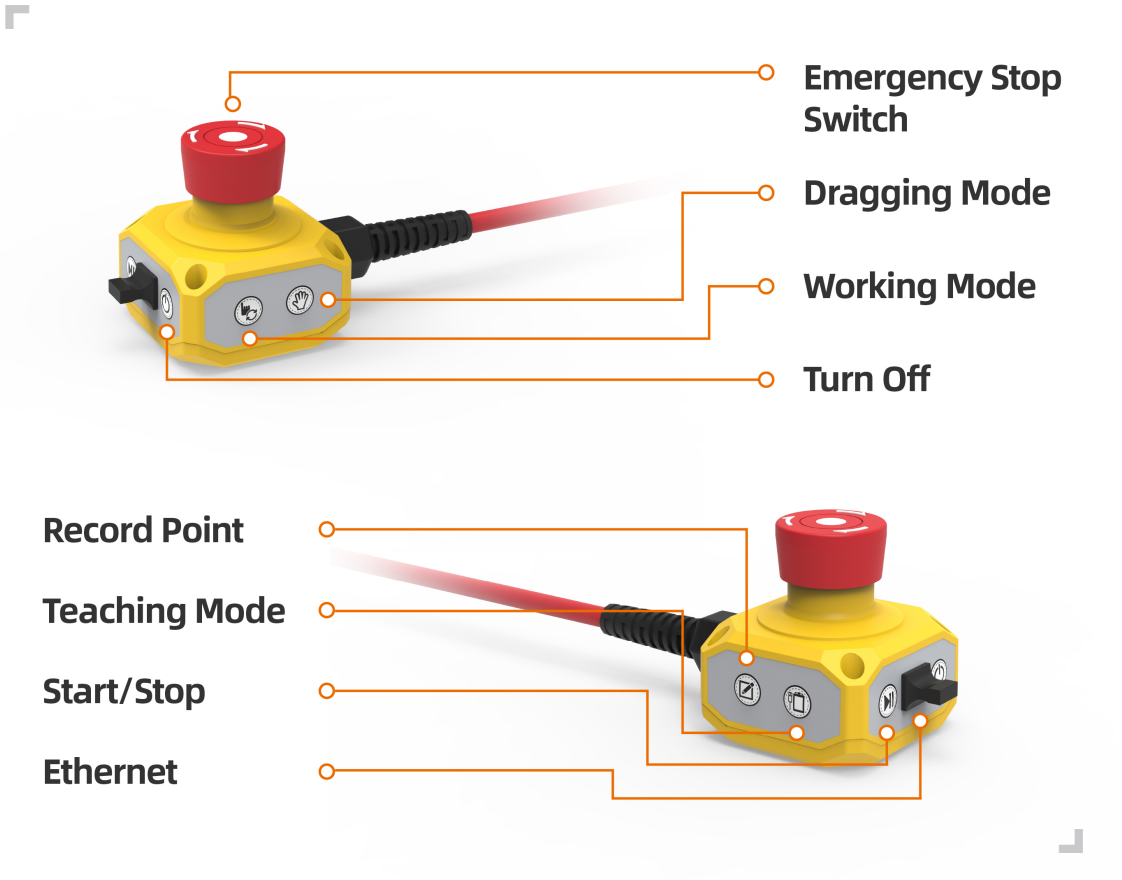


Figure 2.6-2 60 Button Box (POE) (BX02)-V1.0

Emergency stop switch: When pressing the emergency stop switch, the robot enters the state of emergency stop.

Start/Stop: Start/stop running program.

Ethernet: Connect to the web teaching pendant.

Turn off: No enabled.

Record point: Record the teaching point.

Teaching mode: Enter/exit with the teaching pendant state.

Working mode: Automatic/manual mode switch.

Drag mode: Enter/exit drag mode.

1.3.3.6.1.3 60 Button Box (POE) (BX02) - V2.0

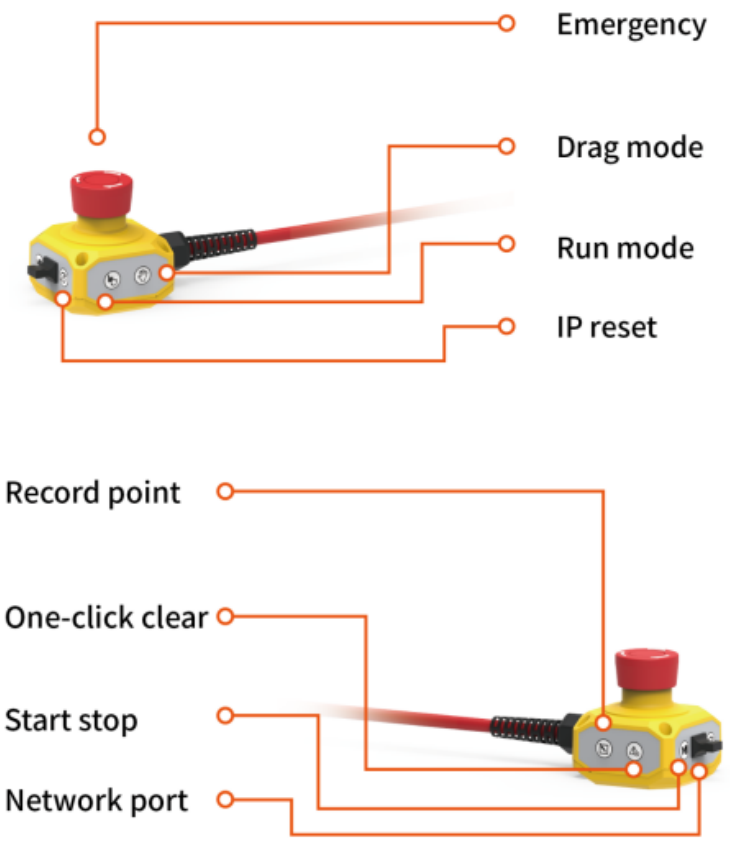


Figure 3.6-3 60 Button Box (POE) (BX02) - V2.0

Emergency stop switch: When the emergency stop switch is pressed, the robot enters an emergency stop state.

Start/Stop: Start/Stop running the program.

Network port: Connect to a web teaching pendant.

IP reset: Reset the network port IP.

Record point: Record teaching points.

One click clear: Clear all recoverable errors.

Operation mode: Automatic/Manual mode switching.

Drag mode: Enter/exit drag mode.

1.3.3.6.2 FR-HMI Teach pendant introduction

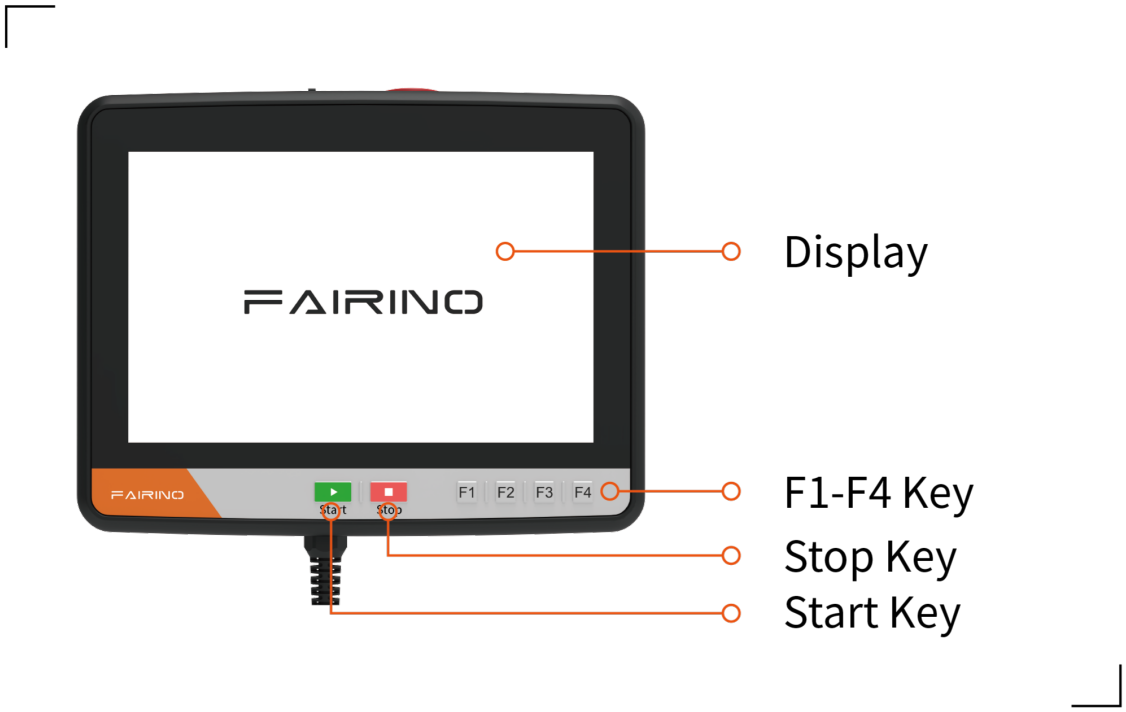


Figure 2.6-3 FR-HMI teaching pendant front

Figure 2.6-4 FR-HMI teaching pendant back

Display: Touch operation and display interface of the teaching pendant.

Start key: Start the program.

Stop key: Stop the currently running program.

Joint button: The joint node of the robot.

Three -bit enable: Manual mode enable robots

Emergency stop switch: When pressing the emergency stop switch, the robot enters the state of emergency stop.

Mode key: Rotate the button to switch the automatic mode.

1.3.3.6.3 End LED definition

Table 3.6-1 The end LED definition table



Function	LED color
When communication is not established	“Off”, “Red”, “Green” and “Blue” alternately
Automatic mode	Blue long bright
Manual mode	Green long bright
Drag Mode	White cyan long bright
Button box record point (only when using button box)	Purple blinks twice
Start running (only when using the button box)	Cyan blue flashes twice
Enter the state of unmatched button box (only when using the button box)	Blue flashes twice
Stop operation (only when using the button box)	Red flashes twice
Error reporting (only when using the button box)	Red long bright
Zero calibration completed	White cyan flashes three times
Enable	Yellow flashes twice

1.3.4 Quick start

1.3.4.1 Install the robot arm and control box

Install the robot arm and control box according to the 3.5 and 3.6 in the hardware installation of Chapter 3.

- Take out the robotic arm and use 4 M8 bolts with a strength of not less than 8.8 levels to install the robot arm. Install the robot arm on a strong and shock -proof surface. If it is fixed with aluminum plate, the thickness of the aluminum plate is not less than 16mm. If the iron plate is fixed, the thickness of the iron plate is not less than 8mm;
- Place the control box on its feet;
- Connect the robot arm ontology to the heavy load interface of the control box;

- Plug the button box aviation plug into the control box exemporary instrument interface.
- Make sure the power button of the control box is turned off (the button is turned to 0) and connect the power cord to the power socket;
- Plug in the power control box plug.

Warning:

(1) If the robot is placed on a strong surface, the robot may dump and cause damage.

(2) Do not quickly switch the power of the control box on and off. It is recommended that the time between switching the power of the control box off and on again be greater than 1 minute.

1.3.4.2 Demonstrate startup control robot

The control box is connected to the robot arm, the teaching box, and the physical electrical input/output terminal of any peripheral equipment. You must open the control box to power the robot's arm.

- Press the power button of the control box to open the control box;
- After starting the robot, at this time the robot is manual mode and has not enabled. If the robot needs to be operated in the manual mode, you need to press the three bits on the teaching pendant to switch off (let go) OFF (press) as the switch as a switch Drag or control the robot movement when you are on your state.
- If you do not need to operate the robot in the manual mode, you can use the key switch to switch the robot working mode on the key to switch the button: automatic, manual, custom;
- When switching the robot's manual state, check whether there is abnormal abnormality inside and outside the safety space and to operate the machine's operation carefully;
- When switching the robot's automatic state, check the safety measures and return to the normal state and operate the machine operation carefully;
- When the teaching pendant cannot be opened normally, please check whether the device connection is normal.

1.3.4.3 Button box control robot movement

Refer to the 3.6.3 of the hardware installation of Chapter 3. The end LED definition of the end to control the robot. The existing button boxes are divided into 60-button box (POE)(BX01), 60-button box (POE)(BX02)-V1.0, 60-button box (POE)(BX02)-V2.0.

Taking the 60-button box (POE)(BX01) as an example, the operation steps are as follows.

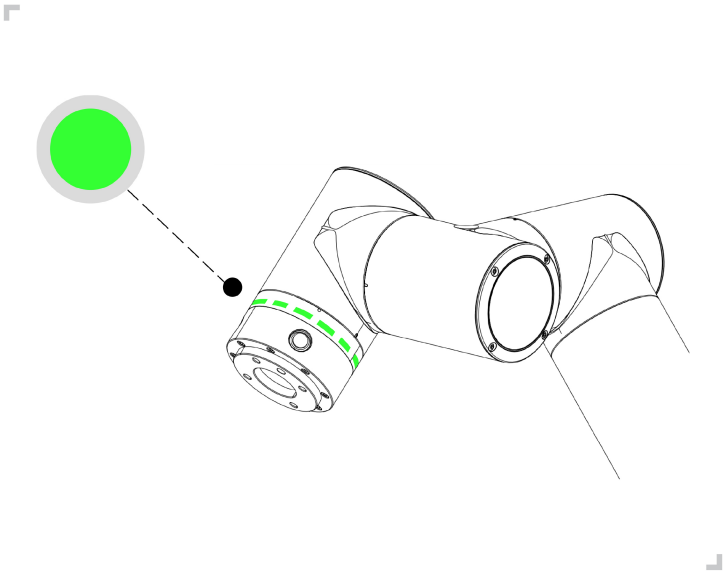
1.3.4.3.1 Not paired with teaching pendant

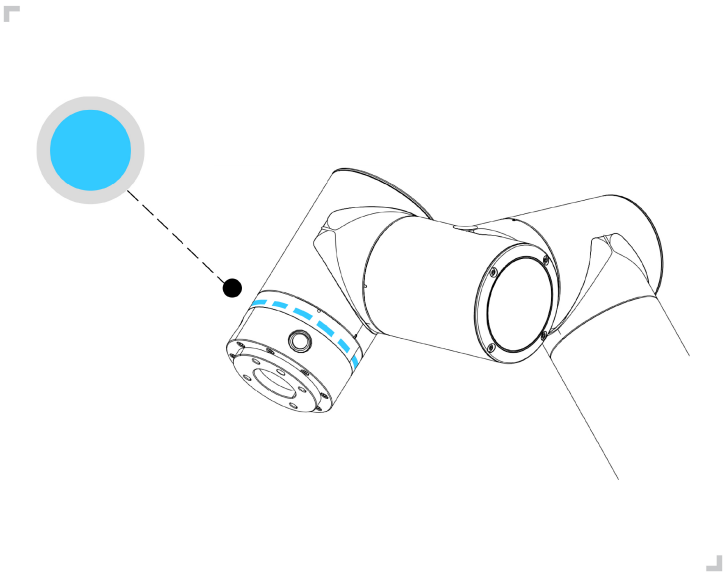
- **Step1**:Turn on the power switch of the robot control box, start the robot, wait for the end LED to display green for a long time before operating the robot, as shown in figure below.

Figure 4.3-1 Green Schematic Diagram of End LED

- **Step2**:Long press the button box "button 2" to enter the mode without teaching pendant, and the end LED flashes blue three times, as shown in figure below.

Figure 4.3-2 end LED blue blue schematic diagram





- **Step3**:Long press button “button 1” Switch the robot to the drag mode. At this time, the end LED is white and blue, as shown in Figure 4.3-3. To any position of the mobile robot, long press the “button 1” to exit the drag mode, the short-press button box “button 2” records P1 point, the end LED purple flashes three times, as shown in Figure 4.3-4.
- **Step4**:Mobile robot, short press button box “button 2” record P2 point, the end LED purple flashes three times, as shown in Figure 4.3-4.

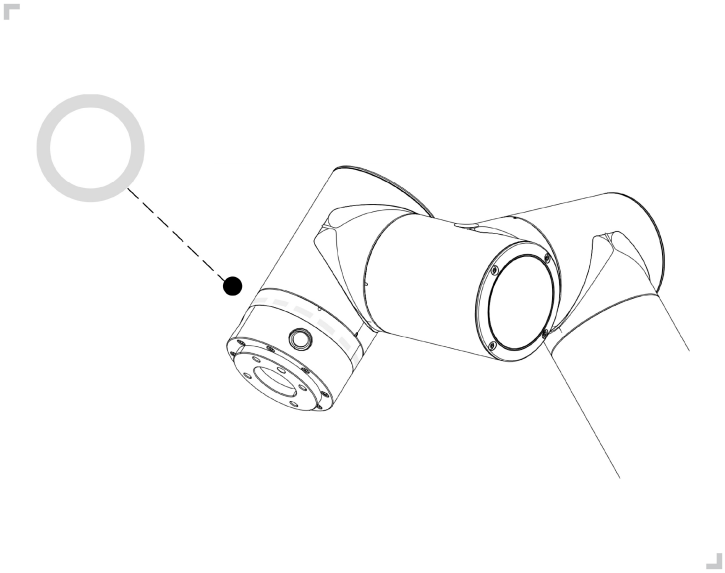


Figure 4.3-3 end LED white green schematic diagram

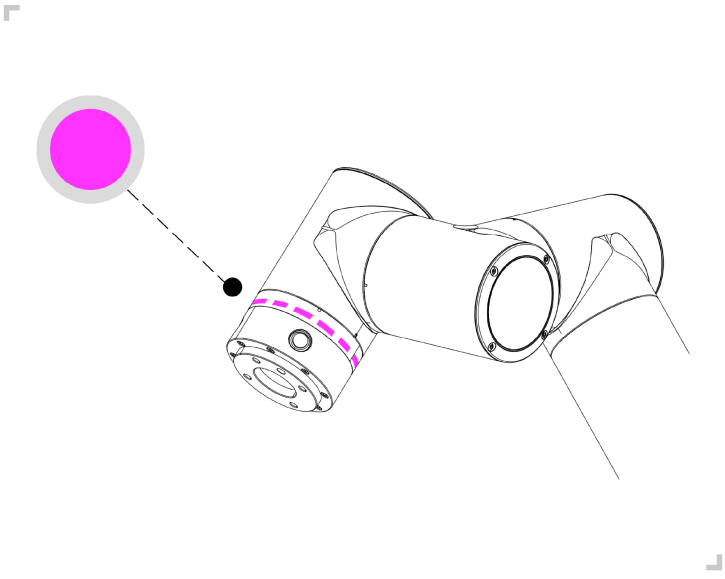


Figure 4.3-4 end LED purple schematic diagram

- **Step5**:The long press button “button 1” exit the drag mode. At this time, it is a manual mode. The end LED is green, as shown in Figure 4.3-5. Short press “Key 1” to switch the robot to the automatic mode. At this time, the end LED is blue, as shown in Figure 4.3-6.

- **Step6**:The short press button “button 3” runs the program, the end LED blue flashes twice, as shown in Figure 4.3-6.

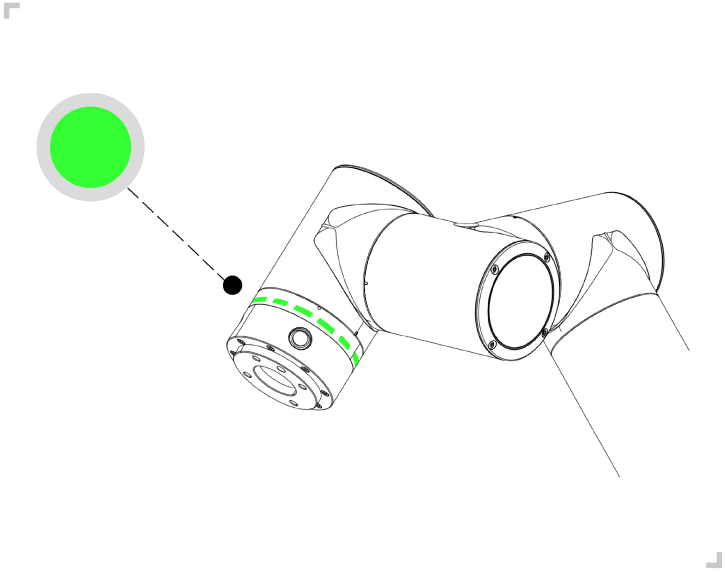


Figure 4.3-5 end LED green schematic diagram

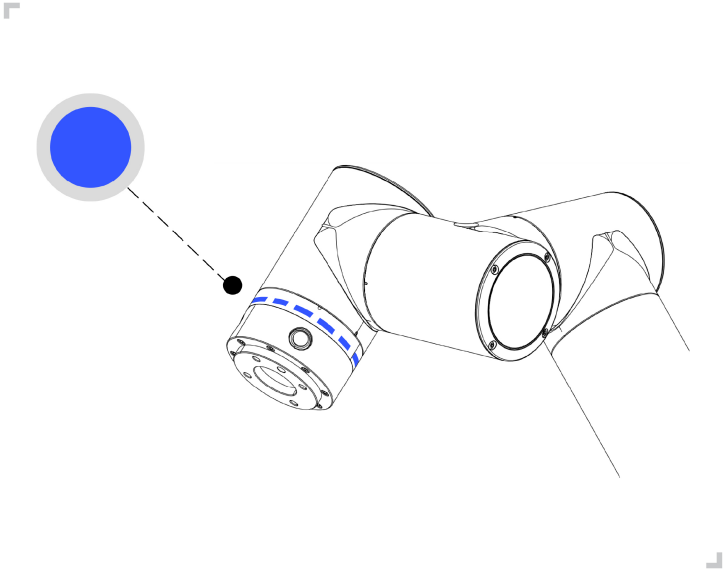
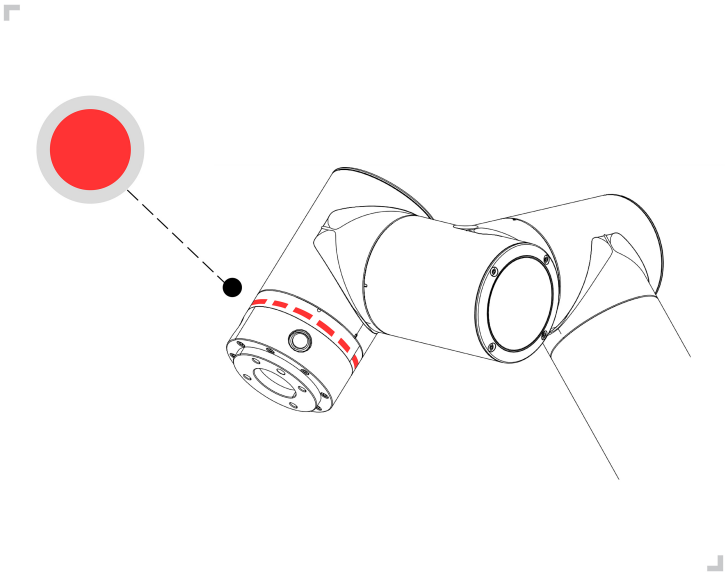


Figure 4.3-6 end LED blue schematic diagram

- **Step7**:Short press the button box “button 3” to stop running the program, and the red LED at the end flashes three times, as shown in figure below.

Figure 4.3-7 end LED red schematic diagram



1.3.4.3.2 Matching the teaching pendant

- **Step1**:Start the robot and wait for the end LED green to stop flashing before operating the robot.
- **Step2**:Open the teaching pendant and enter the program editing interface.
- **Step3**:Select the blank template to create a new program file.
- **Step4**:Short -press button box button 1 Switch the robot to the manual mode. At this time, the end LED is green.
- **Step5**:Long press the button box button 1 to switch the robot to the drag mode. At this time, the end LED is white and green, the mobile robot to any position, the short press button box button 2 records P1, the end LED purple flashes three times, manually add "PTP(p1,100,-1,0)"The instructions to the program file.

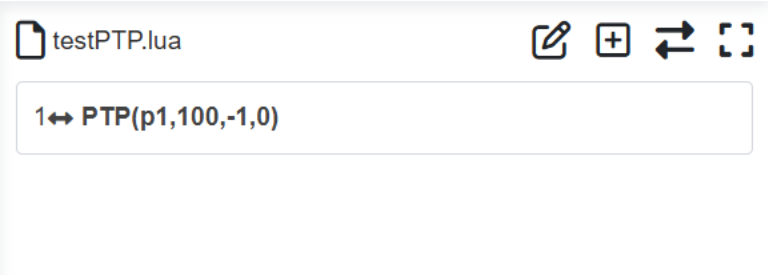
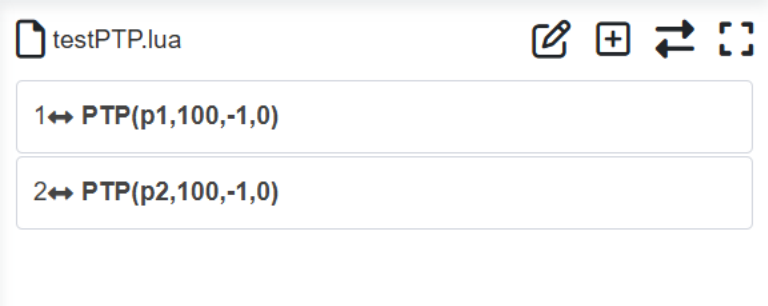


Figure 4.3-8 record and add some P1

- **Step6**:Mobile robot, short -press button box button 2 records P2 point, end LED purple flashes three times, and manually add the "PTP(p2,100,-1,0)" instruction to the program.

Figure 4.3-9 record and add some P2

- **Step7**:Save the program file content.
- **Step8**:Long press the button box button 1 to exit the drag mode. At this time, it is a manual mode. The end LED is green. The short press button box button 1 switch the robot to the automatic mode. At this time, the end LED is blue.
- **Step9**:Short pressing button box button 3 Run the program, the end LED blue flashes twice.



1.3.4.4 Teaching pendant control robot movement

Click the "Program" button in the first-level menu on the left side of the teaching pendant, and click the sub -menu "Coding" to enter the program demonstration interface. The interface mainly implements the writing and modification of the robot teaching program.

After clicking the "New" icon button, the user names the file and selects a template as the content of the new file. Click the newly created to create a successful and open the program file.

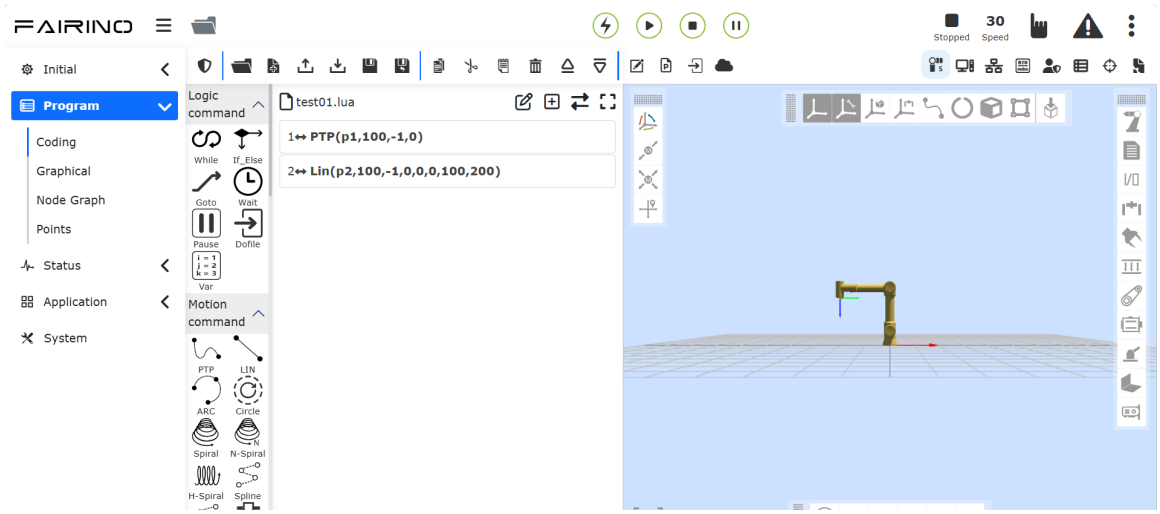


Figure 4.4-1 Demonstration of teaching program running schematic diagram

Warning: Your head and trunk cannot be located in the range (work area) that the robot can reach. Please do not put your fingers in the place where the robot can grasp.

Important:

- Don't let the robot move to yourself or other objects, because it will cause damage to the robot.
- This is just a quick start guide to teach you how to easily use FR collaborative robots. The premise of this guide is that the environment is safe and harmless, and users are cautious. Do not increase the speed or acceleration to the default value. Before the robot enters the operation, risk assessment is always carried out.

1.3.5 Teaching pendant software

1.3.5.1 Basic information

1.3.5.1.1 Introduction

The teaching pendant software is the supporting software developed for the robot and runs on the teaching pendant operating system. Its main functions and technical characteristics are as follows:

- Ability to write teaching programs for robots;
- It can display the coordinates of the robot's position in real time, simulate the physical robot in three dimensions, and control the movement of the robot;
- It can realize single-axis inching and linkage operation of each axis of the robot;
- Able to view control IO status;
- Users can modify passwords, view system information, etc.

1.3.5.1.2 Robot's first activation

1. Power on the control box and connect the Ethernet cable to the PC.
2. Open a web browser on the PC and visit the target website 192.168.58.2; the robot will enter the activation page upon first power-up.

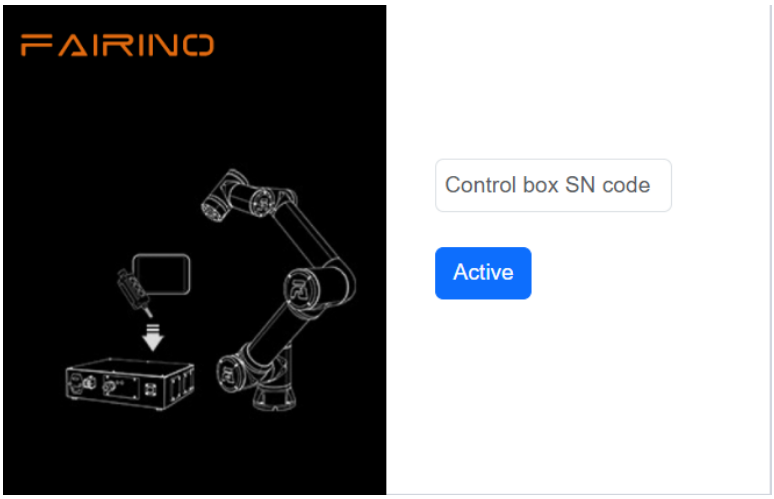
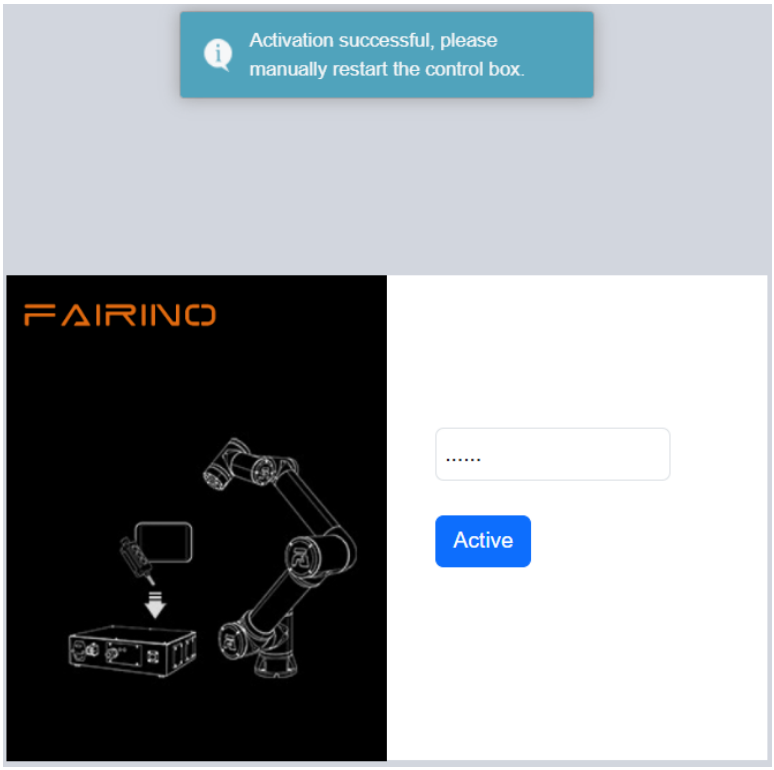


Figure 5.1-1 Activation Interface

3. Enter the correct SN code of the device box and click the "Activate" button after input is complete.
4. The system will verify your SN code. If the input is correct, the activation process will be completed automatically.

Figure 5.1-2 Activation Success Interface

5. After successful activation, please manually restart the control box.
6. After restarting, visit the target website 192.168.58.2 again to enter the login page.



1.3.5.1.3 Start software

1. Power on the control box;
2. The teach pendant opens a browser to access the target website 192.168.58.2;
3. Enter the user name and password and click Login to log in to the system.

1.3.5.1.4 User login and permission update

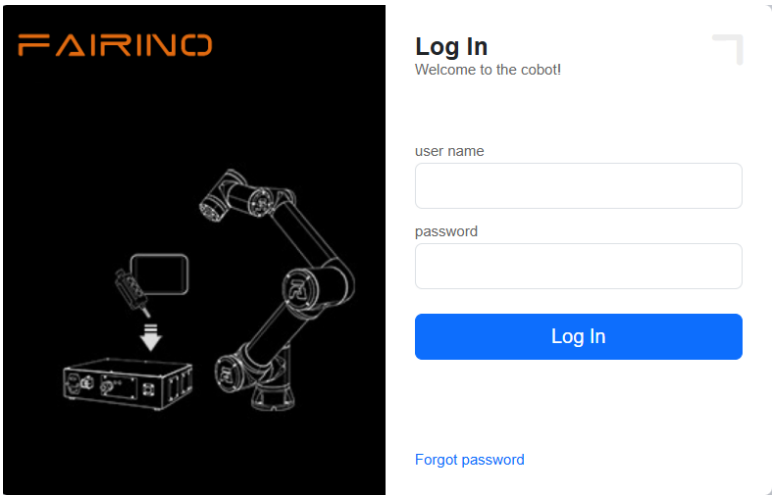
Table 5.1-1 Initial user

Job number	Initial username	Password	Function code
111	admin	123	1
222	MEenginer	222	2
333	PEenginer	333	3
444	programmer	444	4
555	operator	555	5
666	monitor	666	6

Users (refer to 15.2.1 User Management for user management) are divided into six levels by default, administrators have no function restrictions, operators and monitors can use a small number of functions, ME engineers, PE&PQE engineers and technicians & team leaders have some function restrictions, management There are no functional restrictions for administrators. For specific default function code permissions, refer to 15.2.2 Authority management.

The login interface is shown in figure below.

Figure 5.1-3 Login Interface



1.3.5.1.5 Multilingual settings function

- The system currently comes with eight languages: Chinese (), Chinese (), English (English), French (français), Japanese () , Russian () and Italian (italiano).
- The language pack name must be: [Language code].json, for example: es.json, the language code is the ISO 639-1 standard.
- The following is a language comparison table.

Language	Local language name	Language code(ISO 639-1)	Is it built-in
Chinese	()	zh	True
Chinese	()	tc	True
English	en	ja	True
French	français	fr	True
Japanese		ja	True
Korean		ko	True
Russian		ru	True
Italian	italiano	it	True

1.On the login screen (or during the initial activation screen, settings can also be made), select the language at the top right corner.

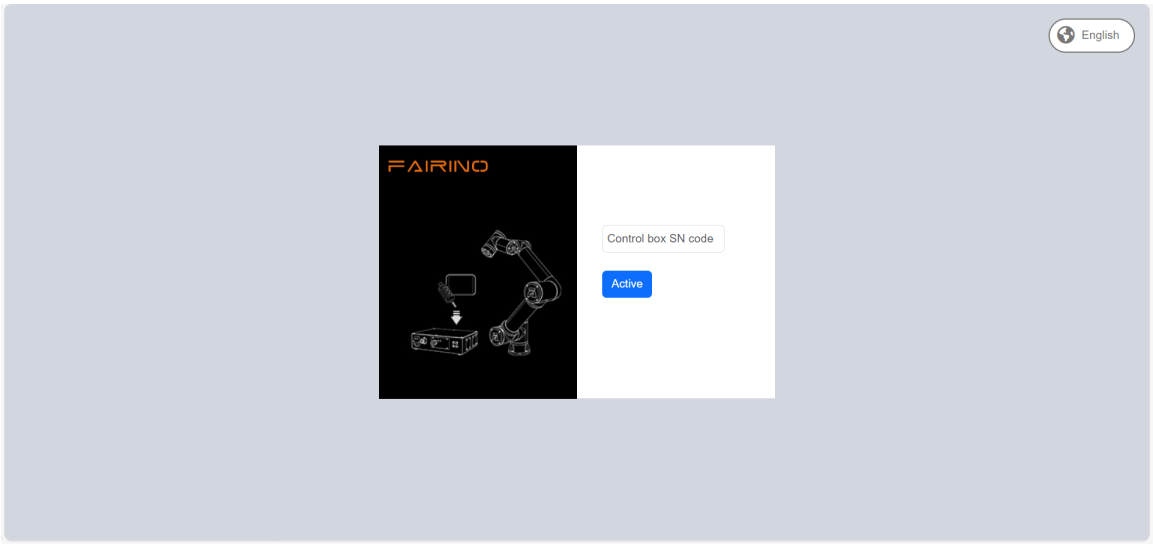


Figure 5.1-5 Language settings on the activation interface

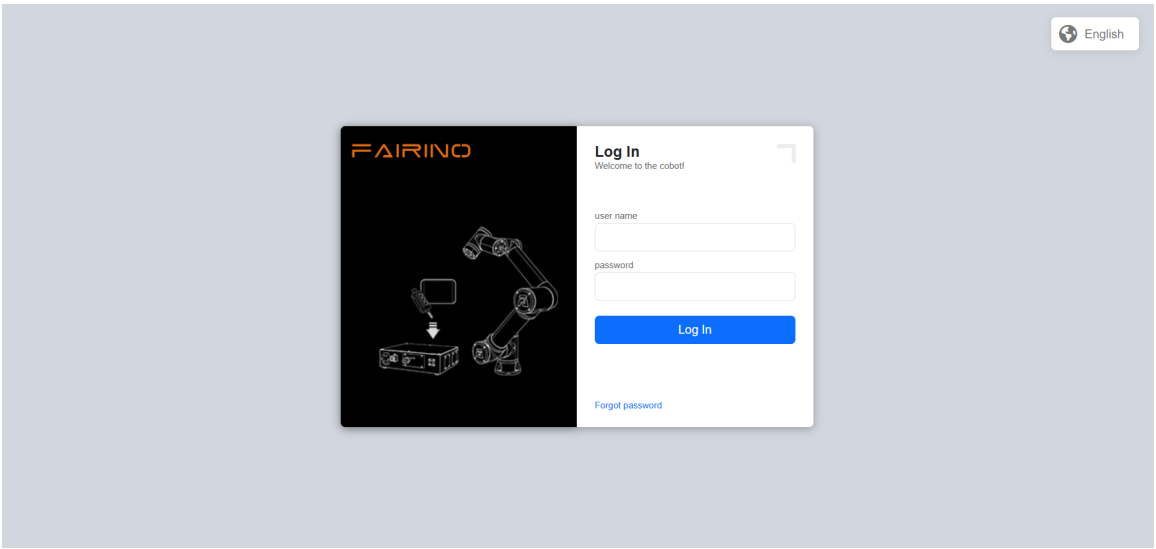


Figure 5.1-6 Language settings on the login interface

2.Take the login interface language settings as an example, select the language. If the following prompts (in different languages) appear, it means the settings are successful. Restart the control box to complete the language settings.



Figure 5.1-7 Set to Chinese

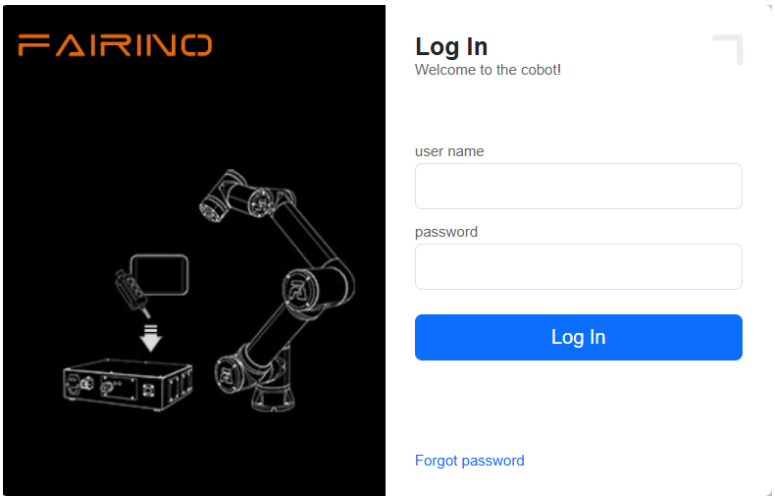


Figure 5.1-8 Set to English

After the login is successful, the system will load the model and other data, and enter the initial page after loading.

1.3.5.2 System initial interface

After the login is successful, the system enters the “Initial”. Mainly include:

1. FAIRINO LOGO;
2. Menu bar zoom button;
3. Menu bar;
4. Robot control area;
5. Robot status area;
6. 3D simulation robot - 3D scene operation;
7. 3D simulation robot - robot body operation;
8. Robot supporting functions;
9. Robot and supporting function status.

The initial interface of the system is shown in the figure below:

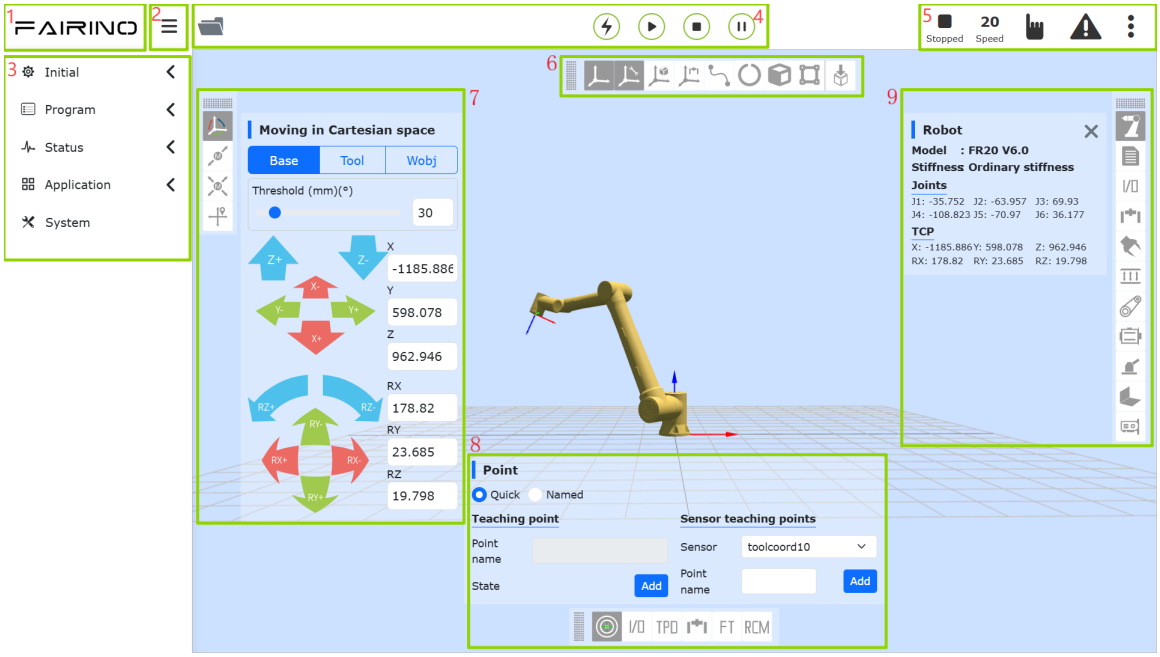


Figure 5.2-1 Schematic diagram of the initial interface of the system

When entering the “Initial”, “Program - Coding”, “Program - Graphical” and “Auxiliary” of the WebApp, the 3D simulation robot model page is half-expanded. Click the expanded icon to re-enter the system initial interface.

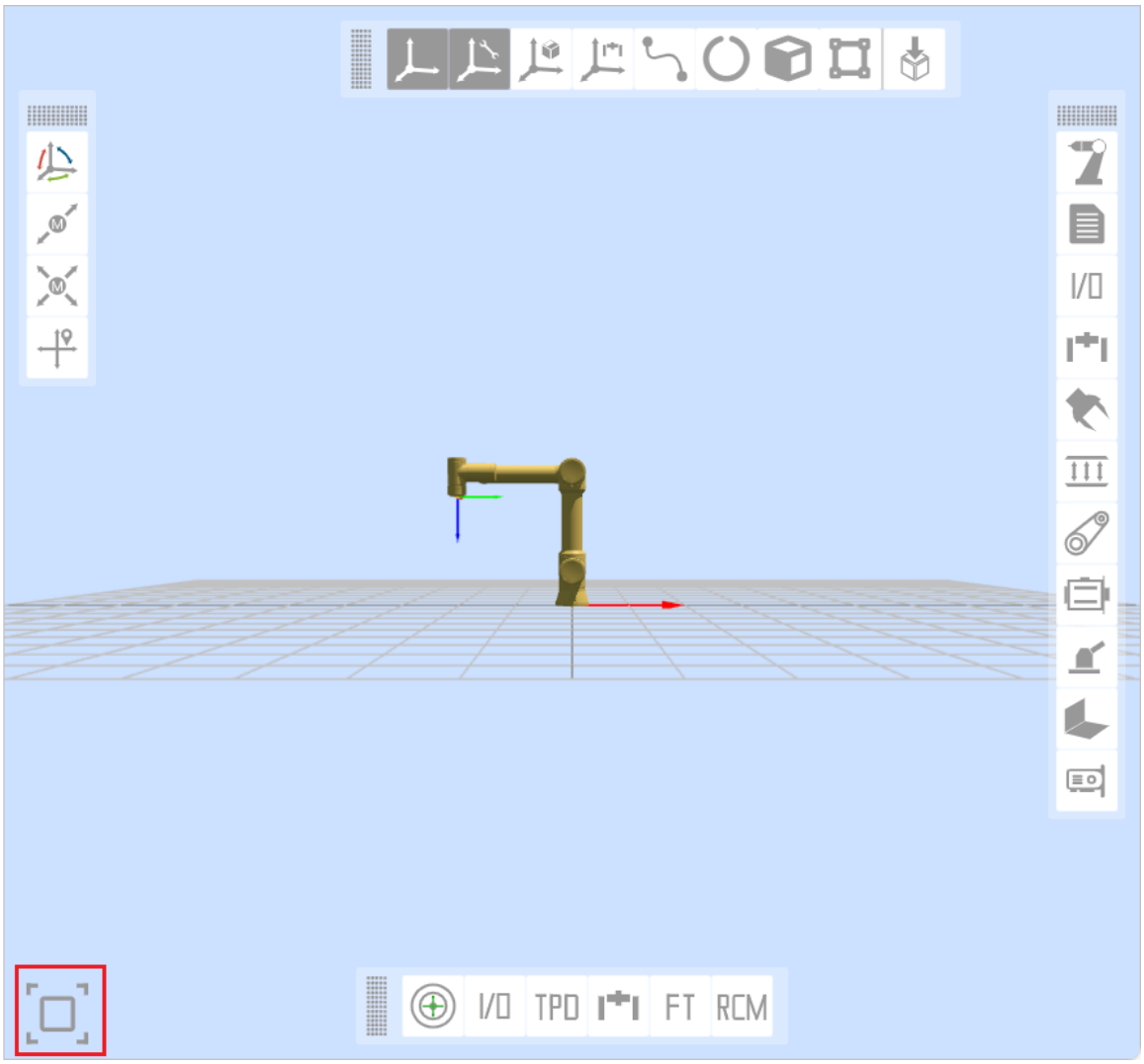


Figure 5.2-2 3D simulation robot model page can be rolled out icon

1.3.5.2.1 Control area



Note:

name:**Enable button**

effect:Enable the robot



Note:

name:**Start button**

effect:Upload and start running the teaching program



Note:

name:**Stop button**

effect:Stop the current teaching program running



Note:

name:**Pause/Resume button**

effect:Pause and resume the current teaching program

Important: The pause instruction is at the end of the program and cannot be judged.

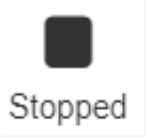
1.3.5.2.2 Status Bar



Note:

name: **Error state**

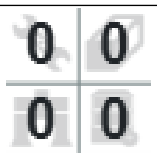
effect: There is an error in the current robot operation, hide when no error.



Note:

name: **robot status**

effect: Stopped-stopRunning-runPause-pauseDrag-drag

Note: 

name: **Robot Tool Coordinate System, Workpiece Coordinate System, Extended Axis Coordinate System, and Load Number**

effect: Top Left – Current Tool Coordinate System Number, Top Right – Current Workpiece Coordinate System Number, Bottom Left – Current Extended Axis Coordinate System Number, Bottom Right – Current Load Number

---

Note: 

name: **Running speed percentage**

effect: The speed of the robot when it is running in the current mode

---

Note: 

name: **Automatic mode**

effect: Robot automatic operation mode.When the global speed adjustment in manual mode and automatic mode is turned on and the speed is specified, the global speed will be automatically adjusted to the specified speed

---

Note: 

name: **Manual mode**

effect: Robot manual mode for robot teaching operations.

Note: 

name: **Robot State Fold/Unfold Button**

effect: Fold/Unfold Tool Coordinate System, Workpiece Coordinate System, Extended Axis Coordinate System, Load, Robot Drag Status, Local/Remote Mode, Robot Connection Status, BOOT Mode, and Account Information Content

Click the collapse/expand button to view the following status information.

Note: 

name: **Tool coordinate system number**

effect: Display the tool coordinate system number of the current application

Note: 

name: **Workpiece coordinate system number**

effect: Display the workpiece coordinate system number currently applied

Note: 

name: **Extended axis coordinate system number**

effect: Displays the currently applied extended axis coordinate system number

Note: 

name: **Payload**

effect: Display the load weight and center of gravity coordinates X, Y, Z of the current application.

Note: 

name: **Drag state**

effect: The current robot can drag



Note:

name: **Drag state**

effect: The current robot is not draggable



Note:

name: **Robot Local Mode**

effect: The current robot is controlled via the control box.



Note:

name: **Robot Remote Mode**

effect: Currently, robots can only be controlled through PLC.



Note:

name: **Connection status**

effect: Robot connected



Note:

name: **Not connected status**

effect: Robot not connected



Note:

name: **Account information**

effect: Display username and permissions and logout user

1.3.5.2.3 Menu Bar

The menu bar is as shown in table below.

Table 5.2-1 Teach pendant menu column

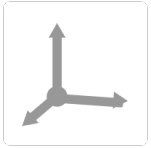
first level	second level
Initial	Base
	Safety
	Peripheral
Program	Coding
	Graphical
	Node Graph
	Points
Status	Log
	Query
Application	Tool App
	Process Package
System	/

1.3.5.3 3D simulation robot

1.3.5.3.1 3D scene operation bar

1.3.5.3.1.1 3D visualization display of robot coordinate system

Create various 3D virtual coordinate systems in the 3D virtual area of the WebAPP robot, taking the display of the base coordinate system as an example, as shown in the figure below. Among them, the X axis is red, the Y axis is green, and the Z axis is blue.



Note:

name:**Base coordinate system**

explain: In the base coordinate system WebAPP, the system robot is displayed in the three-dimensional virtual area by default, and the fixed mark is at the bottom center of the robot base. The 3D virtual base coordinate system can be

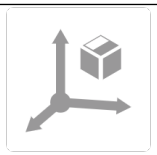
displayed manually.

Note: 

name:**Tool coordinate system**

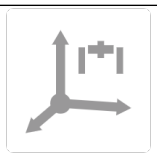
explain: The display of the tool coordinate system is enabled by default and can be manually disabled. After the WebAPP starts and the user logs in successfully, obtain the tool coordinate system name and corresponding parameter data of the current application, and initialize the current tool coordinate system.

Important: When applying other tool coordinate systems during use, after the application tool coordinate system command succeeds, first clear the existing tool coordinate system in the robot's 3D virtual area, and then transfer the newly applied tool coordinate system parameter data into the 3D coordinate system The API is generated to generate the tool coordinate system, and after the generation is completed, it is displayed in the three-dimensional virtual area of the robot.

Note: 

name:**Workpiece coordinate system**

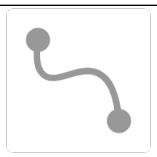
explain: The workpiece coordinate system is closed by default, and it can be displayed manually. The process is consistent with the tool coordinate system.

Note: 

name:**External Axis Coordinate System**

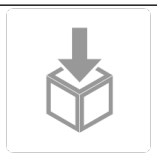
explain: The workpiece coordinate system is closed by default, and it can be displayed manually. The process is consistent with the tool coordinate system.

1.3.5.3.1.2 3D virtual trajectory and import tool model

Note: 

name:**Trajectory drawing**

explain: Click the button to turn on the trajectory drawing function. When running the teaching program, the robot 3D model will depict the trajectory of the robot's movement.

Note: 

name:**Import tool model**

explain: Click the button to pop up the import tool model modal window. After the uploaded file is successfully imported, the tool model can be displayed at the end of the robot. Currently supported tool model file formats are STL and DAE.

1.3.5.3.2 Robot body operation bar

1.3.5.3.2.1 TCP

Base Jog: In the base coordinate system, the robot can be controlled by long pressing the corresponding coordinate system button, move in a straight line on the X, Y, and Z axes or rotate around RX, RY, and RZ. The function of Base jogging is similar to that of single-axis jogging in Joint motion. As shown in figure below.

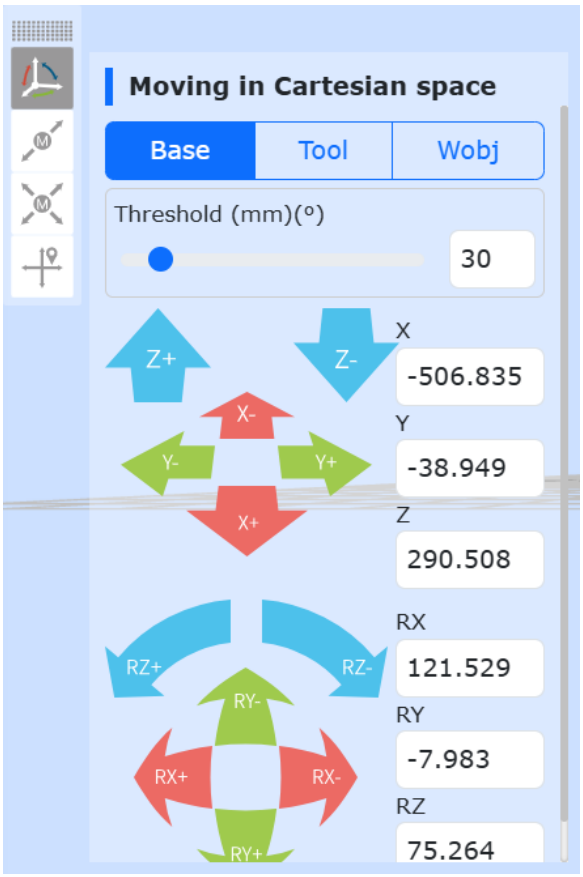


Figure 5.3-1 Base jog diagram

Important: The button can be released at any time to stop the robot. If necessary, press the emergency stop button to stop the robot.

Tool Jog: Select the tool coordinate system, you can click the corresponding coordinate system button to control the robot, move in a straight line on the X, Y, and Z axes or rotate around RX, RY, and RZ. The function of Tool jogging is similar to that of single-axis jogging in Joint motion. As shown in figure below.

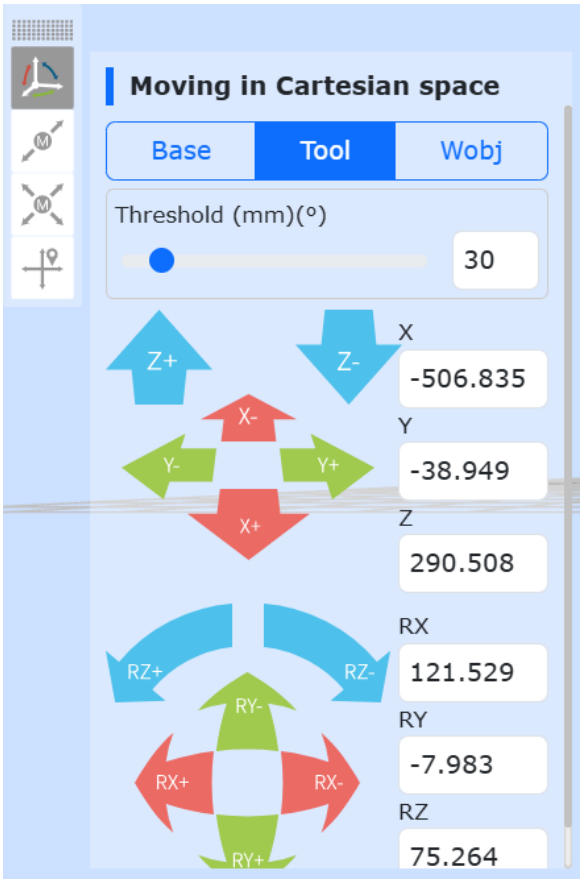


Figure 5.3-2 Tool Jogging Schematic Diagram

Wobj Jog: Select workpiece jogging, you can long press the circular buttons on the left and right to control the robot. In the workpiece coordinate system, move along the X, Y, Z axis in a straight line or rotate around RX, RY, RZ. The function of Wobj jogging is similar to that of single-axis jogging in Joint motion. As shown in figure below.

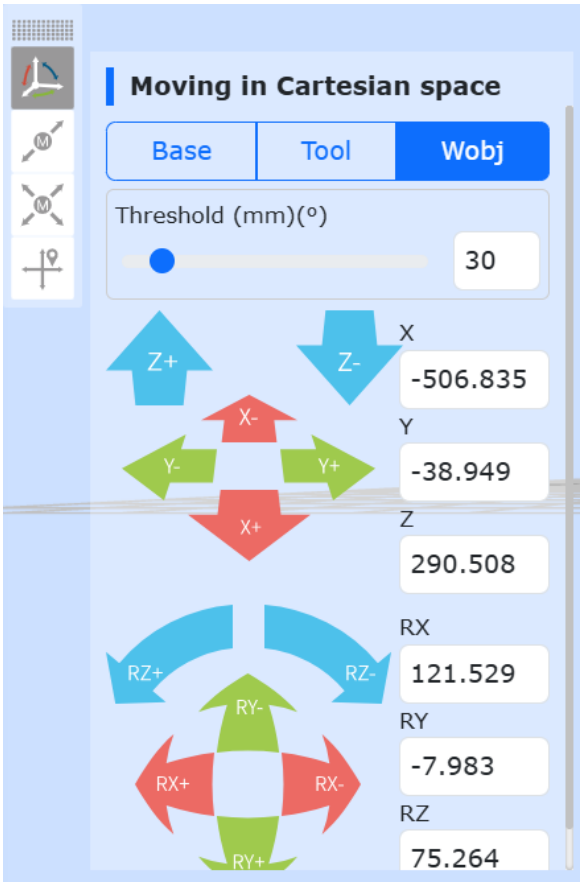


Figure 5.3-3 Wobj jogging diagram

1.3.5.3.2.2 Joint Jog

Under Joint operation, the 6 slider bars in the middle represent the angles of the corresponding axes respectively, and the joint movement is divided into single-axis inching and multi-axis linkage

single axis jog:The user can control the movement of the robot by operating the left and right circular buttons, as shown in figure below. In the manual mode and the joint coordinate system, perform a rotation operation on a joint of the robot. When the robot stops beyond the range of motion (soft limit), you can use single-axis jogging for manual operation to move the robot out of the overrun position. Single-axis jogging is faster and more convenient than other operation modes for rough positioning and large-scale movement.

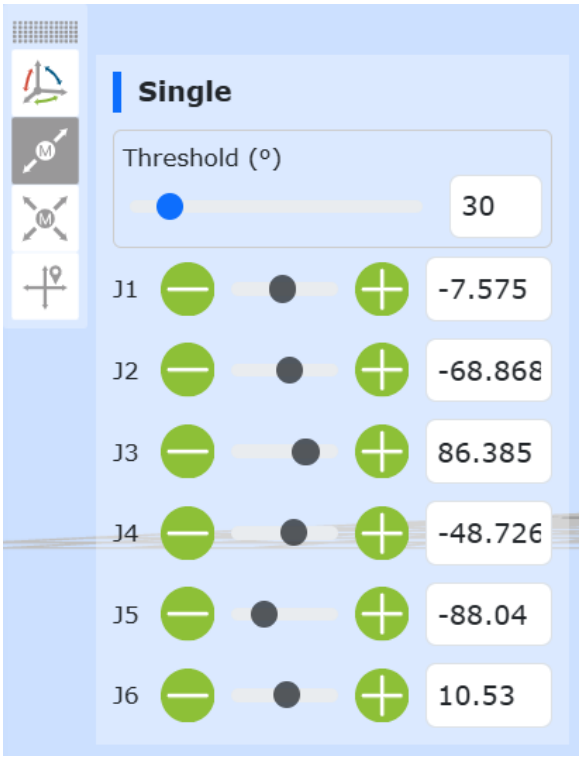


Figure 5.3-4 Schematic diagram of single-axis jogging

Important: Set the "long press movement threshold" (the maximum distance the robot can run when the button is long pressed, and the input value range is 0~300) parameters, long press the round button to control the robot running, if the button is released during the robot running, the robot will stop immediately Movement, if you keep pressing the button without releasing it, the robot will run and stop moving after long pressing the value set by the movement threshold.

Multi-axis linkage:The user can operate the six sliders in the middle to adjust the corresponding target position of the robot, as shown in figure below. The target position can be determined by observing the 3D virtual robot. If the adjusted position does not meet your expectations, click the "Restore" button. Make the 3D virtual robot return to the initial position. After the user determines the target position, he can click the "Apply" button, and the physical robot will perform corresponding movements.



Figure 5.3-5 Schematic diagram of multi-axis linkage

Important: In multi-axis linkage, the setting value of the fifth joint j5 cannot be less than 0.01 degrees. If the expected value is less than 0.01 degrees, you can first set it to 0.011 degrees, and then fine-tune the fifth joint j5 through single-axis jogging.

1.3.5.3.2.3 Move

Select Move, you can directly input the Cartesian coordinate value, click “Calculate joint position”, the joint position is displayed as the calculated result, confirm that there is no danger, you can click “Move to this point” to control the robot to move to the input Cartesian pose.

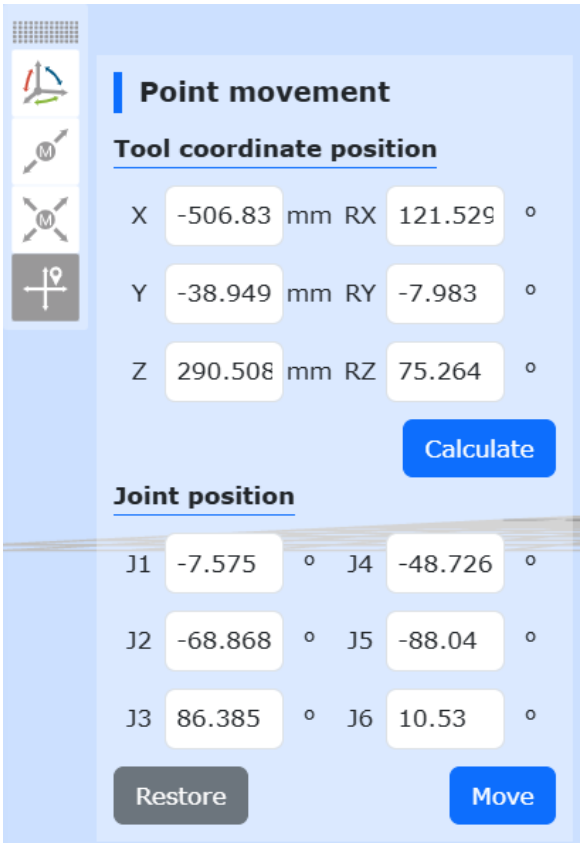


Figure 5.3-6 Move mobile diagram

Important: When a given pose cannot be reached, first check whether the Cartesian space pose exceeds the working range of the robot, and then check whether there is a singular pose in the process from the current pose to the target pose, if there is a singular position, adjust the current pose or process insert a new pose in order to avoid singular poses.

1.3.5.3.3 Robot supporting function bar

1.3.5.3.3.1 Teaching point record

The manual teaching control area is mainly to set the test coordinate system in the teaching mode, and display the angle and coordinate values of each axis of the robot in real time, and can name and save the teaching points.

When saving a teaching point, the coordinate system of the teaching point is the coordinate system of the current robot application.

There are two ways to record teaching points: “quick point recording” and “named point recording”.

- Quick point recording: teaching points are automatically recorded and names are automatically generated;
- Naming points: Customized naming of teaching points, consisting of teaching point prefix+teaching point name;

For the sensor teaching point, select the calibrated sensor type tool, enter the point name, click Add, and the saved point position is the position recognized by the sensor.

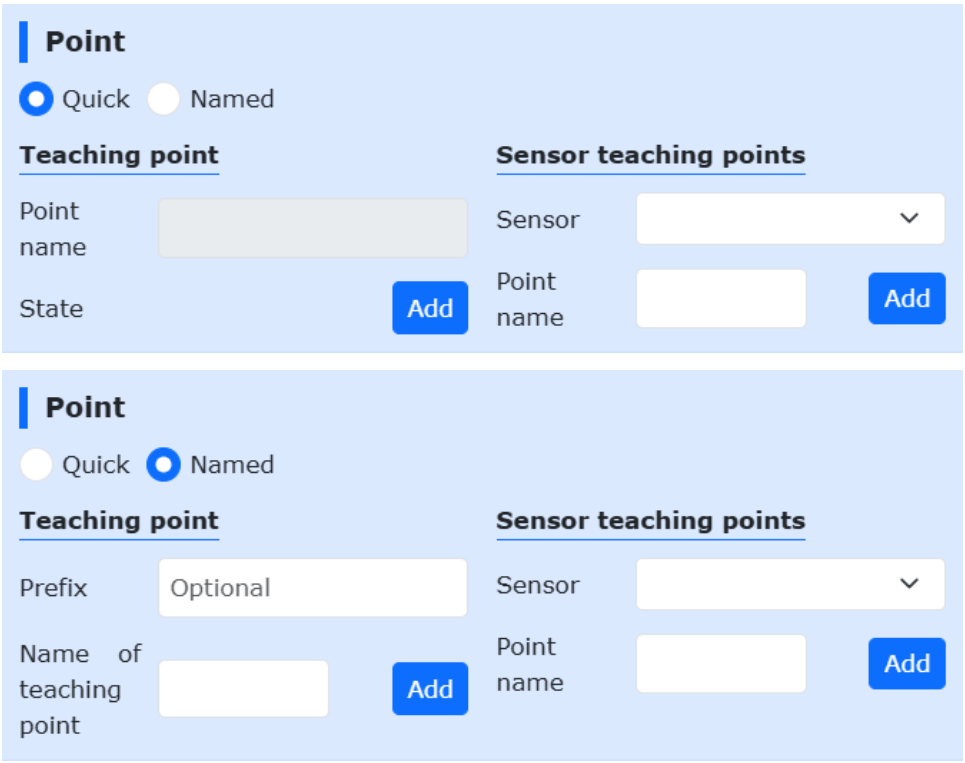


Figure 5.3-7 Schematic diagram of manual operation area

Important: When using it for the first time, please set a small speed value like 30 to familiarize yourself with the movement of the robot to avoid accidents

1.3.5.3.3.2 I/O

In this interface, the digital output, analog output (0-10v) and end tool digital output in the robot control box can be realized. Output, analog output (0-10v) for manual control. As shown in figure below.

- DO operation: select the port number, if the DO is low level, the right operation button will display ON, click the button to set the DO to high level.
- AO operation: Select the port number, enter a value (0-100) in the input box on the right, the value is a percentage, setting 100 means setting the AO port to 10v.

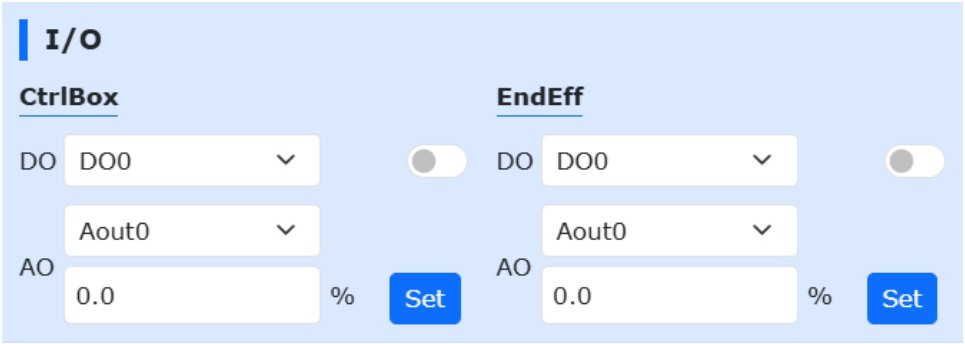


Figure 5.3-8 I/O

1.3.5.3.3.3 TPD (Teach-in programming)

The operation steps of teach programming (TPD) function are as follows:

- **Step1 record initial position**:Enter the operation area on the left side of the 3D model and record the current position of the robot. Set the name of the point in the edit box, click the "Save" button, if the save is successful, it will prompt "Save point successful";
- **Step2 Configure track recording parameters**:Click TPD to enter the "TPD" function item to configure trajectory recording parameters, set the name of the trajectory file, pose type and sampling period, configure DI and DO, and record the corresponding output by triggering DI during the process of recording the TPD trajectory DO;

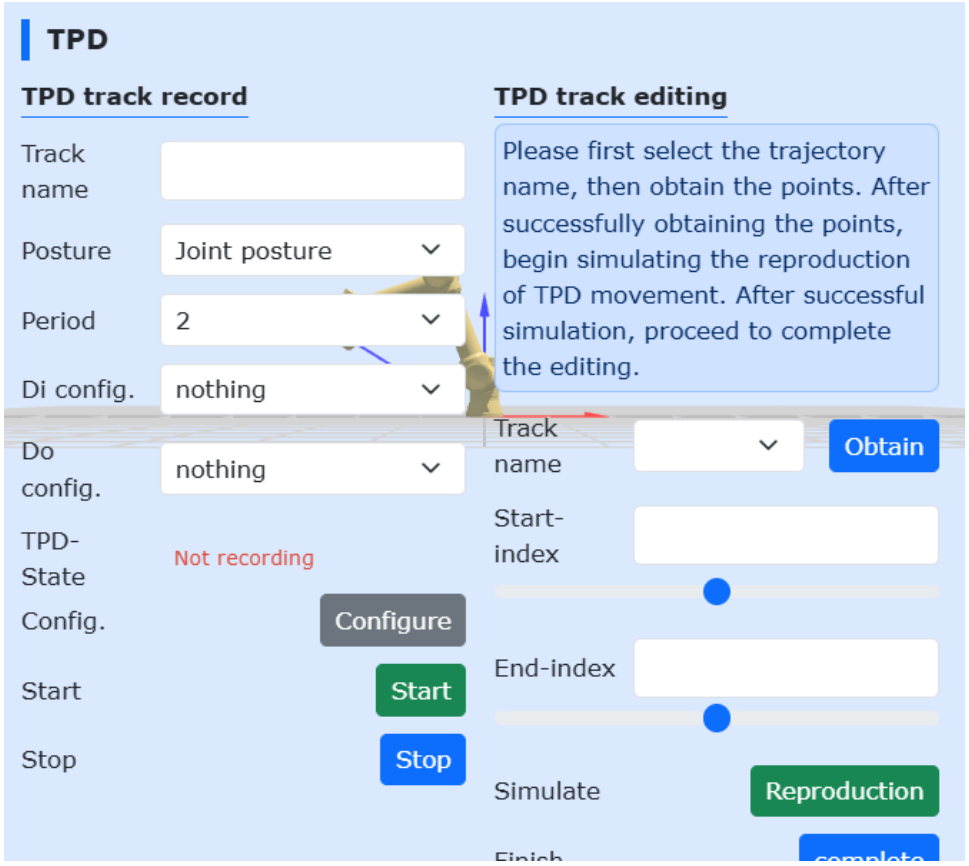


Figure 5.3-9 TPD track record

- **Step3 Check robot mode**:Check whether the robot mode is in manual mode. If not, switch to manual mode. In manual mode, there are two ways to switch to drag teaching mode, one is long press the end button, and the other is interface dragging mode It is recommended to switch the robot from the interface to the drag teaching mode in the TPD record.

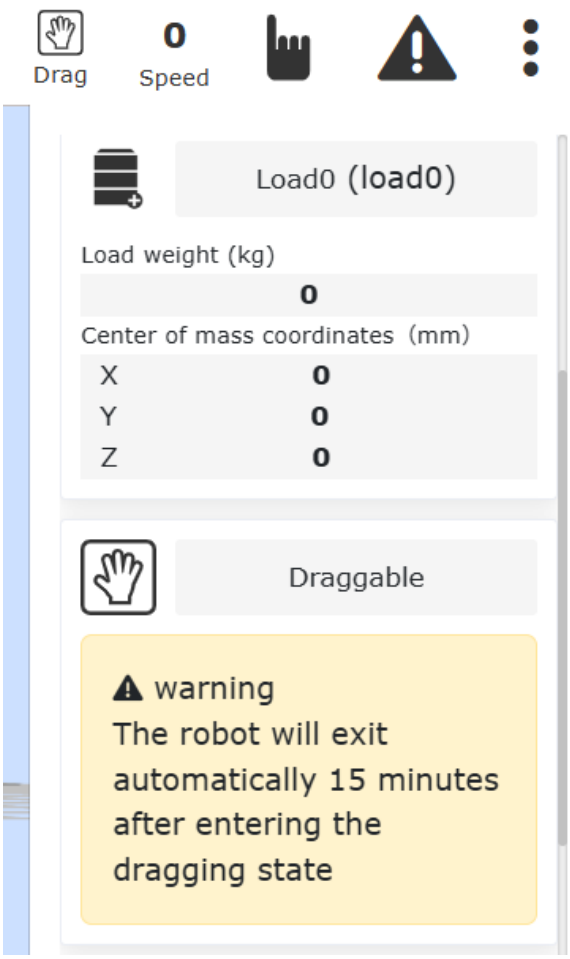


Figure 5.3-10 robot mode

Important: When switching to the drag mode from the interface, first confirm whether the end tool load and the center of mass are set correctly, and whether the friction compensation coefficient is set reasonably, then press and hold the end button to confirm whether the drag is normal, and then switch to the drag mode from the interface after confirming that it is correct.

- **Step4 Start recording**:Click the "Start" button to start track recording, and drag the robot to teach actions. In addition, there is a "TPD recording start/stop" function configuration item in the terminal DI configuration. By configuring this function, the user can trigger the "start recording" trace function through an external signal. It should be noted that to start recording a trace through an external signal, first Configure the TPD track information on the page first.
- **Step5 Stop recording**:After the action teaching is completed, click the "Stop" button to stop the track recording, and then drag the teaching switch button to make the robot exit the drag teaching mode. When the teach pendant receives "stop track recording successful", it means the track recording is successful. Same as step 4, after configuring the "TPD recording start/stop" function, you can trigger the recording to stop by an external signal.
- **Step6 Teach programming**:Click New, select a blank template, click to enter the PTP function programming item, select the initial position point just saved, click the "Add" button, after the application is completed, a PTP instruction will be displayed in the program file; then click to enter the TPD function programming item, Select the track just recorded, set whether it is smooth and the speed scaling, click the "Add" button, after the application is completed, a MoveTPD instruction will be displayed in the program file;

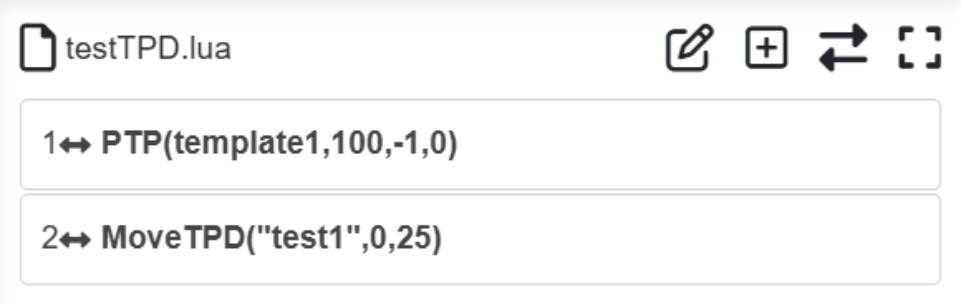


Figure 5.3-11 TPD programming

- **Step7 Trajectory recurrence**:After editing the teaching program, switch to the automatic operation mode, click the "start running" icon on the top of the interface to start running the program, and the robot will start to reproduce the teaching action.
- **Step8 track edit**:The TPD track editing area can visually display and edit the track to achieve pre-analysis and streamlining of the TPD track. Select the corresponding track acquisition point, then the track points recorded by the user will be displayed in the three-dimensional space of the robot, and then the user can drag the "Start" and "End" scroll bars to simulate and reproduce and edit the start and end points of the track.

TPD file deletion and exception handling:

- **Trajectory file deletion**:Click to enter the TPD function item, select the track file to be deleted, and click the "Delete Track" button. If the deletion is successful, you will receive a successful deletion prompt.
- **Exception handling:**
  - **Command points exceeded**:A track can record up to 20,000 points. When the number exceeds 20,000 points, the controller will no longer record the exceeded points, and will send a warning to the teach pendant that the number of command points exceeds the limit. At this time, you need to click to stop recording;
  - **TPD instruction interval is too large**:If the teaching pendant reports an error that the TPD command interval is too large, check whether the robot has returned to the initial position before recording. If the robot returns to the initial position and still reports an error that the TPD command interval is too large, delete the current track and record a new one;
  - If other abnormal conditions occur during TPD operation, the robot operation should be stopped immediately through the teach pendant or the emergency stop button to check the cause.

Important: During the operation of the TPD function, you should strictly follow the corresponding prompts on the teach pendant.

1.3.5.3.3.4 Eaxis move

Select Eaxis to move, this function is the jog function of the extension axis, it needs to be configured to configure the extension axis, use this jog function to control the extension axis, see "Chapter 4 Machine-Robot Peripherals-Extended Axis Peripherals" for details configuration".

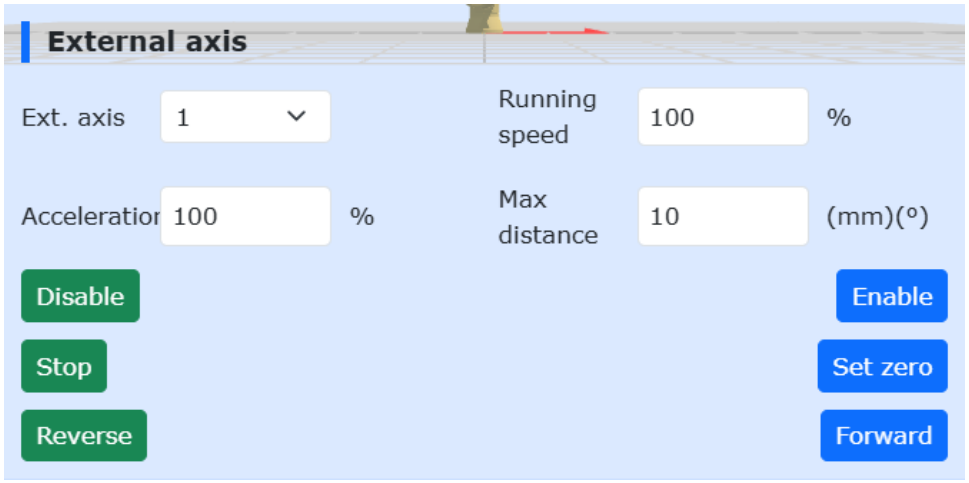


Figure 5.3-12 Eaxis Mobile Diagram

1.3.5.3.3.5 FT

Select the reference frame as a reference for the force sensor when dragging.



Figure 5.3-12 FT diagram

1.3.5.3.3.6 Telecentric fixed point

This function is mainly used in medical penetration. After setting the telecentric fixed point, the robot end always moves at this point.

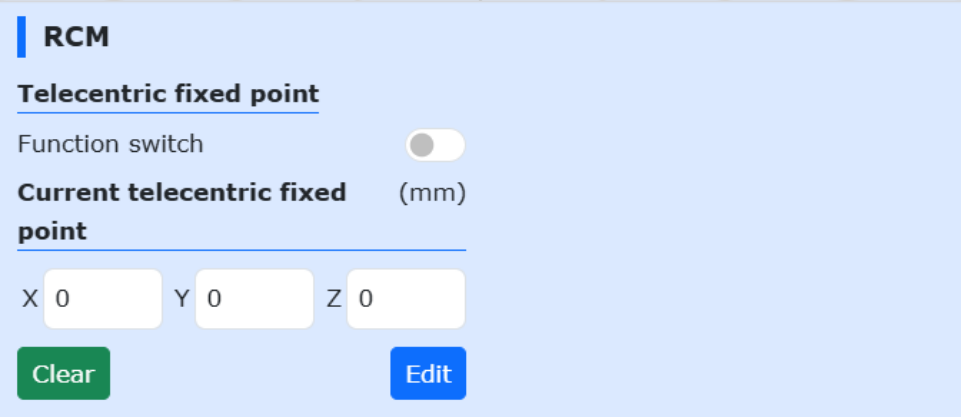


Figure 5.3-13 Schematic diagram of telecentric fixed point

1.3.5.3.4 Robot and supporting function status bar

1.3.5.3.4.1 Robot

Displays the current robot model, stiffness, joints and coordinate data information.

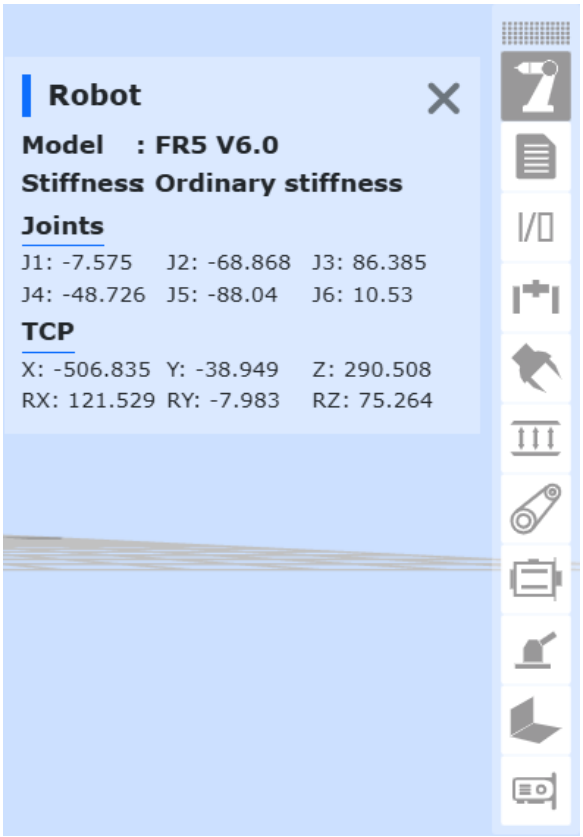


Figure 5.3-14 Robot Status

1.3.5.3.4.2 Program

Display information about the currently running program and subprogram.

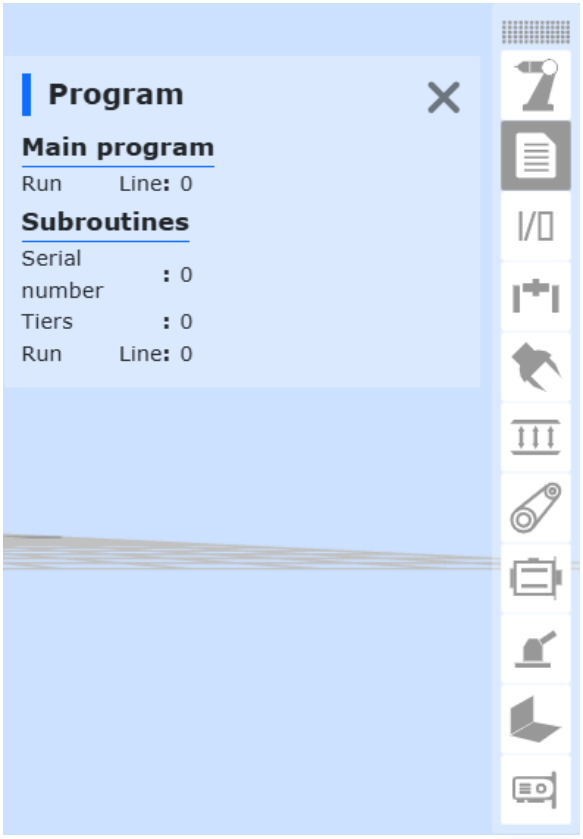


Figure 5.3-15 Program Status

1.3.5.3.4.3 I/O

The status display area on the right side of the 3D model will display the current IO status. In digital input and digital output, if the port level is high, the point will be displayed in green, and if it is low, it will be displayed in white; analog input and analog output The display value is 0-100, and 100 means 10v.

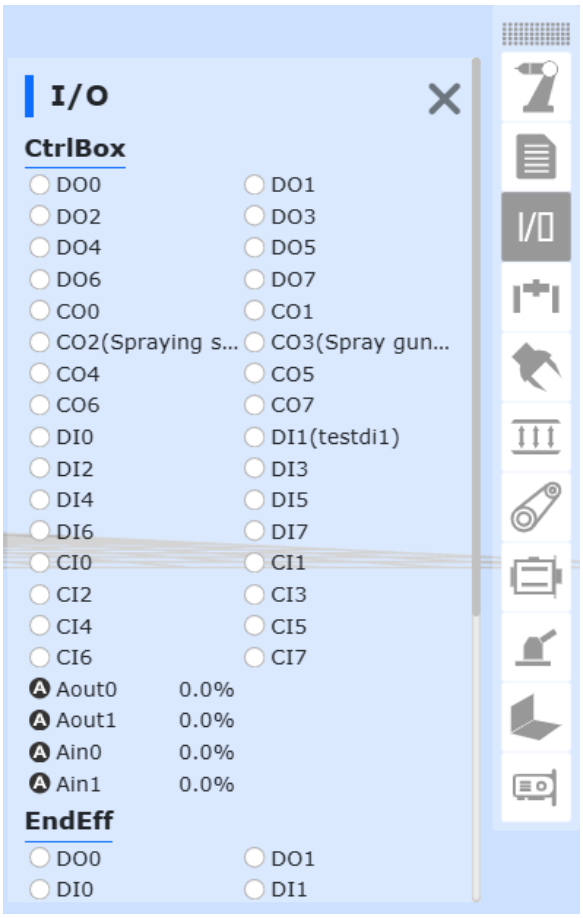


Figure 5.3-16 IO Status

1.3.5.3.4.4 ExAxis

Display the current extended axis (controller + PLC) servo status information.

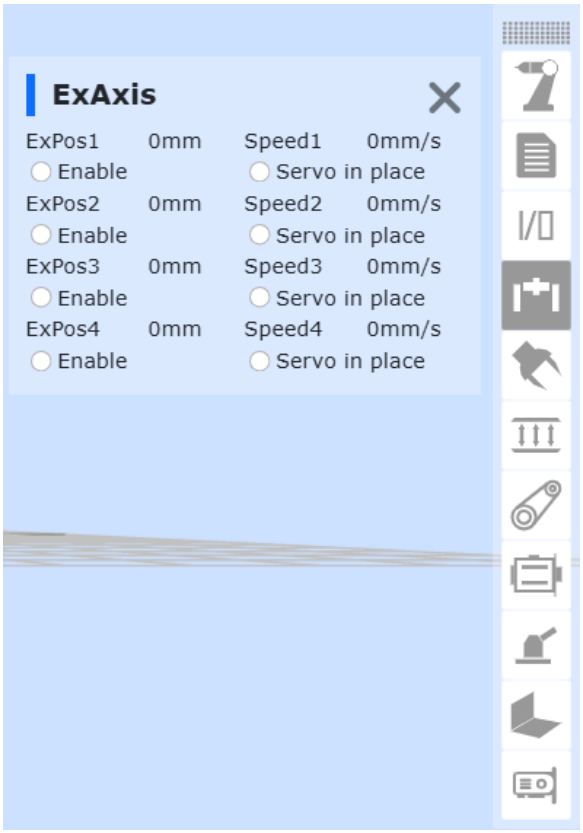


Figure 5.3-17 Extended axis (controller + PLC) status

1.3.5.3.4.5 Gripper

Displays the current gripper status information.

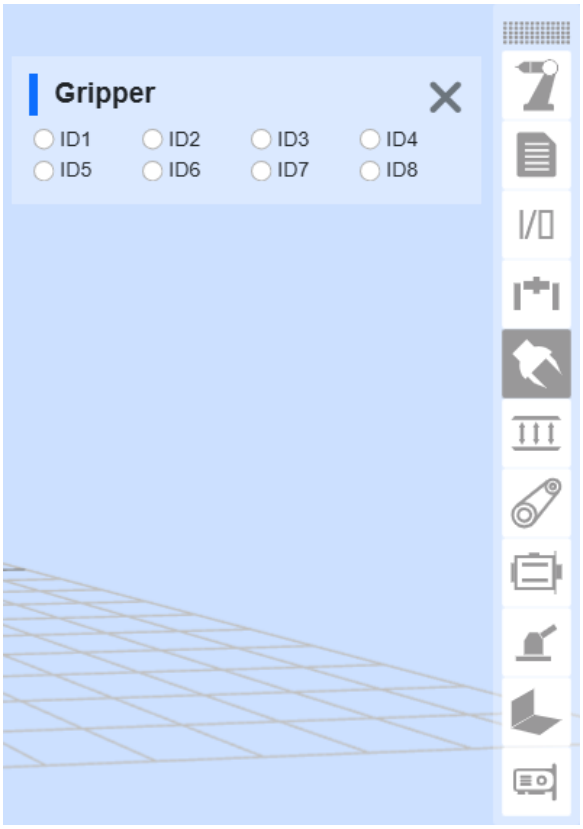


Figure 5.3-18 Gripper status

1.3.5.3.4.6 FT

Display current force control status information.

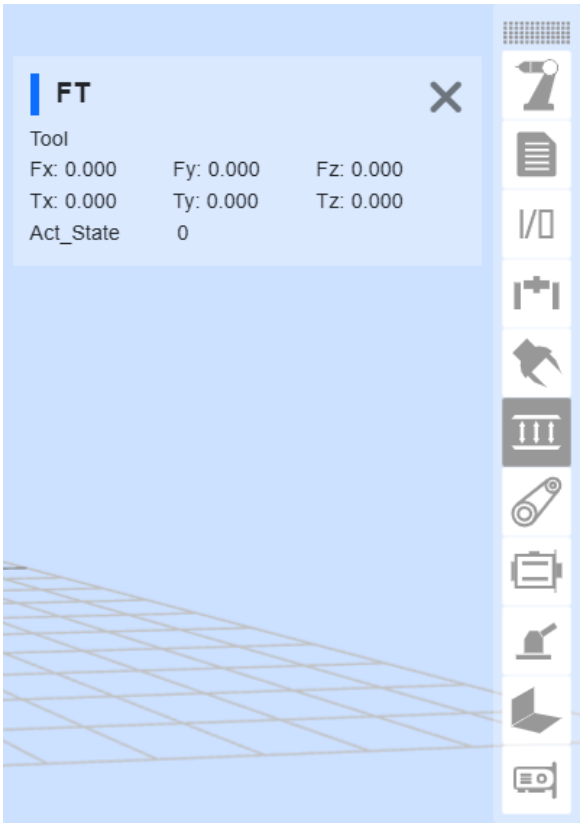


Figure 5.3-19 Force control state

1.3.5.3.4.7 Convery

Displays current conveyor status information.

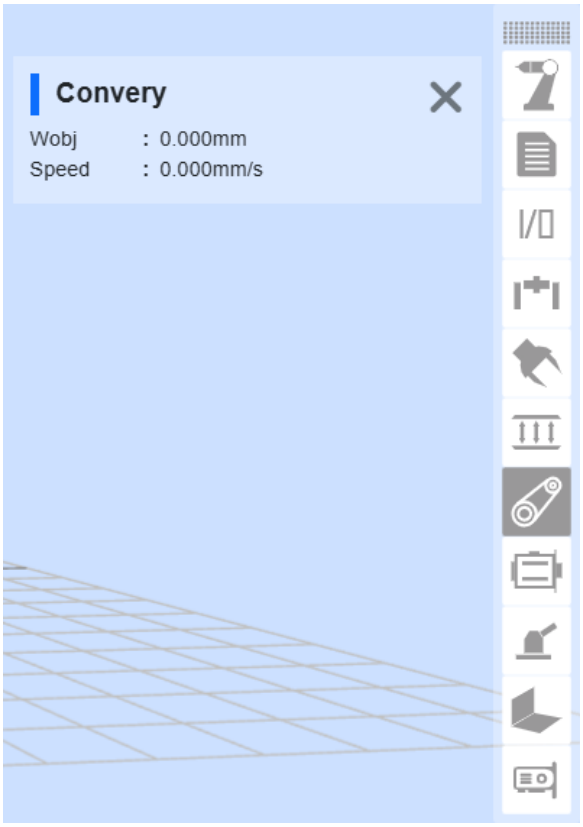


Figure 5.3-20 Conveyor Belt Status

1.3.5.3.4.8 Servo

Displays the current extended axis (controller + servo controller) status information.

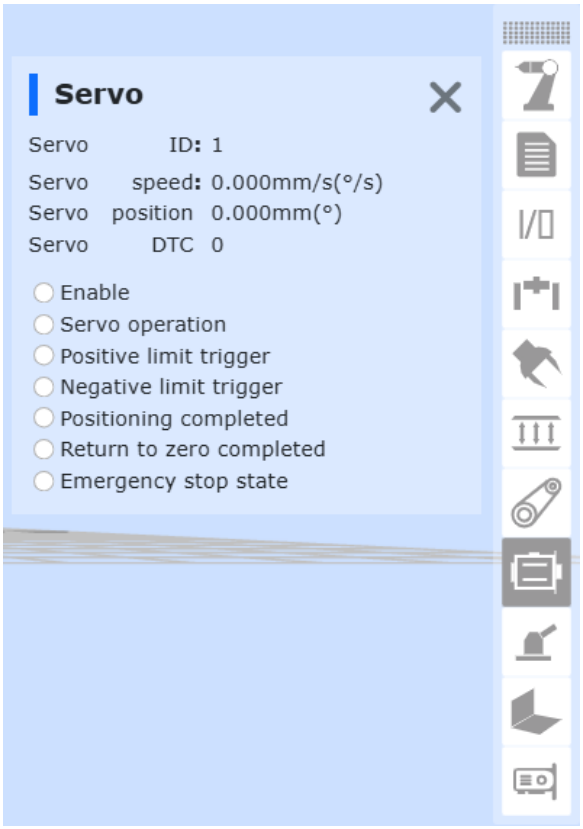


Figure 5.3-21 Extended axis (controller + servo controller) status

1.3.5.3.4.9 Polish

Display current grinding status information.

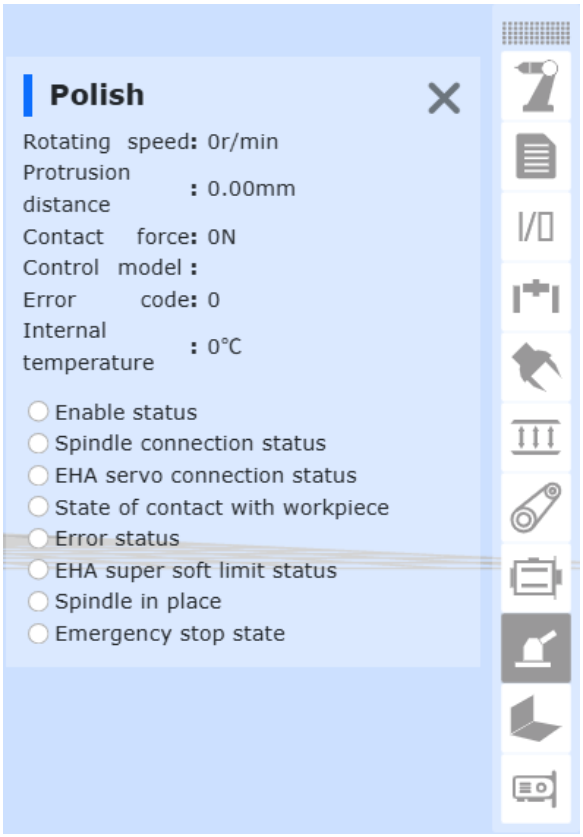


Figure 5.3-22 Polishing status

1.3.5.3.4.10 Weld

Display current welding status information.

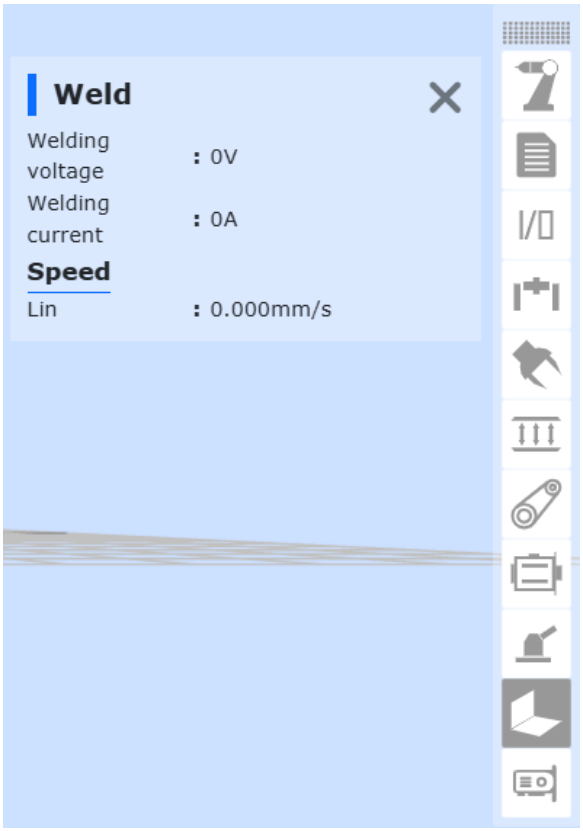


Figure 5.3-23 Welding status

1.3.5.3.4.11 Board I/O

Display the current status information of the board.

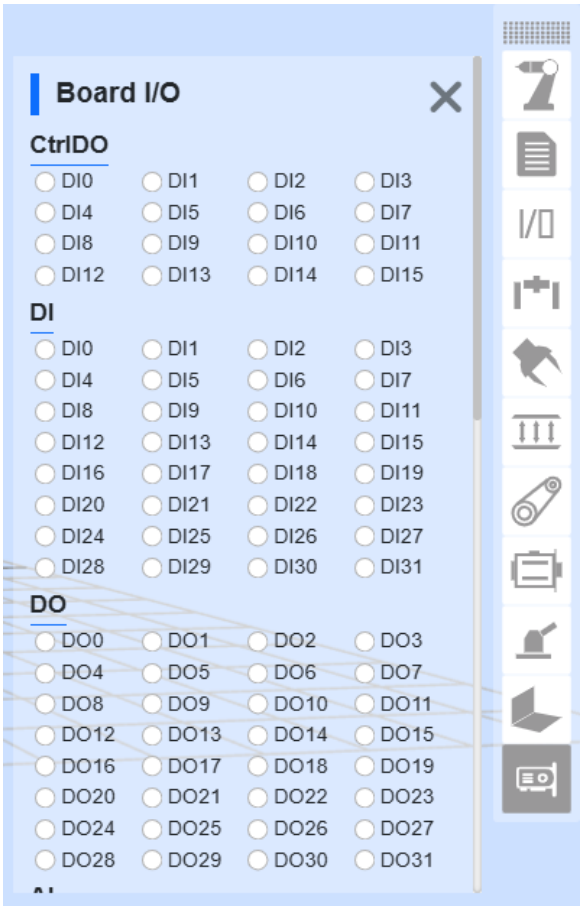


Figure 5.3-24 Board I/O status

1.3.6 Base

1.3.6.1 Installation

1.3.6.1.1 Robot installation method setting and display

On the web-based teaching page, click “Initial Setup” → “Basic” → “Installation”. The page layout is shown as follows. The specific instructions are as follows:

(1) Quick installation is used for common installation settings of robotic arms, corresponding from left to right: formal installation, side installation and reverse installation. When the corresponding button is clicked, the interface will automatically deliver and change the base tilt and rotation angle.

(2) If the required installation method does not meet the requirements of quick installation, you can configure it by setting the base tilt and rotation angle yourself.

(3) Whether it is quick installation or setting the base tilt and rotation angle by yourself, you need to click Apply to take effect.

Note: Please make sure that the set installation method is consistent with the actual robotic arm before dragging, otherwise there will be safety risks.

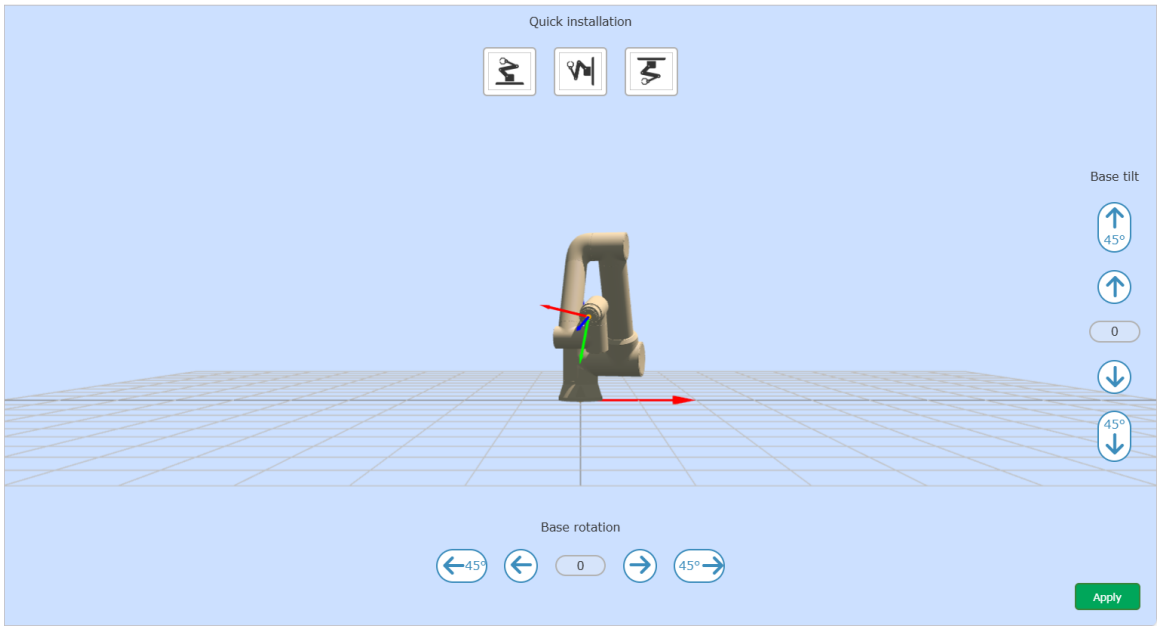


Figure 6.1-1 360 degree free installation

Important: After the installation of the robot is completed, the installation method of the robot must be set correctly, otherwise it will affect the use of the robot's dragging function and collision detection function.

1.3.6.2 Coordinate

1.3.6.2.1 TCP

In the menu bar of "Initial" -> "Base", click "TCP" to enter the tool coordinate page.

Tool coordinates can be modified, cleared, renamed and applied. In the drop-down list of tool coordinate systems, after selecting the corresponding coordinate system(the coordinate system name can be customized), the corresponding coordinate value, tool type and installation location (only displayed under sensor type tools) will be displayed below. After selecting a coordinate system, click the "Apply" button, and the currently used tool coordinate system will become the selected coordinate, as shown below.

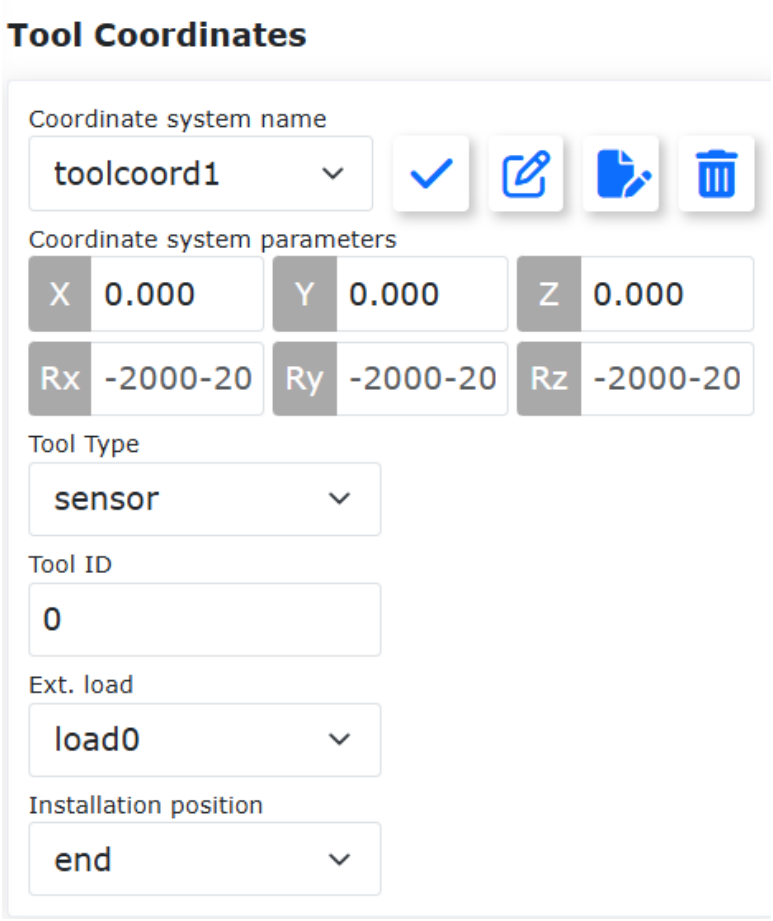


Figure 6.2-1 Set tool coordinates

Click “Modify” to reset the tool coordinate system of the number according to the prompt. Tool calibration methods are divided into four-point method and six-point method. The four-point method only calibrates the tool TCP, that is, the position of the center point of the tool. Its posture defaults to be consistent with the end posture. The six-point method adds two points to the four-point method. , used to calibrate the attitude of the tool, here we take the six-point method as an example to explain.

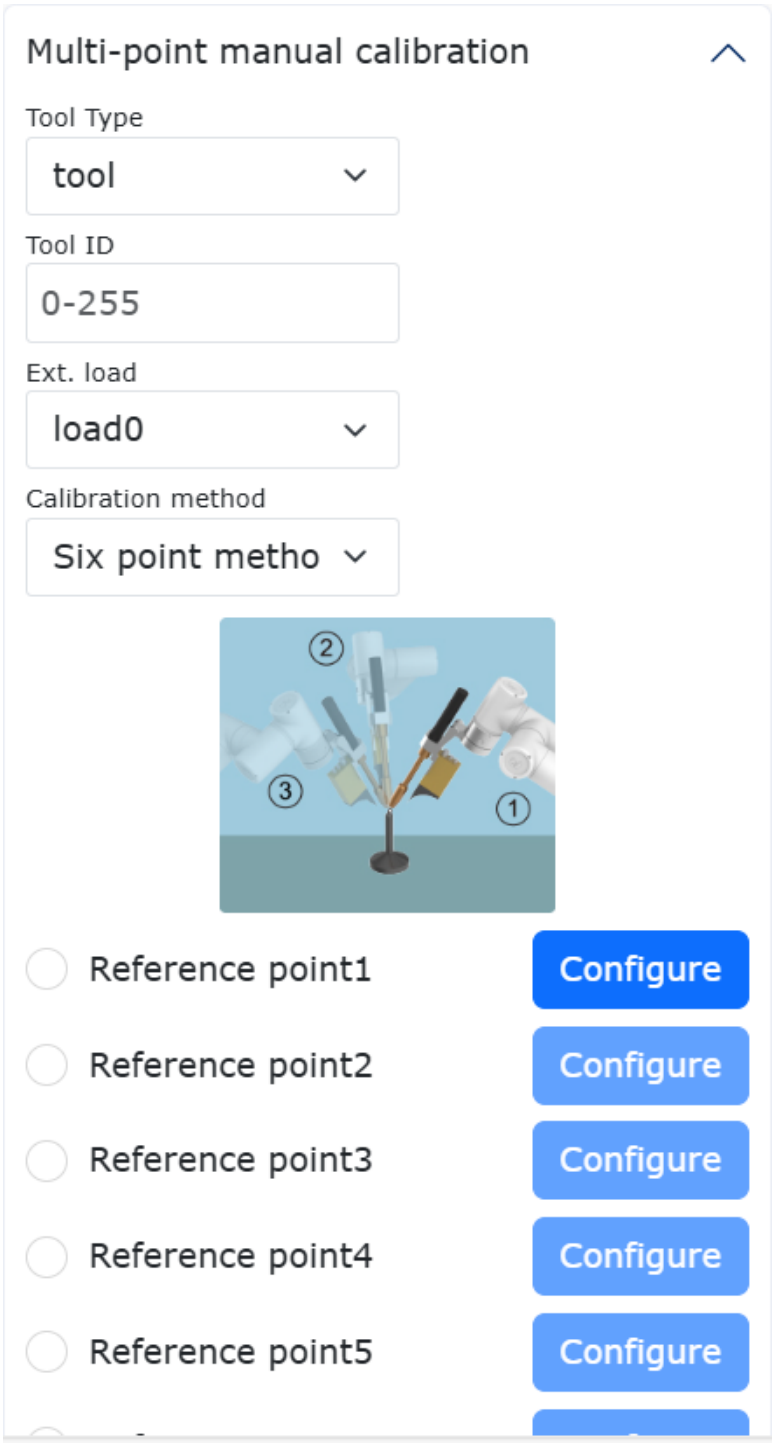


Figure 6.2-2 Set tool coordinates

Select a fixed point in the robot space, move the tool to the fixed point in three different postures, and set 1-3 points in sequence. As shown in the upper left of figure. Move the tool vertically to the fixed point setting point 4, as shown in the upper right of figure. Keep the posture unchanged, use the base coordinates to move, move a certain distance in the horizontal direction, and set point 5, which is the positive direction of the X-axis of the set tool coordinate system. Return to the fixed point, move vertically for a certain distance, and set point 6. This direction is the positive direction of the Z-axis of the tool coordinate system, and the positive direction of the Y-axis of the tool coordinate system is determined by the right-hand rule. Click the Calculate button to calculate the tool pose. If you need to reset it, click

Cancel and press the Modify button to re-create the tool coordinate system.

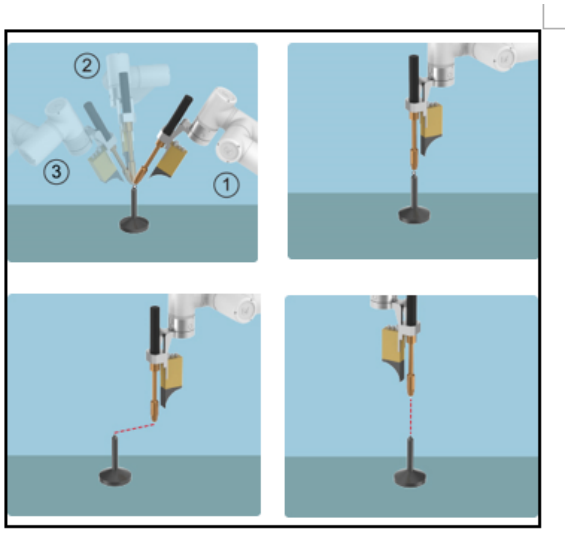


Figure 6.2-3 Schematic diagram of the six-point method

After completing the last step, click “Finish” to return to the tool coordinate interface, and click “Save” to store the tool coordinate system just created.

Important:

1. After the tool is installed at the end, the tool coordinate system must be calibrated and applied, otherwise the position and attitude of the tool center point will not meet the expected values when the robot executes the motion command.
2. The tool coordinate system generally uses toolcoord1~toolcoord19, and toolcoord0 is used to indicate that the position center of the tool TCP is at the center of the end flange. When calibrating the tool coordinate system, it is first necessary to apply the tool coordinate system to toolcoord0, and then select other tool coordinate systems for calibration and application.

1.3.6.2.2 Ext. TCP

Under the menu bar of “Initial” -> “Base”, click “Ext. TCP” to enter the external tool coordinate system interface.

The modification, clearing and application of external tool coordinates can be realized in the external tool coordinate system setting interface.

There are 15 numbers in the drop-down list of the external tool coordinate system, from etoolcoord0~etoolcoord14, after selecting the corresponding coordinate system, the corresponding coordinate value will be displayed below, after selecting a coordinate system, click the “Apply” button, the currently used tool coordinate system Change to the selected coordinates, as shown in figure below.

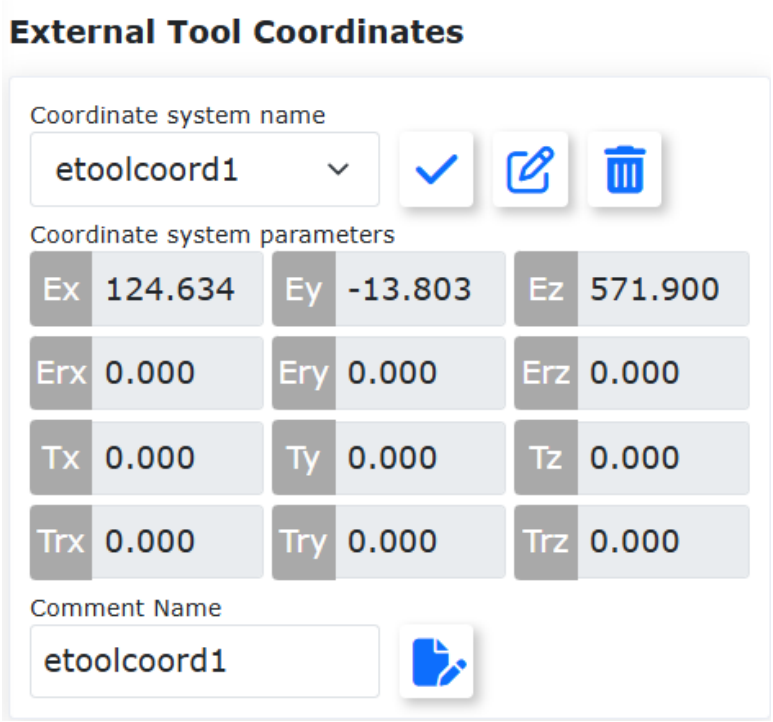


Figure 6.2-4 External tool coordinates

Click “Modify” to reset the tool coordinate system of the number according to the prompt, as shown in figure below.

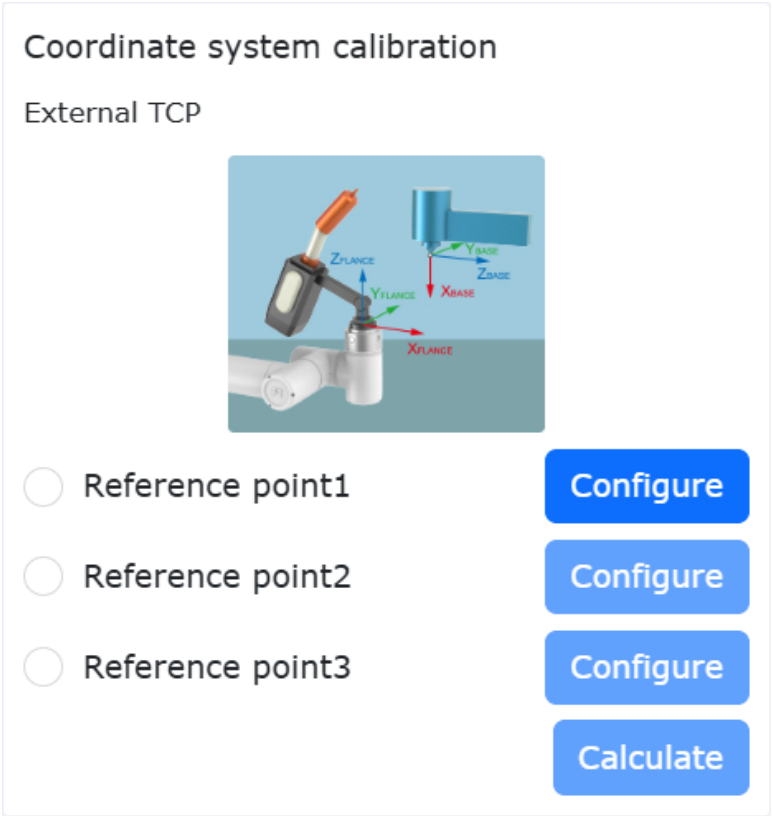


Figure 6.2-5 Schematic diagram of the six-point method

1. Three-point method to determine the external TCP

- **Set point 1**:The TCP of the measured tool is moved to the external TCP, click the Setpoint 1 button;
- **Set point 2**:Move a certain distance from point 1 along the X axis of the external TCF coordinate system, and click the button to set point 2;
- **Set point 3**:Go back to point 1, move from point 1 along the Z axis of the external TCF coordinate system for a certain distance, and click the button to set point 3;
- **Calculate**:Click the calculate button to get the external TCF;

2.Six-point method to determine the tool TCF

- **Set points 1-4**:Select a fixed point in the robot space, move the tool to the selected point from four different angles, and set points 1-4 in sequence;
- **Set point 5**:Go back to the fixed point and move a certain distance along the X axis of the tool TCF coordinate system, and click the Set Point 5 button;
- **Set point 6**:Go back to the fixed point and move a certain distance along the Y axis of the tool TCF coordinate system, and click the set point 6 button;
- **Calculate**:Click the calculate button to get the tool TCF;

If you need to reset, click the Cancel button to go back to the step of creating a new tool coordinate system.

After completing the last step, click "Finish" to return to the tool coordinate interface, and click "Save" to store the tool coordinate system just created.

Important:

1. The use of external tools must be calibrated and applied to the external tool coordinate system, otherwise the position and attitude of the tool center point when the robot executes motion commands will not meet the expected values.
2. The external tool coordinate system generally uses etoolcoord1~etoolcoord14, and the application of etoolcoord0 means that the center position of the external tool TCP is at the center of the end flange. When calibrating the tool coordinate system, the tool coordinate system must first be applied to etoolcoord0, and then other tool coordinate systems should be selected calibration.

1.3.6.2.3 Workpiece

Under the menu bar of "Initial" -> "Base", click "Workpiece" to enter the workpiece coordinates interface. Workpiece coordinates can realize the modification, clearing and application of workpiece coordinates. There are 15 numbers in the drop-down list of the workpiece coordinate system, select the corresponding coordinate system (wobjcoord0~ wobjcoord14), and then the corresponding coordinate value will be displayed in the "Coordinate System Coordinates" below. After selecting a certain coordinate system, click the "Apply" button, and the currently used workpiece coordinate system will change to the selected coordinates, as shown in figure below.

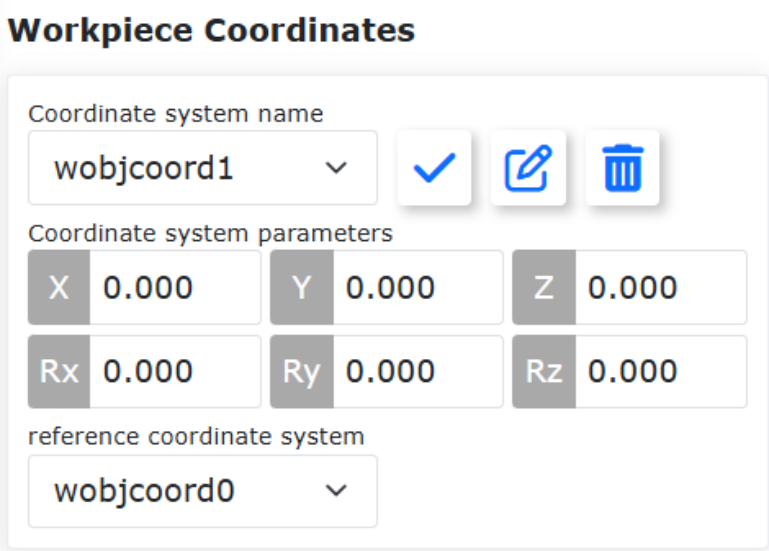


Figure 6.2-6 Set workpiece coordinates

The workpiece coordinate system is generally calibrated based on the tool, and the workpiece coordinate system needs to be established on the basis of the established tool coordinate system. Click “Modify” to reset the workpiece coordinate system of the number according to the prompt. Fix the workpiece and select the calibration method “origin-X-axis-Z-axis” or “origin-X-axis-XY+plane”. The selection of the first two points of the two calibration methods is the same, and the third point is different. One method is to calibrate the Z direction of the workpiece coordinate system, and the second method is to calibrate a point on the XY+ plane, just calibrate according to the diagram. Click the Calculate button to calculate the workpiece pose. If you need to reset it, click Cancel and press the Modify button to re-create the workpiece coordinate system.

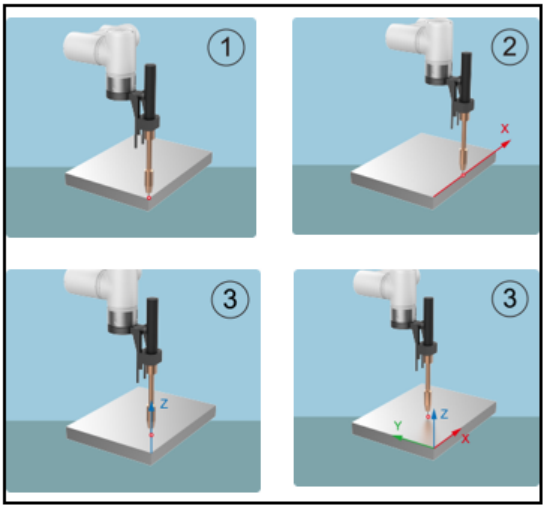


Figure 6.2-7 Schematic diagram of the three-point method

After completing the last step, click “Finish” to return to the workpiece coordinate interface, and click “Save” to store the workpiece coordinate system just created.

Important:

1. The workpiece coordinate system is calibrated based on the tool, and the workpiece coordinate system needs to be established on the basis of the established tool coordinate system.

2. The workpiece coordinate system generally uses wobjcoord1~wobjcoord14, and wobjcoord0 is used to indicate that the origin of the workpiece coordinate system is at the origin of the base coordinates. When calibrating the workpiece coordinate system, it is first necessary to apply the workpiece coordinate system to wobjcoord0, and then select other workpiece coordinate systems for calibration and application.

1.3.6.3 Payload

1.3.6.3.1 End payload

In the menu bar under 'Initial Setup' -> 'Basic' -> 'Load', click 'Trajectory Identification' to enter the trajectory identification interface.

When configuring the end load, please enter the mass of the end tool used and the corresponding center of mass coordinates into the "Load mass" and "Load mass center coordinates X, Y and Z" input boxes and apply.

Important: The load mass cannot exceed the maximum load range of the robot. For the load range corresponding to the specific robot model, please refer to 2.1. Basic parameters.The center of mass coordinate setting range is 0-1000, unit mm.

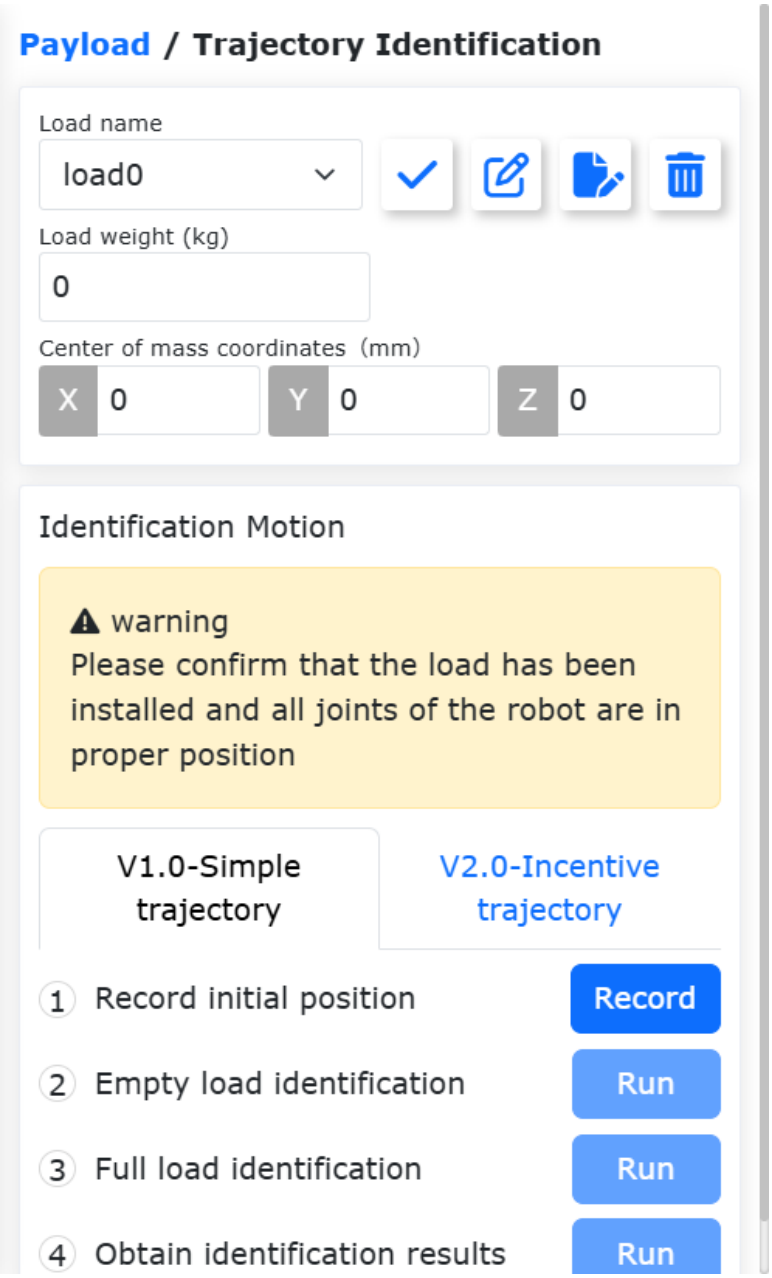


Figure 6.3-1 Schematic diagram of load setting

Important: After the load is installed at the end of the robot, the weight of the end load and the coordinates of the center of mass must be set correctly, otherwise it will affect the drag function of the robot and the use of the collision detection function.

If the user is unsure about the tool mass or center of mass, he can click “Automatic Identification” to enter the load identification function to measure the tool data.

Before taking measurements, make sure the load is installed and then select the version. Click the “Tool Data Measurement” button to enter the load motion test interface.

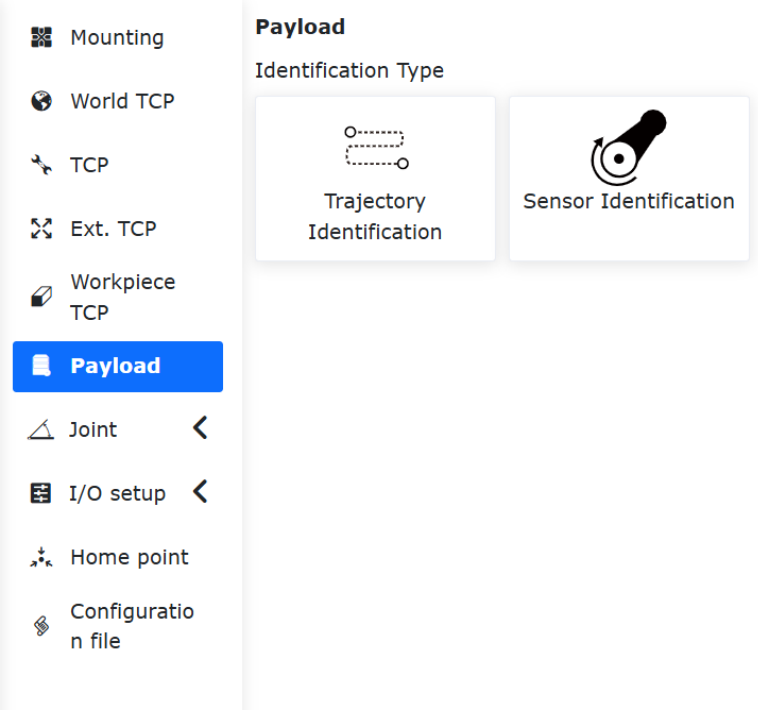


Figure 6.3-2 Load Identification Joint Setup

Click “Load Identification Start” to test. In case of emergency, please stop the movement in time.

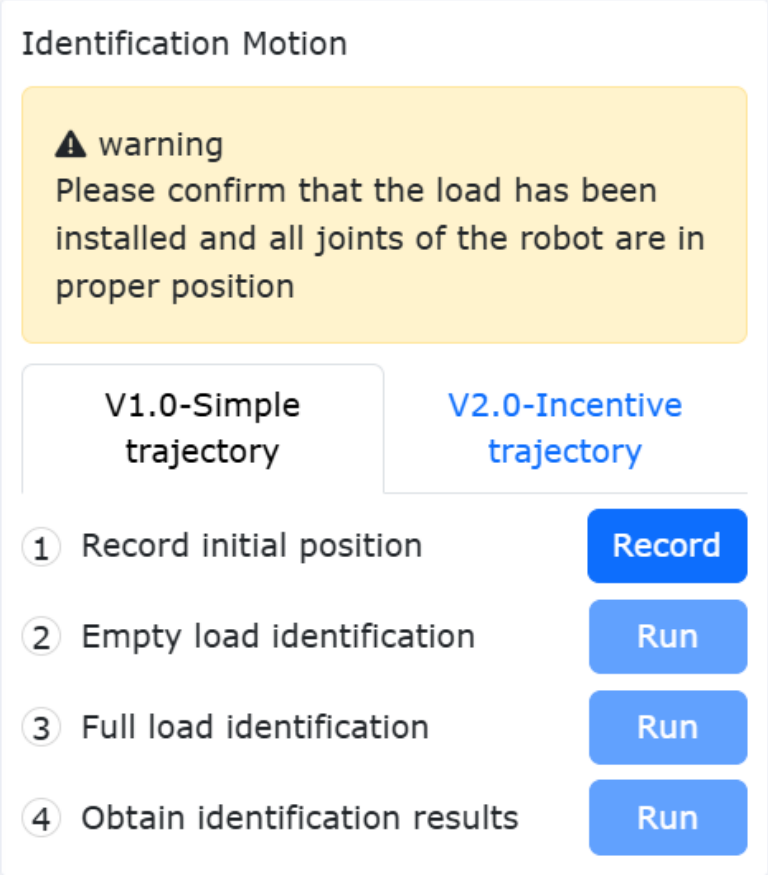


Figure 6.3-3 load identification start

After the exercise is over, click the “Get Identification Result” button to obtain the calculated tool data and display it on the page. If you want to apply it to the load data, click Apply.

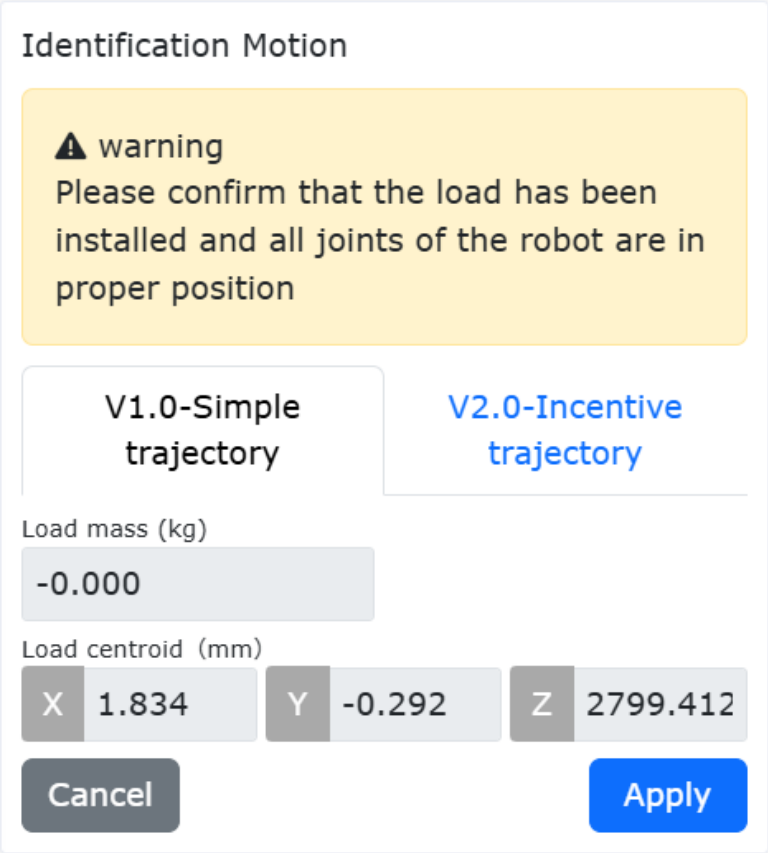


Figure 6.3-4 Load Identification Results

1.3.6.4 Joint

1.3.6.4.1 Soft limit

Under the menu bar of “Initial” -> “Base” -> “Joint”, click “Soft limit” to enter the soft limit interface.

There may be other equipment in the robot’s stroke, and the limit angle can softly limit the robot so that the robot’s movement does not exceed a certain coordinate value and prevent the robot from colliding. Triggering the soft limit to stop the robot is automatically triggered by the robot, and there is no stopping distance.

Administrators can use the default values or enter angle values. Input the angle value to limit the positive and negative angles of the robot joints respectively. When the input value exceeds the robot joint soft limit angle value listed in the robot basic parameter table in 2.1-Basic Parameters, the limit angle will be adjusted to the maximum value that can be set. When the robot reports that the command exceeds the limit, it needs to enter the drag mode and drag the robot joints to within the limit angle. The interface is shown in figure below.

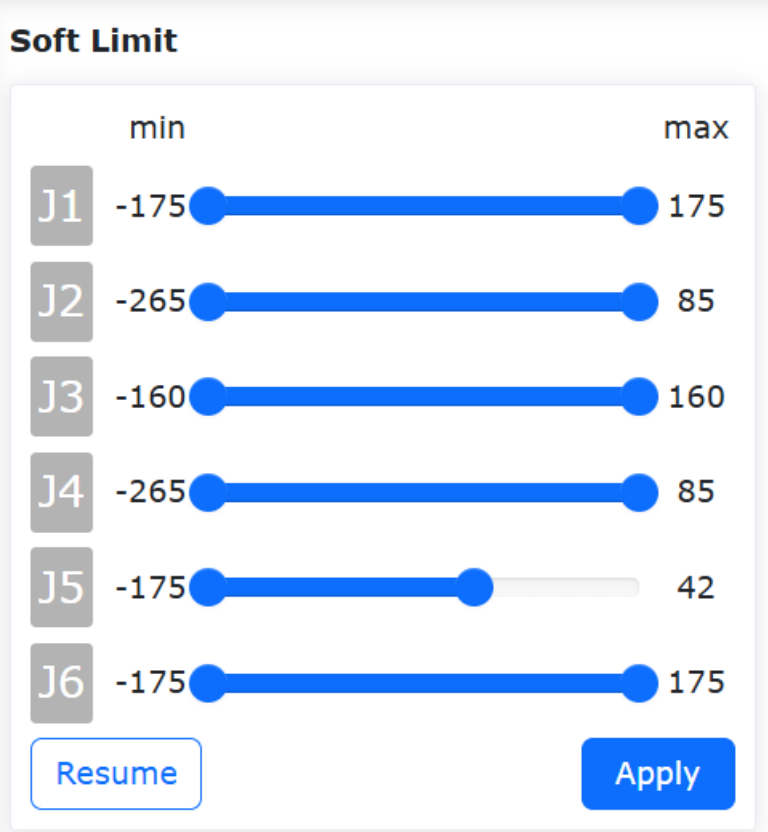


Figure 6.4-1 Schematic diagram of robot limit

1.3.6.4.1.1 Joint Soft Limit Protection

1.3.6.4.1.2 Overview

The Joint Soft Limit Protection function is an active safety mechanism that dynamically restricts operators from exceeding the set soft limit range during drag teaching by real-time monitoring of the robotic arm's joint motion status. This feature ensures soft limits remain effective during drag teaching, thereby enhancing human-robot collaboration safety.

1.3.6.4.1.3 Joint Soft Limit Protection

The Joint Soft Limit Protection function requires matching software package and firmware versions for optimal performance.

1.3.6.4.1.4 Soft Limit Configuration and Function Control

Step1: Log in to the web interface, navigate to “Initial Setup”->”Basic”->”Joints”->”Soft Limits” to access the robot soft limit configuration module.



Figure 6.4-1-2 Robot Soft Limit Configuration Module

Step2: Set appropriate soft limits for each joint based on the robot’s actual working range. Verify whether each joint’s current angular position is within the preset soft limit range. If yes, click “Apply” to implement the preset soft limits; if not, move each joint within the preset range first. Otherwise, an out-of-limit error will appear when clicking “Apply”, as shown below. In this case, perform single-axis jogging or dragging toward the valid range to clear the error.

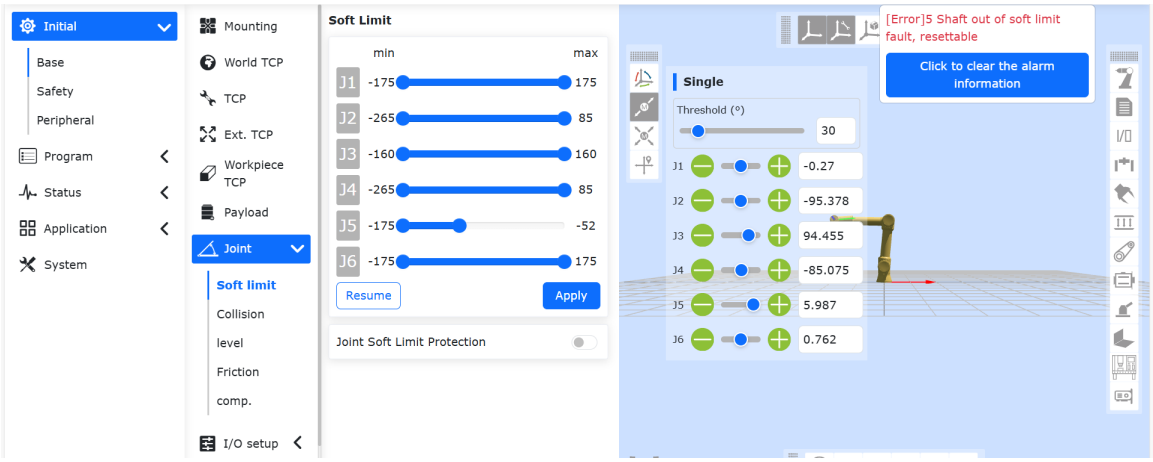


Figure 6.4-1-3 Error Display When Joint Angles Exceed Soft Limit Range

Step3: After successful soft limit configuration, toggle the "Joint Soft Limit Protection" slider to activate this function (see Figure below). During drag teaching, the set soft limits will take effect, and resistance will be felt when approaching the limits.

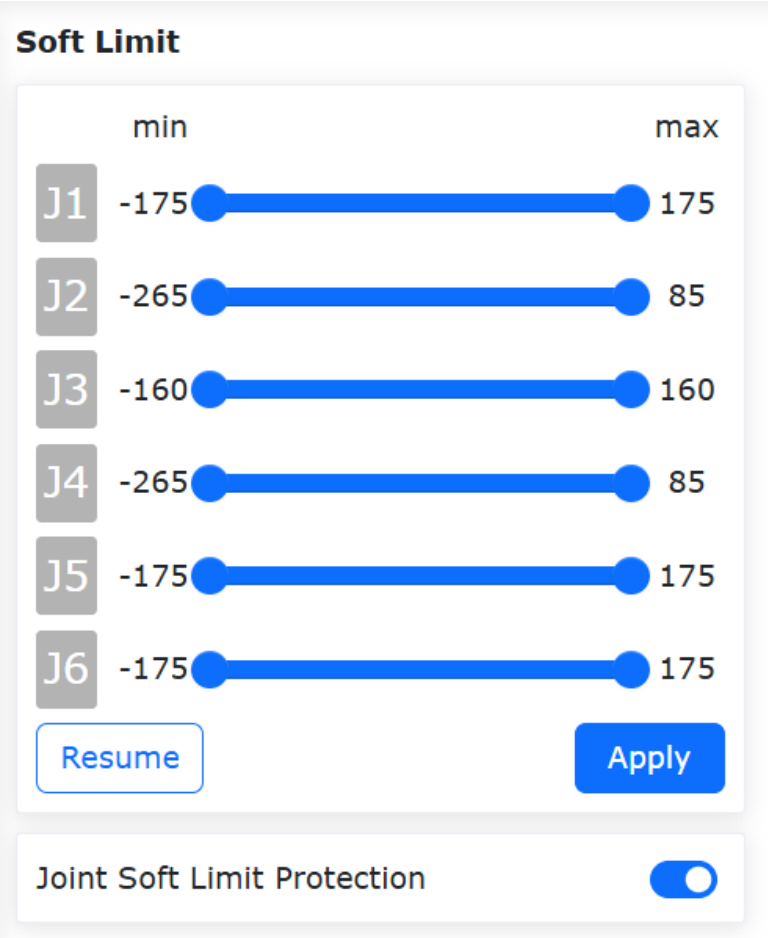


Figure 6.4-1-4 Enabling Joint Soft Limit Protection

Step4: To disable Joint Soft Limit Protection, simply toggle the "Joint Soft Limit Protection" slider.

1.3.6.4.2 Collision level

Under the menu bar of “Initial” -> “Base” -> “Joint”, click “Collision Level” to enter the collision level interface.

The collision level is divided into one to ten levels, and the detection of one to three levels is more sensitive, and the robot needs to run at the recommended speed. At the same time, you can choose to customize the percentage setting, and 100% corresponds to the tenth level. The collision strategy can set the processing method of the robot after the collision, which is divided into error stop and continuous movement, and the user can set it according to the specific use requirements. Such as Figure 6.2-16.

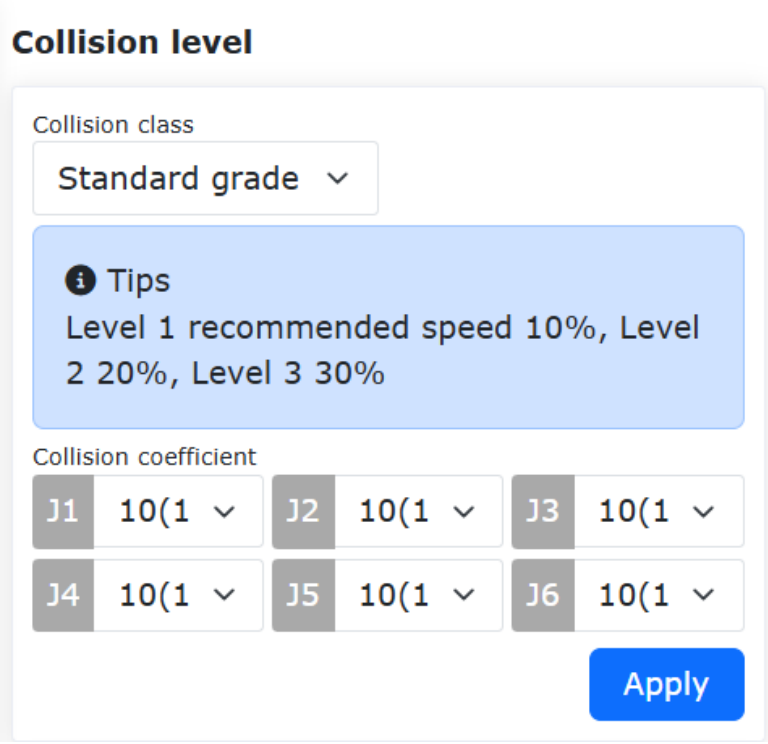


Figure 6.4-2 Schematic diagram of collision level

1.3.6.4.2.1 Post-collision response strategy function

On the basis of the original collision strategy in motion, ” Gravitational moment mode ” and ” Vibration response mode ” are added to ensure the safety of man-machine cooperation.

When the two strategies are triggered, they will switch from automatic mode or manual mode to drag mode. The Gravitational moment mode will move away from the collision point according to the magnitude and direction of the collision force, while the Vibration response mode will return to the collision position after moving away from the collision point. At the same time, collision detection at rest is added.

1.3.6.4.2.2 Collision strategy

FT_Guard command is used to realize collision detection based on force sensor. The previous collision strategies were “collision stop”, “collision pause” and “moving on”. In order to avoid the extrusion force between robot and object after collision, the strategies of “Gravitational moment mode”, “Vibration response mode” and “Collision rebound mode” are added.

When triggered, all three strategies will switch from automatic mode or manual mode to drag mode, and then switch to manual mode. Among them, the Gravitational moment mode will be far away from the collision point according to the magnitude and direction of the collision force; The Vibration response mode will return to the collision position after being far away from the collision point; The collision rebound mode will accelerate away from the collision point according to the set parameters.

1.3.6.4.2.3 Gravitational moment mode

The Gravitational moment mode in collision strategy is set as follows.

Step1:Click “Collision Level” under the menu bar of “Initial” -> “Base” -> “Joint” to enter the corresponding interface.

Step2:In the column of “Collision Strategy”, click the drop-down box to select ” Gravitational moment mode “, and the interface is shown below; Then, click the “Apply” button to enable the function.

Note: In the operation of the robot, if the load mass changes greatly, this strategy is not recommended; This strategy is not recommended if the running speed is too fast.

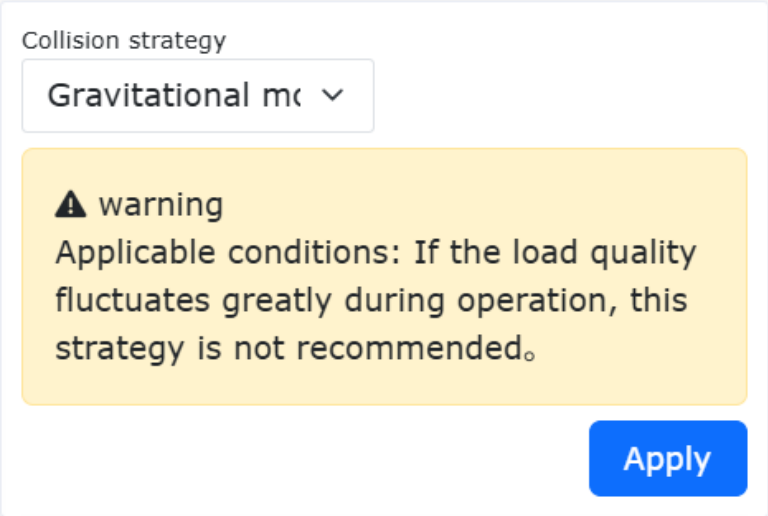


Figure 6.4-3 Gravitational moment mode of Collision Strategy

1.3.6.4.2.4 Vibration response mode

The setting steps of oscillation response mode in collision strategy are as follows.

Step1:Click “Collision Level” under the menu bar of “Initial” -> “Base” -> “Joint” to enter the corresponding interface.

Step2:In the column of “Collision Strategy”, click the drop-down box to select ” Vibration response mode”, and the interface is shown below; Then, click the “Apply” button to enable the function.

Note: It is not recommended to use this strategy if the robot runs too fast.

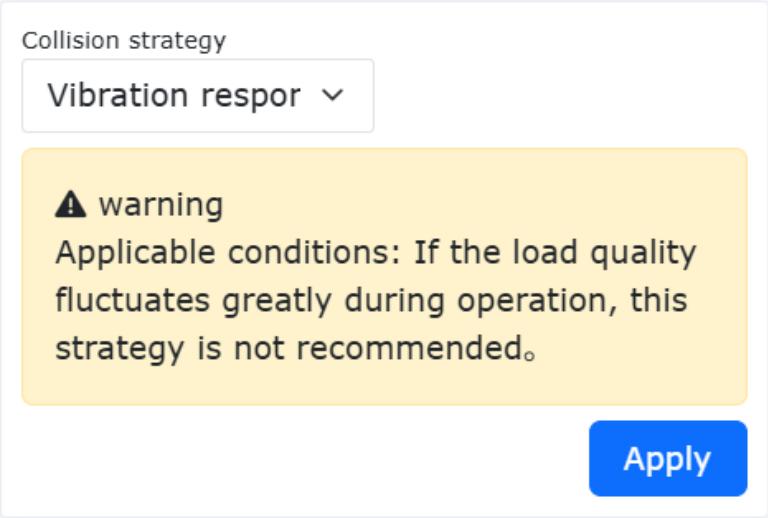


Figure 6.4-4 Vibration response mode of Collision Strategy

1.3.6.4.2.5 Collision rebound mode

The collision rebound mode in collision strategy is set as follows.

Step 1 Click “Collision Level” under the menu bar of “Robot Settings” in the initial setting to enter the corresponding interface.

Step 2 In the column of “Collision Strategy”, click the drop-down box to select “Collision Rebound Mode”, and set the safety time of 1000ms, the safety distance of 150mm, the safety speed of 150mm/s, and the safety factor of each joint is 5. The specific interface is shown below.

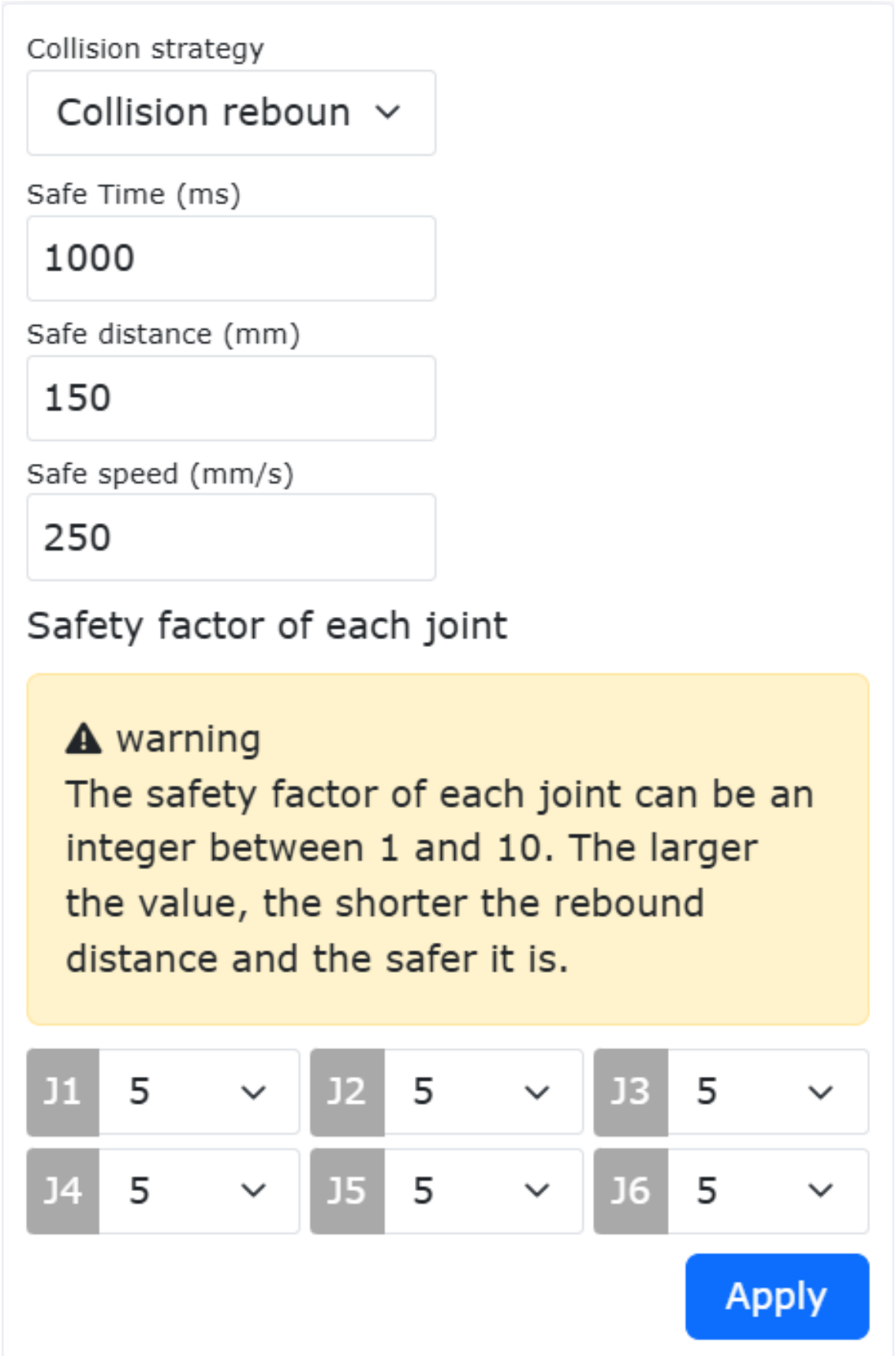

Collision strategy
Collision reboun
Safe Time (ms)
1000
Safe distance (mm)
150
Safe speed (mm/s)
250
Safety factor of each joint
warning
The safety factor of each joint can be an integer between 1 and 10. The larger the value, the shorter the rebound distance and the safer it is.
J1 5
J2 5
J3 5
J4 5
J5 5
J6 5
Apply

Figure 6.4-5 Collision rebound mode of collision strategy

The meaning of each parameter:

- Safe time: indicates the duration in drag mode after switching from automatic mode to drag mode, and the range is [1000-2000] ms;
- Safe distance: indicates the position of the robot away from the collision point after collision, and the range is [150-200] mm;
- Safe speed: indicates the maximum TCP speed of the robot away from the collision point after collision. If the speed limit is exceeded, the rebound force will be restrained, and the range is [50-250] mm/s;
- Safety factor: indicates the attenuation speed of rebound force. The smaller the factor, the faster the attenuation and the faster the rebound speed, and vice versa. The range is [1-10], dimensionless.

1.3.6.4.2.6 FT_Guard Command

FT_Guard command is used to realize collision detection of force sensor. First select the direction of detection, or set all directions. Then, the current force sensor data is obtained as the initial value, and then the maximum threshold and minimum threshold are set to determine the upper and lower limits of collision force triggering, so that the collision detection function setting can be completed. Take the Z-direction configuration as an example. See Figure for detailed settings.

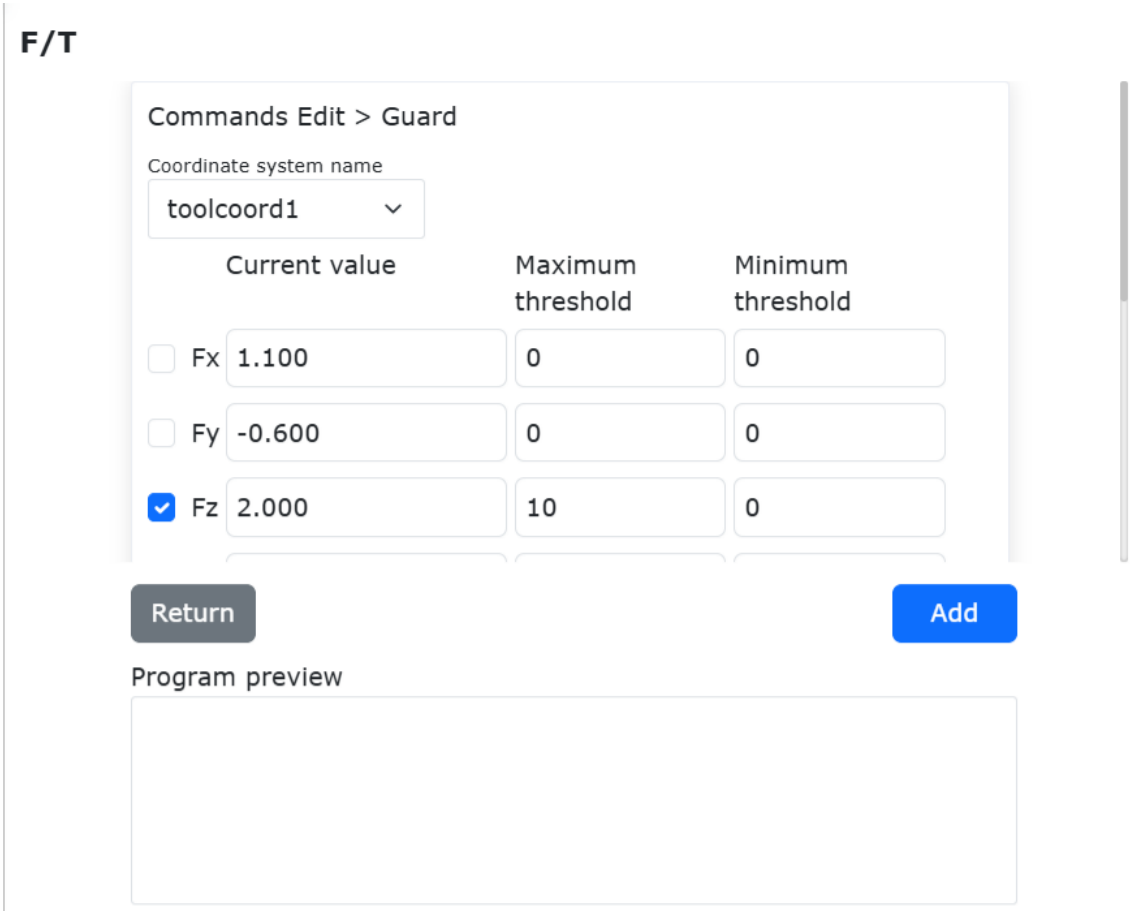


Figure 6.4-6 parameter of FT_Guard command

FT_Guard command is usually used with motion command, such as PTP or LIN, and a simple example is shown below.

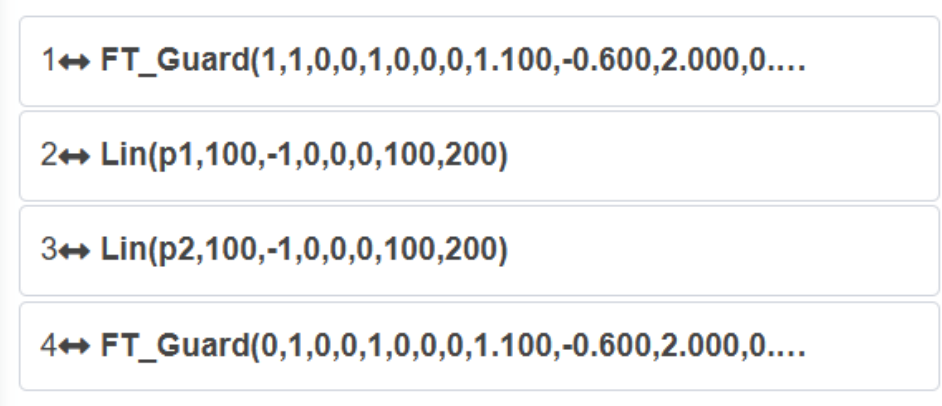


Figure 6.4-7 example of FT_Guard with motion command

The first behavior of figure sets the force sensor collision detection on, and the last behavior turns off the force sensor collision detection function.

1.3.6.4.2.7 Static collision detection

The setup steps of static collision detection are as follows.

Step1:Click "Collision Level" under the menu bar of "Initial" -> "Base" -> "Joint" to enter the corresponding interface.

Step2:Turn on the switch for static collision detection, as shown below. When it is detected that the gap between the joint torque command and the torque feedback is too large, the robot will enter the drag mode to avoid continuous extrusion force.

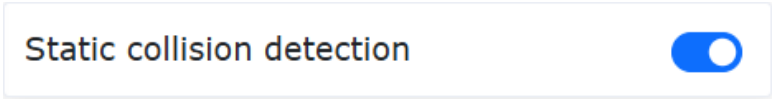


Figure 6.4-8 Static collision detection

1.3.6.4.2.8 Torque Detection Function Before Dragging

1.3.6.4.2.9 Overview

Before the robot enters the drag mode, torque detection is required. The function of this is to prevent abnormal phenomena such as lifting or falling after the robot enters the drag mode due to the operator setting incorrect load parameters or selecting the wrong installation method. If the joint torque is detected to exceed the allowable range, the controller will immediately report an error and prohibit the robot from entering the drag mode.

1.3.6.4.2.10 Torque Detection Before Dragging

Step1: Click "Initial Settings"->"Basic"-> "Joints"->"Collision Level" to enter the collision level setting interface, and turn on the torque detection function before dragging, as shown in Figure 2-1.

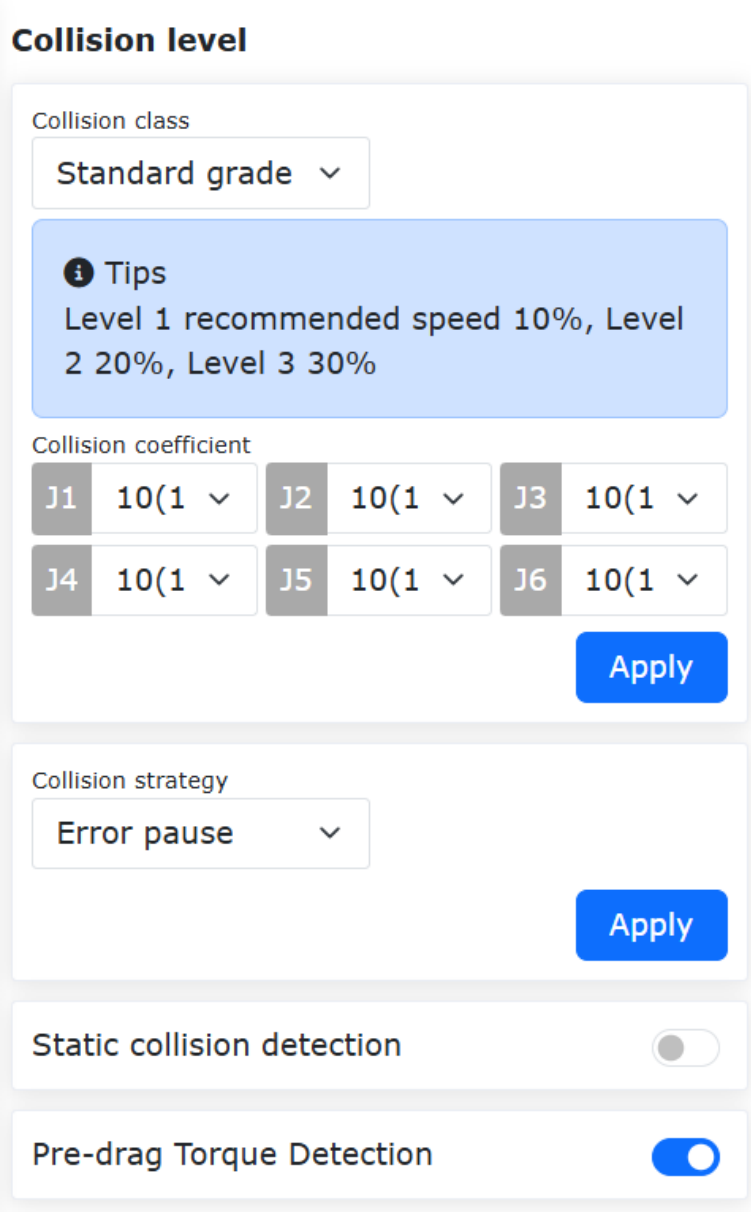


Figure 6.4-9 Torque Detection Function Before Dragging Enabled

Step2: Switch to drag mode. The Web interface enters the drag mode by clicking the robot status area - robot drag status, long pressing the teach pendant "Teach Mode" button, or long pressing the robot end drag button. If the controller reports an error and the robot does not switch to drag mode, as shown in Figure 2-2, check whether the robot load configuration and installation method are correct.

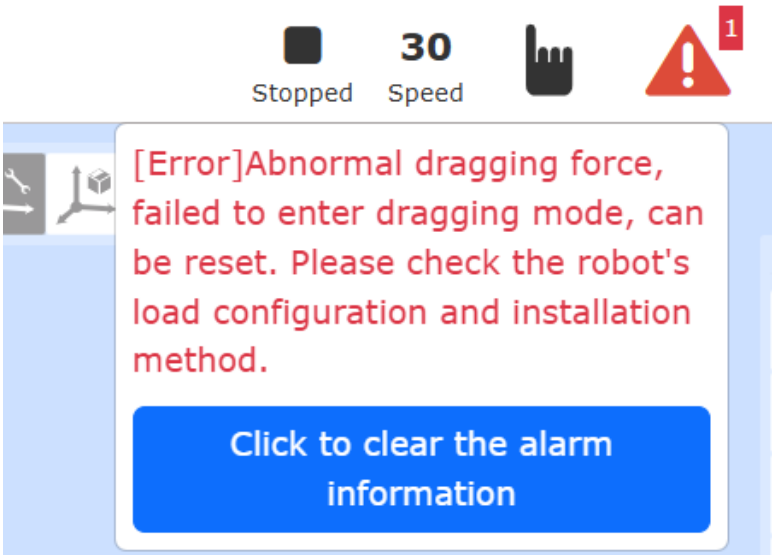


Figure 6.4-10 Torque Exceeds Limit, Controller Reports Error

Step3: Check the load configuration and installation method. Click “Initial Settings”->”Basic”-> “Load”->”End” to check whether the end load configuration on the web interface is the same as the actual installed load; click “Initial Settings”->”Basic”-> “Installation”->”Free Installation” to check whether the installation method on the web interface is the same as the actual installation method.

1.3.6.4.2.11 False Alarm Detection Function

1.3.6.4.2.12 Overview

Collision function optimization builds upon collision detection by adding a false alarm switch, which can help avoid the risk of false collision alarms.

1.3.6.4.2.13 Collision Level Setting

Step 1: Log in to the Web interface, then click “Initial Setup” → “Basic” → “Joint” → “Collision Level” in sequence to enter the collision level setting module.

The higher the collision level, the greater the torque required for a collision to occur, making the collision response less sensitive. The torque values corresponding to the current collision levels, such as 38.4 N·m for Level 10, represent the theoretical joint torque required to trigger a collision on Axis 1.

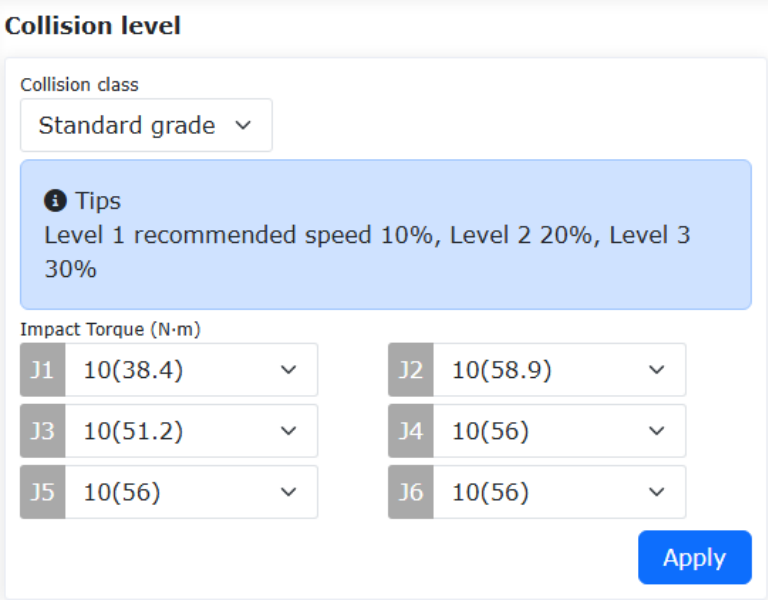


Figure 6.4-11 Collision Level Setting Module

Step 2: The false alarm detection switch is enabled by default. If not needed, the detection switch can be turned "Off", as shown in the figure below.

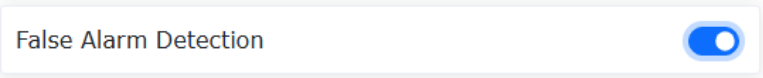


Figure 6.4-12 False Alarm Detection Switch

1.3.6.4.3 Friction compensation

Under the menu bar of "Initial" -> "Base" -> "Joint", click "Friction comp." to enter the friction compensation setting interface.

Friction compensation coefficient:The usage scenario for friction compensation is only in the dragging mode. The friction compensation coefficient can be set from 0 to 1. The higher the value, the greater the compensation force when dragging. The friction compensation coefficient needs to be set separately for each axis according to the different installation methods.

Friction compensation switch:Users can turn on or off friction compensation according to the actual robot and usage habits.

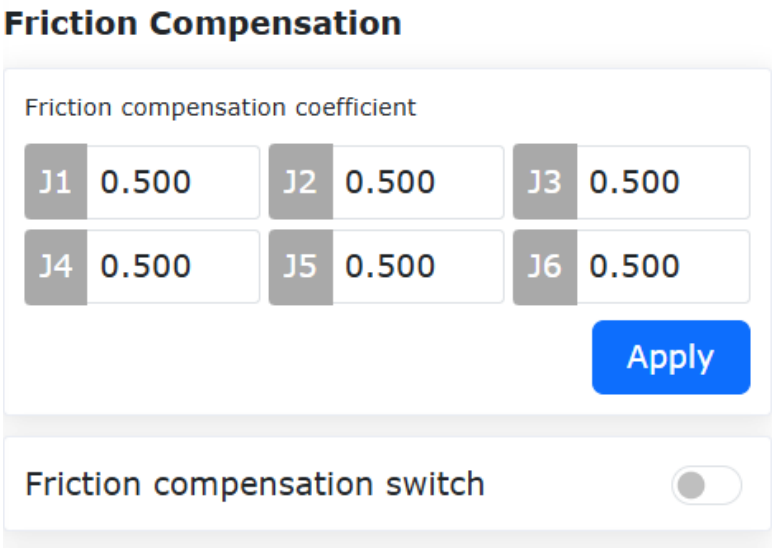


Figure 6.4-11 Friction Compensation Settings

Important: The friction compensation function of the robot needs to be used with caution. According to the actual situation, a reasonable compensation coefficient should be set. Generally, the recommended median value is about 0.5.

1.3.6.4.4 Friction Compensation Coefficient Adjustment Function

1.3.6.4.4.1 Overview

The friction compensation coefficient adjustment function is primarily used to adjust the magnitude of the friction compensation value within the controller.

In drag mode, adjusting the friction compensation coefficient can make robot dragging smoother; in automatic mode, it can improve the goodness of fit between the torque command curve and the torque feedback curve.

1.3.6.4.4.2 Friction Compensation Coefficient Adjustment

The factory default for the friction compensation coefficient is 0.5, which is a general parameter. Users can adjust the friction compensation gain according to their actual conditions to achieve a better experience.

1.3.6.4.4.3 Friction Compensation Coefficient Configuration

Step1: Log in to the web interface. Click sequentially on “Initial Setup” → “Basic” → “Joint” → “Friction Compensation” to enter the friction compensation settings module.

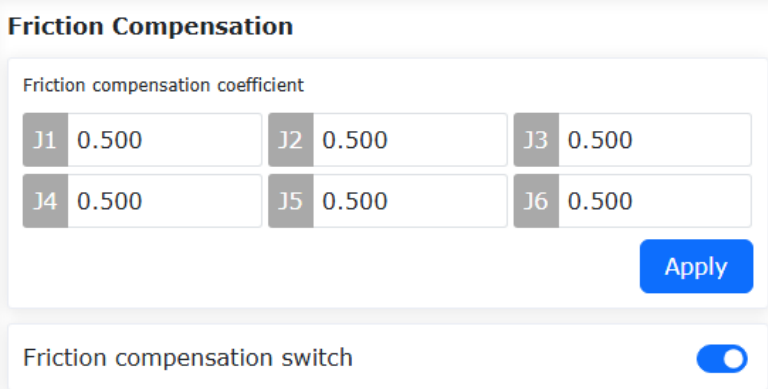


Chart 6.4-12 Friction Compensation Settings Module

Step2: The friction compensation coefficient defaults to 0.5. When the friction compensation switch is turned on as shown in the figure, the dragging feel becomes smoother compared to when it is off. If the function is not enabled, the dragging feel is heavier.

Step3: Parameter Adjustment. The range for the friction compensation coefficient is [0-1]. If the dragging feels slightly heavy, you can increase the parameter for each axis based on the actual situation. If the joint fails to stop during dragging or joint vibration occurs, the parameter for the corresponding axis needs to be decreased.

Step4: To disable the friction compensation function, select "Off" via the compensation switch.

1.3.6.4.5 Dragging Force Compensation

1.3.6.4.5.1 Overview

Dragging force optimization builds upon the current current-loop dragging by compensating additional torque based on the robot's motion trend. This compensates for torque errors introduced by modeling inaccuracies and other factors, resulting in smoother robot dragging.

1.3.6.4.5.2 Robot Dragging Force Optimization

The dragging force optimization feature requires matching software and firmware versions to ensure optimal performance.

1.3.6.4.5.3 Dragging Force Optimization Configuration

Step1: Log in to the web interface, navigate to "Initial Setup" -> "Basic" -> "Joints" -> "Friction Compensation" to access the dragging force compensation settings module, as shown in the figure.

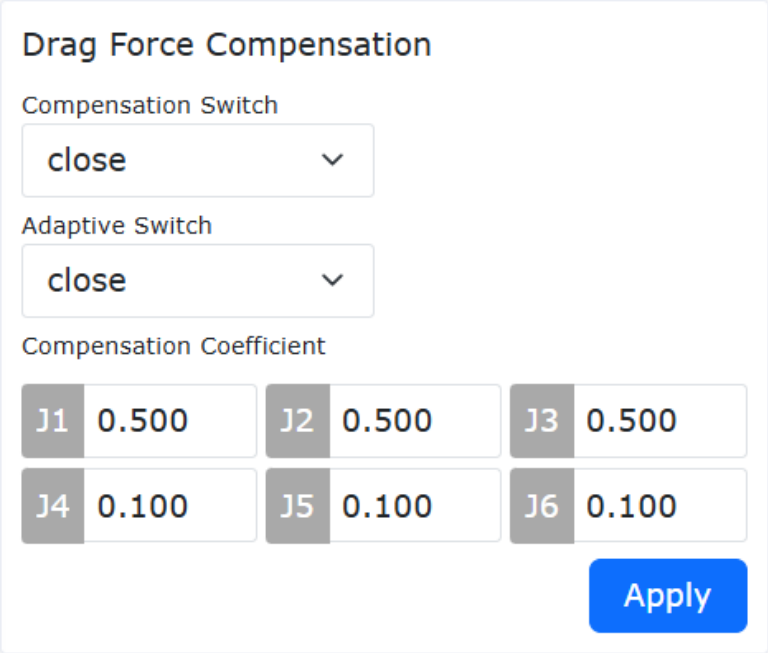


Figure 6.4-13 Dragging Force Compensation Settings Module

Step2: Set "Compensation Switch" to "ON" and "Adaptive Switch" to "OFF", configure parameters as shown in Figure 2-1, then click "Apply" to successfully enable the function. Press the drag button to drag the robot, and the dragging feel will be noticeably smoother than before enabling the function.

Step3: Parameter adjustment - the compensation coefficient range is [0-1]. If dragging feels slightly heavy, increase the parameter for the corresponding axis. If the robot fails to stop during dragging or joint vibration occurs, decrease the parameter for the corresponding axis. During dragging, a damping sensation will be present to decelerate and stop the robot.

Step4: To disable the dragging force compensation function, set "Compensation Switch" to "OFF".

1.3.6.5 I/O setup

1.3.6.5.1 Pausing LUA Program

During the execution of a robot LUA program, clicking the "Pause/Resume" button will pause the LUA program, and the robot's status will change to "Pause". Clicking the button again will resume the program from the paused position, and the robot's status will return to "Running".

All running background processes will also be paused and resumed synchronously during this operation. Different types of LUA instructions behave differently when paused:

Motion Commands: The robot stops immediately when paused. Upon resuming, the robot will continue moving to the target position specified by the command.

Logic Commands (SetDO, GetDI, GetInverseKinRef, etc.): If the program is paused during execution, these commands will complete their current operation before waiting for the LUA program to resume before proceeding to the next command.

Wait Commands (WaitDI, ModbusMasterWaitDI, etc.): If paused during a wait operation, the pause duration does not count toward the wait timeout period.

Sleep Commands (sleep_ms, WaitMs): If paused during a sleep operation, the pause duration does not count toward the specified sleep time.

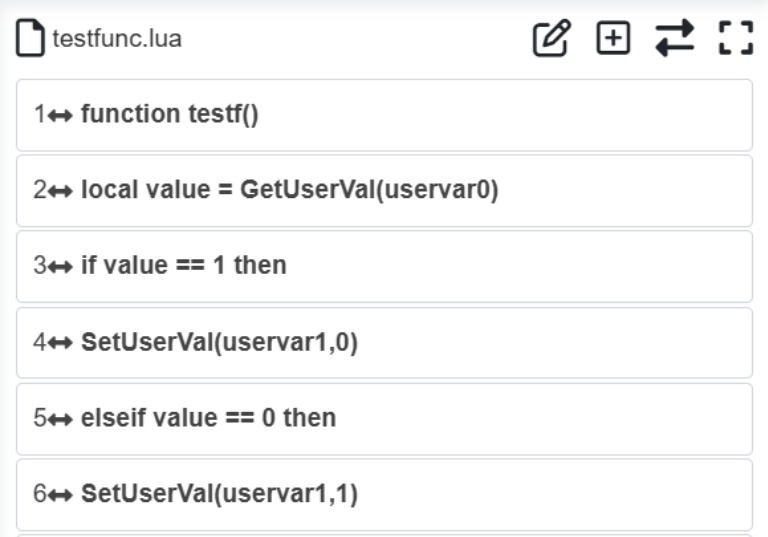


Figure 6.5-1 LUA Program Paused State

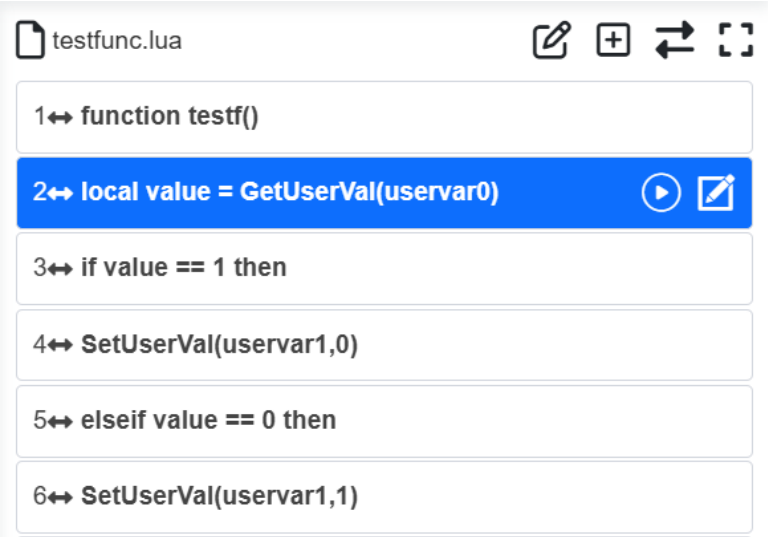


Figure 6.5-2 LUA Program Running State

1.3.6.5.2 I/O configuration

Click “Initial” -> “Base” -> “I/O setup” on the menu bar, and click the “DI Configuration” and “DO Configuration” submenus respectively to enter the DI and DO configuration interface. Among them, the control box CI0-CI7 and CO0-CO7 are configurable, and the terminal DI0 and DI1 are configurable.

1.3.6.5.2.1 DI Configuration

In production, when the collaborative robot needs to connect peripherals or stops suddenly due to failure or other factors, it needs to output DO signal to realize sound and light alarm prompt. The input configurable functions are shown in table below.

Table 6.5-1 Configurable functions of control box input

Function number	Function name	Function description
0	None	None
1	Arc start success signal	The arc starts successfully and the robot outputs the arc start signal to the welder
2	Welding preparation signal	Robot welding preparation success signal
3	Conveyor detection	Conveyor detection switch DI configuration signal
4	Pause	Robot motion pause signal during welding
5	Resume	When the arc is interrupted unexpectedly during robot welding or the operator actively pauses welding, welding interruption will be triggered. After welding interruption, when the external input signal to the robot changes from invalid to valid, the robot automatically resumes welding from the original interrupted position
6	Start	In the DI configuration configurable input, select CIO as “Start” and click “Apply”. The configurable input valid state can be selected as “high level valid”. When the CI0 level changes from low level to high level, the “start” function is triggered to start the program opened in the current teaching program interface. If the interface is not opened, the last saved program will be run; the configurable input valid state can be selected as “low level valid”. When the CI0 level changes from high level to low level, the “start” function is triggered to start the program opened in the current teaching program interface. If the interface is not opened, the last saved program will be run
7	Stop	The robot stops the motion signal during welding
8	Pause/resume	After the robot moves, the pause/resume motion signal is triggered cyclically
9	Start/stop	After the robot moves, the start/stop motion signal is triggered cyclically
10	Foot drag switch	Robot foot drag switch motion signal
11	Move to the operation origin	The robot moves to the operation origin signal with the current robot posture as the operation origin
12	Manual automatic switching (pulse signal)	In the configurable input of DI configuration, select CIO as “Manual automatic switching (pulse signal)”, and click “Apply”. Select “High level valid” for the configurable input valid state. When the CI0 level changes from low level to high level, the “Manual automatic switching (pulse signal)” function is triggered, and the robot switches the running state once; select “Low level valid” for the configurable input valid state. When the CI0 level changes from high level to low level, the “Manual automatic switching (pulse signal)” function is triggered, and the robot switches the running state once
13	Welding wire positioning success	Robot welding wire positioning success signal
14	Motion interruption	Robot motion program interruption signal
15	Start main program	Start robot main program signal
16	Start rewinding	After the robot program is running, the program rewinding start signal.
17	Start confirmation	Robot program start confirmation signal

continues on next page

Table 1 – continued from previous page

Function number	Function name	Function description
18	Laser detection signal X	Robot laser sensor detection signal X
19	Laser detection signal Y	Robot laser sensor detection signal Y
20	External emergency stop input signal 1	Robot external emergency stop input signal 1, only displayed under QX. Under LA, relevant configuration can be made in the "Initial Settings" -> "Safety" -> "I/O Safety" -> "DIO Safety Function Configuration" interface
21	External emergency stop input signal 2	Robot external emergency stop input signal 2, only displayed under QX. Under LA, relevant configuration can be made in the "Initial Settings" -> "Safety" -> "I/O Safety" -> "DIO Safety Function Configuration" interface
22	Level 1 reduction mode	Robot level 1 reduction mode, only displayed under QX. In LA, you can configure the relevant settings in the "Initial Settings" -> "Safety" -> "I/O Safety" -> "DIO Safety Function Configuration" interface
23	Secondary reduction mode	Robot secondary reduction mode, only displayed in QX. In LA, you can configure the relevant settings in the "Initial Settings" -> "Safety" -> "I/O Safety" -> "DIO Safety Function Configuration" interface
24	Third-level reduction mode (stop)	Robot third-level reduction mode (stop), only displayed in QX. Under LA, you can make relevant configurations in the "Initial Settings" -> "Safety" -> "I/O Safety" -> "DIO Safety Function Configuration" interface
25	Resume welding	After the robot is interrupted from welding, the welding operation signal is restored
26	Stop welding	During the robot welding process, the welding operation signal is stopped
27	Auxiliary drag on	Control box DI function configuration force sensor drag function on signal
28	Auxiliary drag off	Control box DI function configuration force sensor drag function off signal
29	Auxiliary drag on/off	Control box DI function configuration force sensor drag function, cycle on/off signal
30	Clear all errors	Clear all error signals triggered by the robot
31	Manual automatic switching (high and low level)	In the configurable input of DI configuration, select CIO as "Manual automatic switching (high and low level)" and click "Apply". The configurable input valid state can be selected as "high level valid". When CI0 is switched to a high level, the "manual automatic switching (high and low levels)" function is triggered, and the robot state is switched to the automatic state; the configurable input valid state can be selected as "low level valid". When CI0 is switched to a low level, the "manual automatic switching (high and low levels)" function is triggered, and the robot state is switched to the automatic state.
32	Enable	Controls robot enable
33	Disable	Controls robot disable
34	Enable/Disable (Rising/Falling Edge)	Rising/Falling edge of input signal triggers robot enable/disable respectively

1.3.6.5.2.2 Adding Configurable Functions to End CI

1.3.6.5.2.3 Overview

Synchronizing all the functions of the robot control box CI to the end CI aims to build a control system that is logically equivalent and physically complementary. The two interfaces are completely equivalent in logical functions and can be used in parallel or selectively, enabling the robot control system to intelligently allocate signal paths according to task scenarios, equipment physical layout, and reliability requirements.

1.3.6.5.2.4 Operation Procedure

Step1: Sequentially click the menu bar buttons “Initial Settings” - “Basic” - “I/O Settings” - “DI”, etc., to enter the DI configuration interface. Select End DI0 and End DI1 to configure the functions of the robot end input.

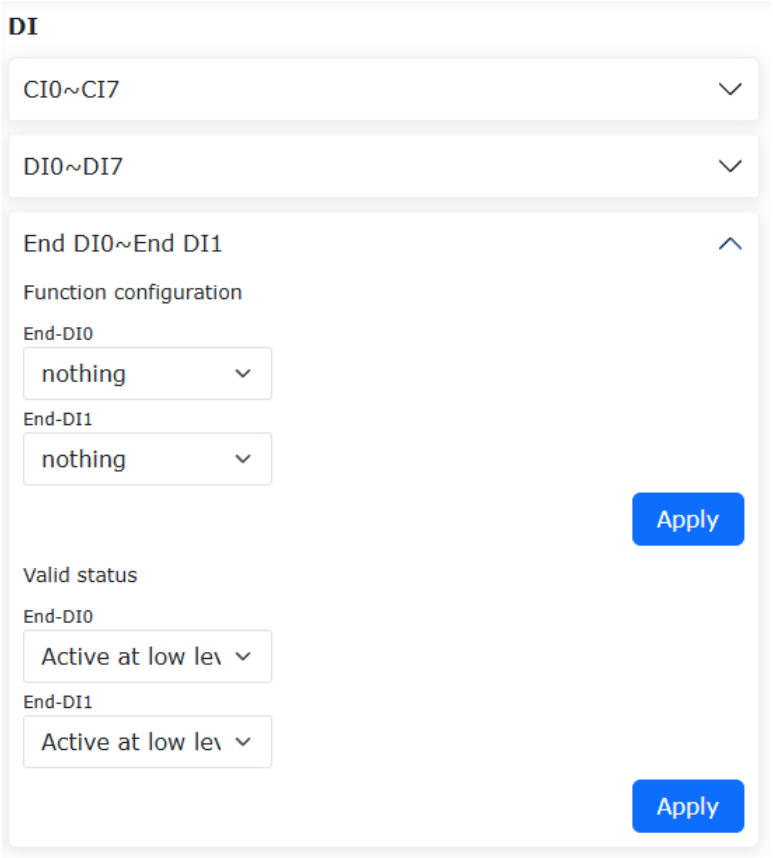


Figure 6.5-3 End DI Parameter Configuration

Step2: The DI signals supported by the robot end are listed in the table below. Users can configure the corresponding signals according to actual usage requirements.

Table 6.5-2 End Input Configurable Functions

Function Number	Function Name	Function Description
0	None	None
1	Drag Mode	Signal to enable drag mode at the robot end
2	Teaching Point Recording	Signal to enable teaching point recording at the robot end, saving the current robot position data
3	Manual/Auto Toggle	Trigger signal for robot manual/automatic mode switching
4	TPD Trajectory Record Start/Stop	After the robot starts TPD motion, the signal to start/stop trajectory recording
5	Pause	Signal to pause robot motion
6	Resume	Signal to resume robot motion
7	Start	Signal to start the robot program
8	Stop	Signal to stop the robot program
9	Pause/Resume	After robot motion, cyclically triggers pause/resume motion signal
10	Start/Stop	After robot motion, cyclically triggers start/stop motion signal
11	Assisted Drag Enable	Control box DI function configures the force sensor drag function enable signal
12	Assisted Drag Disable	Control box DI function configures the force sensor drag function disable signal
13	Assisted Drag Enable/Disable	Control box DI function configures the force sensor drag function, cyclically enable/disable signal
14	Laser Detection Signal X	Robot laser sensor detection signal X
15	Laser Detection Signal Y	Robot laser sensor detection signal Y
16	Move to Operation Origin	Signal to move the robot to the operation origin
17	Motion Interruption	Signal to interrupt the robot motion program
18	Start Main Program	Signal to start the robot main program
19	Start Rewind	After the robot program runs, signal to start program rewind
20	Start Confirmation	Signal to confirm robot program start
21	Resume Welding	After a welding interruption occurs on the robot, signal to resume welding operation
22	Terminate Welding	During robot welding process, signal to terminate welding operation
23	Error Message Clear	Clears all error signals triggered by the robot
24	Manual/Auto Toggle (High/Low Level)	When the configurable input selects “High Level Active”, the robot switches to automatic when the input signal is high level; when the configurable input selects “Low Level Active”, the robot switches to automatic when the input signal is low level
25	Enable	Controls robot enabling
26	Disable	Controls robot disabling
27	Enable/Disable (Rising/Falling Edge)	Rising and falling edges of the valid input signal state respectively trigger robot enable and disable actions
28	Laser Servo Tracking Start/Stop Signal	When robot laser recording and tracking is enabled and the I/O start/stop function is turned on, triggering the corresponding end CI starts laser tracking; releasing the corresponding end CI ends tracking

The default configuration of the control box: CO0 is 1-robot error, CO1 is 2-robot in motion.

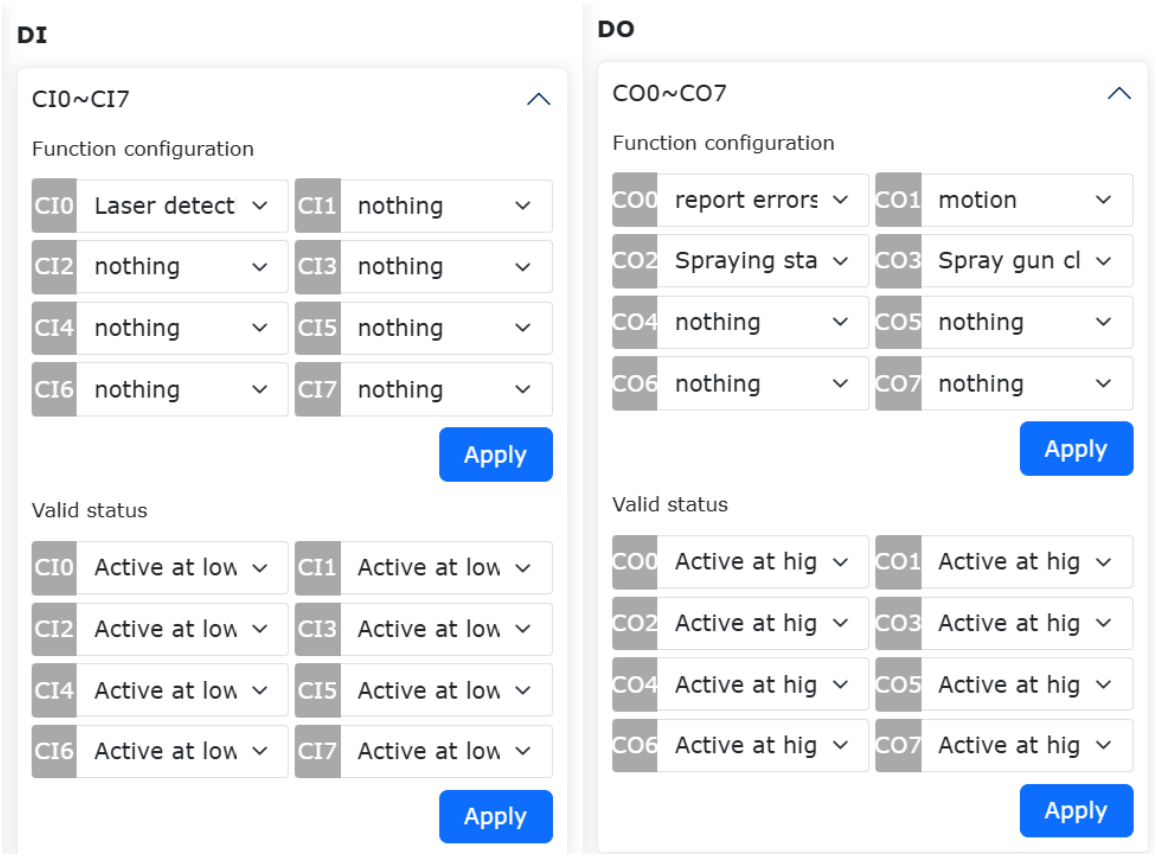


Figure 6.5-3 Control box DI and DO configuration

Terminal DI default configuration: DI0 drag teaching, DI1 teaching point recording.

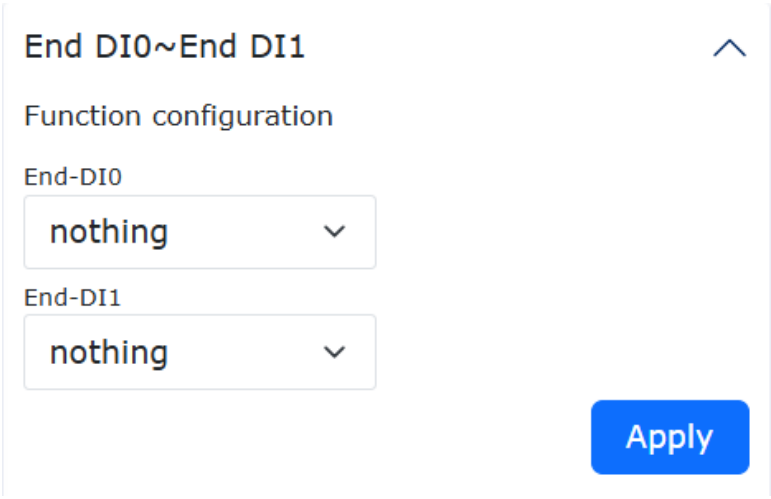
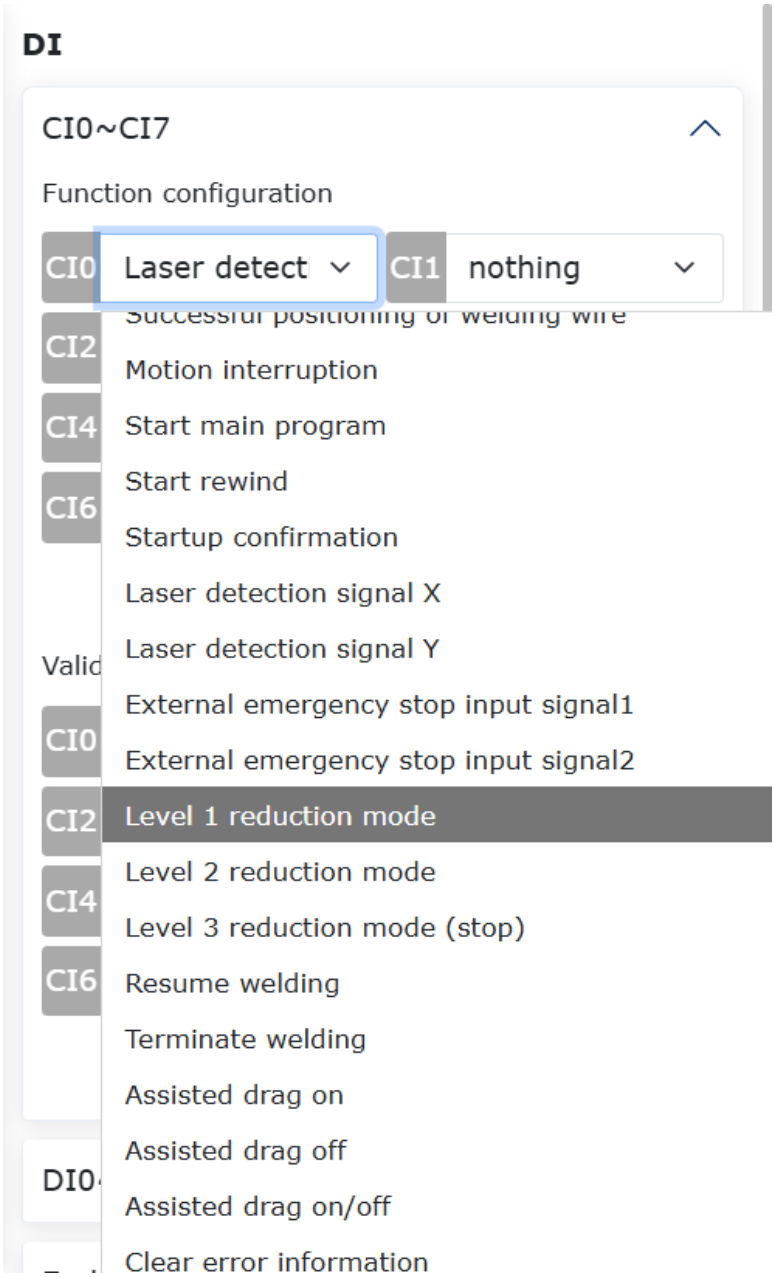


Figure 6.5-4 Terminal DI configuration

After configuration is completed, the corresponding output DO status can be viewed in the control box I/O page under the corresponding state.

Important: The configured DI and DO are prohibited from being used in program programming.

Reduction mode configuration (first level, second level, third level): The first and second level reduction modes can configure the joint speed and terminal TCP speed, and the third level reduction mode is stop and the speed does not need to be configured.



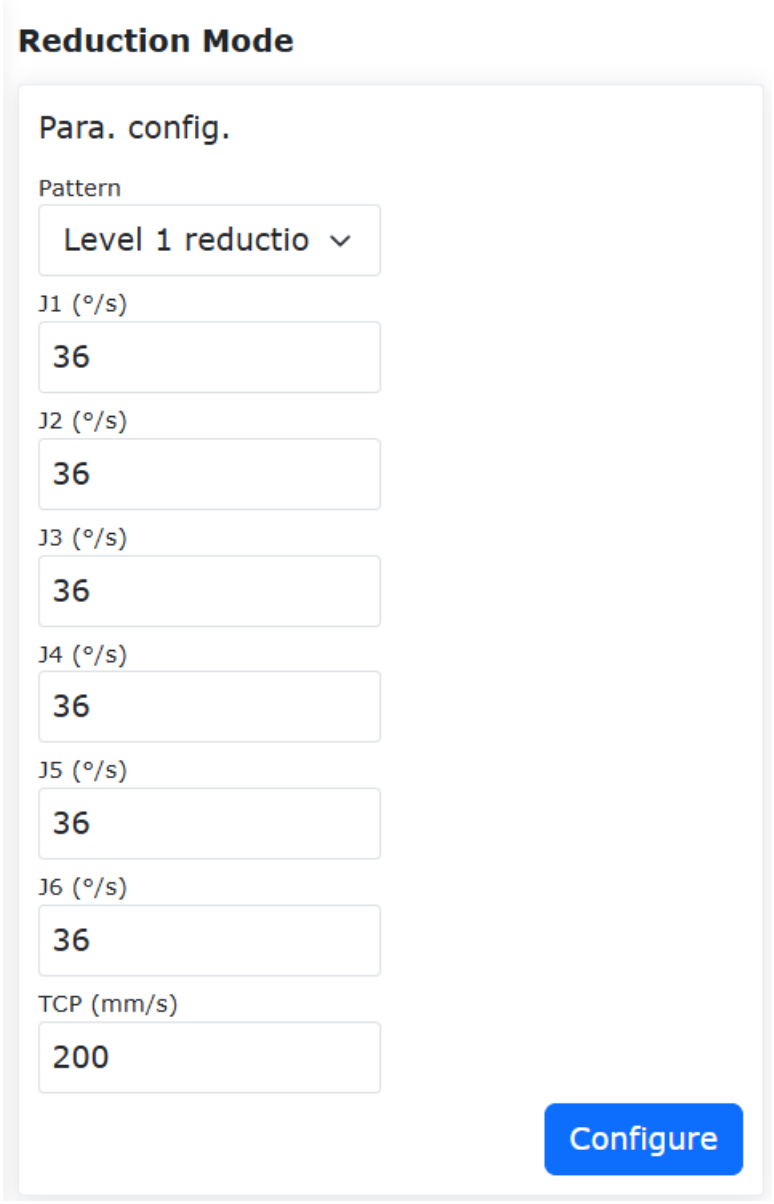


Figure 6.5-5 Reduced mode configuration

1.3.6.5.2.5 DO configuration

The output configurable functions are shown in the following table:

Table 6.5-3 Control box output configurable functions

Function number	Function name	Function description
0	None	None
1	Error	DO output error signal

continues on next page

Table 2 – continued from previous page

Function number	Function name	Function description
2	Movement	Robot movement signal
3	Spray start and stop	Robot spray start and stop operation signal
4	Spray gun cleaning	Robot spray gun cleaning operation signal
5	Arc start	The robot controls the DO output port of the welding machine arc start. When the robot program executes the arc start command, the DO output port corresponding to the welding machine arc start automatically outputs valid
6	Gas supply	The robot controls the DO output port of the welding machine gas supply. When the robot executes the welding gas supply command, the DO output port corresponding to the gas supply automatically outputs valid
7	Forward wire feeding	The robot controls the DO output port of the welding machine forward wire feeding. When the robot executes the forward wire feeding command, the DO output port corresponding to the forward wire feeding automatically outputs valid
8	Reverse wire feeding	The robot controls the DO output port of the welding machine to feed the wire in reverse. When the robot executes the reverse wire feeding command, the DO output port corresponding to the reverse wire feeding automatically outputs and becomes valid
9	JOB input port 1	JOB input port 1 signal
10	JOB input port 2	JOB input port 2 signal
11	JOB input port 3	JOB input port 3 signal
12	Conveyor start and stop	Conveyor movement start and stop operation signal
13	Pause	Robot movement pause signal
14	Arrival at the operation origin	Robot movement to the operation origin signal
15	Enter the interference zone	Robot movement to the interference zone signal
16	Wire position finding start and stop control	Robot wire position finding start and stop control operation signal
17	Robot start completion	Robot start completion signal
18	Program start stop	Robot motion program start stop signal
19	Automatic manual mode	Robot hand automatic mode switching signal
20	Emergency stop output signal 1	Robot emergency stop output signal 1, only displayed under QX. Under LA, you can configure it in the “Initial Settings” -> “Safety” -> “I/O Safety” -> “DIO Safety Function Configuration” interface
21	Emergency stop output signal 2	Robot emergency stop output signal 2, only displayed under QX. Under LA, you can configure it in the “Initial Settings” -> “Safety” -> “I/O Safety” -> “DIO Safety Function Configuration” interface
22	Lua script program run/stop	Robot motion Lua script program run/stop signal
23	Safety status output	Robot safety status output signal, only displayed under QX. In LA, you can configure the relevant settings in the “Initial Settings” -> “Safety” -> “I/O Safety” -> “DIO Safety Function Configuration” interface
24	Protective stop status output	Robot protective stop status output signal, only displayed in QX. In LA, you can configure the relevant settings in the “Initial Settings” -> “Safety” -> “I/O Safety” -> “DIO Safety Function Configuration” interface

continues on next page

Table 2 – continued from previous page

Function number	Function name	Function description
25	Robot in motion	Robot in motion status signal, only displayed in QX. In LA, you can configure the relevant settings in the "Initial Settings" -> "Safety" -> "I/O Safety" -> "DIO Safety Function Configuration" interface
26	Robot reduction mode	Robot reduction mode signal, only displayed in QX. In LA, you can configure the relevant settings in the "Initial Settings" -> "Safety" -> "I/O Safety" -> "DIO Safety Function Configuration" interface
27	Robot non-reduction mode	Robot non-reduction mode signal, only displayed in QX. Under LA, you can configure the relevant settings in the "Initial Settings" -> "Safety" -> "I/O Safety" -> "DIO Safety Function Configuration" interface
28	Reserved	Reserved
29	Command point error	Joint command point error signal
30	Drive error	Drive error signal
31	Soft limit error	Robot exceeds soft limit error signal, need to adjust the corresponding joint soft limit
32	Collision error	Robot collision error signal
33	Active slave number error	Active slave number error abnormal signal
34	Slave error	Slave abnormal error signal
35	I/O error	I/O error signal
36	Gripper error	Gripper related configuration abnormal signal
37	File error	Configuration file loading error signal
38	Singular position error	Error signal when the robot moves to a singular position
39	Driver communication error	Abnormal communication error signal of the robot driver
40	Parameter error	DO high and low level range error
41	External axis exceeds soft limit error	External axis 1-4 exceeds soft limit fault signal
42	Planning & Timeout Warning	Robot planning and timeout alarm status
43	Safety Door Warning	Safety door trigger status
44	Motion Warning	Motion warning status
45	Interference Zone Warning	Robot entering interference zone warning
46	Safety Wall Warning	Robot entering safety wall warning
47	Robot Enabled	Robot enable status

1.3.6.5.3 Control box DO high and low effective configurable function

1.3.6.5.3.1 Overview

During the entire process from the control box power-on to the robot enabling, DO can be configured to the required output state according to the specific usage scenario, which is more flexible and convenient to use.

1.3.6.5.3.2 Operation steps

Enter Initial Settings->Basics->I/O Settings->DO interface, and configure the control box DO output during power-on to the required high/low level.

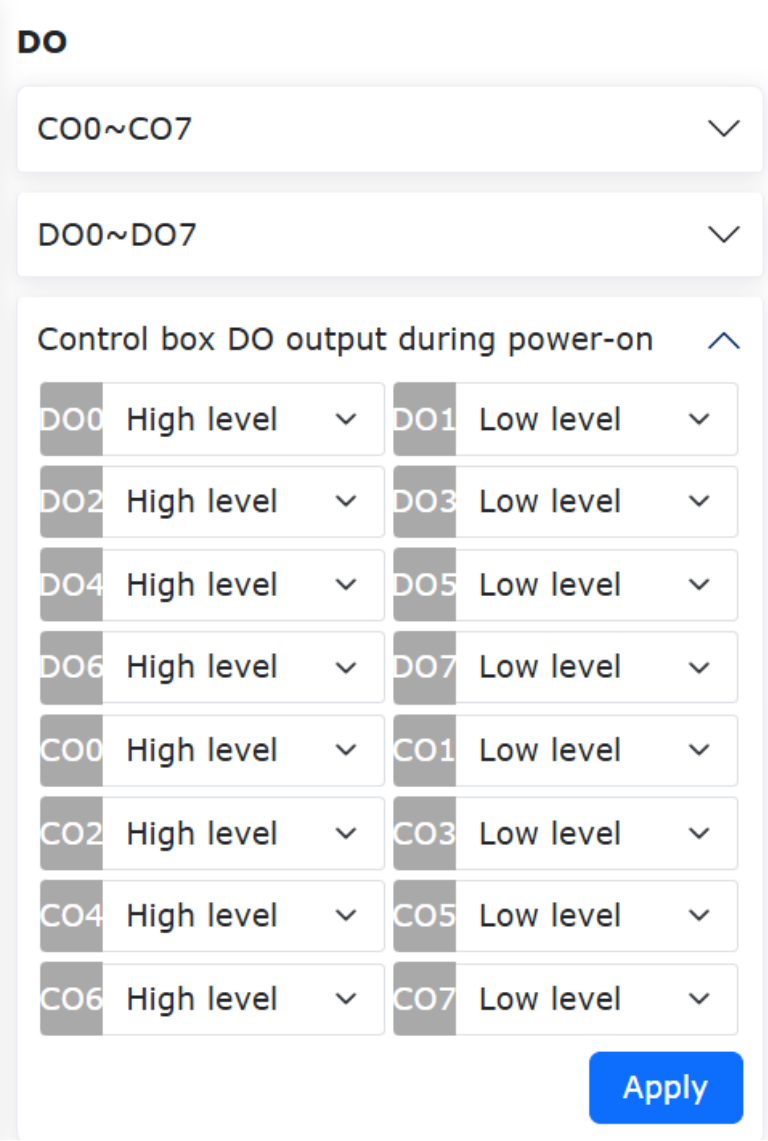


Figure 6.5-6 Control box DO output configuration during power-on

1.3.6.5.4 Alias

Click “Initial - Base - I/O setup” on the menu bar, click on the “Alias” submenu to enter the configuration interface, and configure the given meaning names of the control box and end IO signals according to the actual usage scenario. After successful configuration, the modules related to IO signal content will display corresponding aliases, as follows:

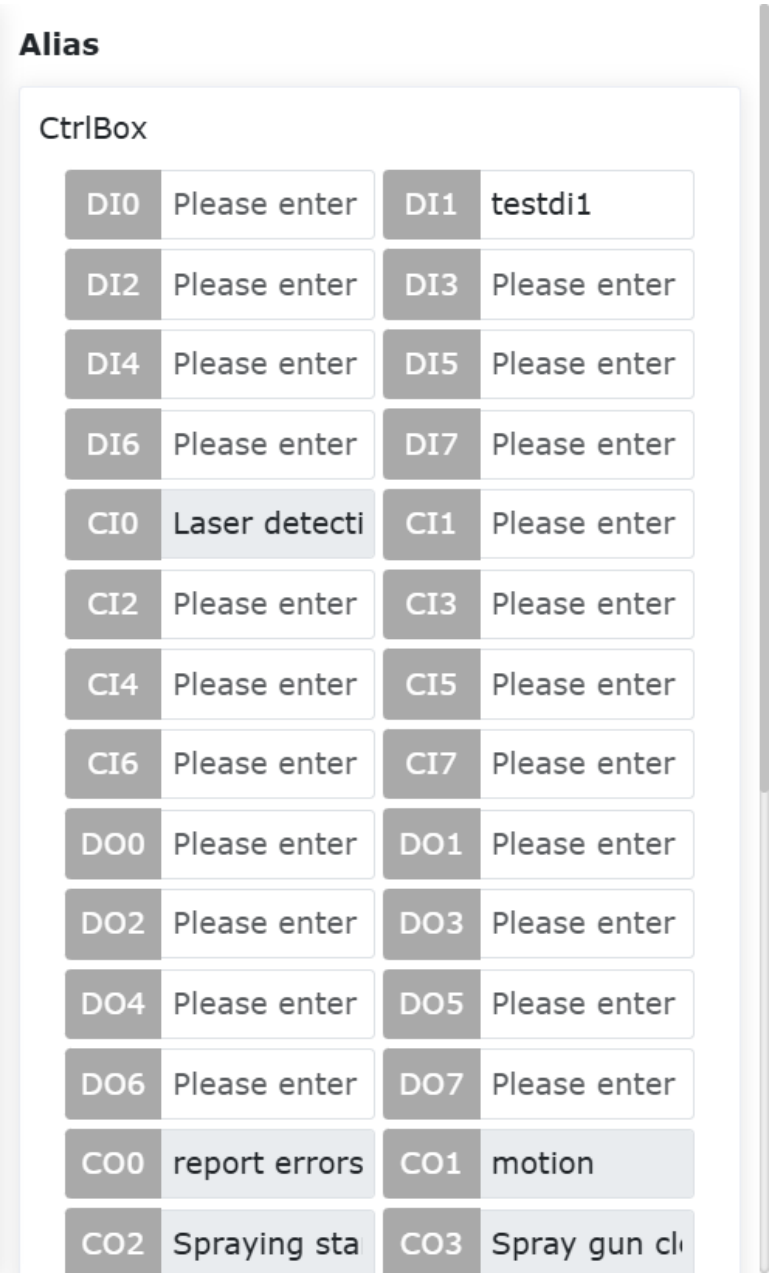


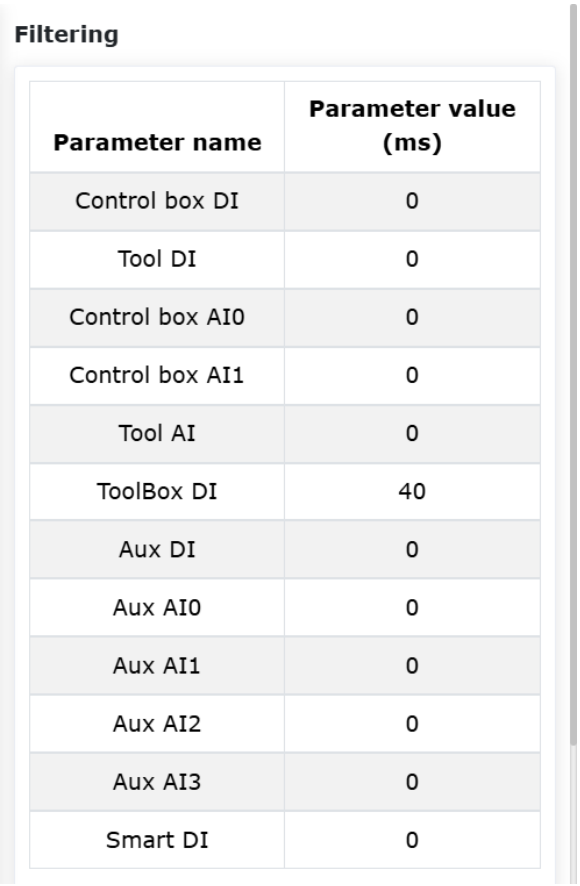
Figure 6.5-7 I/O alias configuration

1.3.6.5.5 Filter

Click “Initial - Base - I/O setup” on the menu bar, and click the “Filter” submenu to enter the IO filter time setting interface. The filter time setting interface includes:

- control box DI filter time
- end board DI filter time
- control box AI0 Filtering time
- AI1 filtering time of the control box
- AI0 filtering time of the end board
- Button box DI filter time
- Extended DI filter time
- Extended AI0 filter time
- Extended AI1 filter time
- Extended AI2 filter time
- Extended AI3 filter time
- Smart DI filter time

Users can view all the filter parameter value tables and set the corresponding parameters according to their needs by selecting the appropriate parameters and inputting the parameter values as shown in the figure below.



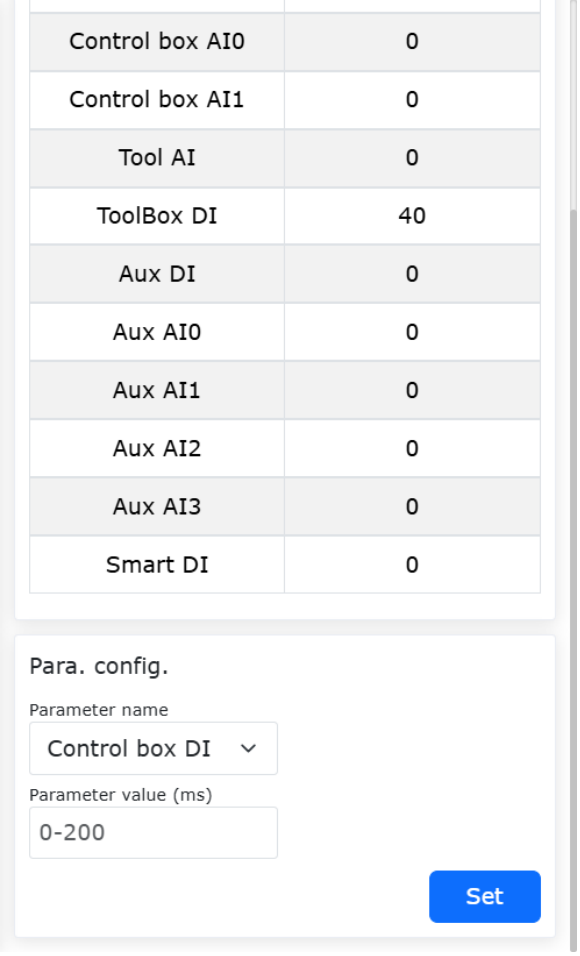


Figure 6.5-8 Filter interface

Important: The I/O filter time range is [0~200], the unit is ms.

1.3.6.5.6 Output reset

Click “Initial - Base - I/O setup” on the left menu bar, click the “Output reset” submenu to enter the configuration interface, and configure whether different outputs need to be reset after stopping/pausing according to the actual need for reset during use. The current output includes:

- control box DO
- control box AO
- End plate DO
- End plate AO
- Expand DO
- Expand AO
- SmartTool DO

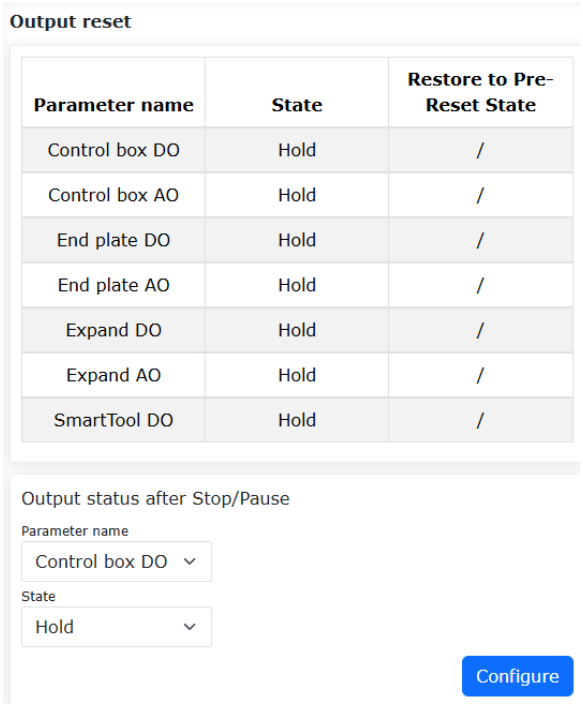


Figure 6.5-9 Output reset configuration

1.3.6.5.7 Pause/Resume DO Reset State Configurable Function

1.3.6.5.7.1 Overview

This function optimizes the existing output reset function by adding configurable options in the I/O settings. It can be set to two types: Hold or Reset, where Reset can be further divided into Restore to Pre-reset State and Do Not Restore to Pre-reset State. Users can set different configuration options according to actual needs.

1.3.6.5.7.2 Operation Process

Step1: Click sequentially on “Initial Setup” - “I/O Settings” - “Output Reset” instruction. Set the output state after stop/pause for DO or AO according to actual usage needs. The state can be set to “Hold” or “Reset”. Only when set to “Reset”, can “Restore to Pre-reset State” be further set.

Step2: Set the state to “Hold”. When clicking pause during Lua program execution, then clicking resume, the DO/AO output state remains unchanged throughout and stays triggered. When clicking stop during Lua program execution, the DO/AO output state remains unchanged. Parameter settings are as shown in the figure below.

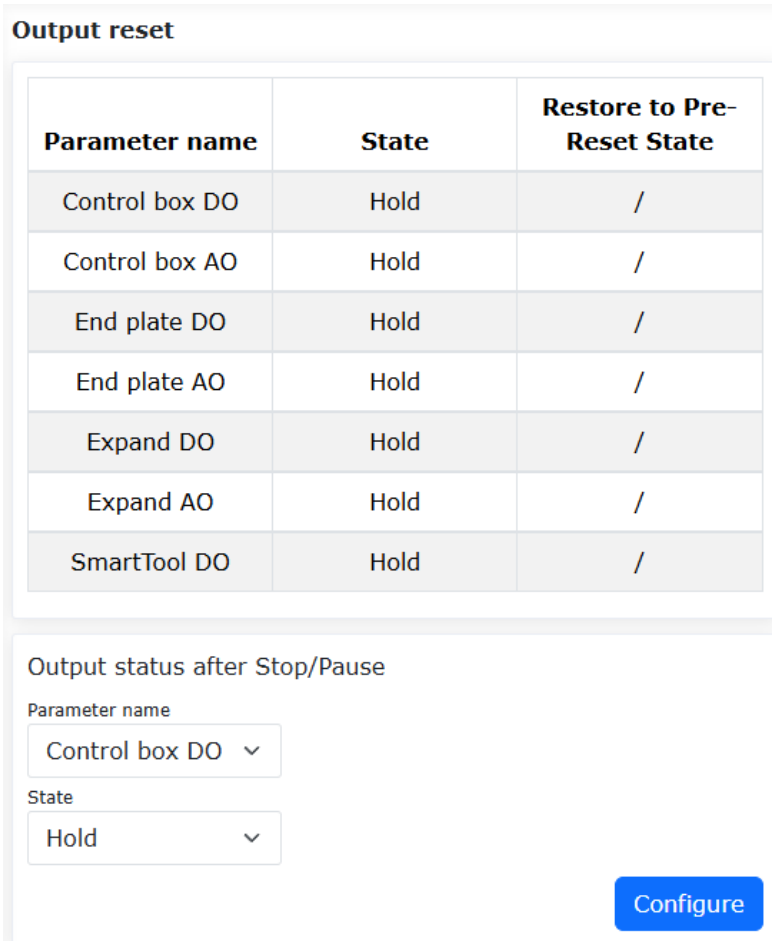


Figure 6.5-10 State Set to “Hold”

Step3: Set the state to “Reset” and “Restore to Pre-reset State” to “No”. When clicking pause during Lua program execution, the DO/AO output state will be reset. When clicking resume, the DO/AO output state remains reset. When clicking stop during Lua program execution, the DO/AO output state will be reset. Parameter settings are as shown in the figure below.

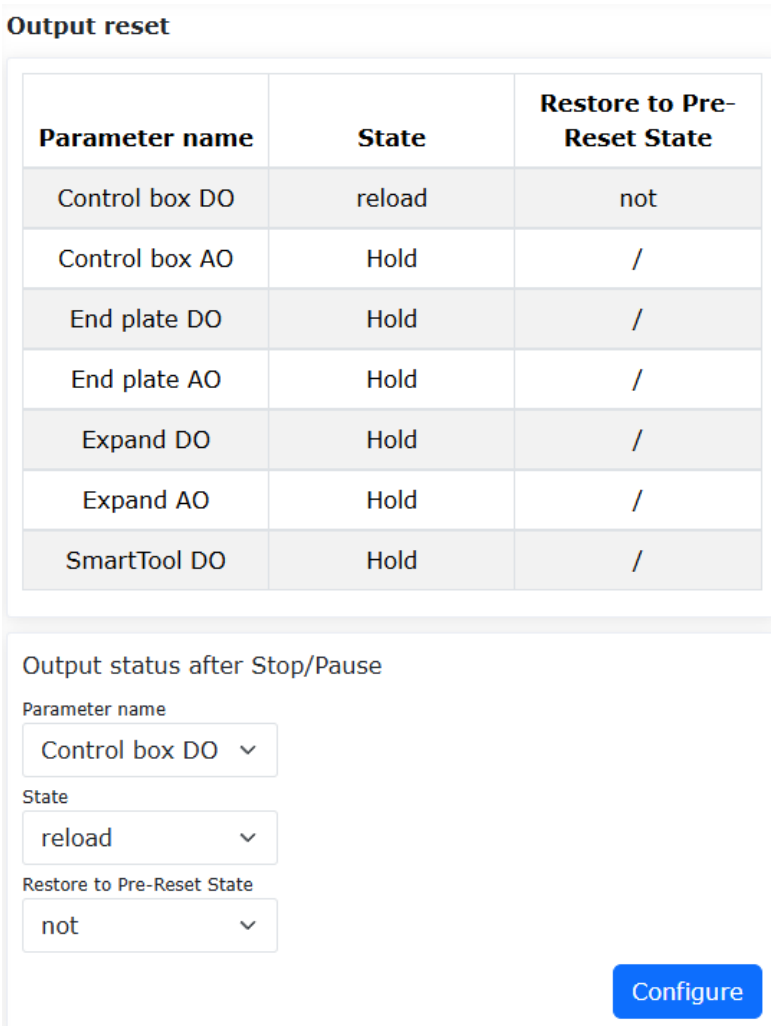


Figure 6.5-11 State Set to “Reset” + “No”

Step4: Set the state to “Reset” and “Restore to Pre-reset State” to “Yes”. When clicking pause during Lua program execution, the DO/AO output state will be reset. When clicking resume, the DO/AO output state will be reloaded. When clicking stop during Lua program execution, the DO/AO output state will be reset. Parameter settings are as shown in the figure below.

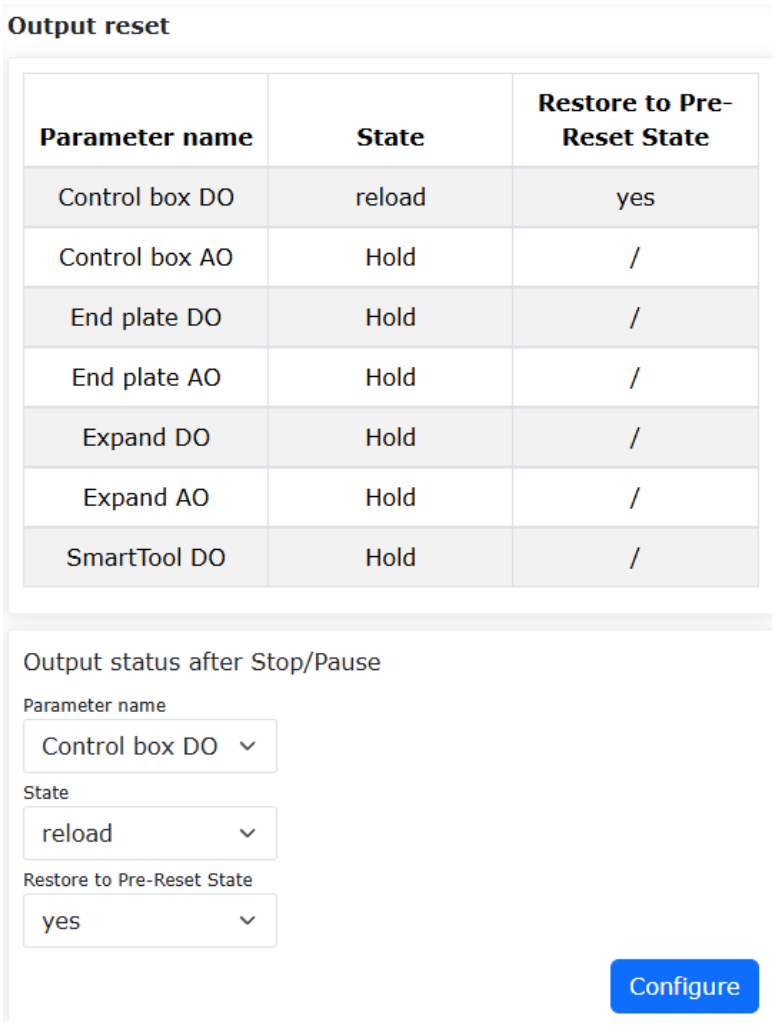


Figure 6.5-12 State Set to “Reset” + “Yes”

1.3.6.6 Joint origin

In the menu bar under ‘Initial Setup’ -> ‘Basics’, click ‘Home point’ to enter the Joint origin configuration interface.

This page displays the name and joint position information of the work origin. The work origin is named pHome. Click “Set” to use the current robot pose as the work origin. Click “Move to this point” to move the robot to the work origin. In addition, the configurable option of moving to the origin of the work is added in the DI configuration, and the configurable option of reaching the origin of the work is added in the DO configuration.

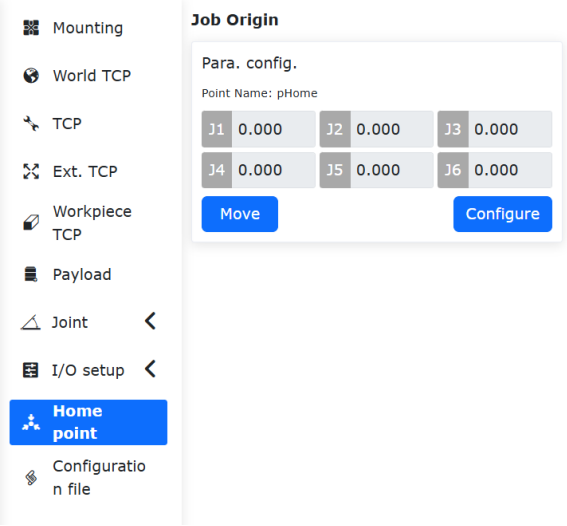


Figure 6.6-1 Joint origin

1.3.6.7 Photoelectric Sensor TCP Auto-Calibration Function

1.3.6.7.1 Overview

When a robot tool collision causes TCP position deviation, the TCP auto-calibration function based on a photoelectric sensor can be enabled. This function quickly recalibrates the tool coordinate system by automatically calculating and compensating for position deviation, significantly reducing downtime and improving equipment operating efficiency and production stability.

1.3.6.7.2 Operation Procedure

Step1: Place the photoelectric sensor within the robot's workspace, and connect the two sets of brown, blue, and black signal wires from the photoelectric sensor device to the two sets of 24V, 0V, and CI0, CI1 ports on the robot control box (any available configurable digital signal input ports will work) or connect them to the two sets of 24V, 0V and End-DI0, End-DI1 ports at the robot end.

Step2: Calibrate the photoelectric sensor coordinate system. The photoelectric sensor coordinate system is essentially a workpiece coordinate system, and its accuracy has a significant impact on the subsequent tool TCP calibration. It can be determined in several ways:

- 1Use the workpiece coordinate system calibration method, with the origin at the intersection of the two laser beams, the two laser beams serving as the X-axis and Y-axis respectively, and the Z-axis perpendicular to the photoelectric sensor pointing outward;
- 2Provided by external measurement equipment (such as a camera);
- 3Calibrate using the photoelectric device configuration within the photoelectric auto-calibration function. This option requires using a tool with known precise dimensions and a roughly accurate workpiece coordinate system to be applied: First click "Initial Setup" -> "Tool Coordinates", apply Tool Coordinate System 0, then click the "Coordinate System Calibration" button for the precise tool coordinate system (using coordinate system 1 as an example), then click "Apply", and finally select "Photoelectric Auto Calibration Function".

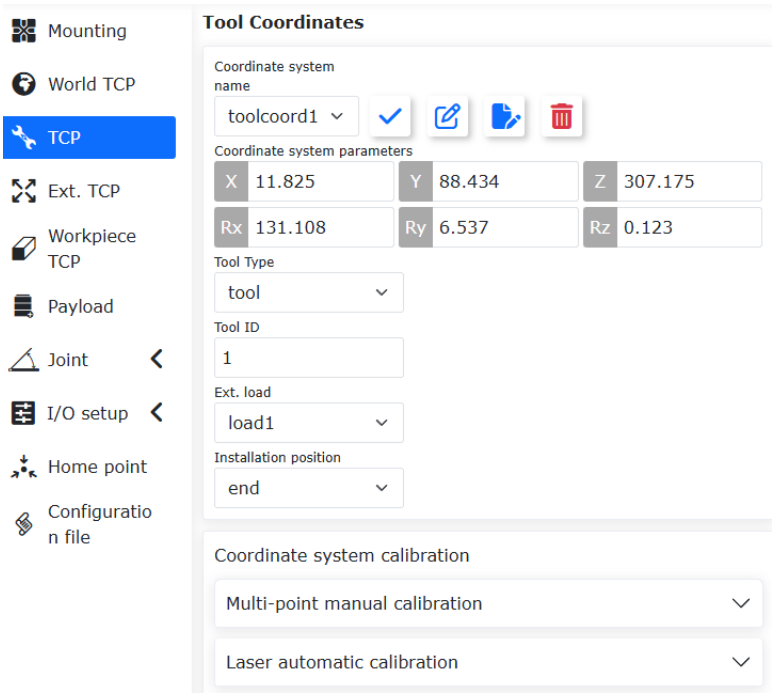


Figure 6.7-1 Photoelectric Auto Calibration Selection

Navigate into the “Photoelectric Device Configured” section, configure the IO trigger signal, set the teach center point, set the offset parameters, then click “Run” to calibrate the sensor’s coordinate system. Subsequently, manually apply the calibration result to a workpiece coordinate system.

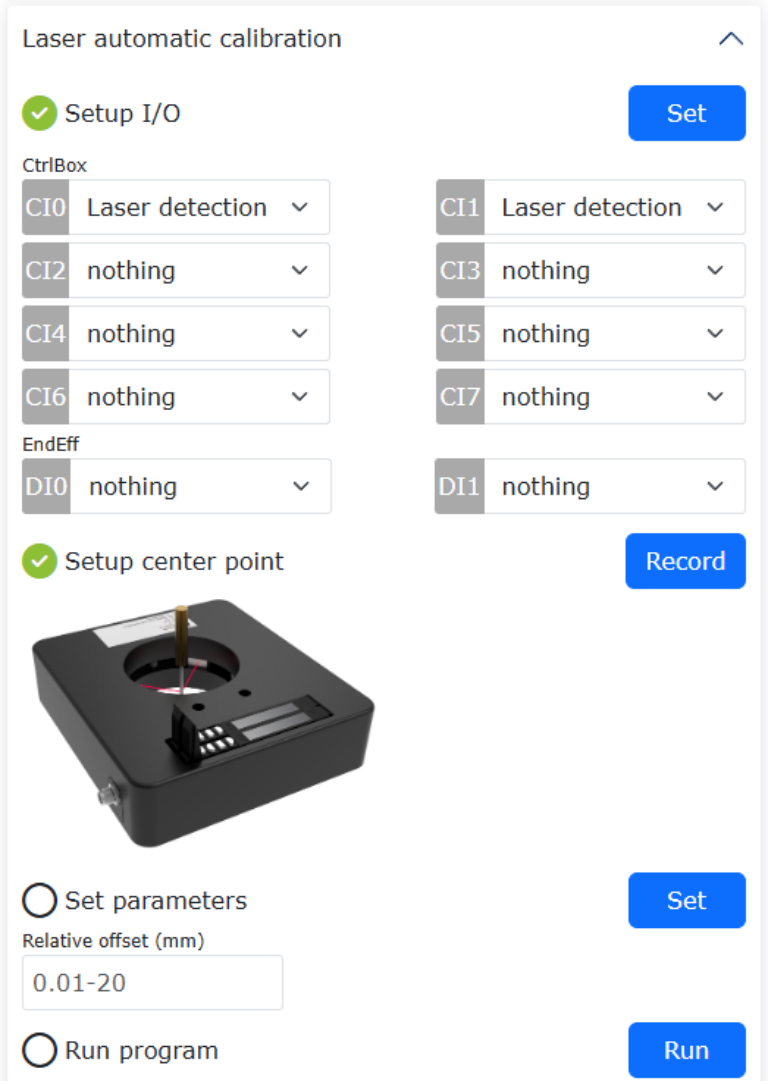


Figure 6.7-2 Calibrate Sensor Coordinate System

Step3: Calibrate the tool coordinate system. After Step2, an accurate workpiece coordinate system has been obtained and applied, and the pre-collision tool coordinate system is known and applied: First click “Initial Setup” -> “Tool Coordinates”, apply Tool Coordinate System 0, then click the “Coordinate System Calibration” button for the pre-collision tool coordinate system (using coordinate system 1 as an example), then click “Apply”, and select “Photoelectric Auto Calibration Function”. Within “Photoelectric Calibration Parameters Configured”, set the calibration parameters.

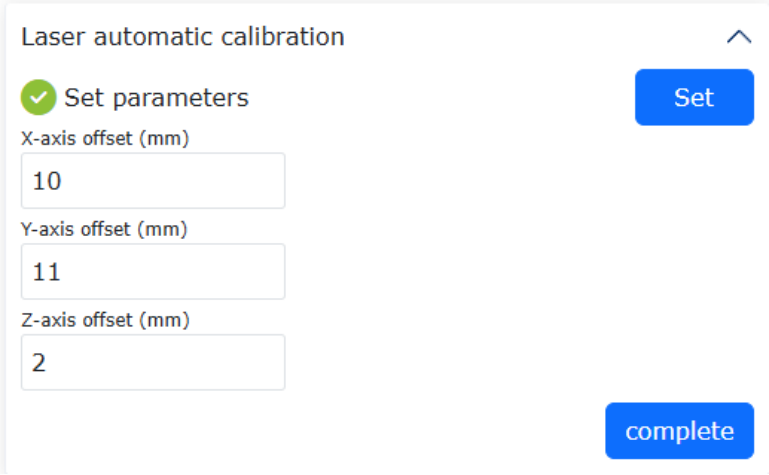


Figure 6.7-3 Photoelectric Auto Calibration Parameter Setup

After setup is complete, click the "Finish" button to return to the previous menu, then click the "Calibrate" button to perform TCP calibration. Once calibration is finished, click the "Save" button to save the calibration result.

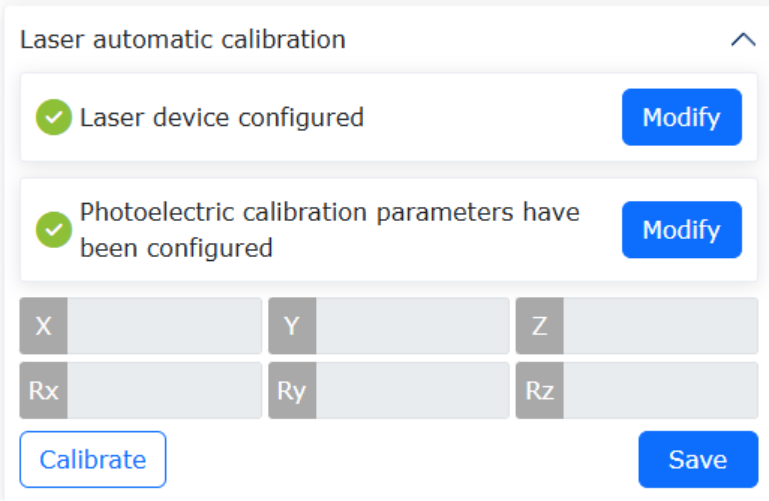


Figure 6.7-4 Photoelectric Auto Calibration and Save

1.3.6.8 TCP Calibration based on flatbed

1.3.6.8.1 Overview

When using the "four-point method" to carry out TCP calibration of tools, it is necessary to manually control the robot movement and realize the exact coincidence of points with the naked eye, and the calibration efficiency and accuracy are affected by the operator's proficiency.

The principle of TCP calibration based on flatbed is as follows: the robot tool is used to touch any position of the flatbed several times, and the TCP of the tool is solved by establishing a calibration model. The whole calibration process is automatically completed, which can improve the efficiency of calibration and reduce the dependence on manual work.

1.3.6.8.2 2The operation flow of TCP calibration function based on flatbed

Fix the calibration flatbed in the working space of the robot, the flatbed should not be shaken, and the electrical conductivity of the flatbed is good. The end of the tool is approximately vertical to the calibration flatbed, and is located 50mm above the flatbed.

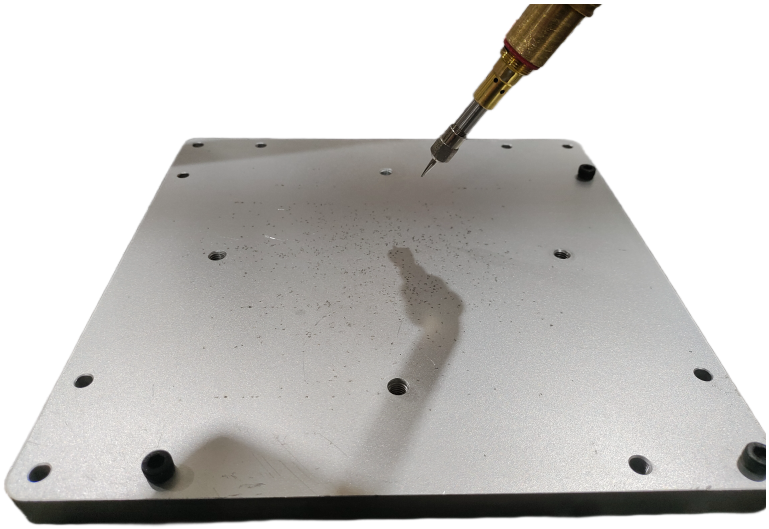


Figure 6.8-1 Calibration layout

In turn, click on the “Program” – “Coding” button, select “FR_CalibrateTheToolTcpPlane. lua” and open the file.

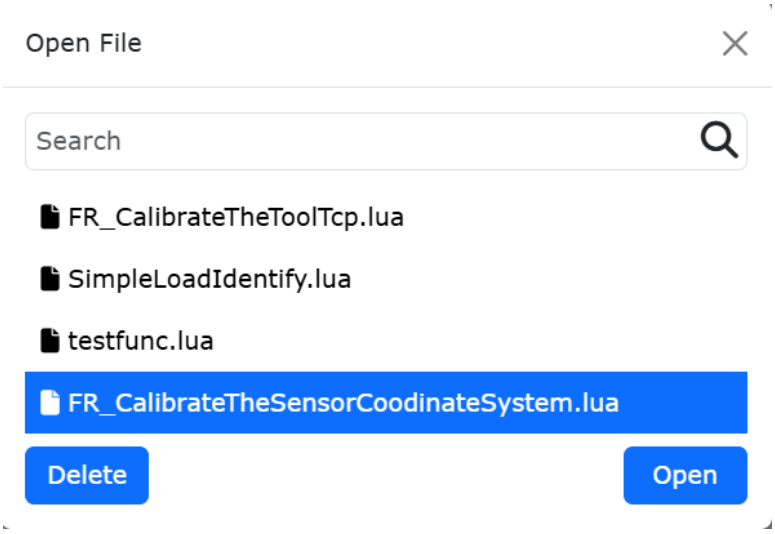


Figure 6.8-2 Opening the calibration file

Click the buttons of “Initial”, “Base” and “Coordinate” and “TCP” successively to enter the “Current tool coordinate system” interface. Select the coordinate system to be calibrated in “Coordinate system name” (take toolcord1 coordinate system as an example), click “Modify” button, you can enter the TCP calibration method selection interface.

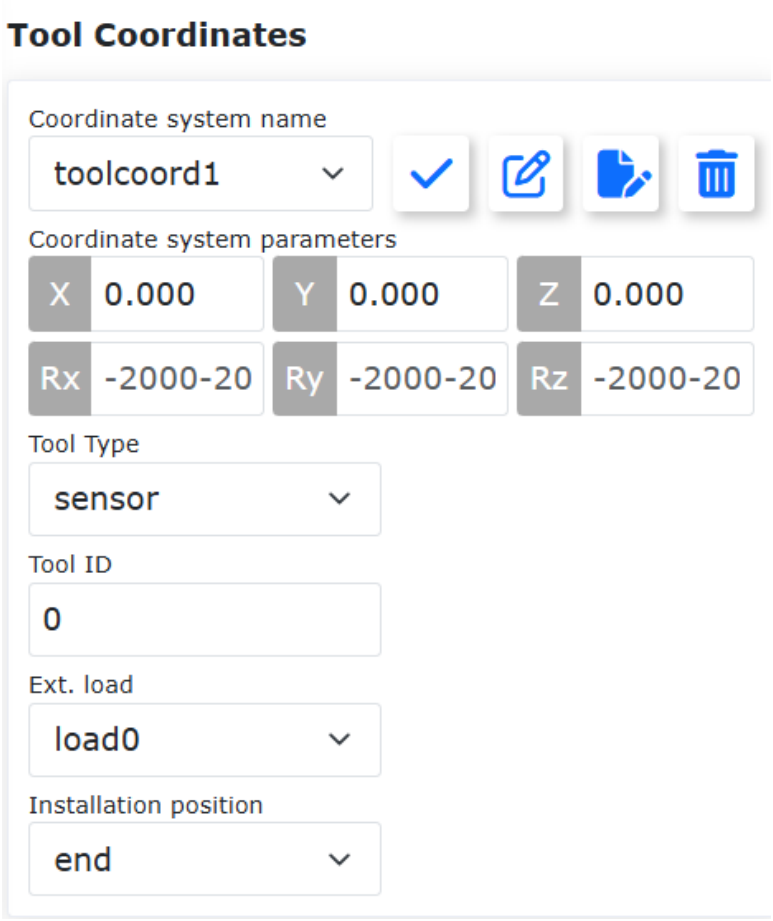


Figure 6.8-3 Setting the tool coordinate system

In the “Modify Wizard”, select “Flatbed Tool Calibration” to enter the tablet tool calibration interface.

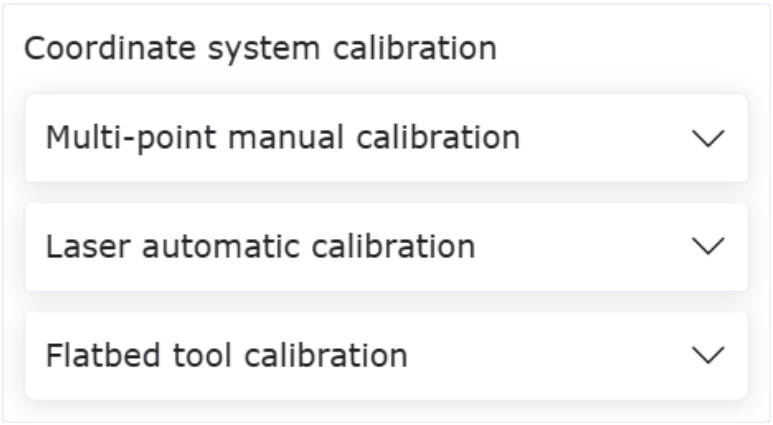


Figure 6.8-4 Selection of calibration method

In the “Flatbed tool calibration” interface, click “Modify” button to configure the flatbed tool, click “Record” button to record the calibration reference point. After the configuration is complete, click the “complete” button to return to the “Flatbed tool calibration” interface..

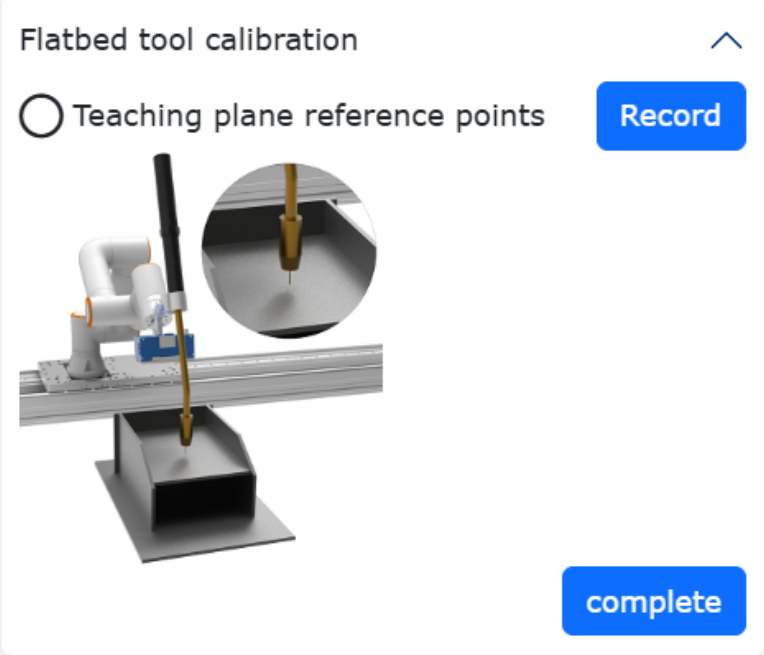


Figure 6.8-5 Configuring the flatbed tool

Click the “Run” button in the “Flatbed tool calibration” interface, the robot will automatically carry out the TCP calibration of the tool, after the calibration is completed, the TCP coordinates of the tool will be displayed, click the “Save” button, the calibration result will be returned to the “Current tool coordinate system” interface.

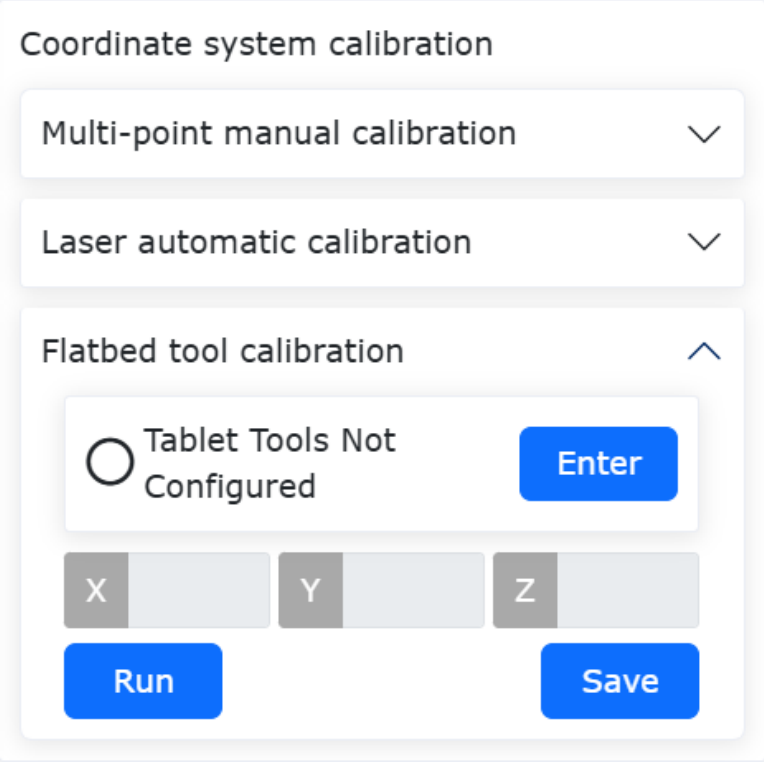


Figure 6.8-6 Calibration results

Click the “Apply” button in the “Current tool coordinate system” interface to save the TCP calibration result of the tool

and apply it.

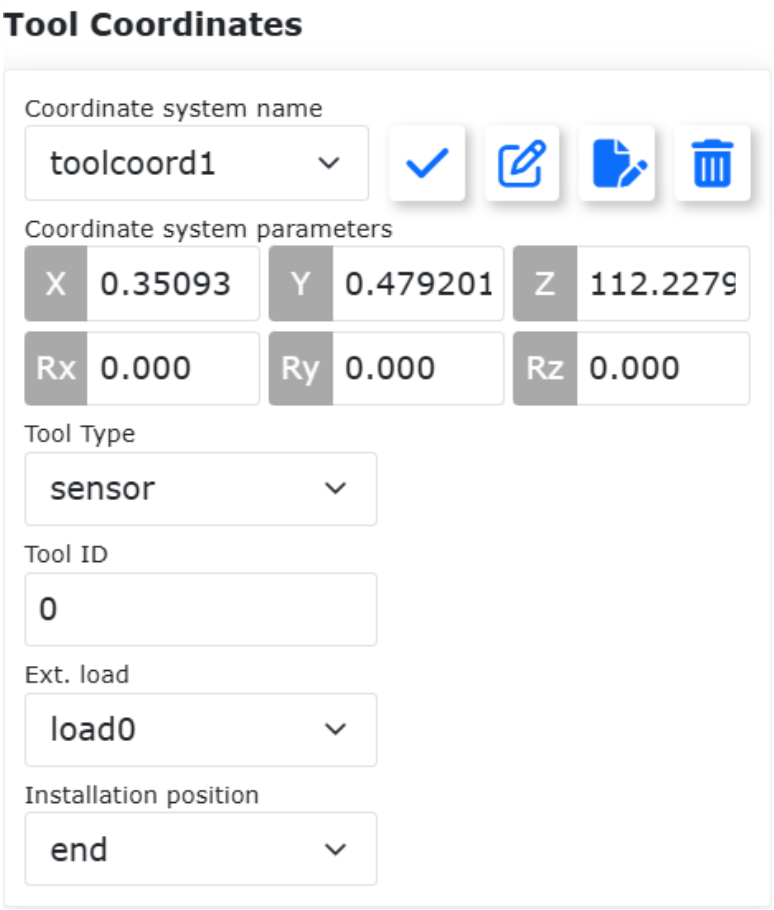


Figure 6.8-7 Calibration result application

1.3.6.9 Control Box Analog Feedback Arc Tracking Function

1.3.6.9.1 Overview

The control box analog feedback arc tracking function collects welding voltage and current analog signals to achieve arc tracking compensation. This function is implemented by configuring the corresponding AI and AO channels for the control box analog signals.

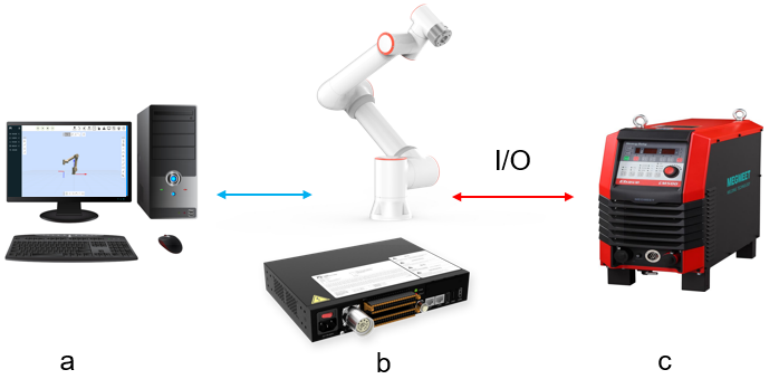


Figure 6.9-1 Topology Diagram of Arc Tracking Function Based on Analog Signal Communication

a represents the computer; b represents the robot and control box; c represents the welder

1.3.6.9.2 Control Box Analog AI Configuration Process

On the robot web control interface, navigate to “Initial Setup” -> “Basic” -> “I/O Settings” -> “AI” to enter the “AI Configuration” interface.

In the “Arc Tracking Channel” section of the “AI Configuration” interface, select “Ctrl-AI0” and “Ctrl-AI1” from the dropdown menus for “Welding Current Control AI” and “Welding Voltage Control AI” respectively, then click “Configure” to complete the control box analog AI configuration.

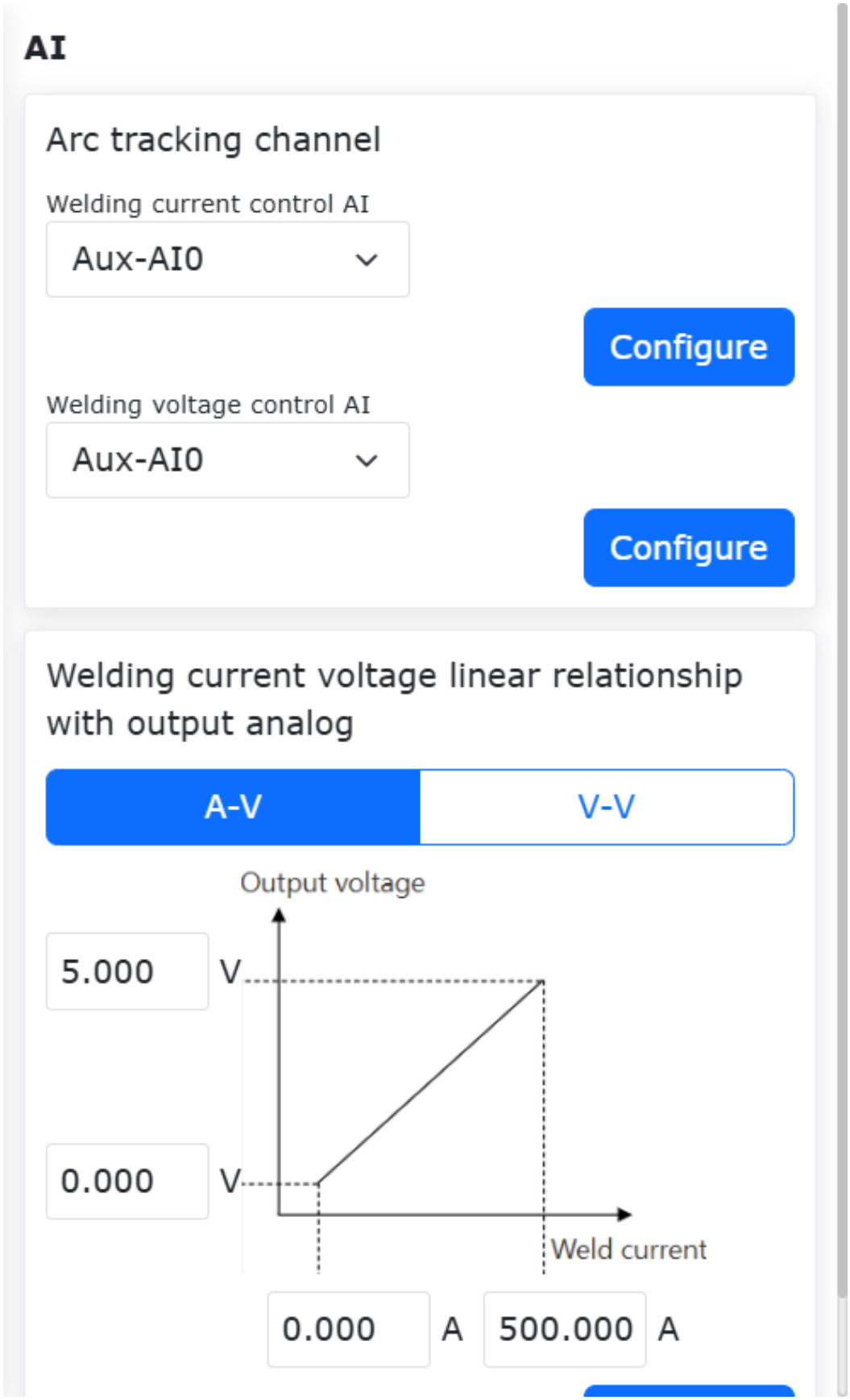

AI
Arc tracking channel
Welding current control AI
Aux-AI0
Configure
Welding voltage control AI
Aux-AI0
Configure
Welding current voltage linear relationship with output analog
A-V
V-V
Output voltage
5.000
V
0.000
V
Weld current
0.000
A
500.000
A

Figure 6.9-2 AI Channel Configuration

In the “Analog Current-Voltage Relationship Graph” section of the AI channel configuration shown above, the parameter settings for the “A-V” and “V-V” interfaces should refer to the analog input/output tables or graphs of the welder being used.

For example, configure the lower and upper limits of the welding current for the control box current analog AI as 0A and 500A respectively; configure the lower and upper limits of the output voltage as 0V and 5V respectively as the parameters for the “A-V” interface in the “Analog Current-Voltage Relationship Graph” section. Click “Configure” to complete the control box current analog AI channel configuration.

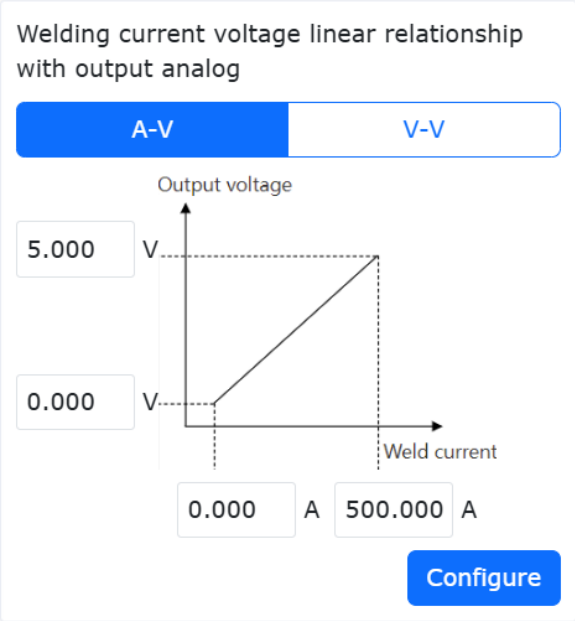


Figure 6.9-3 Control Box Current Analog AI Configuration

For example, configure the lower and upper limits of the welding voltage for the control box voltage analog AI as 0V and 50V respectively; configure the lower and upper limits of the output voltage as 1.018V and 10V respectively as the parameters for the “V-V” interface in the “Analog Current-Voltage Relationship Graph” section. Click “Configure” to complete the control box voltage analog AI channel configuration.

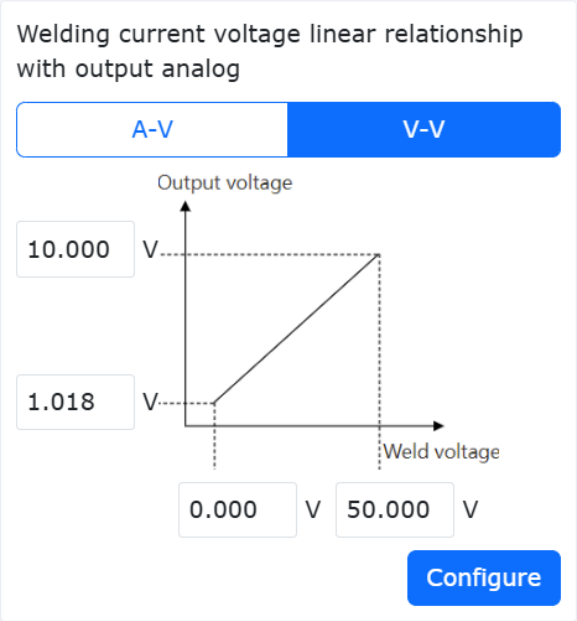


Figure 6.9-4 Control Box Voltage Analog AI Configuration

1.3.6.9.3 Control Box Analog AO Configuration Process

On the robot web control interface, navigate to “Initial Setup” -> “Peripherals” -> “Welder” to enter the “Welder Configuration” interface.

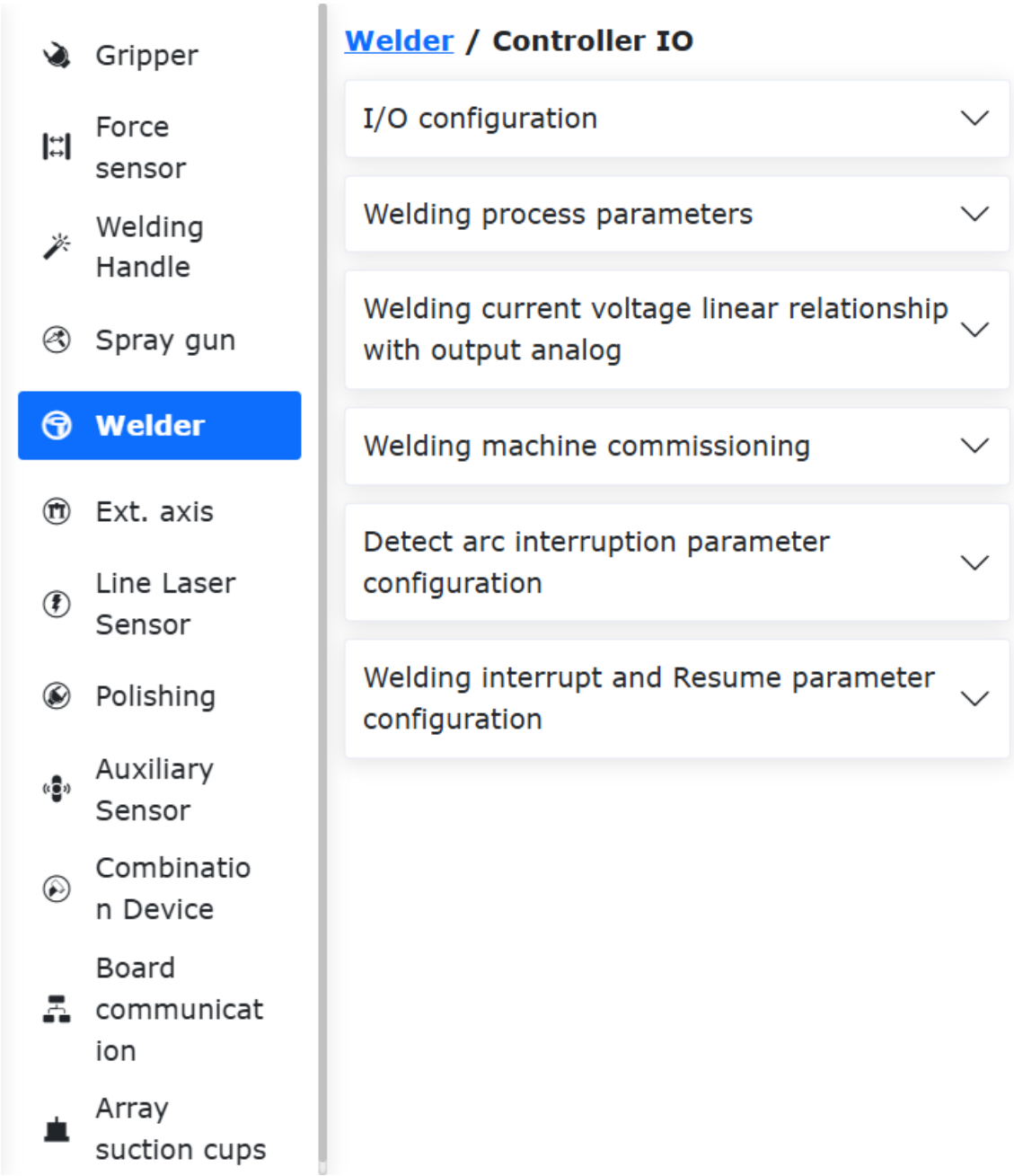


Figure 6.9-5 Welder Configuration

In the “Welding Function I/O Configuration” section of the “Welder Configuration” interface, the parameters for the “DI” and “DO” interfaces can be customized to configure the control box CI and CO channels. Select “Controller I/O” from the “Control Type” dropdown menu to begin the controller analog AO channel configuration process.

In the “Analog Current-Voltage Relationship Graph” section of the “Welder Configuration” interface, the parameter settings for the “A-V” and “V-V” interfaces should refer to the analog input/output tables or graphs of the welder being used.

For example, configure the lower and upper limits of the welding current for the control box current analog AO as 0A and 495A respectively; configure the lower and upper limits of the output voltage as 1V and 10V respectively as the parameters for the control box AO channel current analog configuration. Then select “Ctrl-AO0” from the “Welder Current Control AO” dropdown menu and click “Configure” to complete the control box current analog AO channel

configuration.

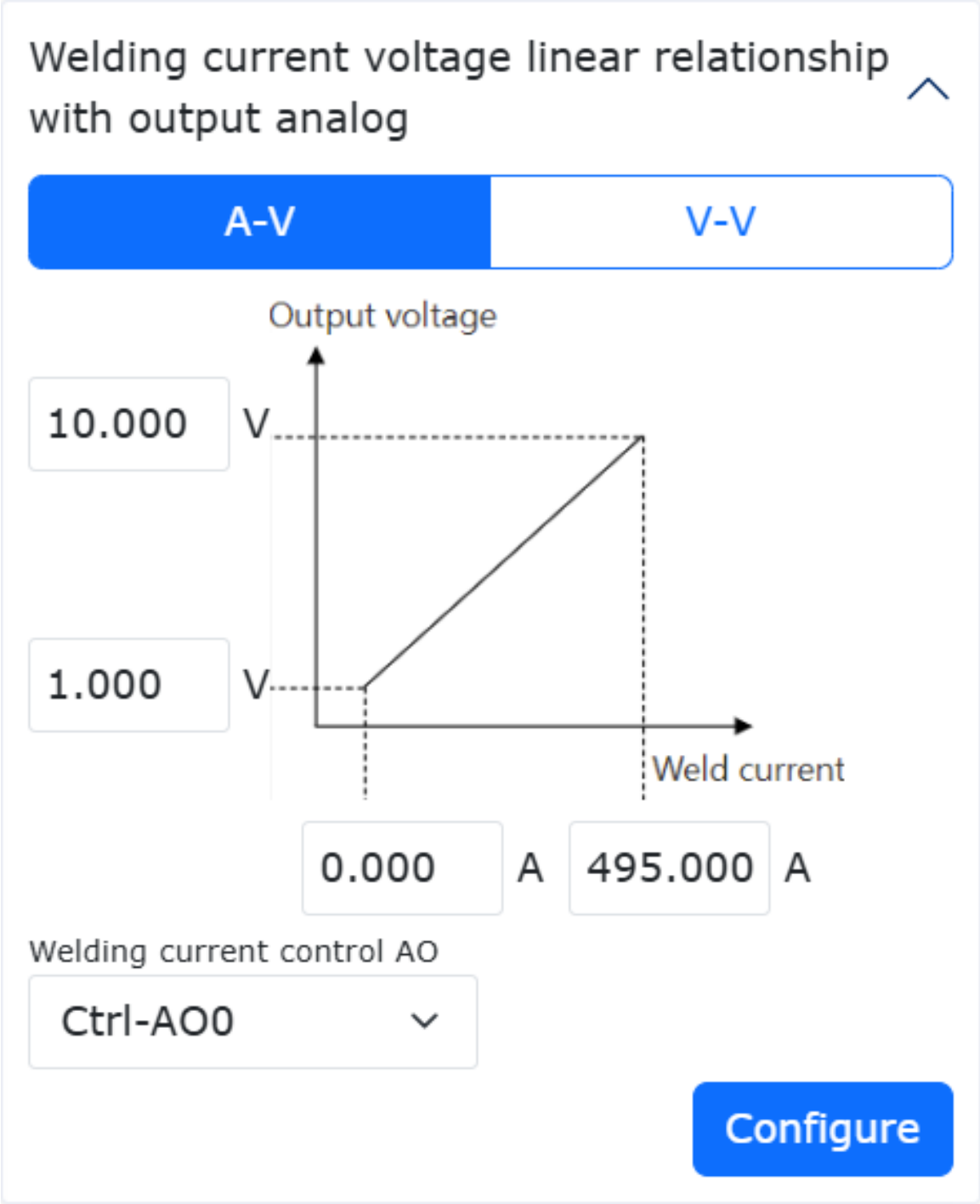


Figure 6.9-6 Control Box Current Analog AO Configuration

For example, configure the lower and upper limits of the welding voltage for the control box voltage analog AO as 10V and 45V respectively; configure the lower and upper limits of the output voltage as 1V and 10V respectively as the parameters for the control box AO channel voltage analog configuration.

Then select “Ctrl-AO1” from the “Welder Voltage Control AO” dropdown menu and click “Configure” to complete the control box voltage analog AO channel configuration.

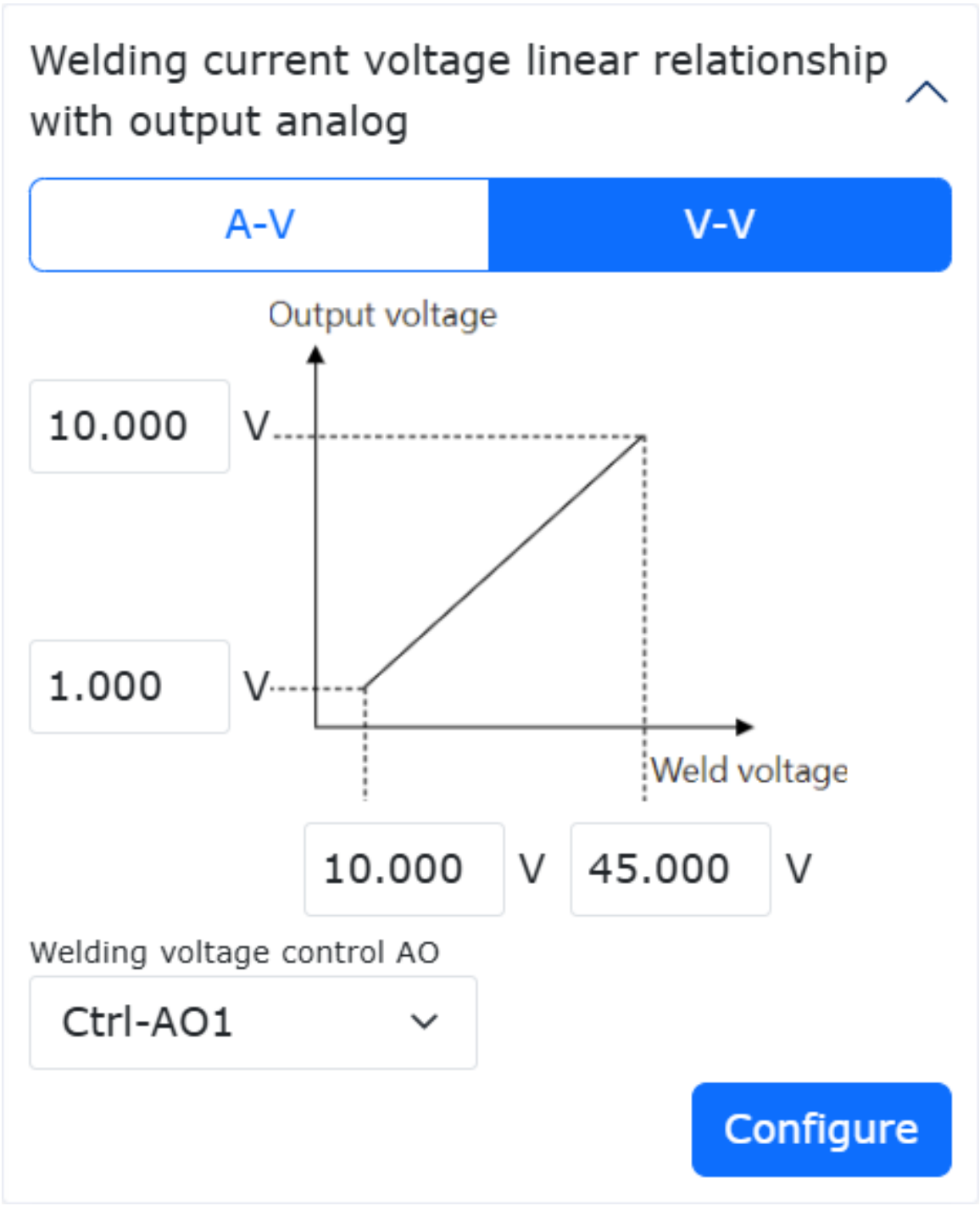


Figure 6.9-7 Control Box Voltage Analog AO Configuration

1.3.6.10 Linear Rack Guideway Collision Detection

1.3.6.10.1 Overview

The Linear Rack Guideway Collision Detection function is used to achieve alarm and emergency stop when the guideway or robot collides with environmental objects during asynchronous or synchronous operation. By monitoring changes in the guideway torque feedback, it judges whether a collision has occurred based on a set threshold. If a collision occurs, the guideway stops immediately, thereby avoiding the application of continuous force by the guideway and robot on the collided object, further enhancing human-robot collaboration safety.

1.3.6.10.2 Linear Rack Guideway Collision Detection Function

To use the Linear Rack Guideway Collision Detection function, the "Rail_Adaptation_Program.lua" program must be executed after the guideway is activated. This ensures the function can adapt to different guideways and loading conditions, obtaining the best collision detection performance. If adaptation is not performed, the collision detection performance will significantly decrease, and the external force required to trigger a collision will be greater.

1.3.6.10.2.1 Linear Rack Guideway Parameter Setting and Enabling

Step1: Log in to the web interface, click "Initial Setup" -> "Peripherals" -> "Extended Axes" in sequence to enter the Extended Axis Coordinate System setup module, as shown in Figure 2-1.

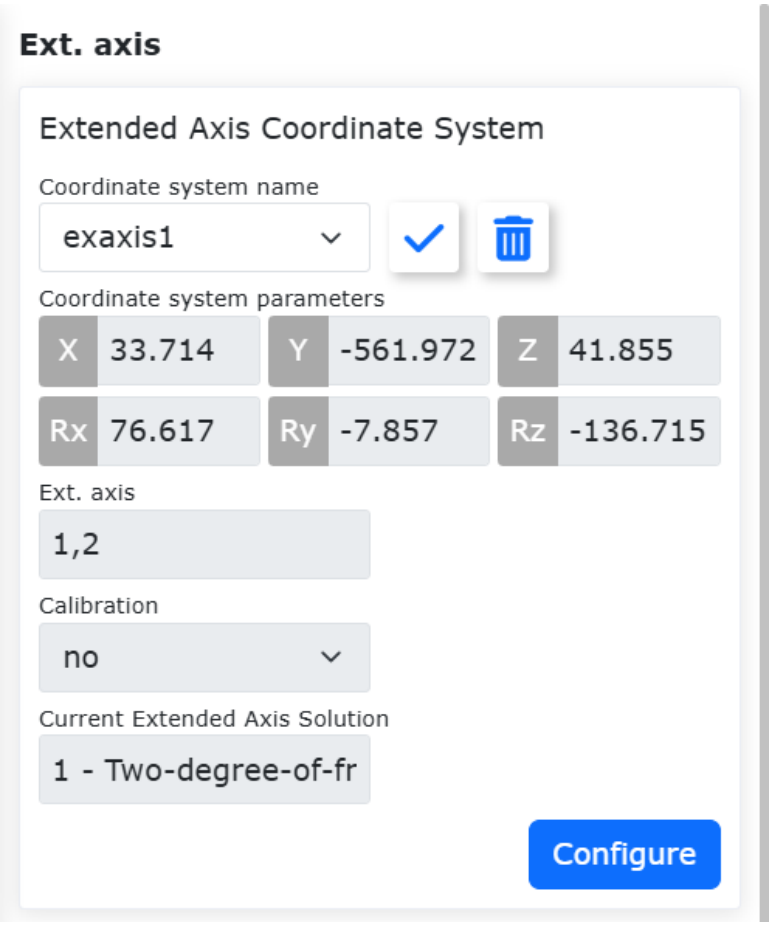


Figure 6.10-1 Extended Axis Coordinate System Setup Module

Step2: Based on the actual working conditions of the extended axis and the robot, set parameters and calibrate as needed. In Figure 2-1, click Edit, set the Extended Axis Coordinate System Name to “exaxis1”, select the Scheme as “0-Single Degree of Freedom Linear Slide”, and select the Extended Axis Number as “1”. If the guideway and robot only run asynchronously, calibration may not be necessary. If synchronous operation is required, calibration is mandatory. Refer to the relevant user manual or consult professional personnel for the calibration process. After setting the parameters, click “Save” and apply the corresponding coordinate system, as shown in Figure 2-2.

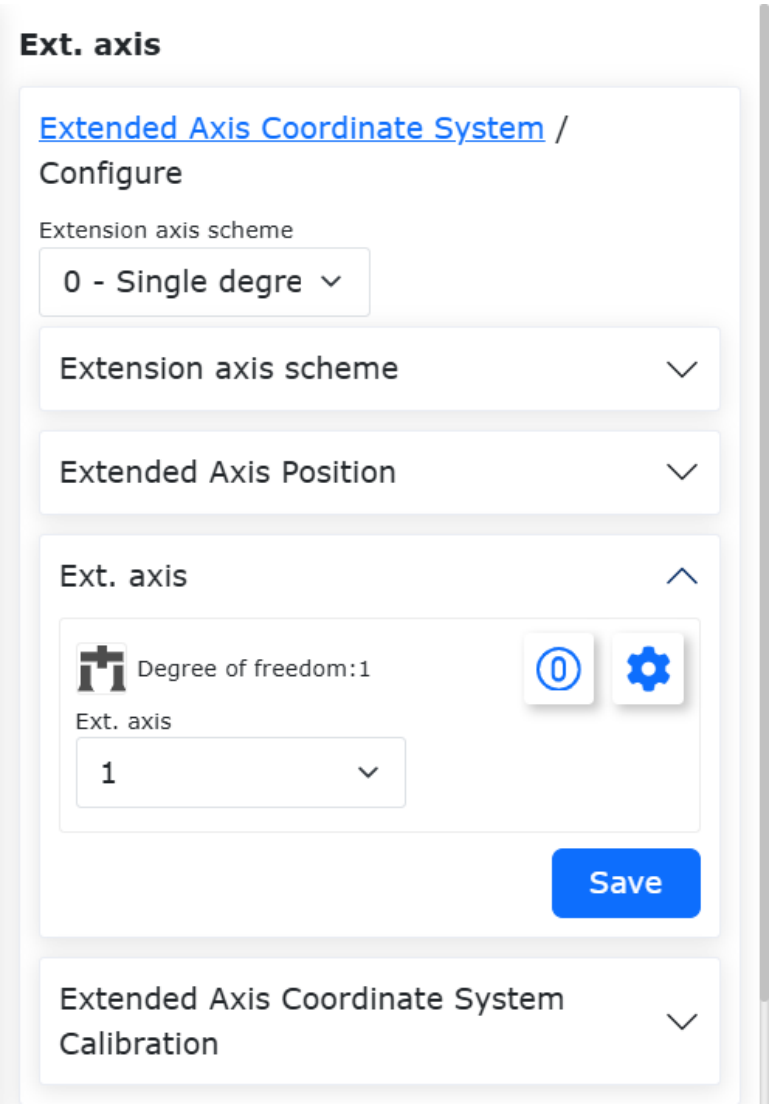


Figure 6.10-2 Extended Axis Coordinate System Parameter Setting

Step3: Establish UDP communication between the extended axis and the robot, and ensure the extended axis PLC program can send back the torque feedback data of the extended axis drive motor (after the gearbox) to the robot controller. Click “Initial Setup” -> “Peripherals” -> “Extended Axes” in sequence to enter the UDP Communication Configuration page. Select the coordinate system set in Step 2 and apply it. Click the “Edit” icon for UDP Communication Configuration to configure and load the communication settings. The PLC and laptop IP addresses must be in the same subnet as the controller, as shown in Figure 2-3. Note: It is necessary to ensure that the extended axis PLC program can send back the torque feedback data of the extended axis drive motor (after the gearbox) to the robot controller, and the sampling period should ideally be 1ms, with a maximum not exceeding 4ms, otherwise the collision detection function will be invalid.

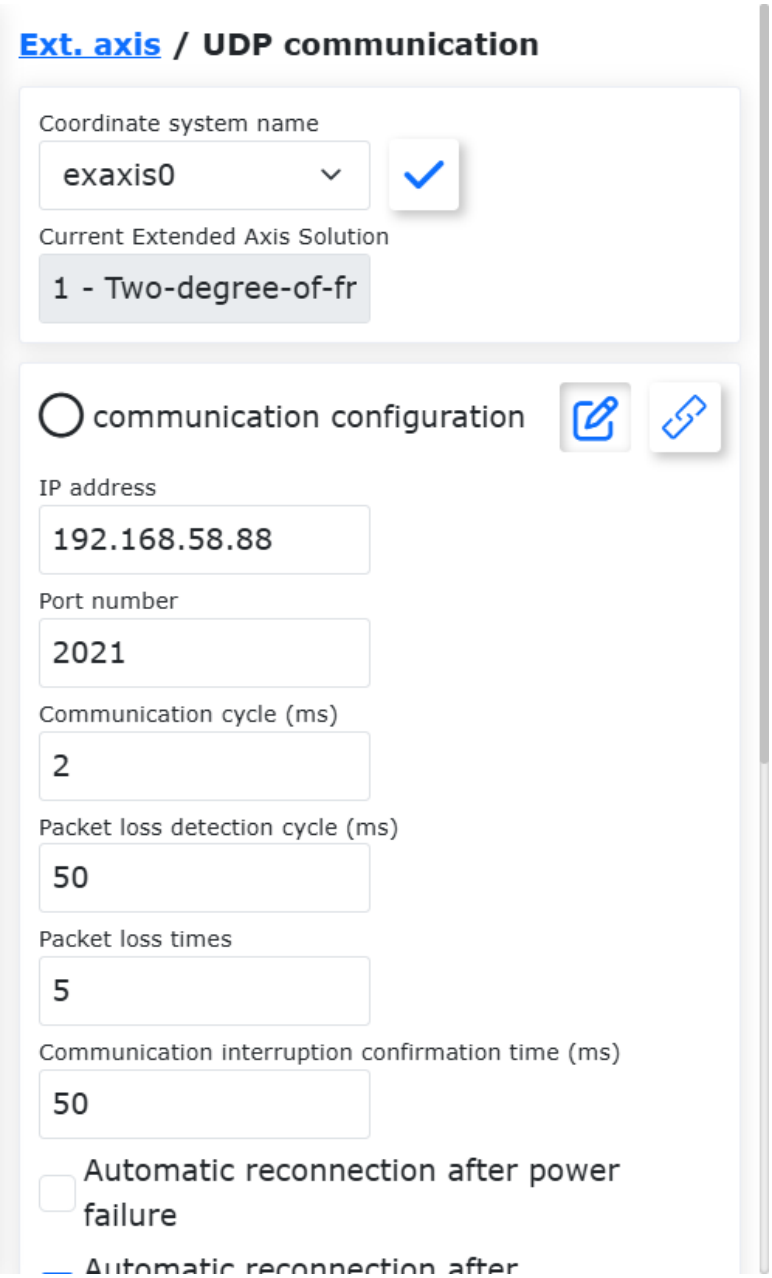


Figure 6.10-3 UDP Communication Configuration Page

Step4: Configure UDP Extended Axis parameters. The UDP Extended Axis parameter setting page is shown in Figure 2-4. Select the Axis Type as "Linear Guideway", Axis Direction as "Positive", and configure the remaining parameters according to the actual situation. Among them, the Lead and Encoder Resolution are fixed and determined by the guideway; the upper limits of Running Speed and Acceleration are affected by the motor performance. The upper limits used in this function test are as shown in Figure 2-4. Please contact professional personnel when configuring different upper limits.

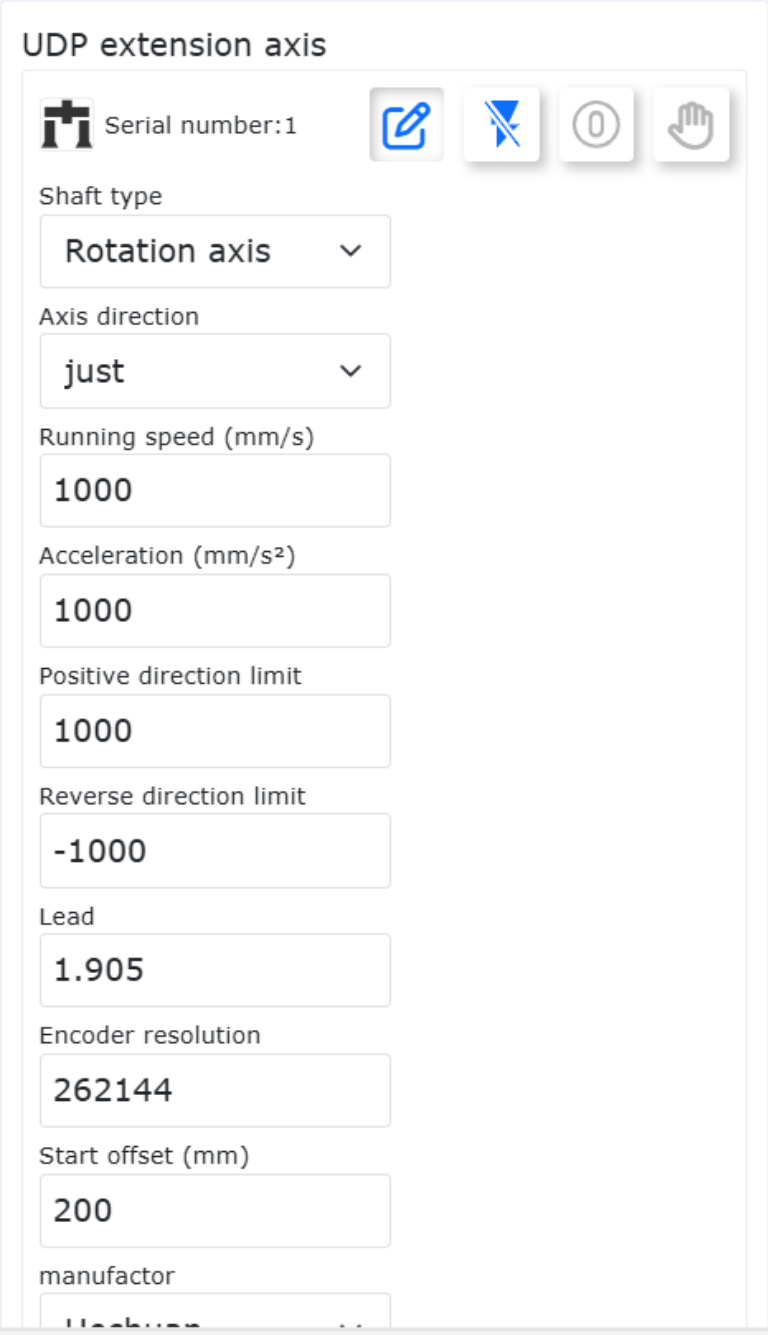


Figure 6.10-4 UDP Extended Axis Parameter Setting

Step5: Enable the linear rack guideway and move it to the starting point. Enable the linear rack guideway via the "Remove Enable" button in Figure 2-4 or the "Servo Enable" button in Figure 2-5. If the slider on the guideway is far from the starting point, move it to the starting point using "Reverse Rotation" or "Forward Rotation" (note: the running speed needs to be away from 15%). After moving to the starting point, click "Zero Point Setting" and perform homing using the "Return to Zero from Current Position" method.

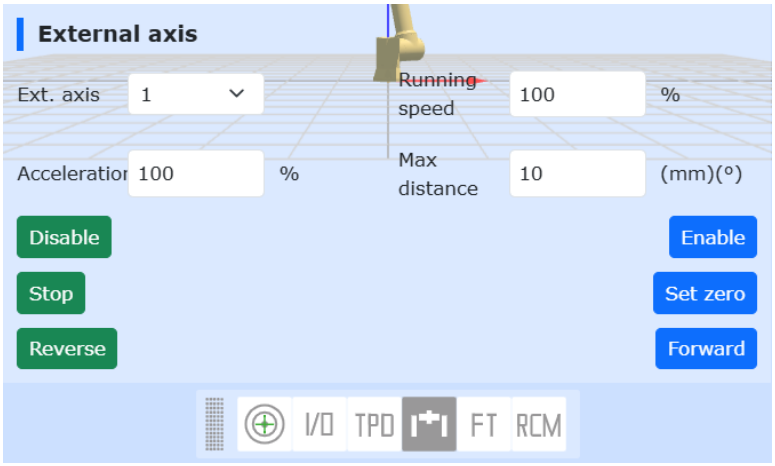


Figure 6.10-5 Enable Linear Rack Guideway and Move

1.3.6.10.2.2 Enabling Linear Rack Guideway Collision Detection Function

Step1: Ensure the installation method of both the guideway and the robot is upright. Before enabling the linear rack guideway collision detection function, check if the installation method is upright. Specifically, first ensure the physical installation of the guideway and robot is upright. Then, click “Initial Setup” -> “Basic” -> “Installation” in sequence to enter the Free Installation page. If both “Base Rotation” and “Base Tilt” are 0, the software is set to upright installation; otherwise, they must be set to 0. If they are not 0, the interface will prompt an error, as shown in Figure 2-6.

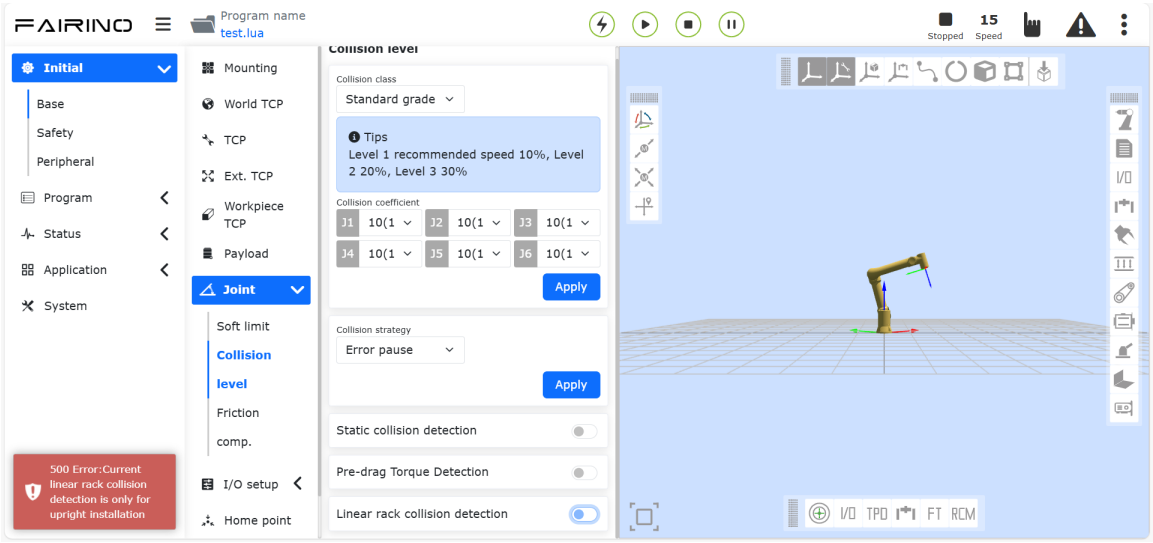


Figure 6.10-6 Error Prompt if Installation Method is Not Upright

Step2: Enable the linear rack guideway collision detection function and set parameters. Click “Initial Setup” -> “Basic” -> “Joints” -> “Collision Level” in sequence to enter the Collision Level setting page. Click the slider for the “Linear Rack Guideway Collision Detection” function, then set the Gear Radius and Slider Mass. The Gear Radius can be calculated from the Lead and Reduction Ratio. The Slider Mass does not include the robot and its carried end load. The guideway level has 11 options, where Level1 is the easiest to trigger a collision, and Level10 is the most difficult. After the controller is powered on, and before executing the adaptation program, set the collision level to “Off” first.

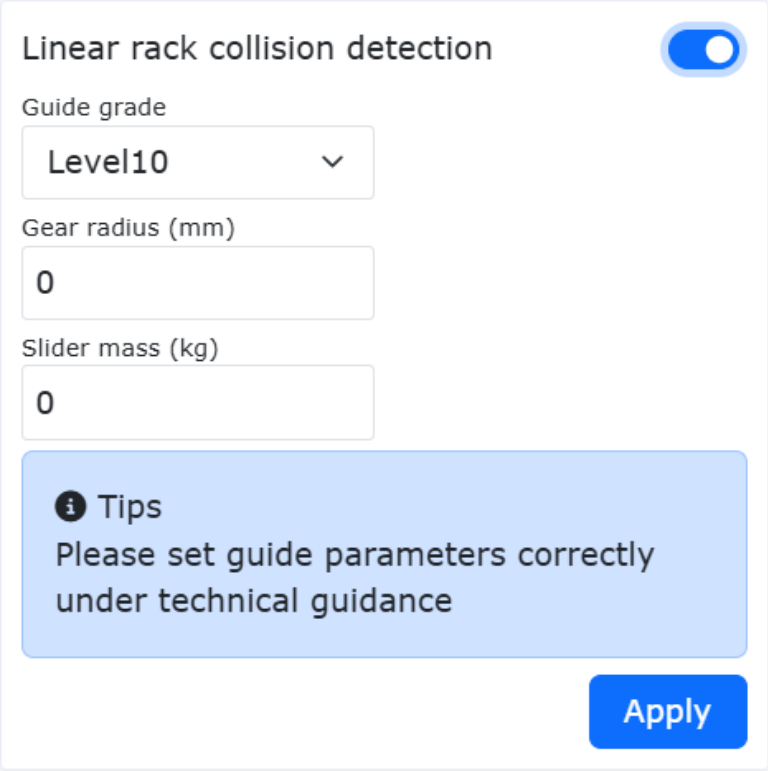


Figure 6.10-7 Linear Rack Guideway Collision Detection Function

Step3: Execute the "Rail_Adaptation_Program.lua" program to adapt to the current guideway. After each controller restart, the "Rail_Adaptation_Program.lua" program must be executed (to prevent changes such as robot type from affecting the dynamic characteristics of the guideway). Before executing the program, ensure the guideway collision level is "Off". In automatic mode, run the lua program at 100% interface speed. Wait for the program to complete one cycle, indicating adaptation is complete, then it can be stopped.



Figure 6.10-8 Execute "Rail_Adaptation_Program.lua" Program to Adapt to Current Guideway

Step4: Reasonably set the guideway collision level and execute tasks. Users can set the guideway collision level reasonably based on the motor drive performance and task running speed. If the guideway and robot are running asynchronously, colliding with the robot or guideway can trigger "Axis 8 Collision Fault, Resettable". At this time, the guideway stops running, as shown in Figure 2-9. If the guideway and robot are running synchronously, colliding with the robot can trigger an alarm, causing the guideway to stop running, and the robot will react according to the set collision strategy.

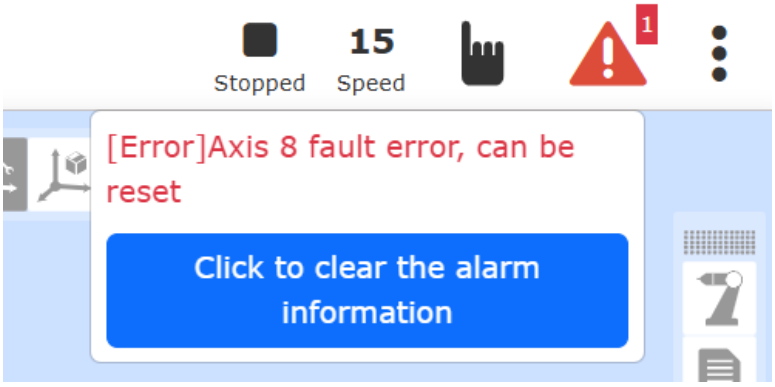


Figure 6.10-9 Guideway Triggering Collision Fault

1.3.6.11 Force Sensor Loaded Zeroing and Open Posture Compliance Admittance Parameters

1.3.6.11.1 Overview

The Force Sensor Loaded Zeroing function is used to quickly clear the sensor's zero drift data when the robot, carrying the quick changer, replaces the load without disassembling the quick changer. The Open Posture Compliance Admittance Parameters allow customers to adjust the posture based on the actual torque magnitude in constant force control.

1.3.6.11.2 Force Sensor Loaded Zeroing

Step1: Install and activate the force sensor. Under the "Initial Setup" -> "Peripherals" -> "Force Sensor" menu, click "Adapted Devices" to enter the configuration interface. After the force sensor configuration is completed, select the configured force sensor number, click the "Reset" button. After the page pops up indicating the command was sent successfully, click the "Activate" button. Check the activation status in the force sensor information table to determine if activation was successful. Additionally, the force sensor will have initial values. Users can choose "Zero Calibration" and "Remove Zero" according to usage needs, as shown in Figure 2-1. Force sensor zero calibration requires ensuring the force sensor is installed vertically downward, and no load is configured below the sensor.

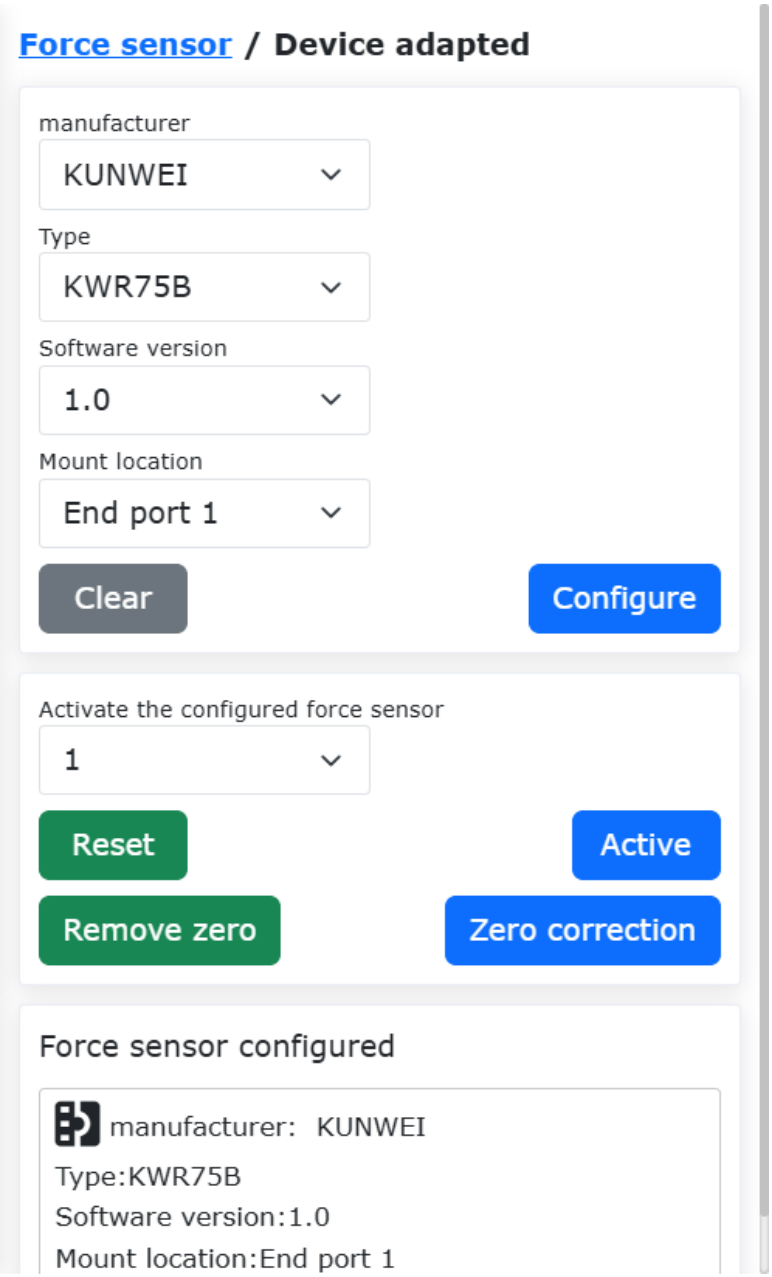


Figure 6.11-1 Force Sensor Configuration Information

Step2: Force sensor load identification. Under the “Initial Setup” -> “Basic” -> “Load” menu, click “Auto Identification” to enter the Force/Torque Sensor Load interface. Perform sensor automatic zeroing, record the initial position, then switch the robot to automatic mode, and click “Auto Zero”, as shown in Figure 2-2.

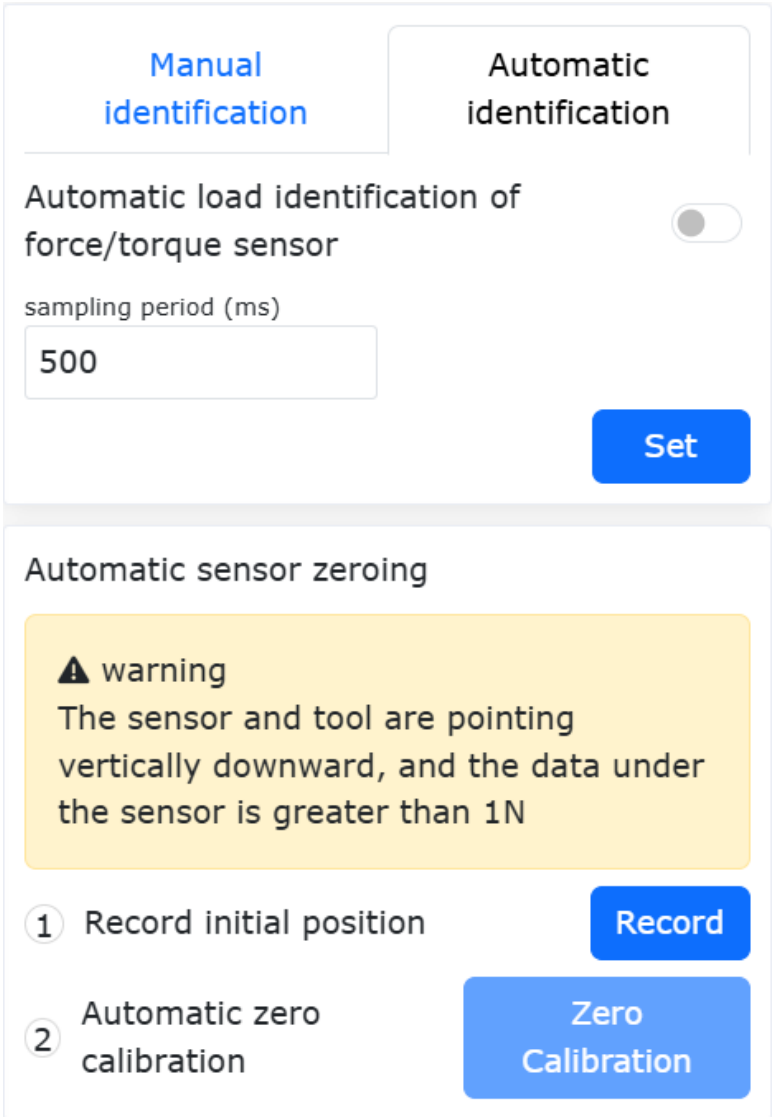


Figure 6.11-2 Force Sensor Load Identification

Step3: If the actual load at the end of the force sensor is replaced, input the load weight and center of mass coordinates in the Load Configuration module, click “Apply” to update the load configuration and complete the force sensor loaded zeroing.

1.3.6.11.3 Open Posture Compliance Admittance Parameters

Step1: Click “Initial Setup” -> “Basic” -> “Tool Coordinate” to enter the Tool Coordinate System setting interface. Select “Coordinate System Name” and set the coordinate system parameters corresponding to the end tool, as shown in Figure 3-1.

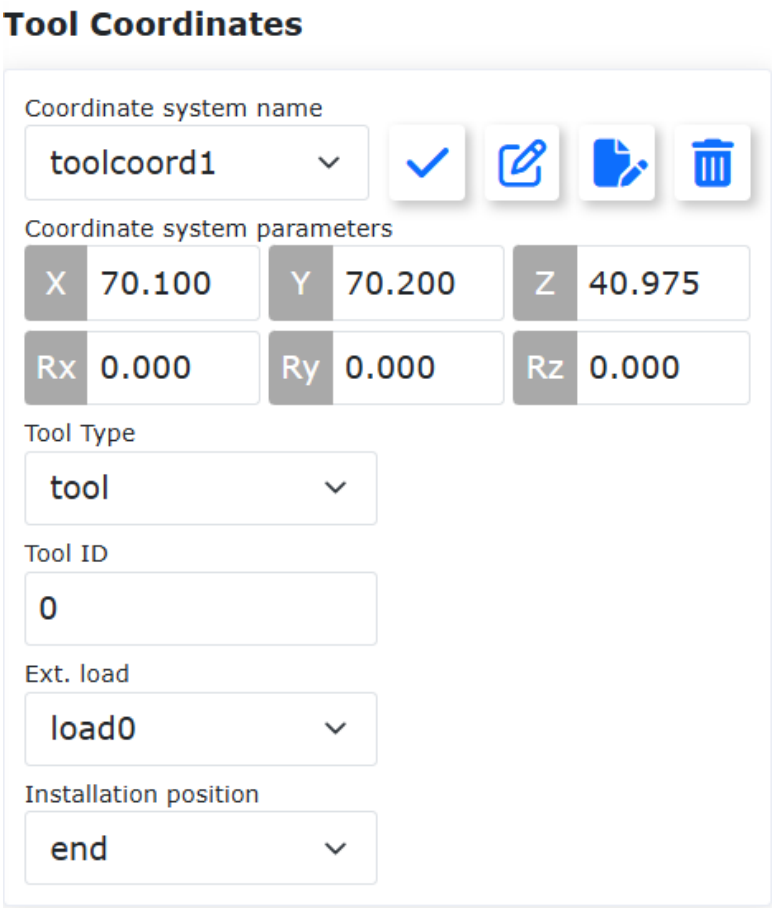


Figure 6.11-3 Tool Coordinate System Setting

Step2: Click “Teach Program” -> “Program Programming”, write a constant force control lua script, select “Force Control Set” -> “Control”, add a force control motion command, set Posture Compliance to “Enable”, set the Inertia Coefficient and Damping Coefficient, and set the Maximum Adjustment Angle as the threshold for the posture compliance angle, as shown in Figure 3-2.

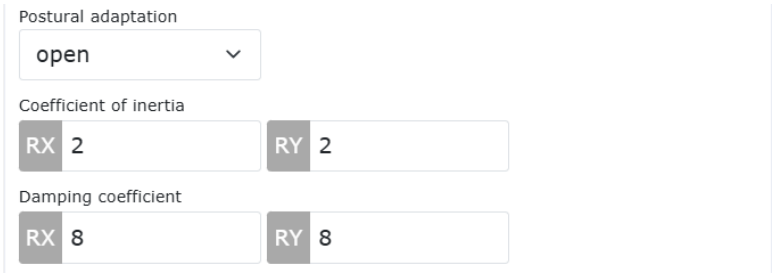


Figure 6.11-4 Open Posture Compliance Admittance Parameters

Step3: On the web interface, click “FT”, set the force sensor reference coordinate system, select the reference coordinate system as “Custom Coordinate System” and set the corresponding coordinate system parameters to “0”, as shown in

Figure 3-3.

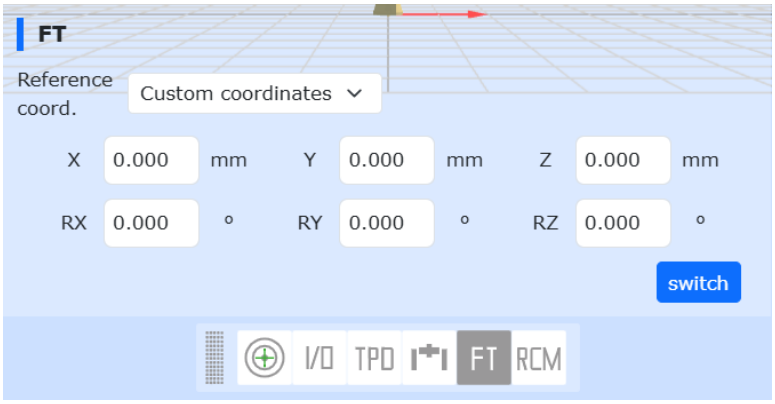


Figure 6.11-5 Set Force Sensor Reference Coordinate System

Step4: Run the script and observe the posture compliance effect. The inertia parameter adjusts the acceleration response and anti-disturbance capability; the larger the inertia, the more obvious the robot lag. The damping coefficient affects the smoothness during posture compliance; the larger the damping, the more difficult the posture compliance.

1.3.7 Safety

1.3.7.1 Stop mode

Click “Initial” -> “Safety” in the menu bar, and then click the “Stop mode” submenu to enter the configuration interface, set the safety stop mode and safety stop policy parameters function.

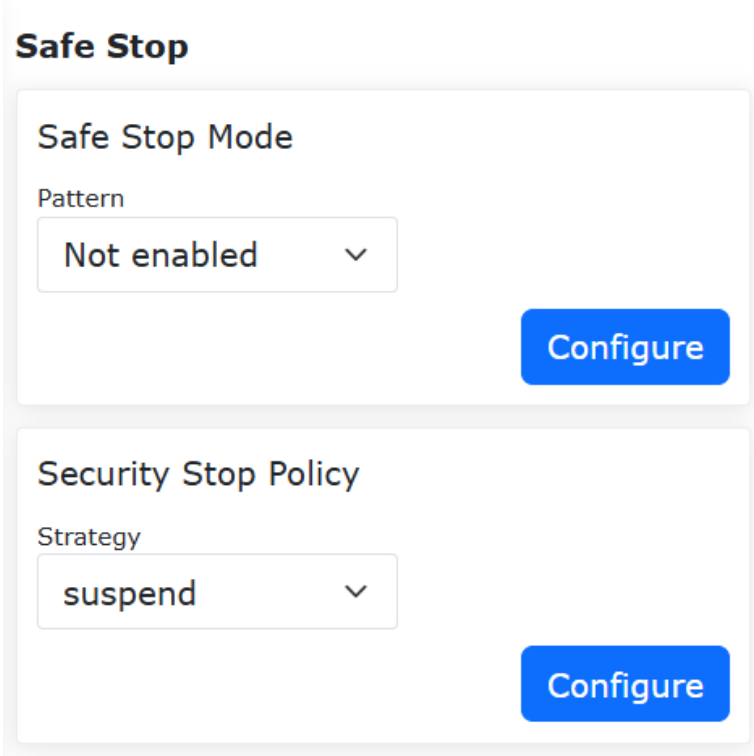


Figure 7.1-1 Safe Stop Configuration

1.3.7.1.1 Configurable Dual-Channel Safety Stop + Reduced Mode

1.3.7.1.1.1 Overview

When the safety stop trigger mode is set to “Dual-Channel”, both channels must be cleared and the warning must be manually cleared on the operation interface before the robot can be reset. Additionally, a reduced mode option is added to the strategy configuration. When this strategy is selected by the user, the robot will enter reduced mode motion.

1.3.7.1.1.2 Operation Procedure

Step1: Click the “Initial Setup” -> “Safety” -> “Safety Stop” button. The trigger mode can be selected as either “Default” or “Dual-Channel”. The difference between the two is: In “Default” mode, the interface error is automatically cleared after triggering and recovery. In “Dual-Channel” mode, the interface error must be cleared manually after triggering and recovery. The “Safety Stop Strategy” can be selected as “Stop”, “Pause”, “Level 1 Reduced Mode”, or “Level 2 Reduced Mode”. Detailed explanations are as follows: When “Stop” is selected, the robot will stop its current motion. When “Pause” is selected, the robot will pause its current motion and will resume after recovery and error clearing. When “Level 1 Reduced Mode” is selected, the robot will enter level 1 reduced mode motion. When “Level 2 Reduced Mode” is selected, the robot will enter level 2 reduced mode motion.

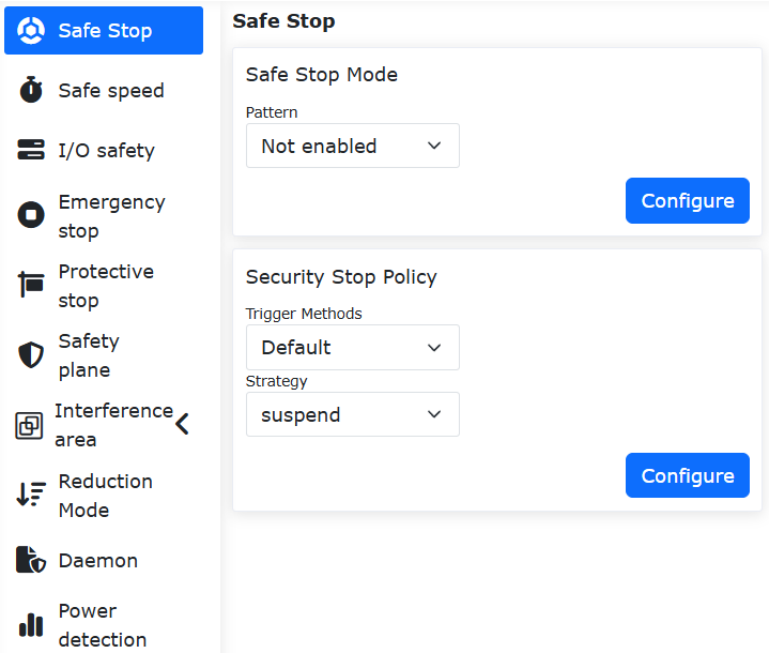


Figure 7.1-2 Safety Stop Strategy Settings

Step2: Since when the trigger mode is selected as “Default”, the interface error can be automatically cleared after triggering and recovery, it does not require much introduction. Therefore, the main focus is on the operation when the trigger mode is selected as “Dual-Channel”: After triggering and recovery, you must manually click the “Clear” operation in the upper right corner before the robot can be reset.

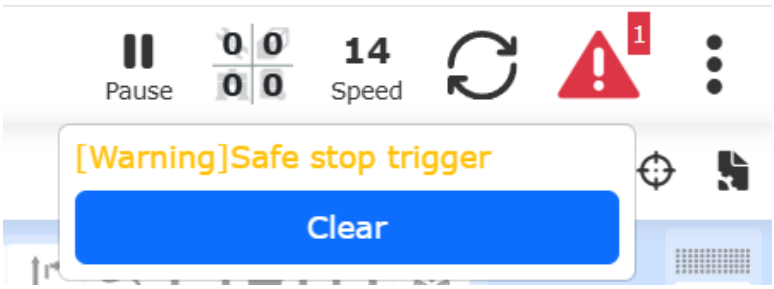


Figure 7.1-3 Manually Clearing a Safety Stop Trigger Operation

1.3.7.2 Safe speed

Click “Initial” -> “Safety” in the menu bar, and then click the “Safe speed” submenu to enter the configuration interface, set the safe speed.

Note: TCP manual speed is less than 250mm/s.

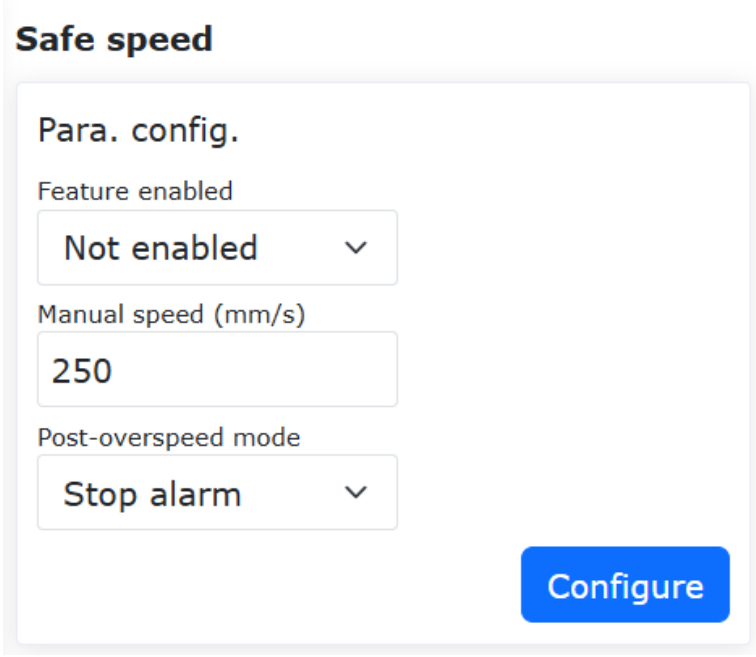


Figure 7.2-1 Safe manual speed configuration

1.3.7.3 I/O safety

Click “Initial” -> “Safety” in the menu bar, and then click the “I/O safety” submenu to enter the configuration interface.

The HMI provides the setting of the safety status of 16 digital inputs and 16 digital outputs, which can be set to valid or invalid status. When the controller determines that it is in a safe state, the 16 digital inputs and 16 digital outputs are set to a safe state.

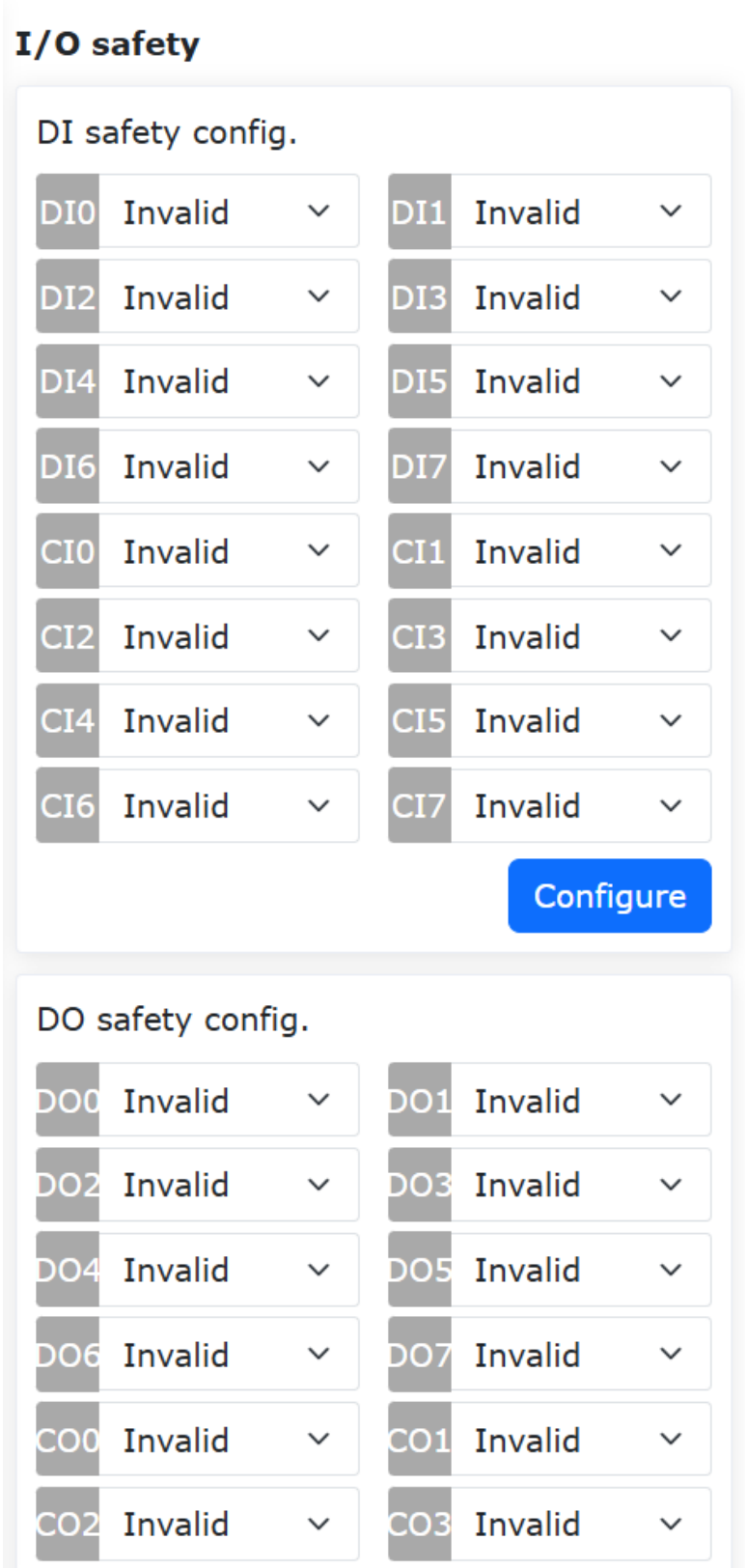


Figure 7.3-1 DIO safety status configuration

On Linux:

The I/O safety function is provided in “DIO Safety”. The safety function is dual-channel DI or DO. When a safety DI signal is detected or the safety status flag is triggered, DO is output.

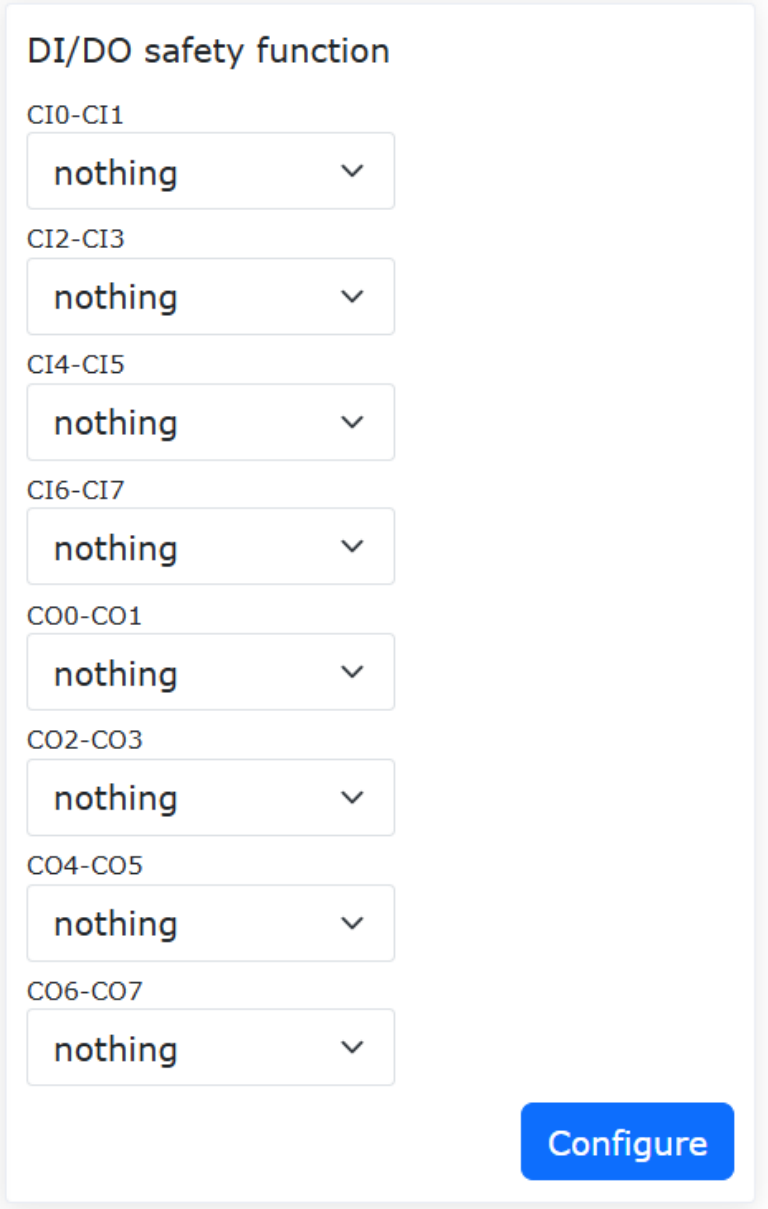


Figure 7.3-2 DIO safety function configuration

1.3.7.4 Emergency stop

Click “Initial” -> “Safety” in the menu bar, and then click the “Emergency stop” submenu to enter the configuration interface.

Emergency stop types 0, 1a, 1b, 2 can be set, stop time limit can be set, and stop distance limit can be set.

- Send the control box board through the controller, and the emergency stop type 0 control box board directly cuts off the power;
- Emergency stop type 1a is to cut off the power supply of the main body after deceleration stop;

- Emergency stop type 1b is to not cut off the power supply of the main body after deceleration stop, and the main body is disabled;
- Emergency stop type 2 indicates pressing the emergency stop button, the robot decelerates to stop and maintains enable status. After releasing the emergency stop, the robot should be able to operate normally.

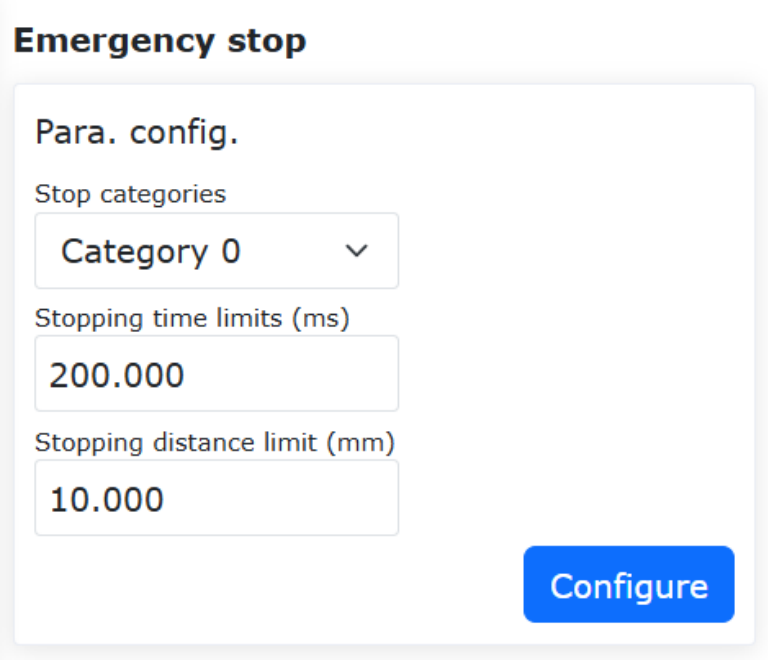


Figure 7.4-1 Emergency stop configuration

1.3.7.4.1 Safety Stop Recovery Optional Auto Enable Function

1.3.7.4.1.1 Overview

After the robot experiences a Category 1b emergency stop, it provides two modes for the user to choose from: Manual Enable and Auto Enable. When Manual Enable is selected, the user needs to change the robot's operation mode to Automatic after releasing the emergency stop button and manually click the enable button to enable the robot. When Auto Enable is selected, the robot will enable automatically after the user releases the emergency stop button.

1.3.7.4.1.2 Operation Process

Step1: Click the "Initial Setup" -> "Safety" -> "Emergency Stop" button. Select "Category 1b" for "Stop Type", and set the "Stop Time Limit" and "Stop Distance Limit" parameters as needed. The "Enable Strategy After E-stop Reset" can be set to "Manual Enable" or "Auto Enable", as shown in Figure 2-1.

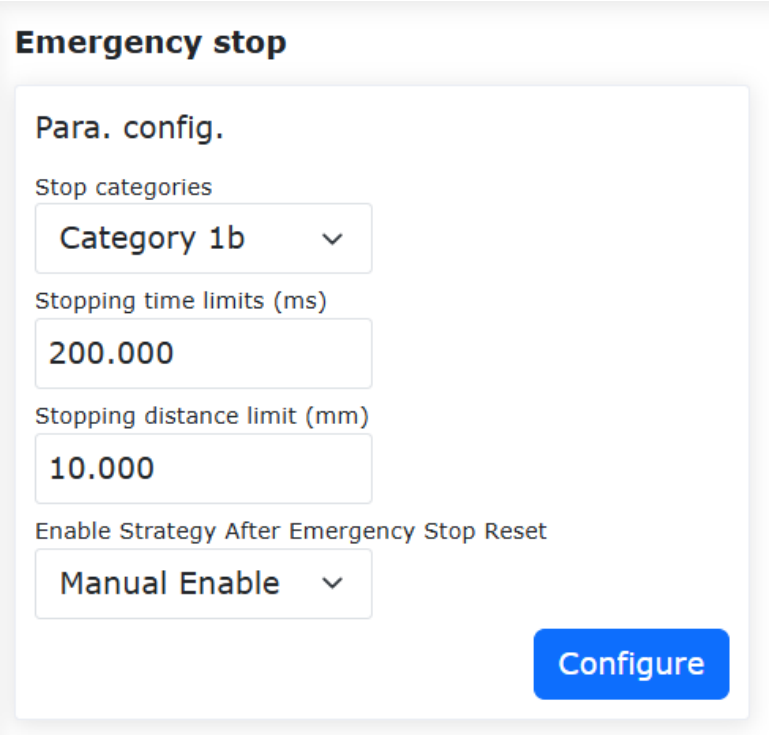


Figure 7.4-2 Enable Strategy Setting

Step2: When "Auto Enable" is selected, the robot will enable automatically after the user releases the emergency stop button. When "Manual Enable" is selected, the user must manually click the enable button in Automatic mode after releasing the emergency stop button to enable the robot, as shown in Figure 2-2.

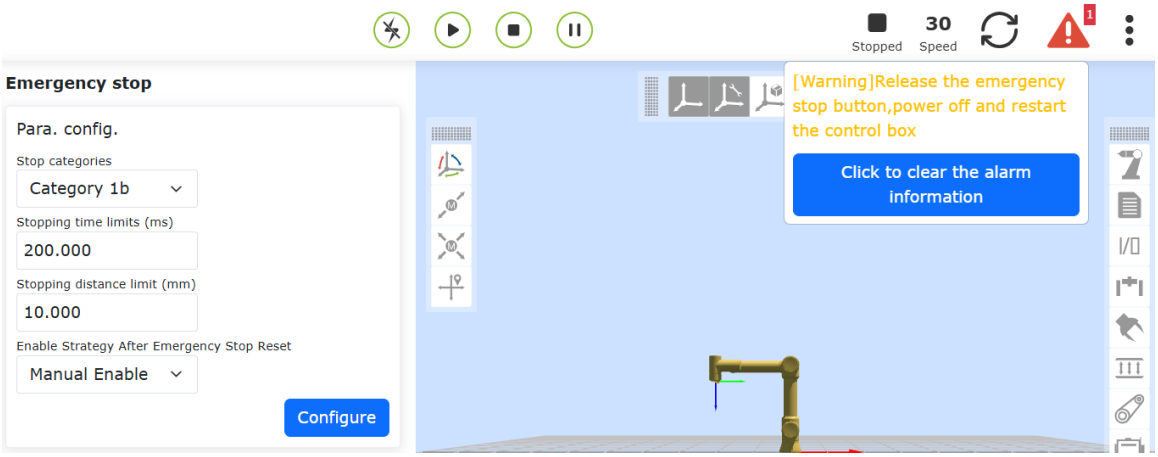


Figure 7.4-3 Manual Enable Operation

1.3.7.5 Protective stop

Click “Initial” -> “Safety” in the menu bar, and then click the “Protective stop” submenu to enter the configuration interface.

Protective stop type 0, 1, 2. Protective stop type 0: the control box board directly cuts off the power. Protective stop type 1: the control box board first notifies the controller to control the robot to stop and then the controller feeds back to cut off the power of the control box board. Protective stop type 2: the control box board notifies the controller to control the robot to stop.

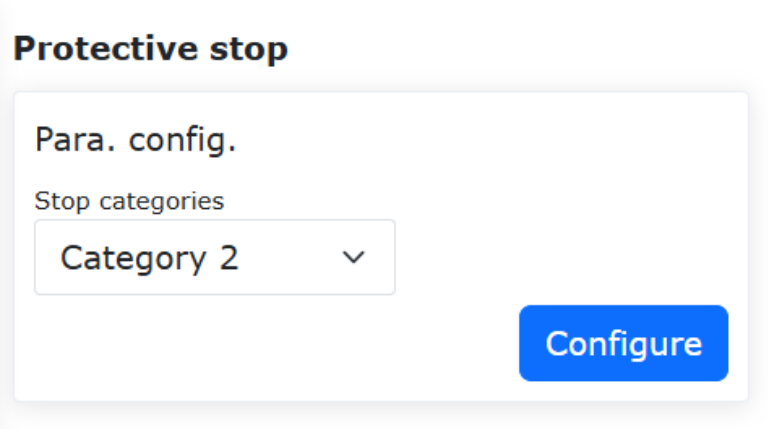


Figure 7.5-1 Protective shutdown configuration

Important: The safety data status flag and control box carrier board fault feedback are obtained through the Web terminal and the controller status feedback. When the flag bit is 1, the safety data status is abnormal in the WebAPP alarm status. After the control box carrier board fault is obtained, the specific error information is displayed in the WebAPP alarm status according to the error code.



Figure 7.5-2 WebAPP alarm status

1.3.7.6 Interference zone configuration

In the menu bar of “Initial” -> “Safety”, click “Interference Zone” to enter the interference area configuration function interface. Click on the “Shaft interference” card to enter the interface, then toggle the “Feature Enabled” slider.

First of all, we need to configure the interference mode and the operation of entering the interference area. The interference mode is divided into “Shaft interference” and “cube interference”. When enabled, the activation sign will be displayed. First, enter the interference zone motion configuration “continue motion” or “stop”.

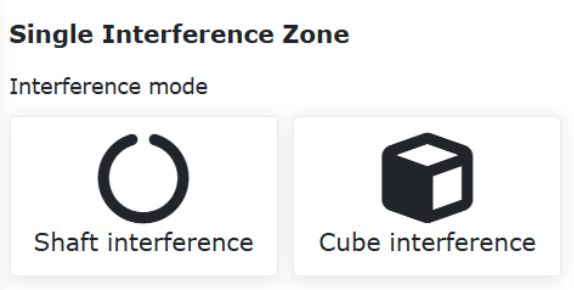


Figure 7.6-1 Interference zone configuration

Next, set the configuration of dragging into the interference area. Users can set the strategy after entering the interference area in drag mode according to their needs, without restricting dragging, impedance callback and switching back to manual mode.

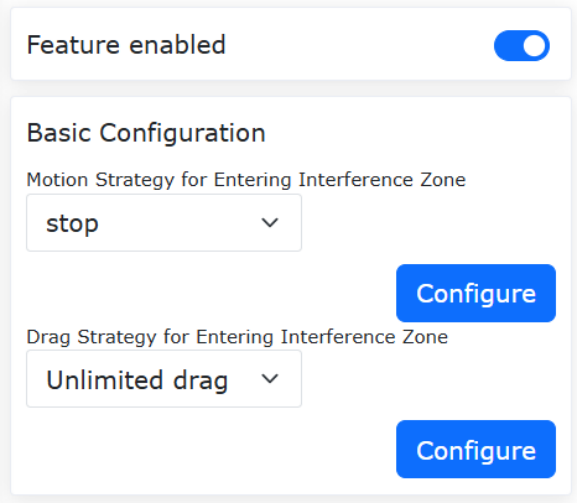


Figure 7.6-2 Interference area drag configuration

To select Shaft interference, you need to configure the parameters of Shaft interference. The detection method is divided into two types: “command position” and “feedback position”. The interference area mode is divided into two types: “interference within the range” and “interference outside the range”. Next Set the range of each joint and whether each joint range is enabled, you can enter the value, or you can record the current position of the robot through the “Robot Teaching” button, and finally click Apply.

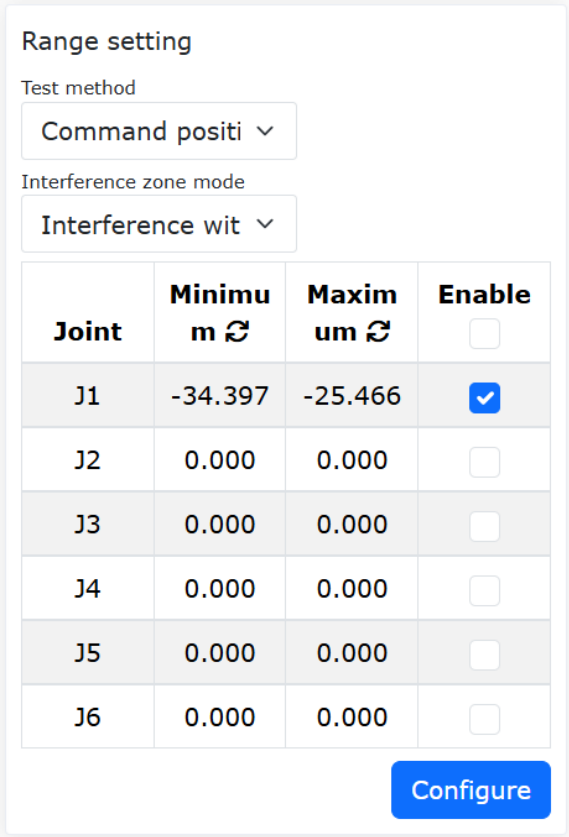


Figure 7.6-3 Shaft interference configuration

To choose cube interference, you need to configure the parameters of cube interference. The detection method is divided into two types: “command position” and “feedback position”. The interference area mode is divided into “interference within range” and “interference outside range”. The system is divided into “base coordinates” and “workpiece coordinates”, which can be selected and set according to actual use. Next, set the range setting. The range setting is divided into two methods. First, look at the first method “two-point method”, which is composed of two diagonal vertices of the cube. We can record the position through input or robot teaching. Finally click Apply.

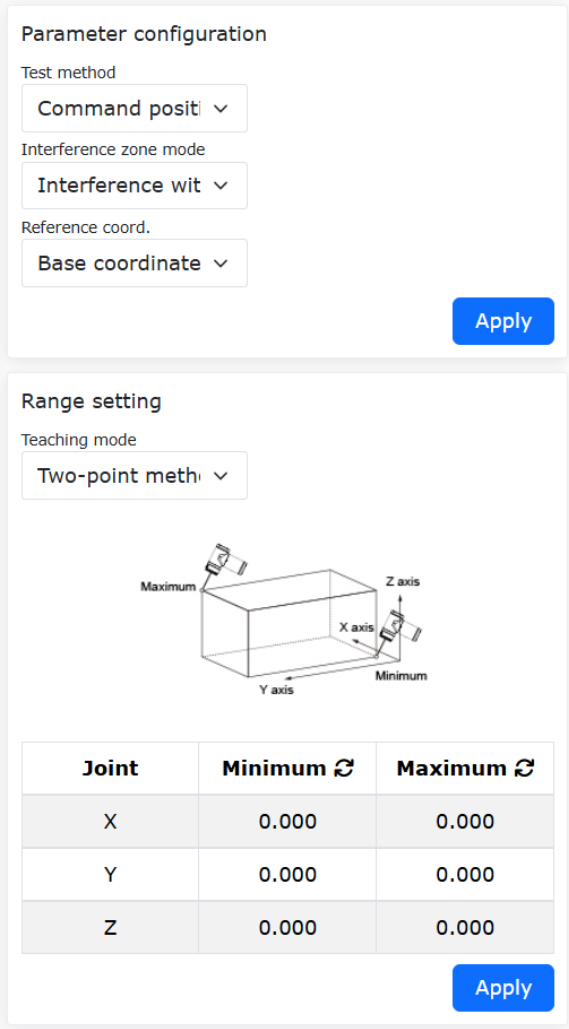


Figure 7.6-4 Cube Interferometric Configuration

Next, look at the second method “center point + side length”, that is, the center point of the cube and the side length of the cube form an interference area, and we can record the position through input or robot teaching. Finally click Apply.

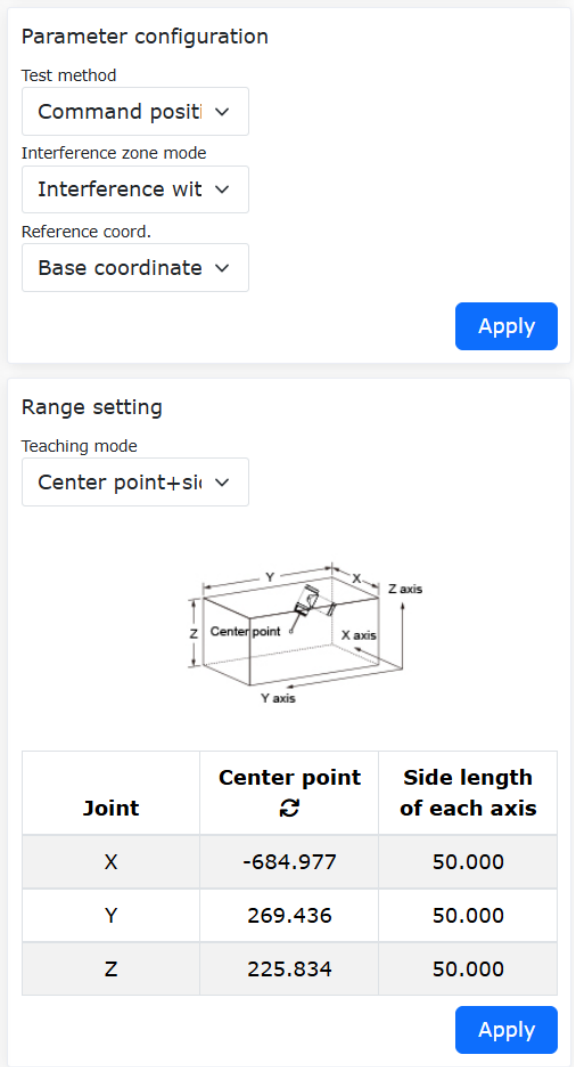


Figure 7.6-5 Cube Interferometric Configuration

1.3.7.6.1 Safety Callback Function for Force Sensor-Assisted Dragging into Shaft interference Zone

1.3.7.6.1.1 Overview

The safety callback function for force sensor-assisted dragging into Shaft interference zones automatically switches the robot to dragging mode with impedance callback effect when entering an interference zone during force sensor-assisted dragging, and reverts to force sensor-assisted dragging when exiting. This satisfies various user scenarios during force sensor-assisted operations.

1.3.7.6.1.2 Operation Procedure

1.3.7.6.1.3 Joint Limit Ring

Step1: Log into the web interface, toggle the "Joint Limit Ring" switch, and the joint limit rings will appear on robot joints as shown below.

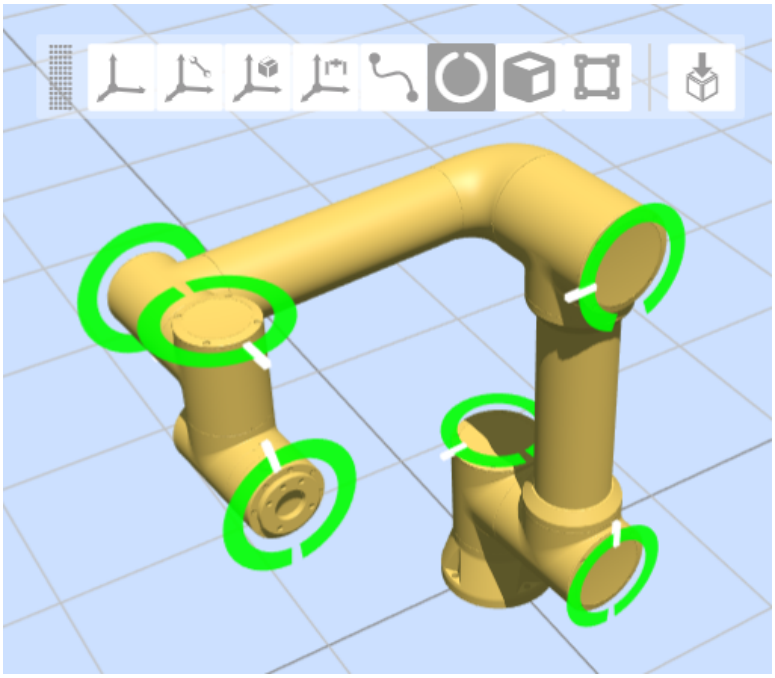


Figure 7.6-6 Joint Limit Ring on Web Interface

Step2: The white marker on the ring indicates actual joint angle; the gap represents soft limit positions (gap size varies with limit settings); rings remain stationary relative to joints during motion.

1.3.7.6.1.4 Shaft interference Configuration

Step1: Configure and activate Shaft interference. Navigate to: "Initial" → "Safety" → "Interference Zone" → "Single", select "Shaft interference" and toggle "Enable".

Step2: Set "Motion Strategy" to "Continue Motion", select "Dragging Strategy" as "Impedance Callback" and configure parameters (e.g., recommended value "5" for callback force intensity).

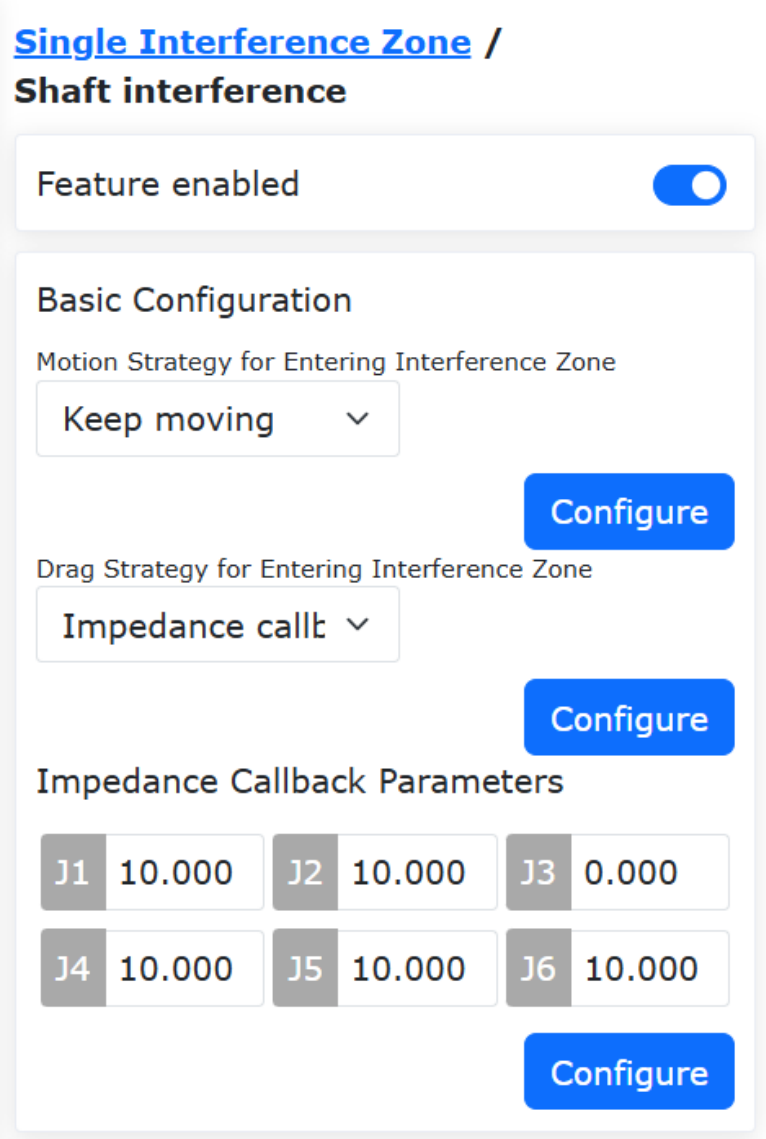


Figure 7.6-7 Shaft interference Configuration

Step3: Set interference ranges. Choose "Feedback Position" detection mode, select "Inside Range" or "Outside Range" interference mode, then enable ranges for each axis.

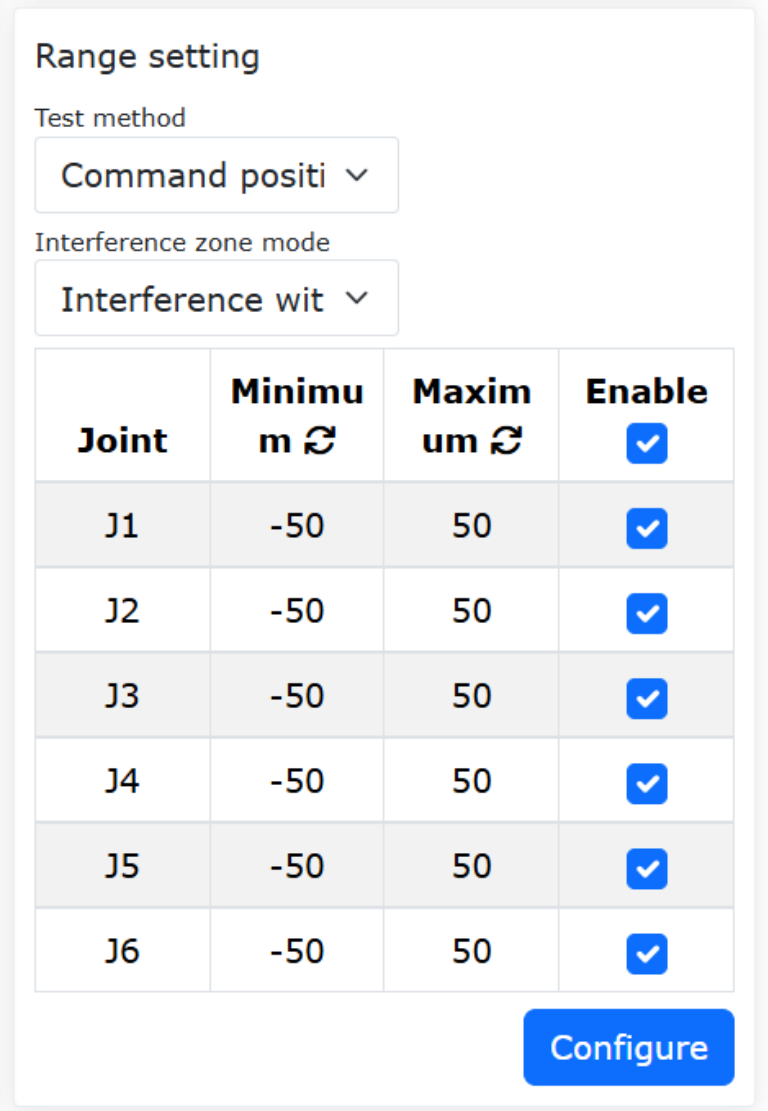


Figure 7.6-8 Interference Range Configuration

Step4: In “Inside Range” mode, green indicates free movement zones and yellow shows interference zones on the web interface.

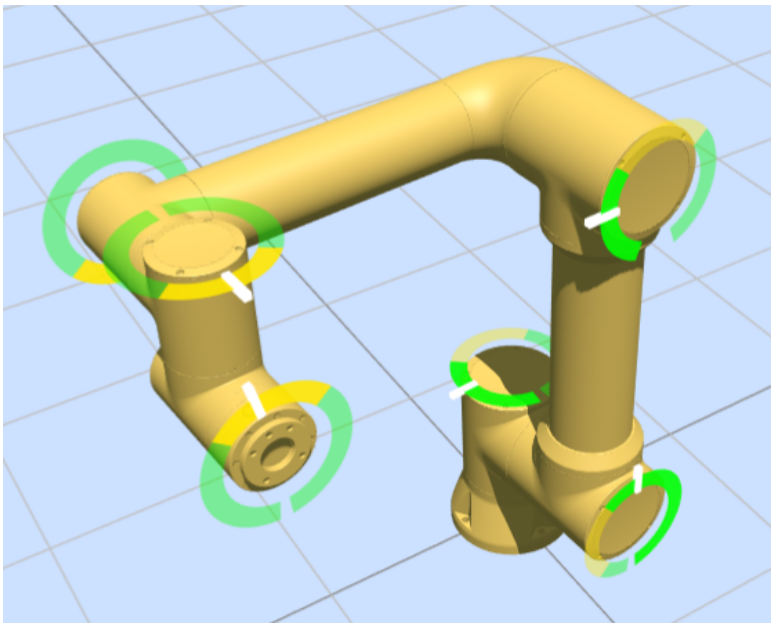


Figure 7.6-9 Limit Ring Display (Inside Range)

Step5: In “Outside Range” mode, the color scheme reverses while maintaining the same display logic.

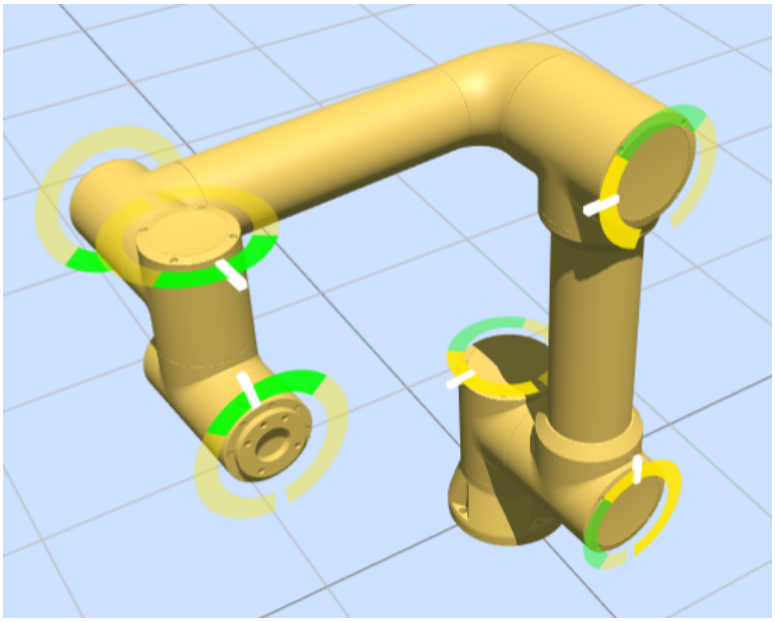


Figure 7.6-10 Limit Ring Display (Outside Range)

1.3.7.6.1.5 Entering Shaft interference Zone with Force Sensor Assistance

Step1: Enable force sensor assistance at: "Auxiliary Applications"→"Tool App"→"Drag Lock", then activate interference zone options.

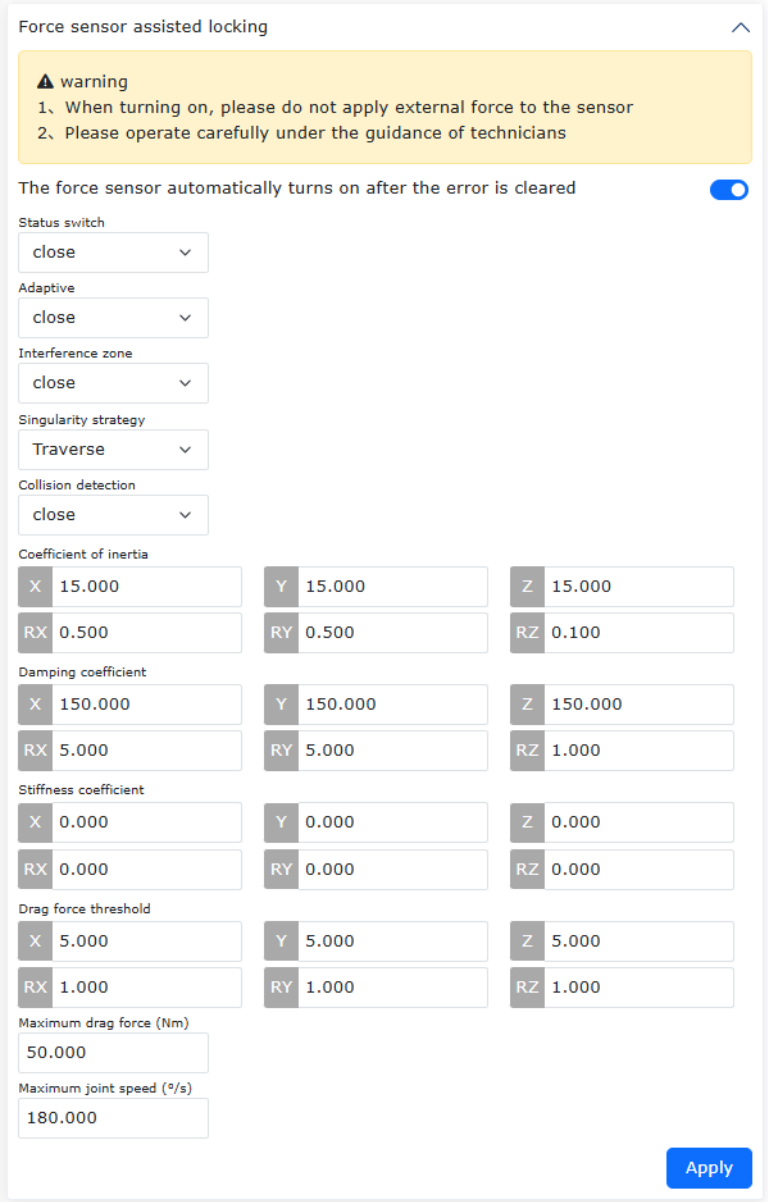


Figure 7.6-11 Force Sensor Drag Configuration

Step2: During force-assisted dragging, the system automatically switches to current-loop dragging with impedance callback when entering interference zones, then reverts upon exit.

1.3.7.6.1.6 Cuboid Interference Configuration

Step1: Configure cuboid interference at: "Initial" → "Safety" → "Interference Zone" → "Single". Click on the "Cube Interference" card to enter the interface, and turn on the "Feature Enabled" slider.

Step2: Set "Motion Strategy" to "Continue Motion" and "Dragging Strategy" to "Unrestricted Dragging".

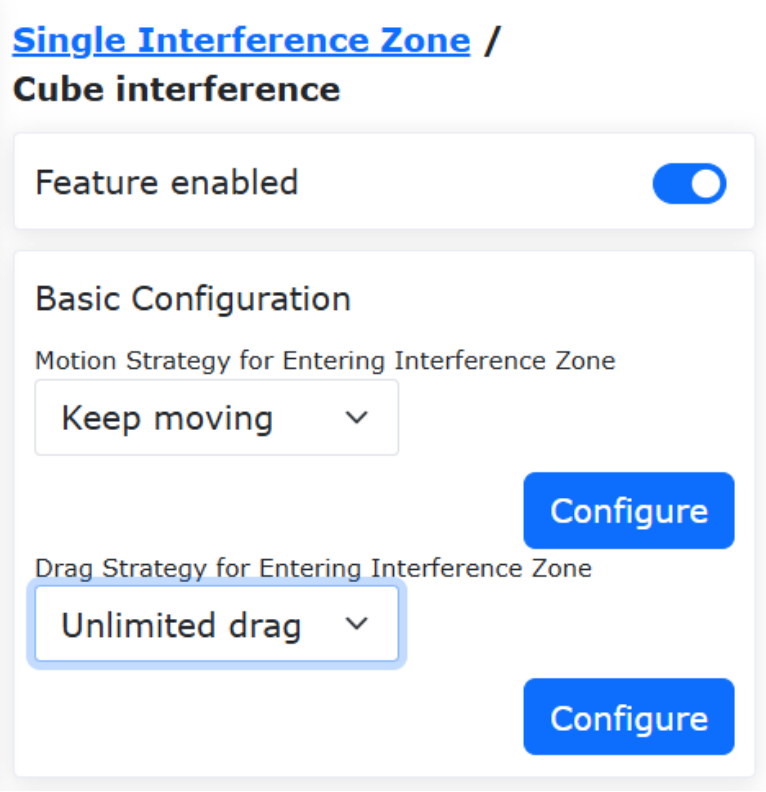


Figure 7.6-12 Cuboid Interference Settings

Step3: Configure parameters including "Base Coordinate" reference and "Two-Point" or "Center+Edge Length" teaching methods.

Step4: Virtual cuboids appear on the web interface (40% opacity yellow/green for normal state, 90% when triggered).

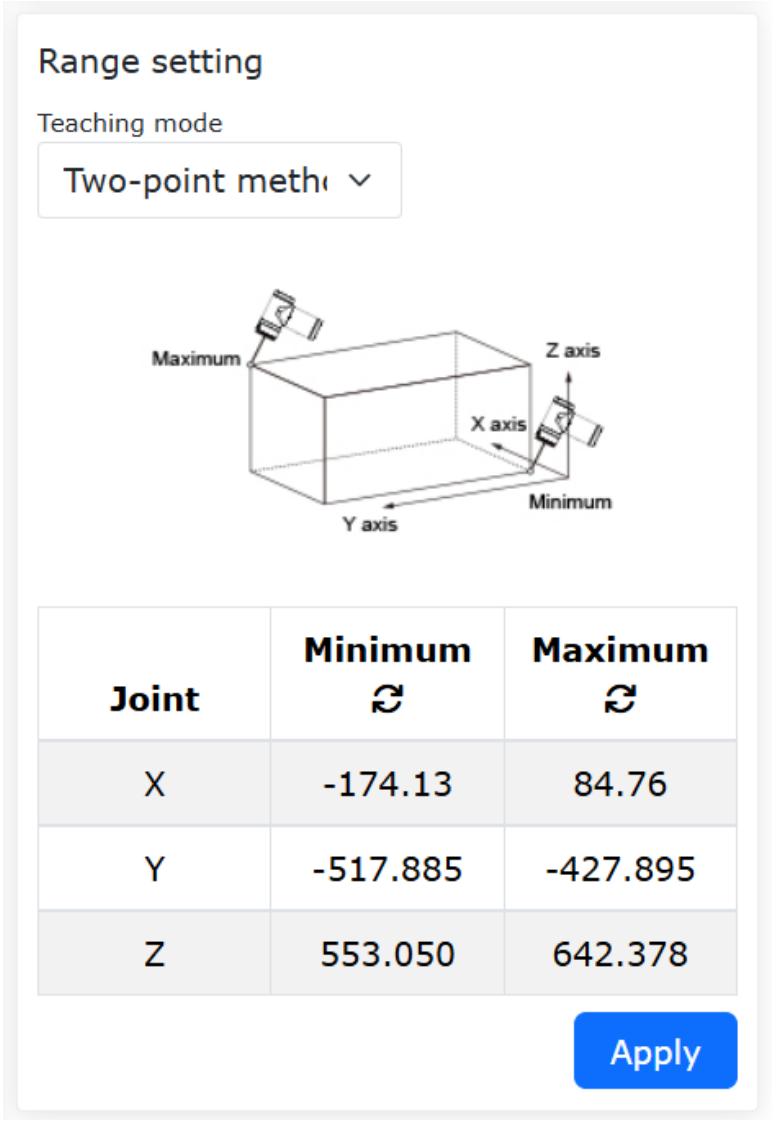


Figure 7.6-13 Two-Point Teaching Method

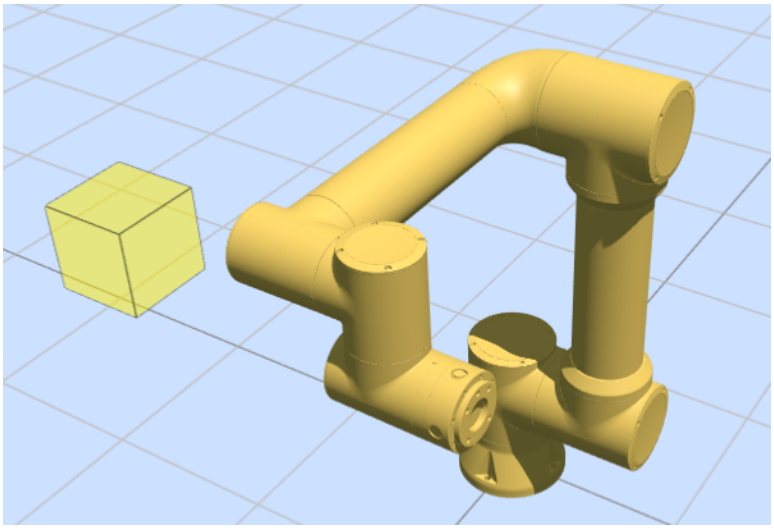


Figure 7.6-14 Web interface virtual wall display

Step5 The teaching method for selecting the interference zone of the cube is "center point + side length." Teach the robot a point, set the side lengths along the X, Y, and Z axes with the taught point as the center, as shown in the figure below. After clicking "Apply," a virtual cube will appear on the web interface.

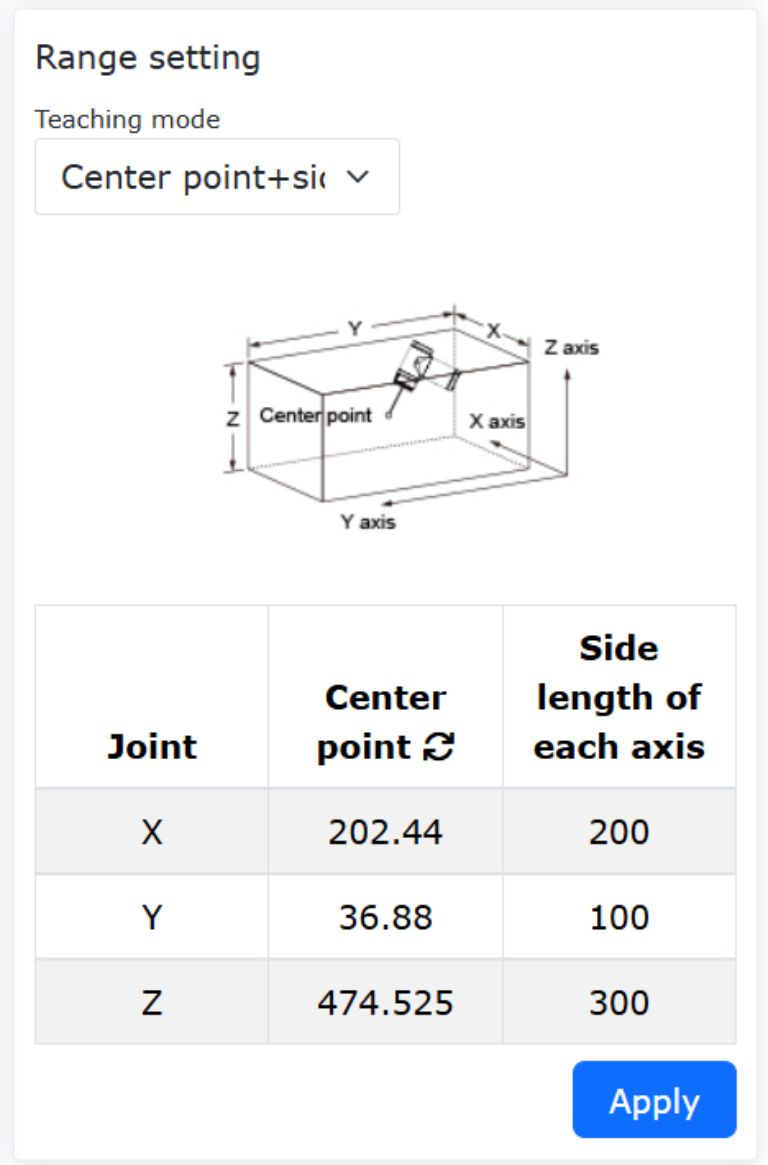


Figure 7.6-15 "Center point + side length" to set cube interference zone

Step6 Set the "interference zone mode" to "interference within range." When the robot end is outside the cubic range, the virtual cube on the web interface appears as yellow with 40% transparency. When the robot end is within the cubic range, the cube turns yellow with 90% transparency, and an "Entering interference zone" warning appears, as shown in the figure below.

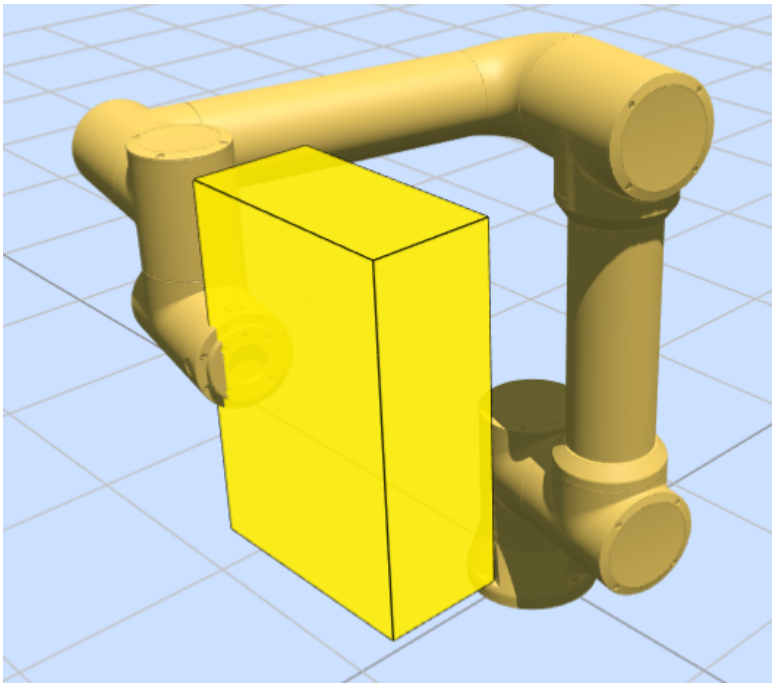


Figure 7.6-16 “In-range interference” mode when entering the cubic interference zone

Step7 Set the “Interference Zone Mode” to “Interference Outside the Range”. When the robot end is within the cube range, the virtual cube on the web interface is displayed as 40% transparency green. When the robot end is outside the cube range, the cube appears as 90% transparency green, and an “Entering Interference Zone” warning is displayed, as shown in the figure below.

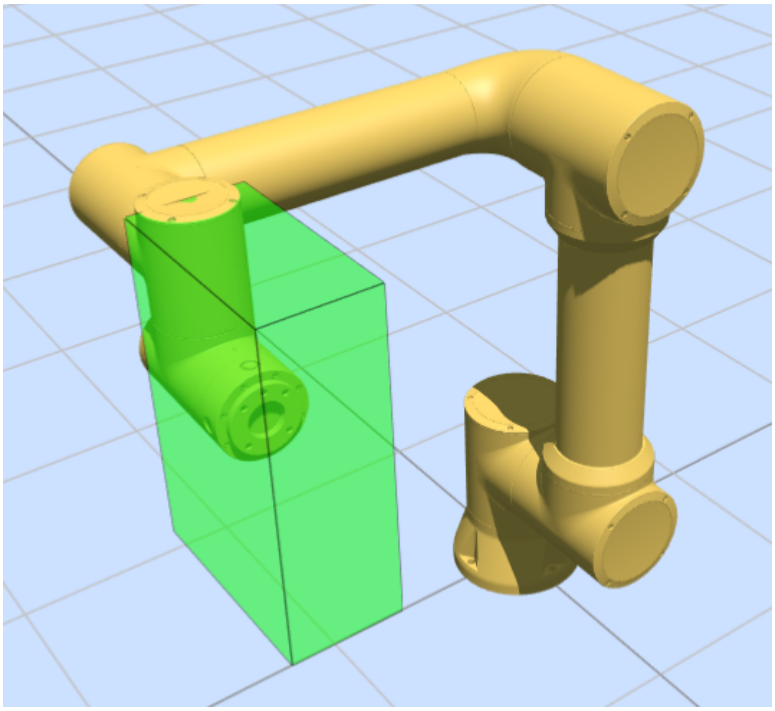


Figure 7.6-17 “Out-of-Range Interference” mode cube interference zone display

1.3.7.6.1.7 Safety plane

Step1: Configure up to 8 safety walls at: “Initial”→”Safety”→”Safety plane”. Enabled walls appear as 40% orange translucent objects.

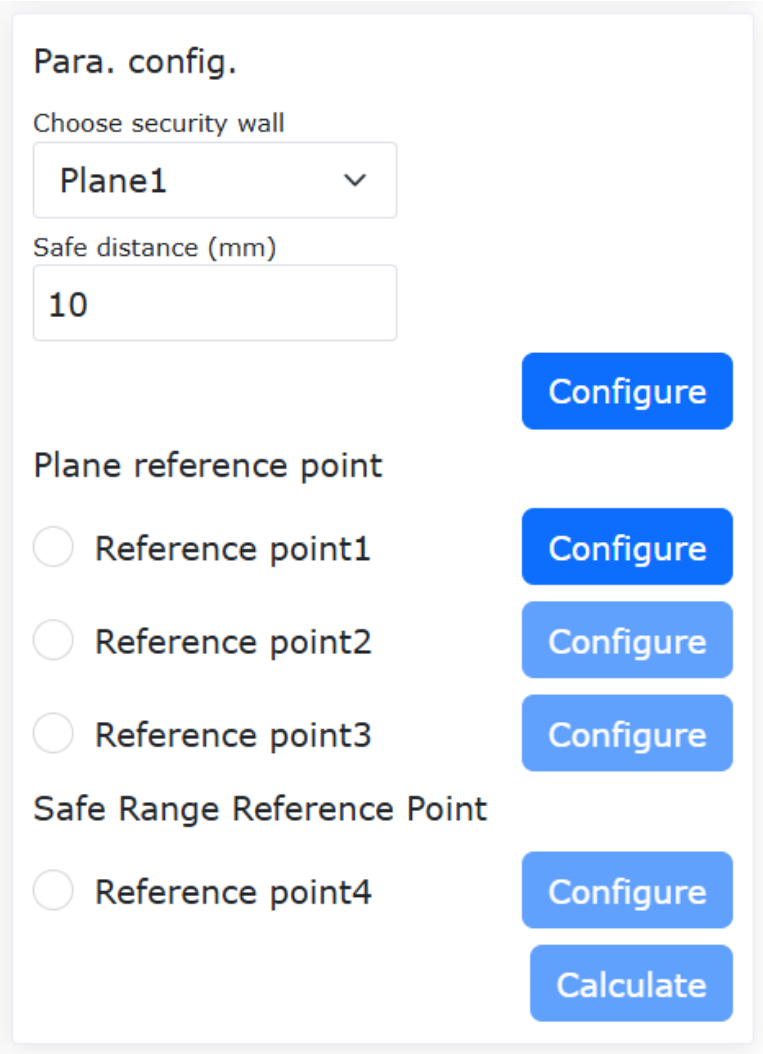


Figure 7.6-18 Safety Wall Setup

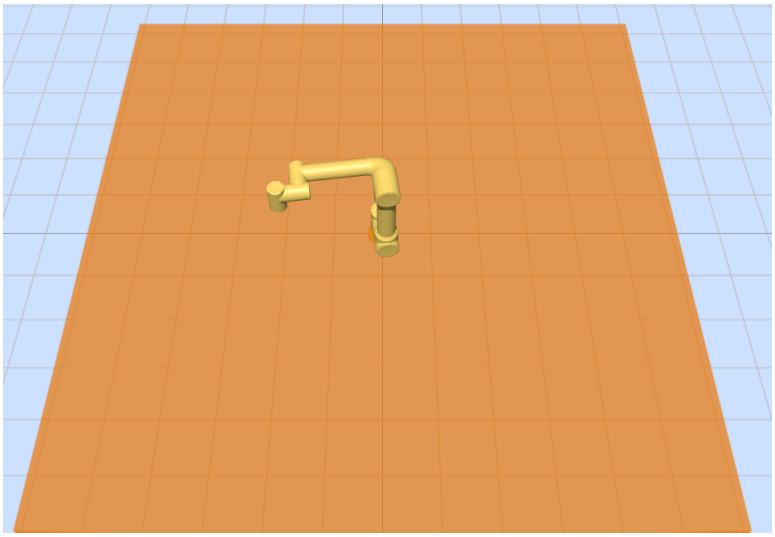


Figure 7.6-19 Web interface virtual wall display

Step2: Walls turn 90% opaque orange with warnings when breached.

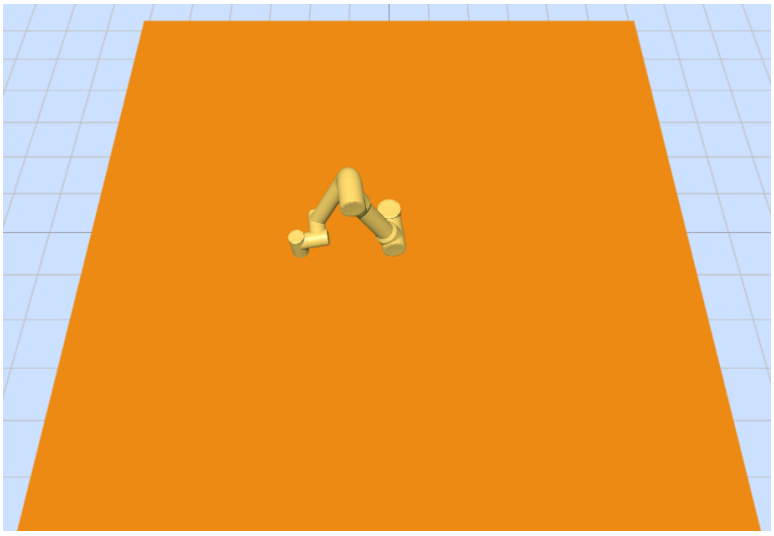


Figure 7.6-20 Triggered Safety Wall

1.3.7.6.2 Cube Interference Function

1.3.7.6.2.1 Overview

The cube interference function supports defining and activating multiple independent cube interference zones simultaneously. The position and dimensions of each interference zone in three-dimensional space can be independently configured. Furthermore, each interference zone is equipped with an individual CO (Controller Output) trigger signal output, capable of outputting corresponding trigger signals based on the robot's real-time position.

1.3.7.6.2.2 Operation Procedure

Step1: Enable the cube interference function and perform basic configuration. Sequentially click the commands “Initial Settings” -> “Safety” -> “Interference Zone” -> “Cube Interference”. Use the slider switches to control whether each cube interference zone is enabled, and perform the basic configuration.

Among these settings, the motion strategy upon entering the interference zone can be selected as “Continue Motion” or “Stop”. When “Continue Motion” is selected, the robot will display a warning but continue moving upon entering the interference zone. When “Stop” is selected, the robot will display a warning and stop moving upon entering the interference zone. The drag strategy upon entering the interference zone can be selected as “Unrestricted Dragging,” “Impedance Callback,” or “Switch Back to Manual Mode”.

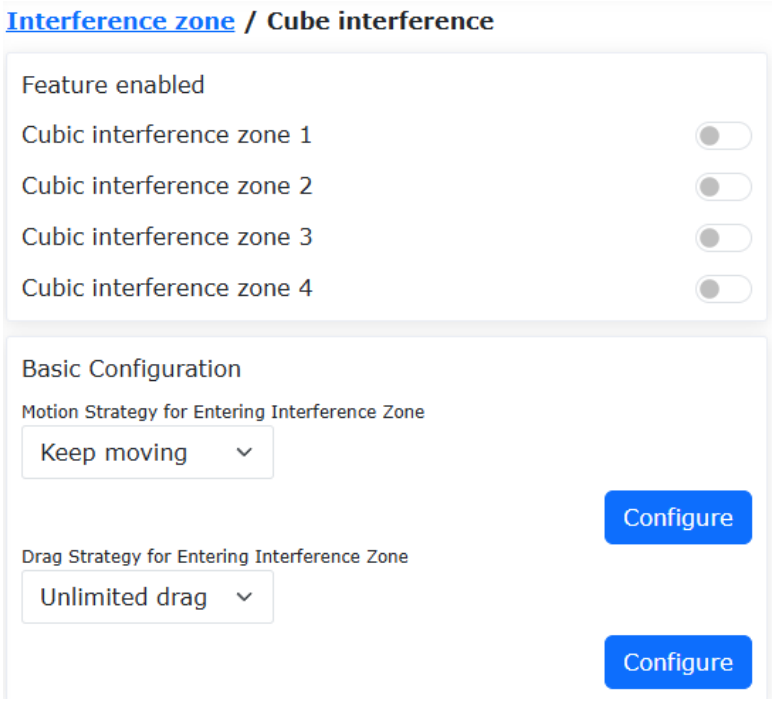


Figure 7.6-21 Cube Enable Control and Basic Configuration

Step2: Configure the cube interference zones. Different configuration parameters can be set for each interference zone ID. It is important to note:

(1) The detection method must be selected based on actual functional requirements as “Command Position” or “Feedback Position”.

(2) When the interference zone mode is selected as “Interference Outside Range,” it is only effective for a single interference zone.

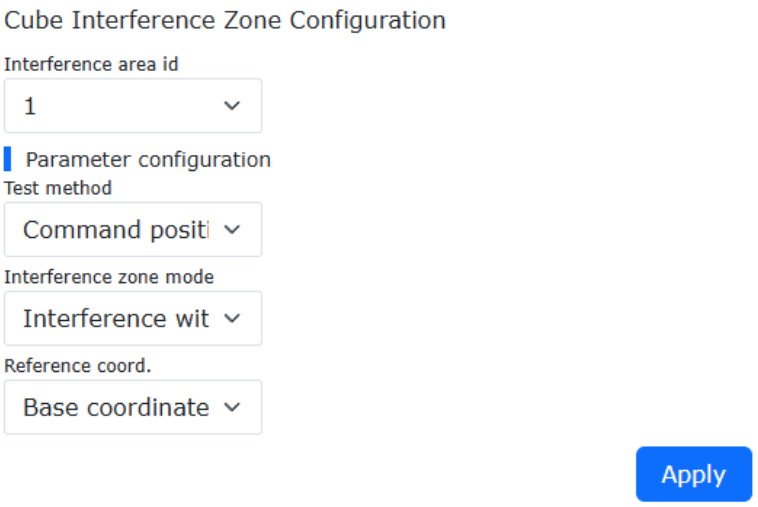


Figure 7.6-22 Cube Interference Zone Configuration

Step3: Set the interference zone range. The range can be set by choosing the "Two-Point Method" or "Center Point + Side Lengths" method to generate the cube interference zone. The "Two-Point Method" generates the zone by specifying two opposite vertices of the cube. The "Center Point + Side Lengths" method generates the zone by specifying the center point and three side lengths of the cube.

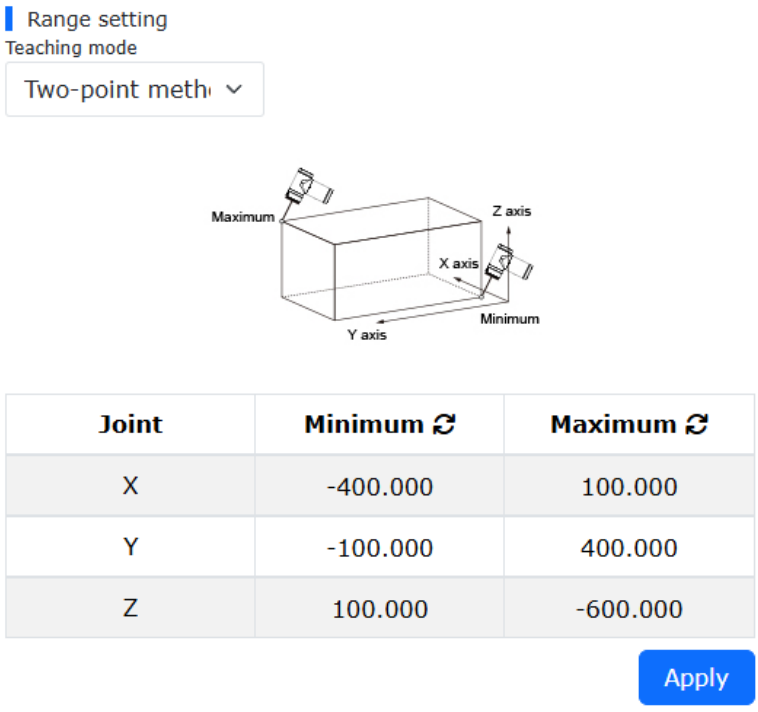


Figure 7.6-23 Generating Interference Zone via "Two-Point Method"

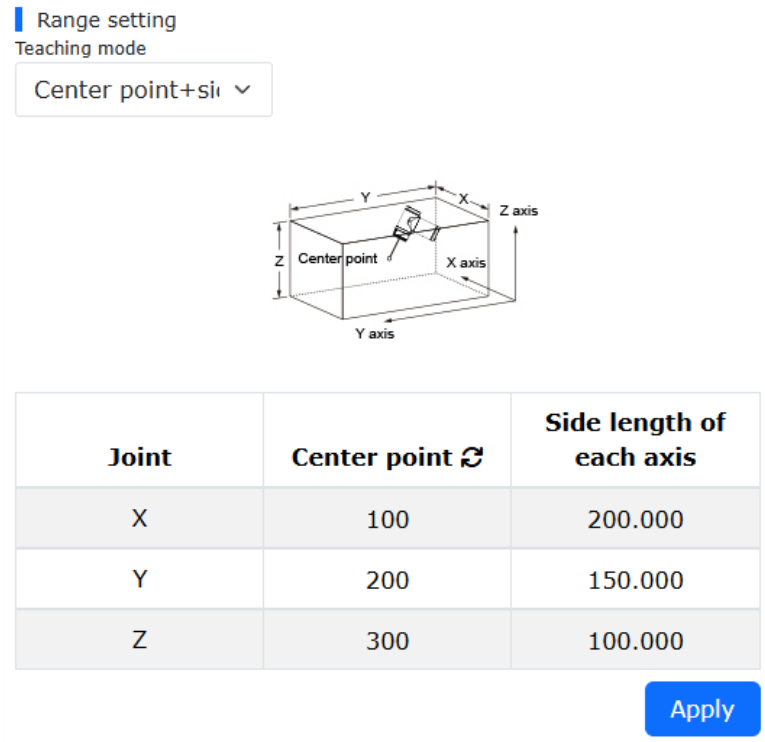


Figure 7.6-24 Generating Interference Zone via “Center Point + Side Lengths”

Step4: Configure CO signals. Sequentially click the commands “Initial Settings” -> “Basic” -> “I/O Settings” -> “DO” to configure the corresponding CO output for each cube.

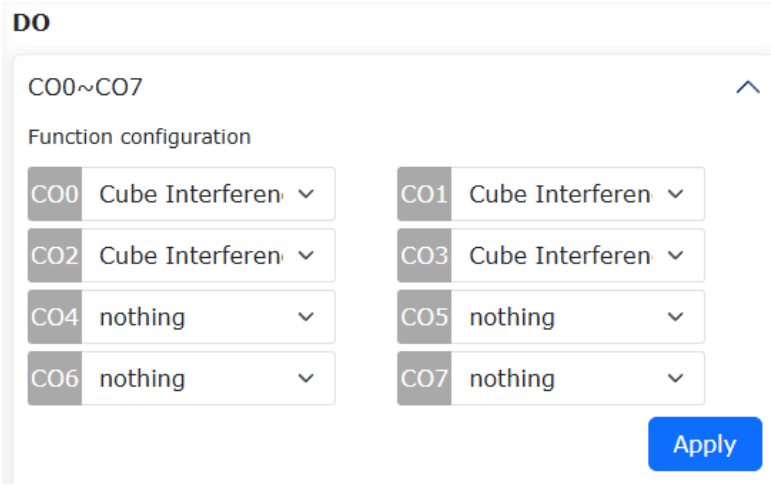


Figure 7.6-25 CO Output Configuration

Step5: Each cube interference zone will be displayed on the robot interface according to its set ID number. When the robot’s end-effector center point enters an interference zone, the interface will display an “Entered Interference Zone” warning, and the corresponding CO interface will output a signal.

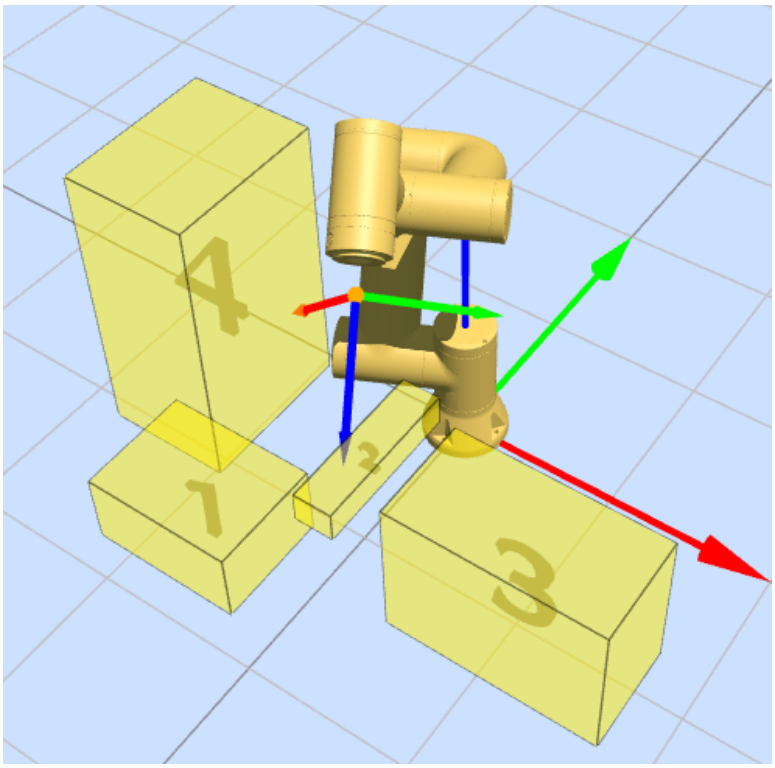


Figure 7.6-26 Multi-Cube Interference Zone Interface Display

1.3.7.7 Reduction Mode

Click on the “Initial” -> “Safety” in the menu bar, then select the “Reduction Mode” submenu to enter the configuration interface, and choose the “Level 1/2 reduction Mode” to configure joint speed and end TCP speed.



Figure 7.7-1 Reduction Mode

1.3.7.8 Safety plane

Click “Initial” -> “Safety” in the menu bar, and then click the “Safety plane” submenu to enter the configuration interface.

- **Safety Plane Configuration**:Click the enable button to enable the corresponding security plane. When the security plane is not configured with a security range, an error will be prompted. Click the drop-down box, select the security plane you want to set, and automatically bring out the security distance (you can not set it, the default value is 0), and then click the “Setting” button to set it successfully.

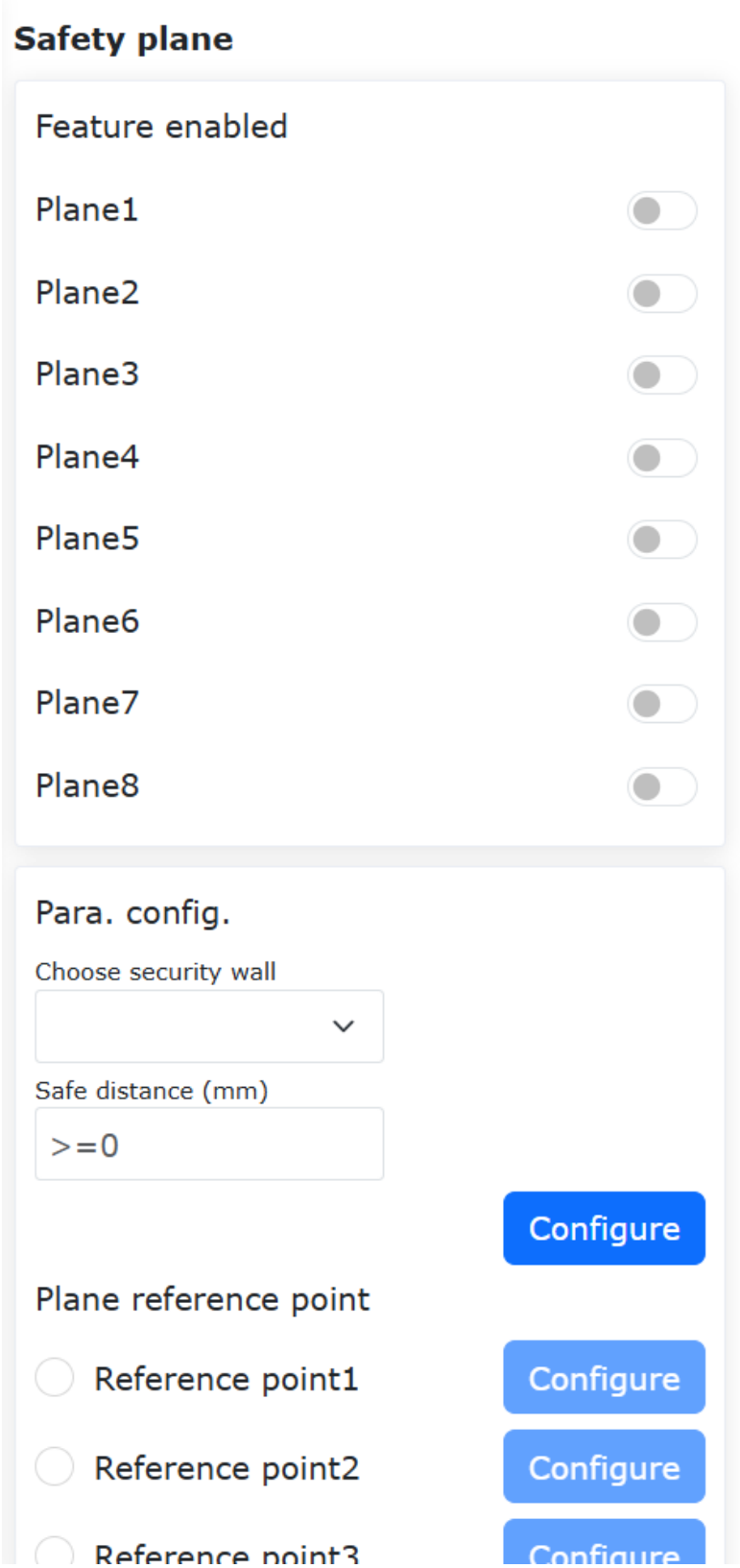


Figure 7.8-1 Safety Plane Configuration

- **SSafety Plane Reference Point Configuration**:After selecting a security plane, four reference points can be set.

The first three points are plane reference points, which are used to confirm the plane of the safety wall set. The fourth point is the safety range reference point, which is used to confirm the safety range of the set safety wall.

Important: If the reference point is set successfully, the green light will be on. Otherwise, the yellow light is on. It turns green until the reference point is set successfully. When the four reference points are all set successfully, the safety range can be calculated, and the safety range parameter point status will return to the default after the calculation is successful.

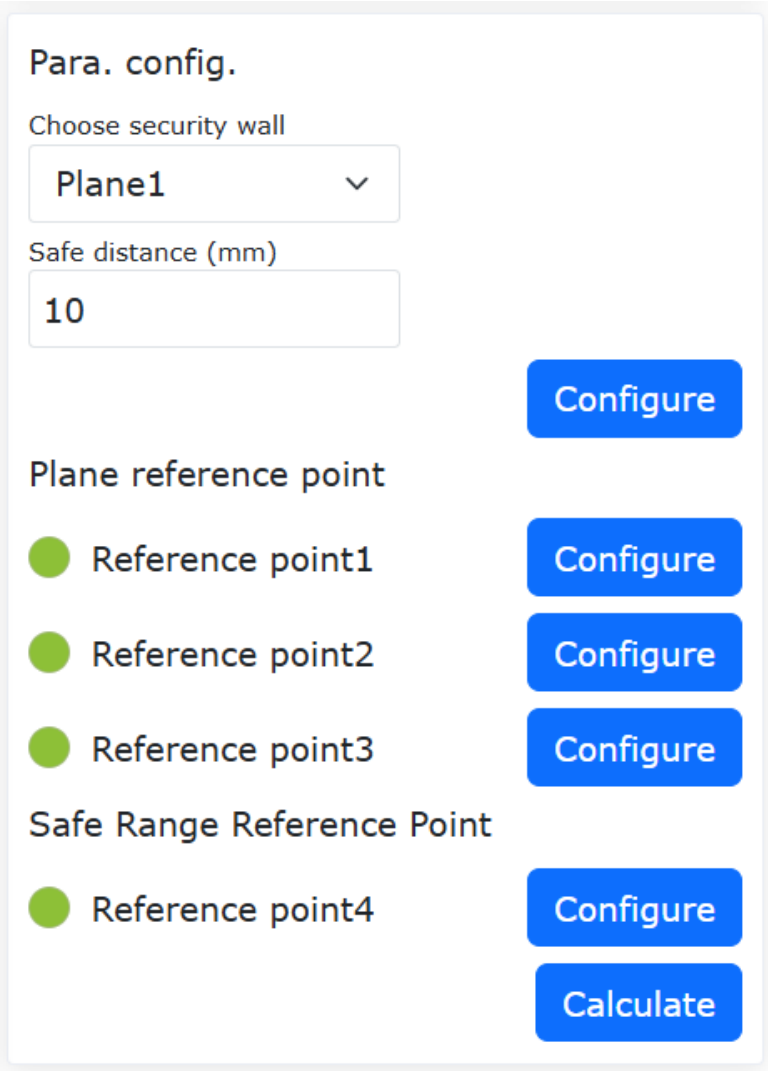


Figure 7.8-2 Safe range reference point setting

- Apply effects: The successfully configured security plane is enabled. Drag the robot, if the TCP at the end of the robot is within the set safety range, the system is normal. If it is outside the set safety range, an error will be prompted.

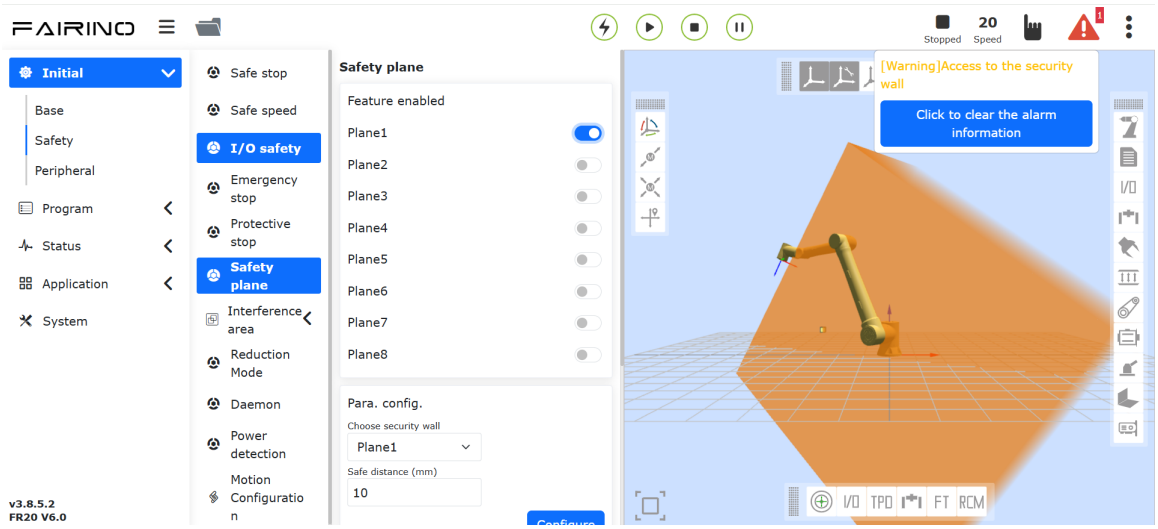


Figure 7.8-3 The effect picture after the security range is set successfully

1.3.7.9 Daemon

Click “Initial” -> “Safety” in the menu bar, and then click the “Daemon” submenu to enter the configuration interface.

The user clicks the “function enabled” button to open or close the setting of the daemon. Select “Unexpected Situation” and “Background Program”, and click the “Settings” button to configure the parameters of the unexpected situation handling logic.

Enable the security background program and set the unexpected scene and background program. When the user starts to run the program and the unexpected situation matches the set unexpected situation, the robot will execute the corresponding background program to play a role of security protection.

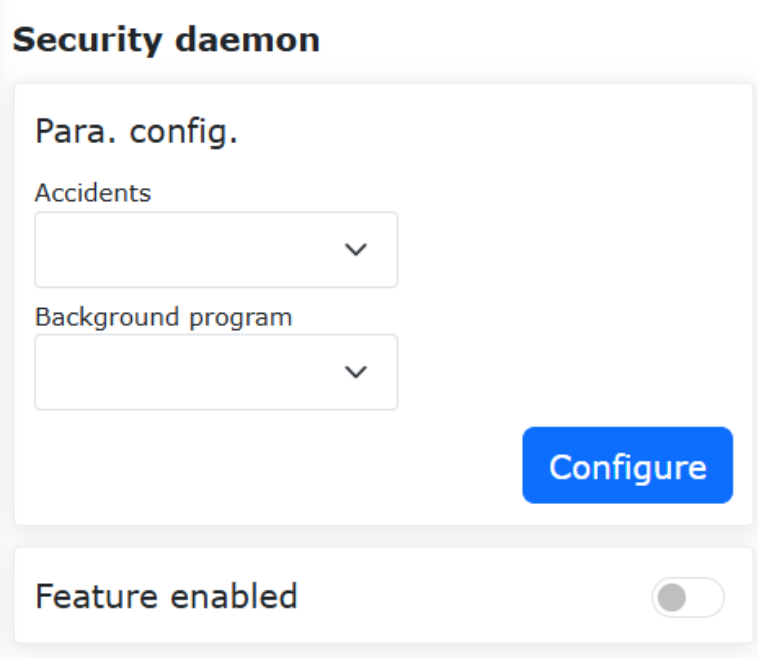


Figure 7.9-1 Daemon

1.3.7.10 Direction limit (Only used in Linux systems)

Click “Initial” -> “Safety” in the menu bar, and then click the “Direction limit” submenu to enter the configuration interface.

Tool direction limit is a protective function that acts on the Cartesian space of the robot tool end to limit the range of motion of the robot end posture, including function enablement settings, reference tool direction settings, and maximum offset angle settings. The maximum offset angle defines the maximum angle limit between the Z axis of the Cartesian coordinate system of the tool end and the reference tool direction, which can usually be understood as a conical space.

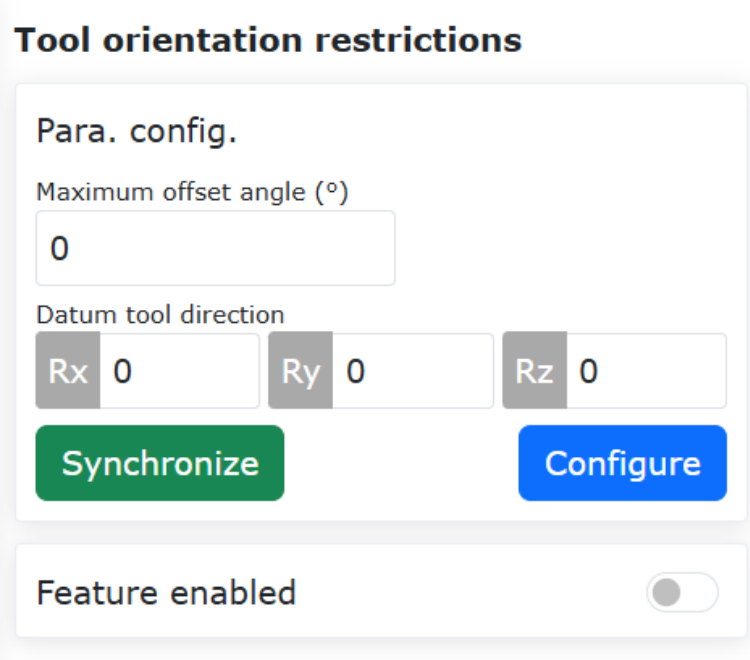


Figure 7.10-1 Direction limit

1.3.7.11 Robot limit (Only used in Linux systems)

Click “Initial” -> “Safety” in the menu bar, and then click the “Robot limit” submenu to enter the configuration interface.

Robot limits include momentum and power, where the momentum limit is used to limit the robot’s maximum momentum, and the power limit is used to limit the mechanical work done by the robot.

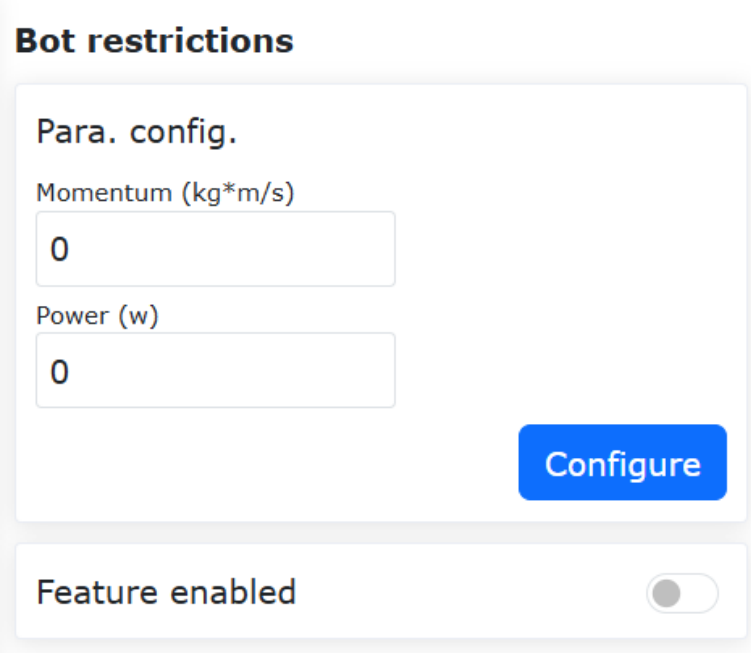


Figure 7.13-1 Robot limit

1.3.7.12 Power detection (Only used in QX systems)

Click “Initial” -> “Safety” in the menu bar, and then click the “Power detection” submenu to enter the configuration interface.

When acting directly on the current loop of the robot (only with the command servoJT), it is used to limit the work done by the robot. When it is detected that the integral of the robot speed and torque exceeds the limit, power protection is performed.

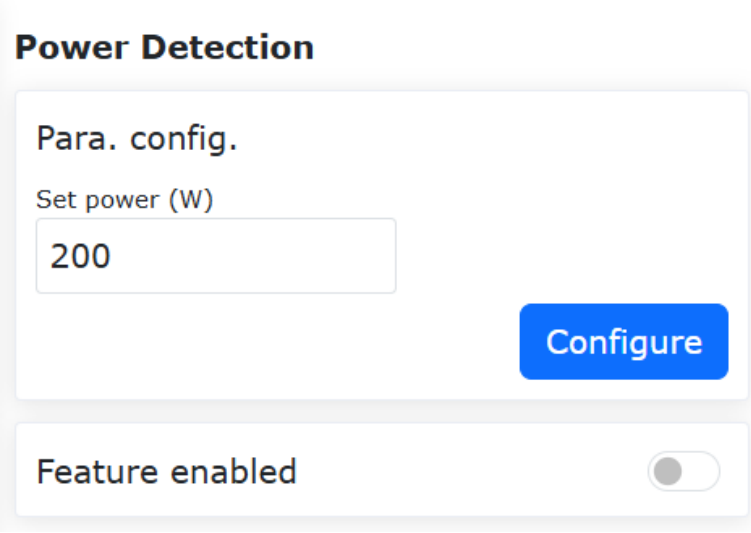


Figure 7.12-1 Power detection

1.3.7.13 Motion Configuration

1.3.7.13.1 T-Shaped Velocity Optimization + Blending Smoothing Function

1.3.7.13.1.1 Overview

Performing blending between two trajectory segments can avoid frequent start-stop issues caused by complete stops, thereby improving the robot's motion efficiency.

This function mainly applies to blending between PTP-PTP, LIN-LIN, ARC-ARC, LIN-ARC, and ARC-LIN commands. Blending between other commands is not effective.

1.3.7.13.1.2 Operation Process

Since the operation methods for each command are similar, this manual uses PTP-PTP blending as an example to explain the operation method. This function can be implemented in two ways: using Lua commands or using the motion configuration switch.

1.3.7.13.1.3 Using Lua Commands

Step 1: Select the teaching points for the PTP function. This manual uses "A0" to "A5" as the names of the teaching points.

Step 2: Click "Teaching Program" -> "Program Programming," select the "Point-to-Point" command under "Motion Commands," choose the teaching point in the "Command Edit" section, set the debugging speed, select "Acceleration Smoothing Mode" for motion protection, and set the "Smooth Transition" parameter at the points where smoothing is required.

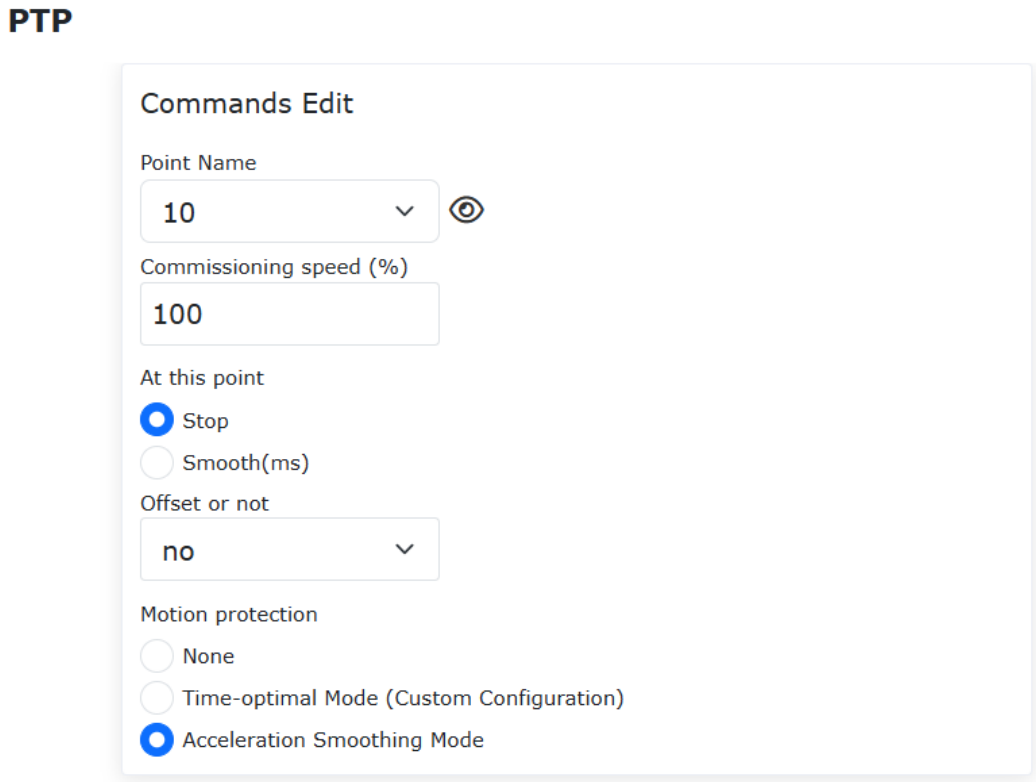


Figure 7.13-1 Blending Command Settings for Acceleration Smoothing PTP

Step 3: Generate and run the Lua program to implement PTP-PTP blending. This method only applies the optimized T-shaped velocity to commands between *AccSmoothStart()* and *AccSmoothEnd()*, while using the original T-shaped velocity for other commands.

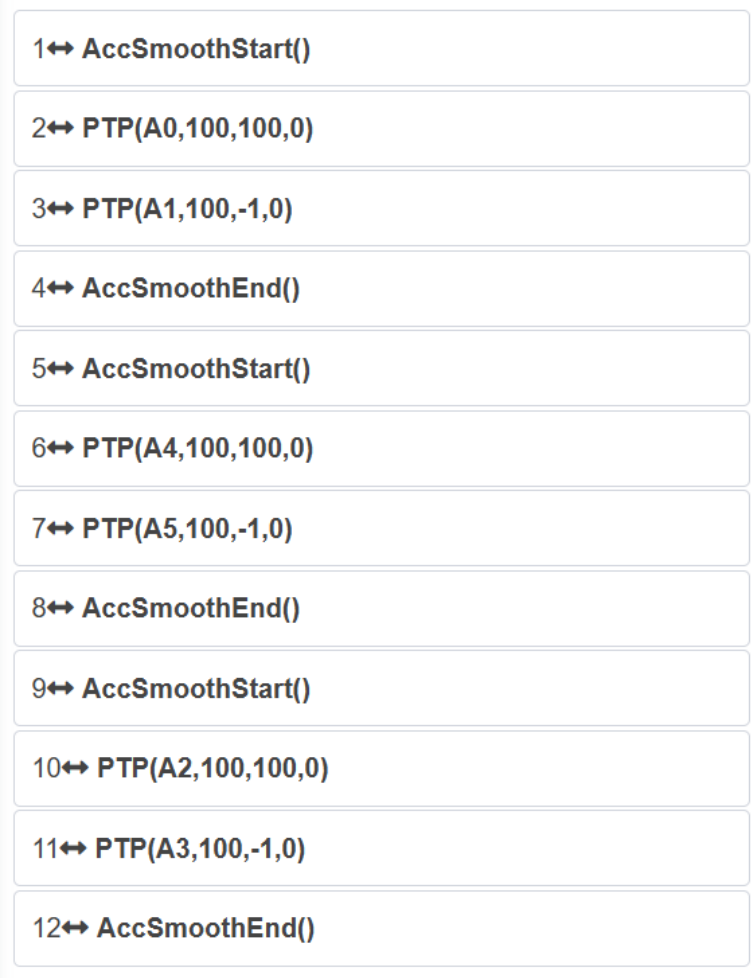


Figure 7.13-2 Typical Program for PTP-PTP Blending Using Lua Commands

1.3.7.13.1.4 Using Motion Configuration Switch

Step 1: Click “Initial” -> “Safety” -> “Motion Configuration,” and turn on the “Acceleration Smoothing Mode” switch.

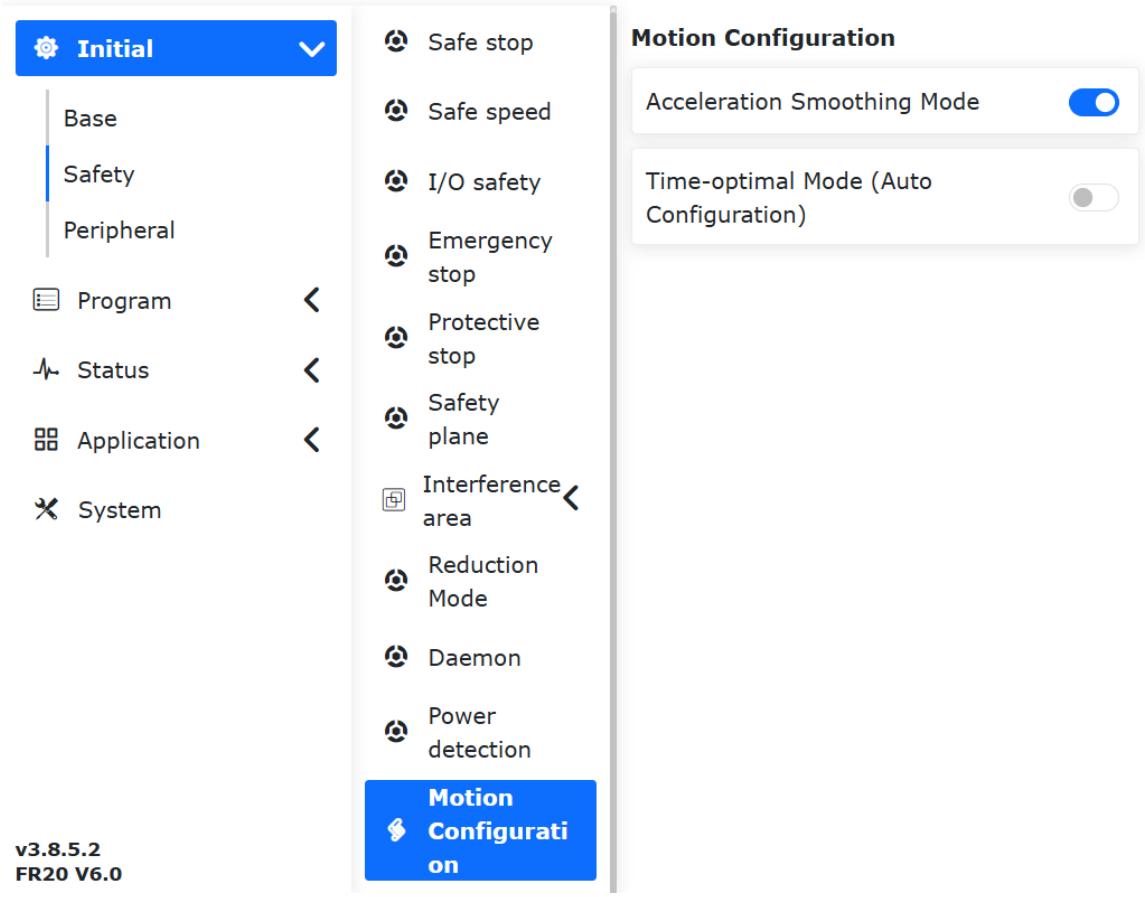


Figure 7.13-3 Acceleration Smoothing Mode Configuration Switch Settings

Step 2: Select the teaching points for the PTP-PTP function. This manual uses “A0” to “A5” as the names of the teaching points.

Step 3: Click “Teaching Program” -> “Coding,” select the “Point-to-Point” command under “Motion Commands,” choose the teaching point in the “Command Edit” section, set the debugging speed, select “None” for motion protection, and set the “Smooth Transition” parameter at the points where smoothing is required.

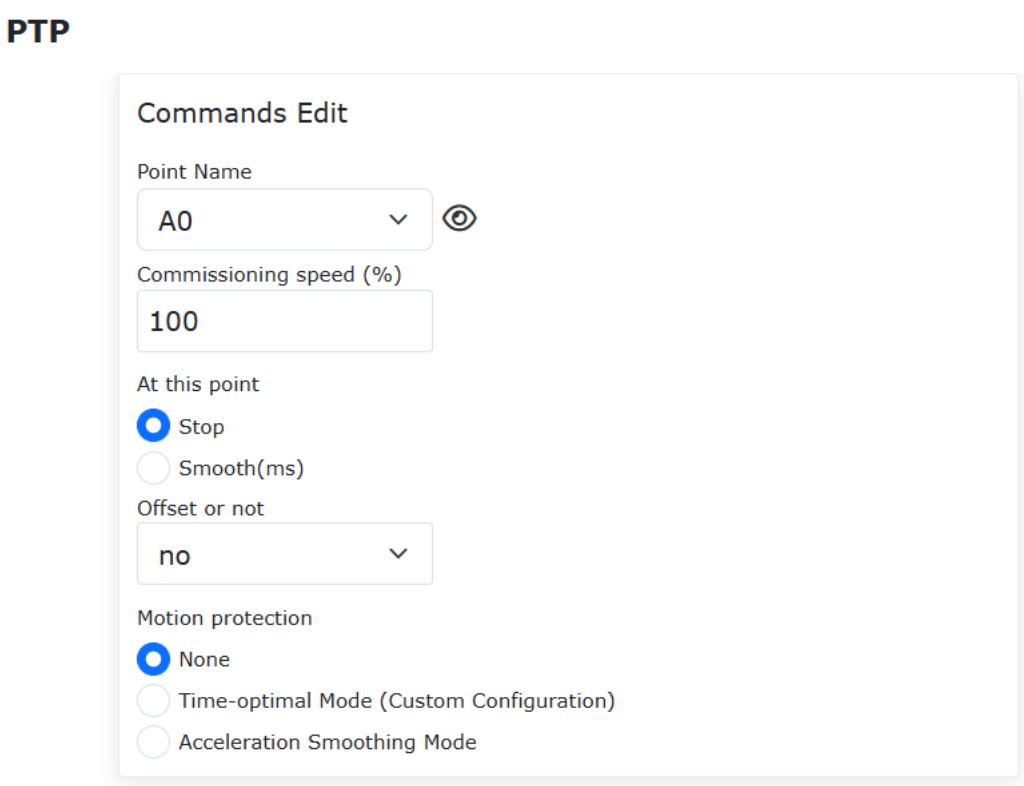


Figure 7.13-4 Blending Command Settings for Regular PTP

Step 4: Generate and run the Lua program to implement PTP-PTP blending. The typical program is the same as a regular PTP program. This method applies the optimized T-shaped velocity to all commands.

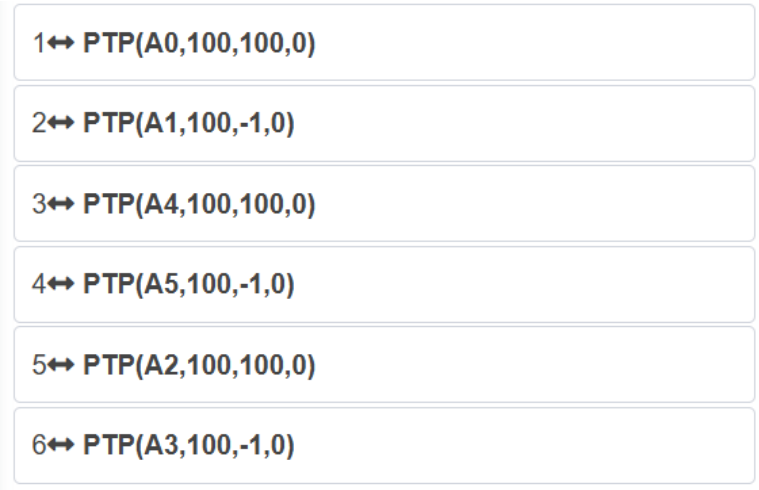


Figure 7.13-5 Typical Program for PTP-PTP Blending Using Configuration Switch

1.3.7.13.2 FIR Adaptive Parameter Function + FIR Pause/Resume Function

1.3.7.13.2.1 Overview

The robot's time-optimal mode parameter adaptive configuration function eliminates the need to manually debug and configure parameters. This function adaptively configures the parameters of the time-optimal mode based on the robot's current operating state, improving debugging efficiency.

1.3.7.13.2.2 Operation Process

The usage of basic robot motion commands (PTP, LIN, and ARC) is similar. This example uses the time-optimal mode PTP motion command as the primary example.

Step 1: On the robot's web control interface, navigate to "Initial" -> "Safety" -> "Motion Configuration" to enter the "Motion Configuration" interface.

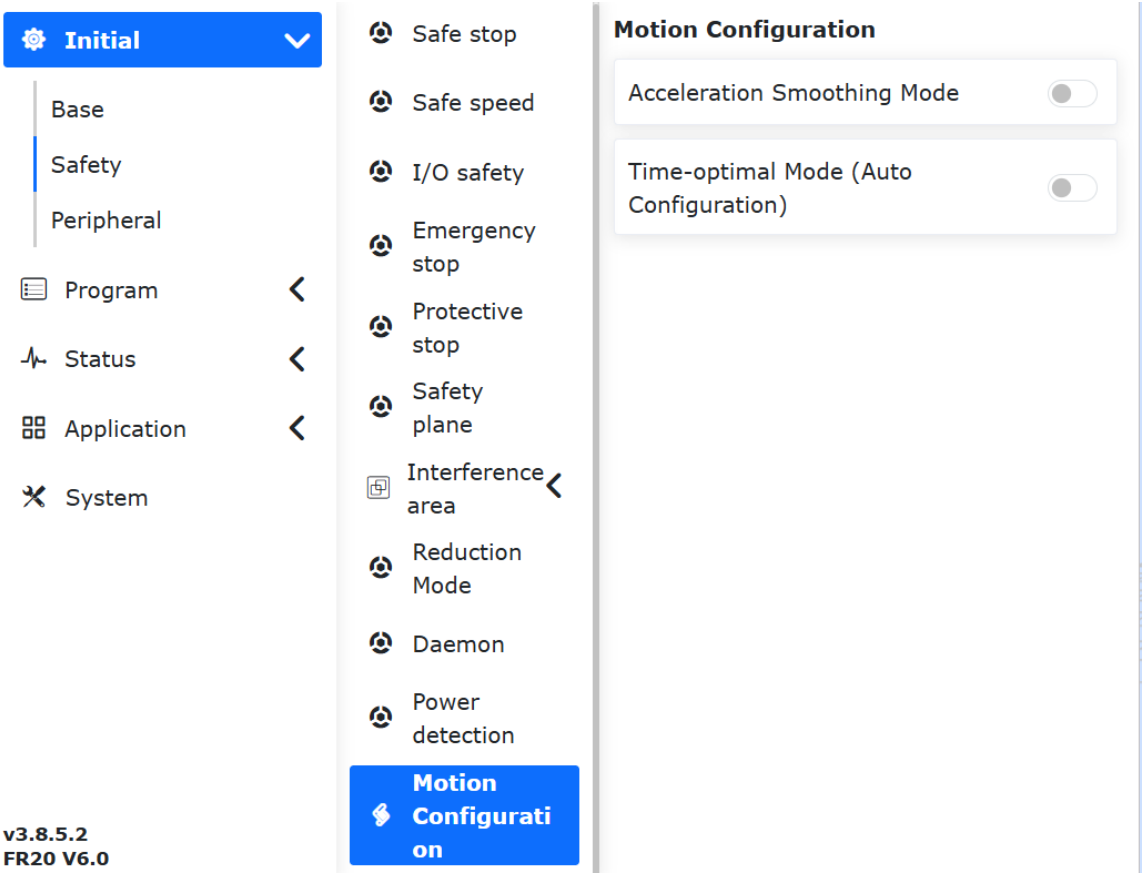


Figure 7.13-6 Motion Configuration Interface

Step 2: In the "Motion Configuration" interface, click the "Time-Optimal Mode" switch to enter the "Time-Optimal Mode" interface.

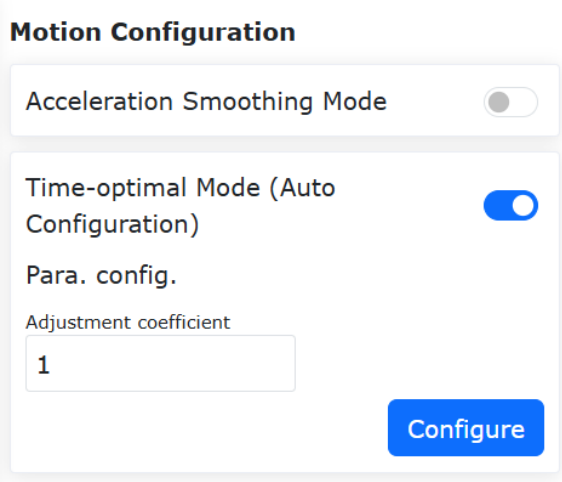


Figure 7.13-7 Time-Optimal Mode Interface

Note: In the "Parameter Configuration" section of the "Time-Optimal Mode" interface, the "Adjustment Coefficient" can be set from -100 to 100, representing a scaling ratio to control the time-optimal degree of motion commands. The default value is 1.

Step 3: Determine the teaching points for the PTP motion. This example uses "A0" to "A5" as the names of the teaching points.

Step 4: On the robot's web control interface, navigate to "Teaching Program" -> "Program Programming" to enter the "Motion Commands" interface.

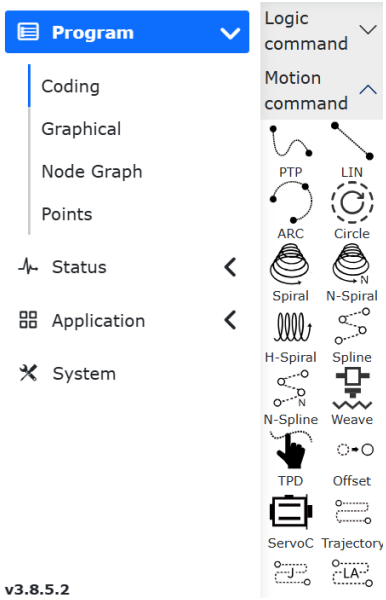


Figure 7.13-8 Motion Commands Interface

Step 5: In the "Motion Commands" interface, click "Point-to-Point" to enter the "PTP" command editing interface. Select the teaching point from the "Point Name" dropdown, set the desired speed ratio in the "Debugging Speed" section, choose "Stop" in the "At This Point" section, select "No" in the "Offset" dropdown, and choose "None" in the "Motion Protection" section. Then, click "Add."

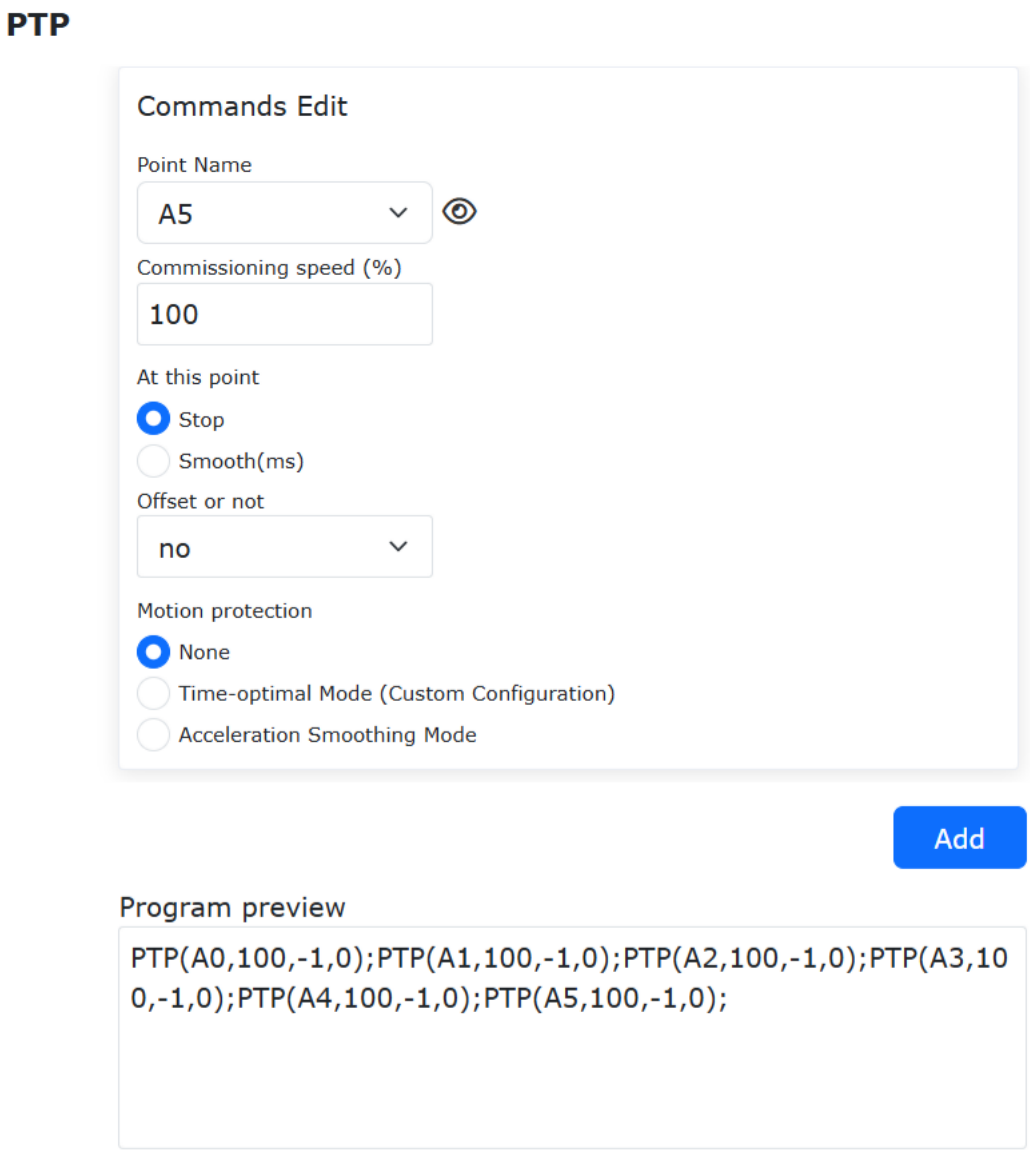


Figure 7.13-9 PTP Motion Command Editing Interface

Step 6: In the "PTP" motion command editing interface, click "Apply" to automatically generate the corresponding Lua program.

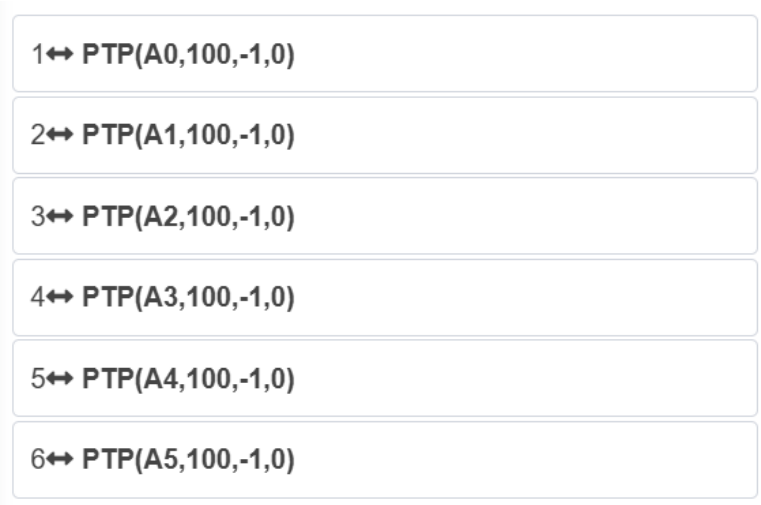


Figure 7.13-10 Typical Time-Optimal Mode PTP Motion Lua Program

Note: The typical time-optimal mode PTP motion Lua program is the same as a regular PTP motion Lua program, except that the "Time-Optimal Mode" function is enabled in Step 2.

When the "Time-Optimal Mode" function switch is enabled, all basic robot motion commands (PTP, LIN, and ARC) operate in time-optimal mode. Disabling the switch restores the commands to their basic state. The "Acceleration Smoothing Mode" function switch cannot be enabled simultaneously in this interface.

1.3.8 Peripherals

1.3.8.1 End-Effector Lua Custom Open Protocol

1.3.8.1.1 Overview

A hardware interface is provided at the robot end for connecting peripherals via 485 communication. Currently supported peripherals include grippers, rotary grippers, force sensors, welding handles, and other devices. All these end-effector devices can be adapted by writing a Lua open protocol to achieve protocol adaptation, enabling control of the peripheral and obtaining its status.For the SmartTool welding handle, users can also choose to log in to the web page to configure key functions and automatically generate an open protocol file. The generated protocol will be automatically applied to the end.

1.3.8.1.2 Operation Steps

Step1: Navigate to System Settings -> About -> Firmware Upgrade interface, select the end-effector firmware .bin file, and upgrade the end-effector firmware.

Important: First, confirm whether the end-effector firmware version FV2.010.06 or later meets the requirements. If the version does not meet the requirements, upgrade the corresponding software firmware; otherwise, firmware upgrade is not necessary.

Before uploading the end-effector firmware upgrade package, it is necessary to disable the robot and then enter boot mode.

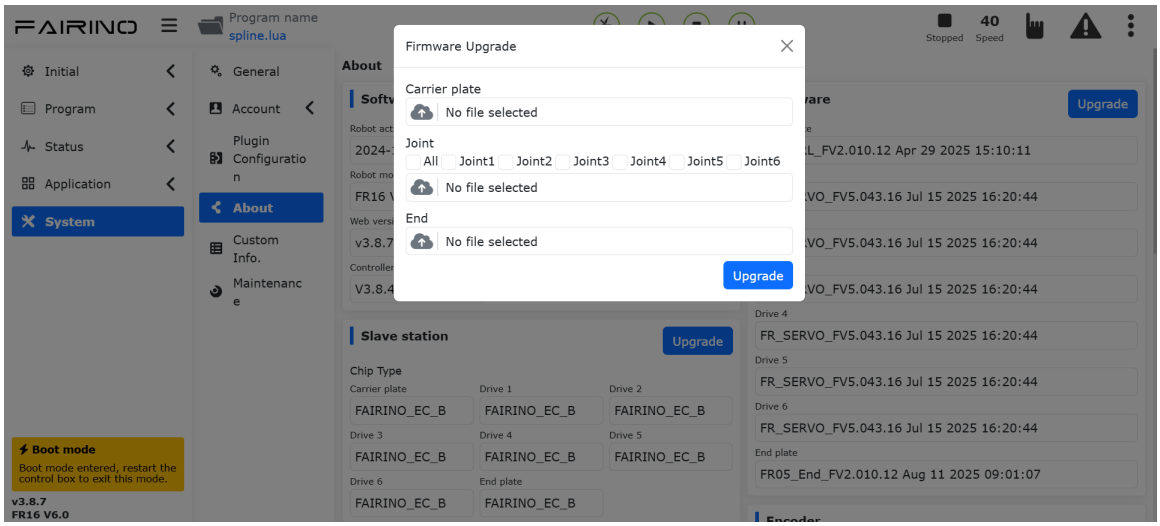


Figure 8.1-1 Upgrade End-Effector Firmware

Step2: Open the WebApp, click “Initial Setup”, “Peripherals” in sequence, and select the end-effector peripheral to be configured (e.g., gripper). The control type for peripherals includes two options: pre-adapted devices and peripheral open protocol:

- **Pre-adapted Devices**: Use the robot controller for communication. No upload or application is required.
- **Peripheral Open Protocol**: Users write a Lua-based open protocol for the end-effector to be adapted to achieve communication control. The end-effector protocols are divided into two categories: one is protocols uploaded by the user, and the other is built-in protocols preset in the robot.Starting from version 3.9.2, users do not need to perform verification and encryption operations on the Lua protocol to be uploaded to the end using additional software; they can upload it directly. Previously verified and encrypted protocols can still be uploaded and used normally. The robot will actively distinguish whether the file has been verified and encrypted. If it has not been verified, the robot will verify and encrypt it before uploading and applying it to the end. If it is already encrypted, it will be uploaded and applied to the end directly.

Figure 8.1-2 Gripper Control Type

Step3: Enter the content interface for Peripherals -> Gripper/Force Sensor/Welding Handle. Click on the “Custom Protocol” card to enter the interface. Upload the Lua end-effector open protocol, select the Lua end-effector open protocol to be uploaded, and perform the upload operation.

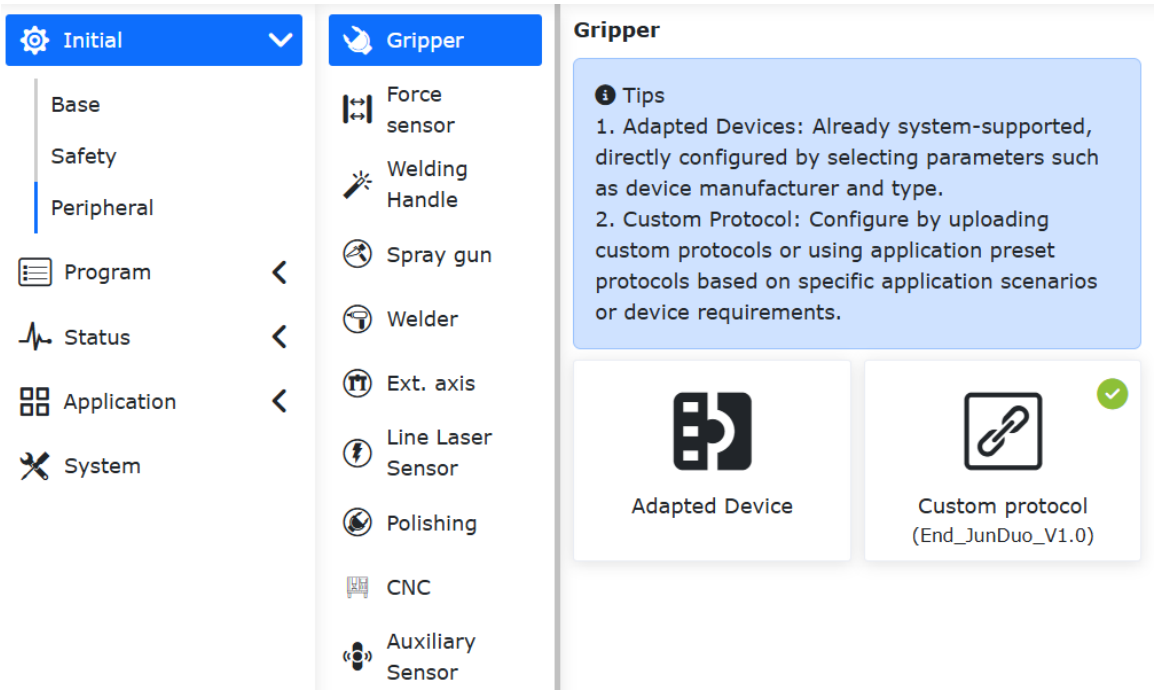
Important: The uploaded file name must start with *AXLE_LUA_*.

Step4: Configure the end-effector communication parameters, including baud rate, data bits, stop bits, etc. After configuration, click the “Configure” button.

Figure 8.1-3 Configure End-Effector Communication Parameters

The detailed end-effector communication parameters are as follows:

- **Baud Rate**: Supports 1-9600, 2-14400, 3-19200, 4-38400, 5-56000, 6-67600, 7-115200, 8-128000; The end-effector Rs485 driver chip is a low-speed 485, baud rate cannot be >200k;
- **Data Bits**: Supports (8,9), commonly 8 is used;
- **Stop Bits**: 1-1, 2-0.5, 3-2, 4-1.5, commonly 1 is used;
- **Parity Bit**: 0-None, 1-Odd, 2-Even, commonly 0 is used;



- **Timeout**: 1~1000ms, this value needs to be set reasonably in combination with the peripheral device;
- **Timeout Retries**: 1~10, mainly for timeout retransmission, reducing sporadic exceptions and improving user experience;
- **Periodic Command Interval**: 1~1000ms, mainly used for the time interval between each issuance of periodic commands;

Step5: Enable the End-Effector Lua by clicking the "Enable" button.

Figure 8.1-4 Enable End-Effector Lua

When an exception occurs in the Lua file, a "End-Effector Lua File Exception" warning is prompted, and "Ignore/Recover" processing can be performed. Turn off the Lua enable button to close the warning prompt.

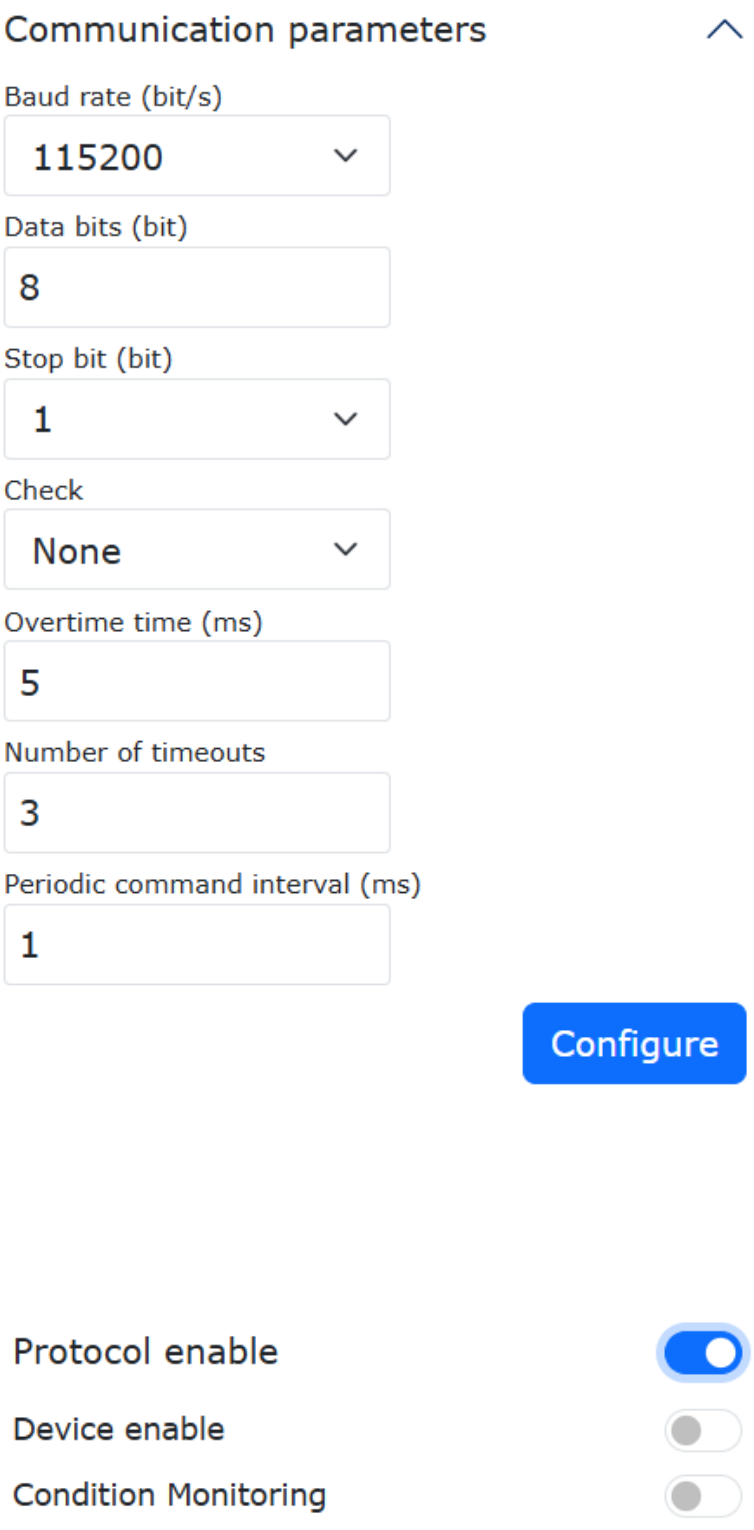
Figure 8.1-5 Lua File Exception

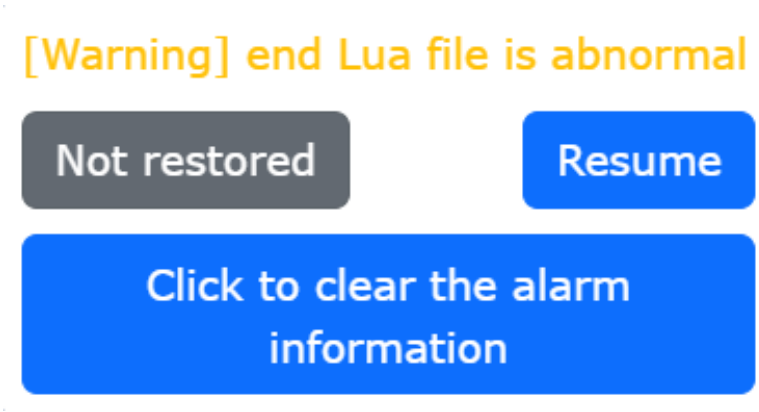
When the device type is a gripper, status monitoring can be performed.

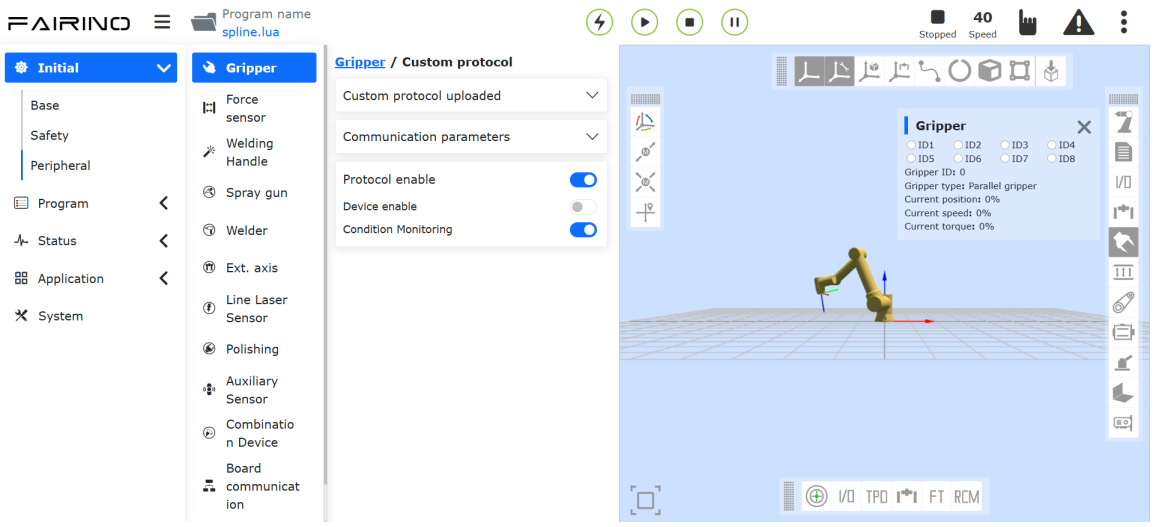
Turn on "Status Monitoring": The gripper status bar on the right displays real-time status information such as gripper running speed, torque, position, etc.

Turn off "Status Monitoring": The gripper data status bar on the right closes.

Figure 8.1-6 Status Monitoring


Communication parameters
Baud rate (bit/s)
115200
Data bits (bit)
8
Stop bit (bit)
1
Check
None
Overtime time (ms)
5
Number of timeouts
3
Periodic command interval (ms)
1
Configure
Protocol enable
Device enable
Condition Monitoring


[Warning] end Lua file is abnormal
Not restored
Resume
Click to clear the alarm information


FAIRINO
Program name
spline.lua
Stopped
40
Speed
Initial
Base
Safety
Peripheral
Program
Status
Application
System
Gripper
Force sensor
Welding Handle
Spray gun
Welder
Ext. axis
Line Laser Sensor
Polishing
Auxiliary Sensor
Combinatio n Device
Board communicat ion
Gripper / Custom protocol
Custom protocol uploaded
Communication parameters
Protocol enable
Device enable
Condition Monitoring
Gripper
ID1 ID2 ID3 ID4
ID5 ID6 ID7 ID8
Gripper ID: 0
Gripper type: Parallel gripper
Current position: 0%
Current speed: 0%
Current torque: 0%
I/O TPD FT RCM

1.3.8.2 Gripper

In the “Initial” -> “Peripherals” -> “Gripper” interface, grippers can currently be used via Pre-Adapted Devices and the End-Effector Lua Custom Open Protocol.

1.3.8.2.1 Pre-Adapted Devices

Step1: Click “Pre-Adapted Devices” to enter the end-effector peripheral configuration interface. The gripper configuration information includes gripper manufacturer, gripper type, software version, and mounting position. Users can configure the corresponding gripper information according to specific production needs. If the user needs to change the configuration, they can first select the corresponding gripper number, click the “Clear” button to clear the corresponding configuration, and reconfigure according to the requirements;

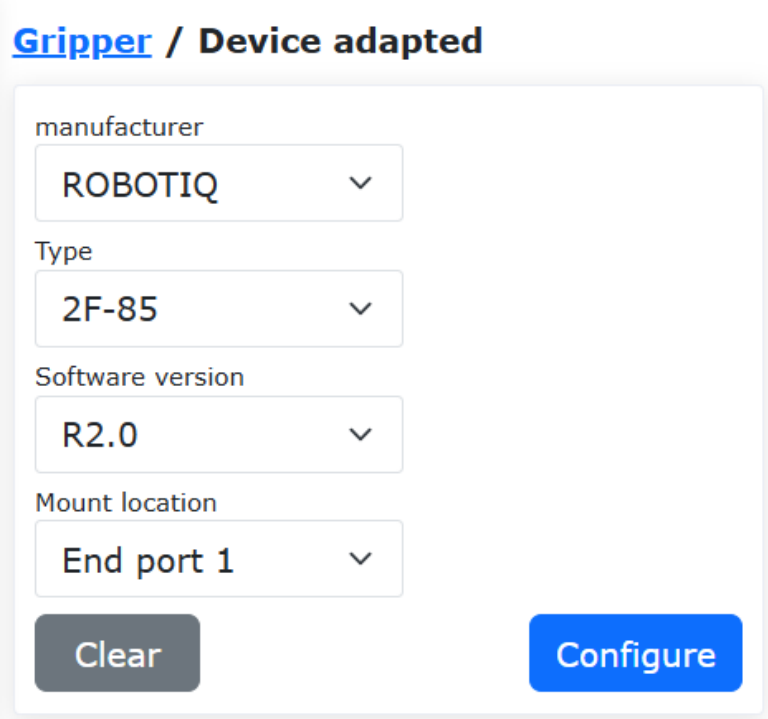


Figure 8.2-1 Gripper Configuration

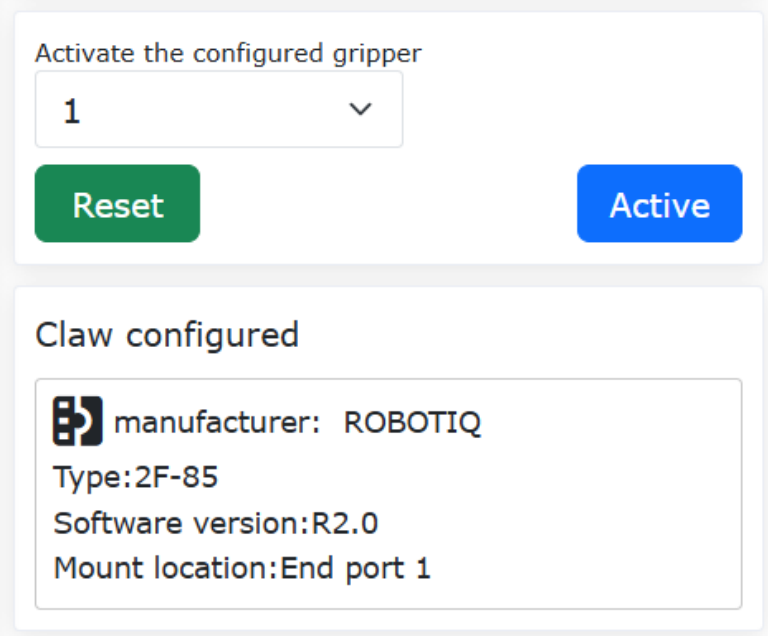
Important: Before clicking to clear the configuration, the corresponding gripper should be in an inactive state.

Step2: After the gripper configuration is completed, the user can view the corresponding gripper information in the gripper information table at the bottom of the page. If a configuration error is found, click the “Clear” button to reconfigure the gripper;

Figure 8.2-2 Gripper Configuration Information

Step3: Select the configured gripper, click the “Reset” button, after the page pops up indicating the command was sent successfully, then click the “Activate” button. Check the activation status in the gripper information table to determine whether the activation was successful;

Important: When activating the gripper, the gripper must not be holding any object.

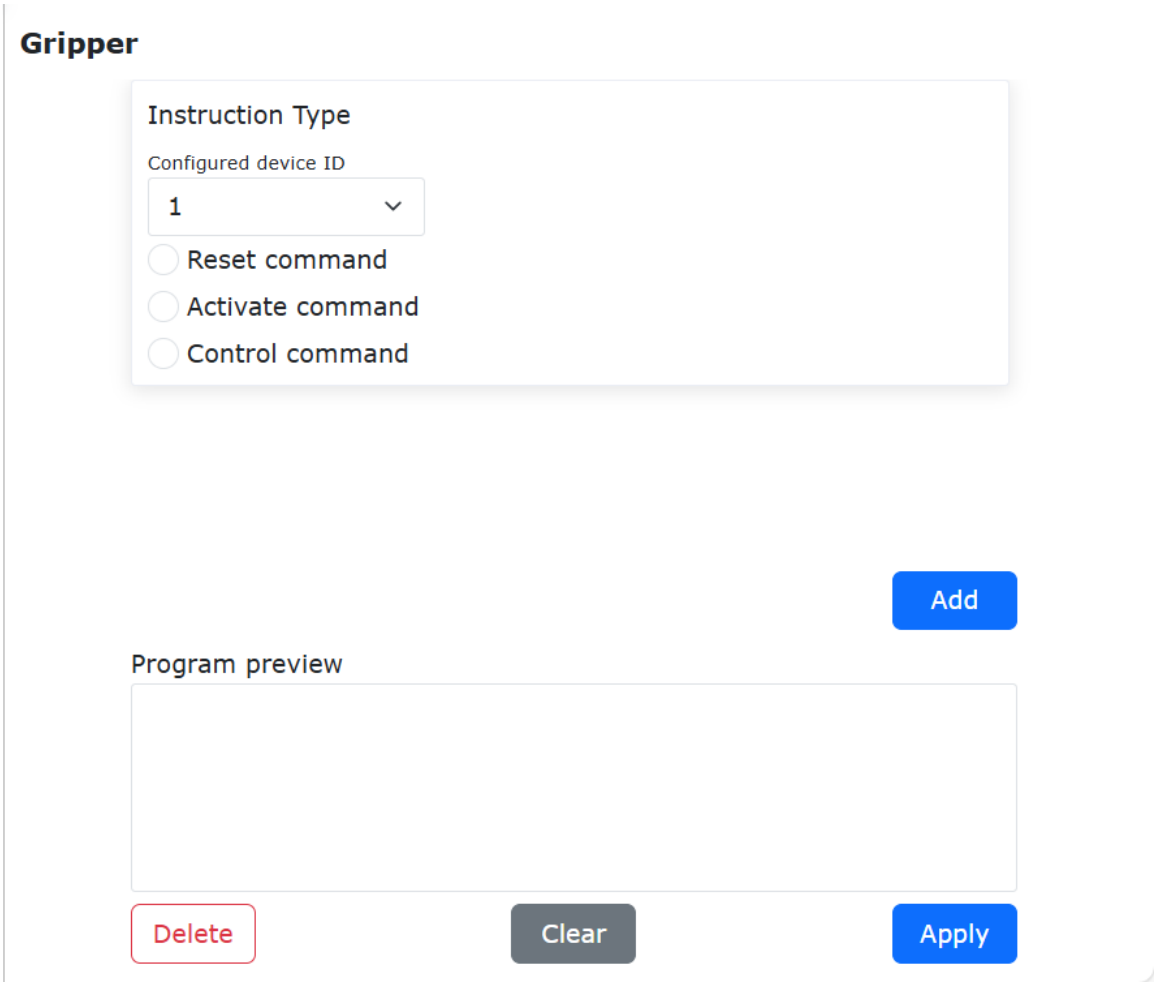


Step4: In the program teaching command interface, select the "Gripper" command. In the gripper command interface, the user can select the gripper number they want to control (grippers that have been configured and activated), set the corresponding open/close state, open/close speed, open/close torque, and the maximum waiting time for the gripper action. After completing the settings, click Add to apply. Additionally, gripper activation and reset commands can be added to activate/reset the gripper when running the program.

Figure 8.2-3 Gripper Command Editing

1.3.8.2.1.1 Gripper Program Teaching

No.	Command Format	Comment
1	PTP(template2,100,-1,0)	# Wait for gripping point
2	PTP(template1,100,-1,0)	# Gripping point
3	MoveGripper(1,255,255,0,1000,0)	# Gripper close
4	PTP(template2,100,-1,0)	/
5	PTP(template3,100,-1,0)	# Wait for placement point
6	PTP(template3,100,-1,0)	# Placement point
7	MoveGripper(1,0,255,0,1000,0)	# Gripper open


Gripper
Instruction Type
Configured device ID
1
Reset command
Activate command
Control command
Add
Program preview
Delete
Clear
Apply

1.3.8.2.2 Gripper Lua End-Effector Protocol Configuration

Open the WebApp, click “Initial Setup”, “Peripherals”, “Gripper”, “Custom Protocol” in sequence. Click “Protocol Management” to configure the end-effector protocol.

The filename uploaded by the user must start with “AXLE_LUA_End”. After uploading, the protocol name in the list will change to start with “Custom_End”. This type of protocol can be downloaded and deleted. Files with duplicate names uploaded by the user will automatically be overwritten with the latest Lua.

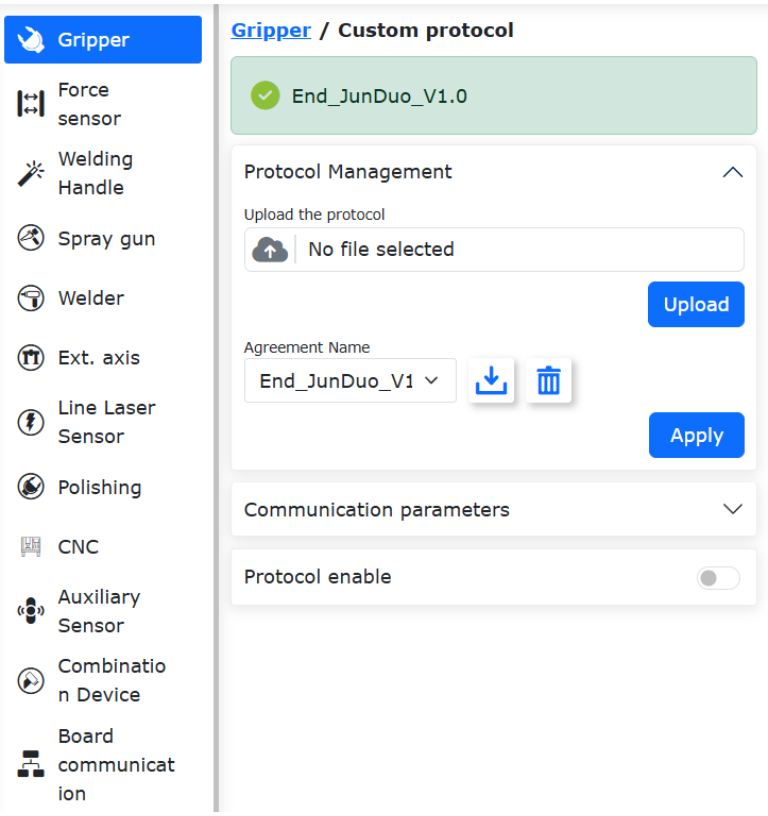


Figure 8.2-4-1 Gripper Custom Protocol Upload

The robot’s preset built-in protocols start with *End_* as a prefix. They can only be downloaded, not deleted. The built-in protocols for peripherals (gripper, rotary gripper, suction cup) are shown in the figure below.

Figure 8.2-4-2 Gripper (Rotary Gripper, Suction Cup) Preset Built-in Protocol

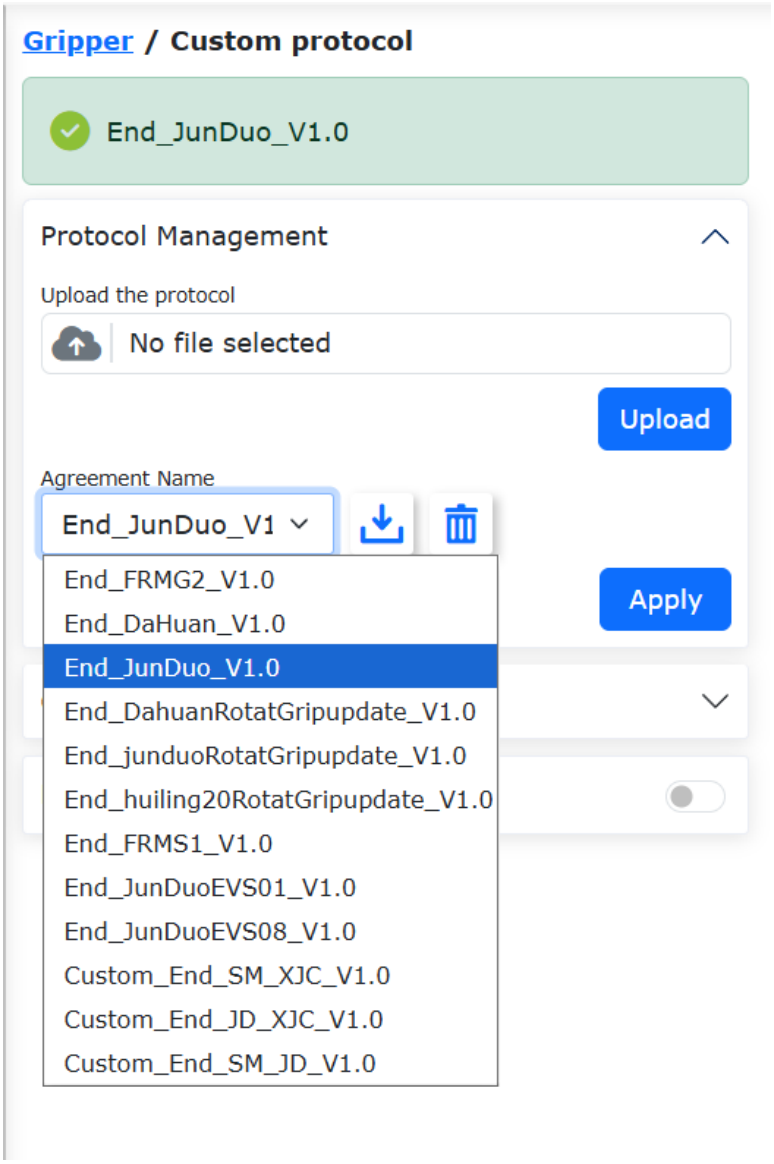
After ensuring the correct protocol is selected, you can disable the robot and apply the open protocol. After application, the robot will automatically enter boot mode and apply the selected protocol to the end-effector. When the page prompts “Upgrade successful, please restart the control box”, you can power cycle the control box.

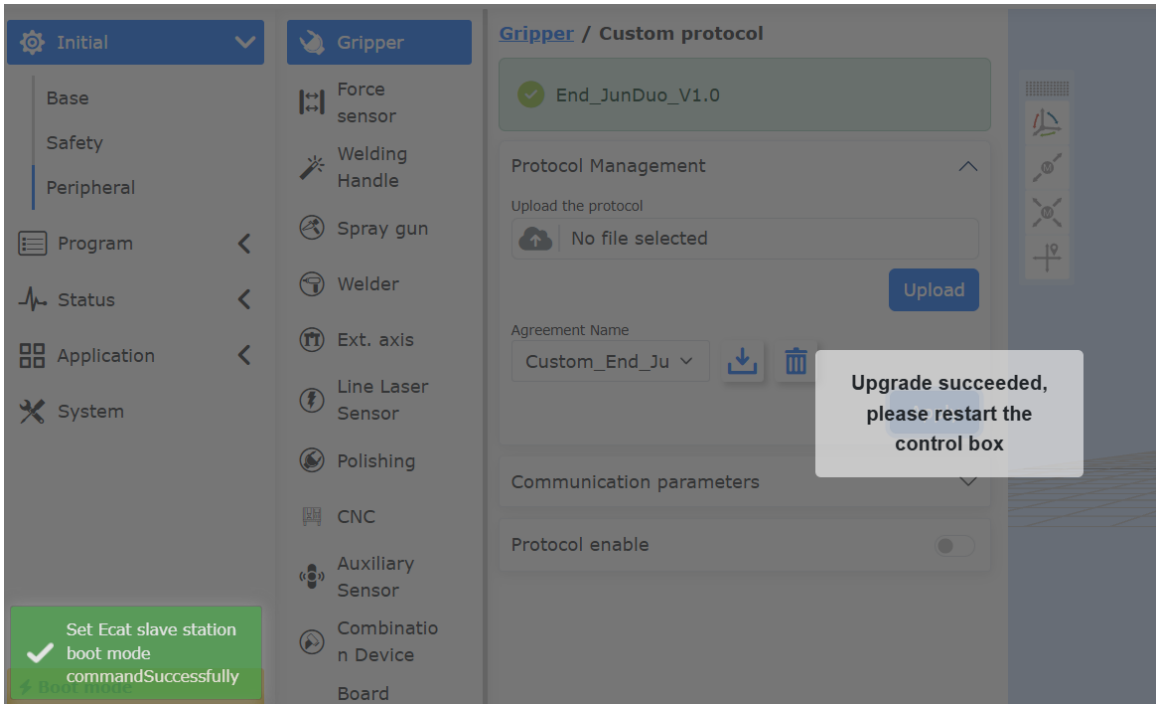
Figure 8.2-4-3 Applying the End-Effector Open Protocol to the End-Effector Board

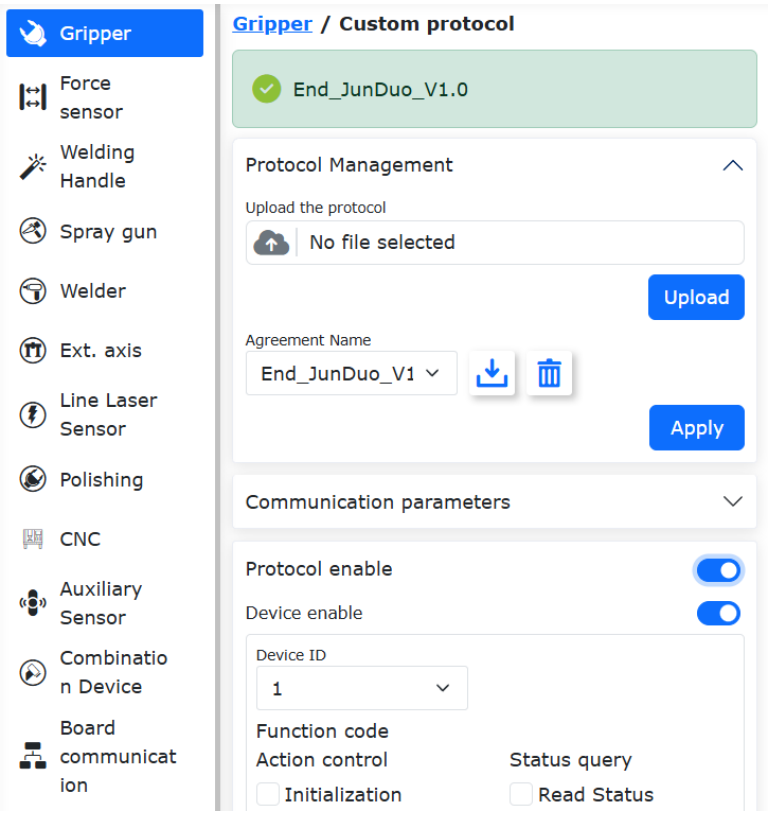
After restarting and entering the WebApp page, the page will display the name of the currently applied protocol. After clicking to enable the end-effector protocol and enabling the device, the end-effector protocol will start running. The Device ID is the Modbus slave address of the end-effector peripheral and needs to be used in conjunction with the content in the protocol.

Figure 8.2-4-4 Gripper End-Effector Protocol Configuration Display and Enablement

The end-effector board will verify the uploaded Lua protocol. When there is an issue with the Lua file, it will show a “End-Effector Lua File Abnormal” warning. You can choose “Do Not Recover/Recover”. Turn off the Lua enable button to close the warning prompt.


Gripper / Custom protocol
End_JunDuo_V1.0
Protocol Management
Upload the protocol
No file selected
Upload
Agreement Name
End_JunDuo_V1
Apply
End_FRMG2_V1.0
End_DaHuan_V1.0
End_JunDuo_V1.0
End_DahuanRotatGripupdate_V1.0
End_junduoRotatGripupdate_V1.0
End_huiling20RotatGripupdate_V1.0
End_FRMS1_V1.0
End_JunDuoEVS01_V1.0
End_JunDuoEVS08_V1.0
Custom_End_SM_XJC_V1.0
Custom_End_JD_XJC_V1.0
Custom_End_SM_JD_V1.0


Initial
Base
Safety
Peripheral
Program
Status
Application
System
Gripper
Force sensor
Welding Handle
Spray gun
Welder
Ext. axis
Line Laser Sensor
Polishing
CNC
Auxiliary Sensor
Combinatio n Device
Board
Gripper / Custom protocol
End_JunDuo_V1.0
Protocol Management
Upload the protocol
No file selected
Upload
Agreement Name
Custom_End_Ju
Upgrade succeeded, please restart the control box
Communication parameters
Protocol enable
Set Ecat slave station boot mode commandSuccessfully


Gripper
Force sensor
Welding Handle
Spray gun
Welder
Ext. axis
Line Laser Sensor
Polishing
CNC
Auxiliary Sensor
Combinatio n Device
Board communicat ion
Gripper / Custom protocol
End_JunDuo_V1.0
Protocol Management
Upload the protocol
No file selected
Upload
Agreement Name
End_JunDuo_V1
Apply
Communication parameters
Protocol enable
Device enable
Device ID
1
Function code
Action control
Initialization
Status query
Read Status

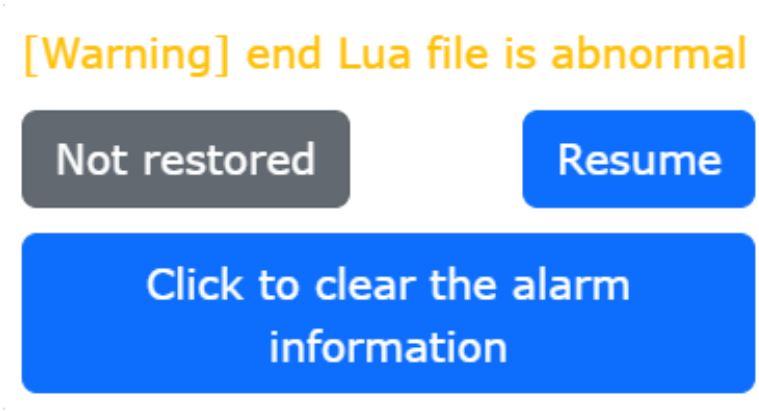


Figure 8.2-4-5 Gripper End-Effector Protocol Configuration Display and Enablement

1.3.8.2.2.1 Example of a Gripper Peripheral's Lua End-Effector Peripheral Protocol

```
function Getbit(X,Bit)--Getbit(), extracts the corresponding bit from a byte.␣
↪Parameters: X: the byte from which to extract the bit; Bit: the bit position to␣
↪extract (range 0-7)
return ((X&(1<<Bit))>>Bit)
end

function GetOneByte(U32)--GetOneByte(), extracts the data 0x1234, gets its low byte,␣
↪returns 0x34
return ((U32>>0)&0xFF)
end

function GetTwoByte(U32)--GetTwoByte(), extracts the data 0x1234, gets its high byte,␣
↪returns 0x12
return ((U32>>8)&0xFF)
end
function GetThreeByte(U32)--GetThreeByte(), extracts the data 0x56781234, extracts and␣
↪returns 0x78
return ((U32>>16)&0xFF)
end
function GetFourByte(U32)--GetFourByte(), extracts the data 0x56781234, extracts and␣
↪returns 0x56
return ((U32>>24)&0xFF)
end
X,Speed,Torque=0,0,0
while(1)
do
IwdgTaskHandle()
MainLoop()
UpDownLoadHandle()
SdoRwPara()
EndErrClear()
local BFlag=LuaBreak()
if(BFlag==1)then
break
```

(continues on next page)

(continued from previous page)

```
end--From here to the end of the file LuaGc(), end is fixed syntax

T1={0x01,0x06,0x03,0xE8,0x00,0x09,0xC9,0xBC}--Populate gripper commands (Modbus RTU␣
↪commands). T1-T5 are respectively: gripper action execution command, gripper␣
↪initialization command, gripper position command, gripper speed command, gripper␣
↪torque command
--/Command parsing: T1[1]=0X01, is the gripper address; T1[2]=0x06, write single holding␣
↪register function code; T1[3], T1[4]: 0x03,0xE8, address of the register to operate␣
↪for the action execution command; T1[5],T1[6]: 0x00,0x09, data to write to the␣
↪register; T1[7],T1[8]: 0xC9,0xBC, CRC checksum, needs to be modified according to the␣
↪gripper user manual
T2={}
T3={}
T4={}
T5={}

T7={0x01,0x03,0x07,0xD0,0x00,0x01,0x84,0x87}--T7-T12, gripper status reading commands,␣
↪respectively: read gripper status command, read gripper initialization command, read␣
↪gripper error code command, read gripper position command, read gripper speed command,␣
↪read gripper torque command
T8={}
T9={}
T10={}
T11={}
T12={}
Rcmd1,Rcmd2,Rcmd3,Rcmd4=GetGripCmd()--Fixed usage, no need for modification. Rcm2 is the␣
↪gripper address sent by the controller, Rcmd4 is the data sent by the controller
if(Rcmd1==1) then
T1[1]=Rcmd2
T2[1]=Rcmd2
T3[1]=Rcmd2
T4[1]=Rcmd2
T5[1]=Rcmd2

T7[1]=Rcmd2
T8[1]=Rcmd2
T9[1]=Rcmd2
T10[1]=Rcmd2
T11[1]=Rcmd2
T12[1]=Rcmd2                    --**Gripper address update
if (Rcmd3==1) then              --Gripper action execution command
T1[7],T1[8]=CrcValue(T1[1],T1[2],T1[3],T1[4],T1[5],T1[6])--Calculate Modbus RTU command␣
↪CRC value, two bytes
EndTxGripData(T1[1],T1[2],T1[3],T1[4],T1[5],T1[6],T1[7],T1[8])--End-effector sends␣
↪command to gripper
DelayMs(10)                                                 --Delay 10ms
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()--End-effector returns the received␣
↪gripper feedback data to Lua. Specific feedback content needs to be checked in the␣
↪gripper user manual
GripStateBack(Rxd3)
end
if (Rcmd3==2) then
```

(continues on next page)

(continued from previous page)

```
T2[7],T2[8]=CrcValue(T2[1],T2[2],T2[3],T2[4],T2[5],T2[6])
EndTxGripData(T2[1],T2[2],T2[3],T2[4],T2[5],T2[6],T2[7],T2[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
GripStateBack(Rxd3)
end
if(Rcmd3==3) then
X=Rcmd4
T3[5]=0x00
T3[6]=X
T3[7],T3[8]=CrcValue(T3[1],T3[2],T3[3],T3[4],T3[5],T3[6])
EndTxGripData(T3[1],T3[2],T3[3],T3[4],T3[5],T3[6],T3[7],T3[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
GripStateBack(Rxd3)
end
if (Rcmd3==4) then
Speed=Rcmd4
T4[5]=Torque
T4[6]=Speed
T4[7],T4[8]=CrcValue(T4[1],T4[2],T4[3],T4[4],T4[5],T4[6])
EndTxGripData(T4[1],T4[2],T4[3],T4[4],T4[5],T4[6],T4[7],T4[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
GripStateBack(Rxd3)
end
if(Rcmd3==5) then
Torque=Rcmd4
T5[5]=Torque
T5[6]=Speed
T5[7],T5[8]=CrcValue(T5[1],T5[2],T5[3],T5[4],T5[5],T5[6])
EndTxGripData(T5[1],T5[2],T5[3],T5[4],T5[5],T5[6],T5[7],T5[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
GripStateBack(Rxd3)
end
if(Rcmd3 == 7) then
T7[7],T7[8]=CrcValue(T7[1],T7[2],T7[3],T7[4],T7[5],T7[6])
EndTxGripData(T7[1],T7[2],T7[3],T7[4],T7[5],T7[6],T7[7],T7[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
RxdCrcH,RxdCrcL = CrcValue(Rxd1,Rxd2,Rxd3,Rxd4,Rxd5)
if((A==8)and(Rxd1==Rcmd2)and(Rxd2==0x03)and(Rxd3==0x02)and(Rxd6==RxdCrcH)and(Rxd7==RxdCrcL))then
GripStateBack(Rxd4)
end
end
if(Rcmd3==8) then
T8[7],T8[8]=CrcValue(T8[1],T8[2],T8[3],T8[4],T8[5],T8[6])
EndTxGripData(T8[1],T8[2],T8[3],T8[4],T8[5],T8[6],T8[7],T8[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
RxdCrcH,RxdCrcL = CrcValue(Rxd1,Rxd2,Rxd3,Rxd4,Rxd5)
```

(continues on next page)

(continued from previous page)

```
if((A==8)and(Rxd1==Rcmd2)and(Rxd2==0x03)and(Rxd3==0x02)and(Rxd6==RxdCrcH)and(Rxd7
↪==RxdCrcL)) then
GripStateBack(Rxd5)
end
end
if(Rcmd3 == 9) then
T9[7],T9[8]=CrcValue(T9[1],T9[2],T9[3],T9[4],T9[5],T9[6])
EndTxGripData(T9[1],T9[2],T9[3],T9[4],T9[5],T9[6],T9[7],T9[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
RxdCrcH,RxdCrcL = CrcValue(Rxd1,Rxd2,Rxd3,Rxd4,Rxd5)
if((A==8)and(Rxd1==Rcmd2)and(Rxd2==0x03)and(Rxd3==0x02)and(Rxd6==RxdCrcH)and(Rxd7==RxdCrcL))
↪then
GripStateBack(Rxd5)
end
end
if(Rcmd3 == 10) then
T10[7],T10[8]=CrcValue(T10[1],T10[2],T10[3],T10[4],T10[5],T10[6])
EndTxGripData(T10[1],T10[2],T10[3],T10[4],T10[5],T10[6],T10[7],T10[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
RxdCrcH,RxdCrcL = CrcValue(Rxd1,Rxd2,Rxd3,Rxd4,Rxd5)
if((A==8)and(Rxd1==Rcmd2)and(Rxd2==0x03)and(Rxd3==0x02)and(Rxd6==RxdCrcH)and(Rxd7==RxdCrcL))
↪then
GripStateBack(Rxd4)
end
end
if(Rcmd3 == 11) then
T11[7],T11[8]=CrcValue(T11[1],T11[2],T11[3],T11[4],T11[5],T11[6])
EndTxGripData(T11[1],T11[2],T11[3],T11[4],T11[5],T11[6],T11[7],T11[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
RxdCrcH,RxdCrcL = CrcValue(Rxd1,Rxd2,Rxd3,Rxd4,Rxd5)
if((A==8)and(Rxd1==Rcmd2)and(Rxd2==0x03)and(Rxd3==0x02)and(Rxd6==RxdCrcH)and(Rxd7==RxdCrcL))
↪then
GripStateBack(Rxd5)
end
end
if(Rcmd3 == 12) then
T12[7],T12[8]=CrcValue(T12[1],T12[2],T12[3],T12[4],T12[5],T12[6])
EndTxGripData(T12[1],T12[2],T12[3],T12[4],T12[5],T12[6],T12[7],T12[8])
DelayMs(10)
A,Rxd1,Rxd2,Rxd3,Rxd4,Rxd5,Rxd6,Rxd7=EndRxGripData()
RxdCrcH,RxdCrcL = CrcValue(Rxd1,Rxd2,Rxd3,Rxd4,Rxd5)
if((A==8)and(Rxd1==Rcmd2)and(Rxd2==0x03)and(Rxd3==0x02)and(Rxd6==RxdCrcH)and(Rxd7==RxdCrcL))
↪then
GripStateBack(Rxd4)
end
end
end
LuaGc()
end
```

1.3.8.2.2.2 Device Enable

Step1: Enable Gripper -> Select Gripper ID -> Check the function codes adapted for the gripper -> Click Configure. The configured device displays the Gripper ID and function codes.

Figure 8.2-4 Configure Gripper

Note: The current supported device address range for the end-effector open function for grippers is 1~8. Before use, adjust the gripper device address via the gripper manufacturer's upper computer software.

The selected function codes should be queried from the product manual provided by the gripper manufacturer to match the gripper's adapted functions, and should correspond to the end-effector Lua function codes. For details, please refer to the "End-Effector Lua Adaptation Gripper Instruction Manual".

Step2: Select Gripper ID -> Reset -> Activate. The gripper performs an initialization. For specific initialization details, please refer to the product manual provided by the gripper manufacturer.

Figure 8.2-5 Activate Gripper

Step3: Enter Teach Program -> Program Programming -> Add Gripper Motion Command.

Figure 8.2-6 Add Gripper Motion Command

Figure 8.2-7 Gripper Motion Command Example

1.3.8.2.2.3 Multiple Grippers

Activation and motion control refer to the gripper steps.

Figure 8.2-8 Configure Multiple Grippers

Note: The current supported device address range for the end-effector open function for grippers is 1~8. Before use, adjust the gripper device address via the gripper manufacturer's upper computer software.

1.3.8.2.2.4 Rotary Gripper

Step1: Enable Gripper -> Select Gripper ID -> Check the function codes adapted for the gripper -> Click Configure. The configured device displays the Gripper ID and function codes.

Figure 8.2-9 Configure Gripper and Function Codes

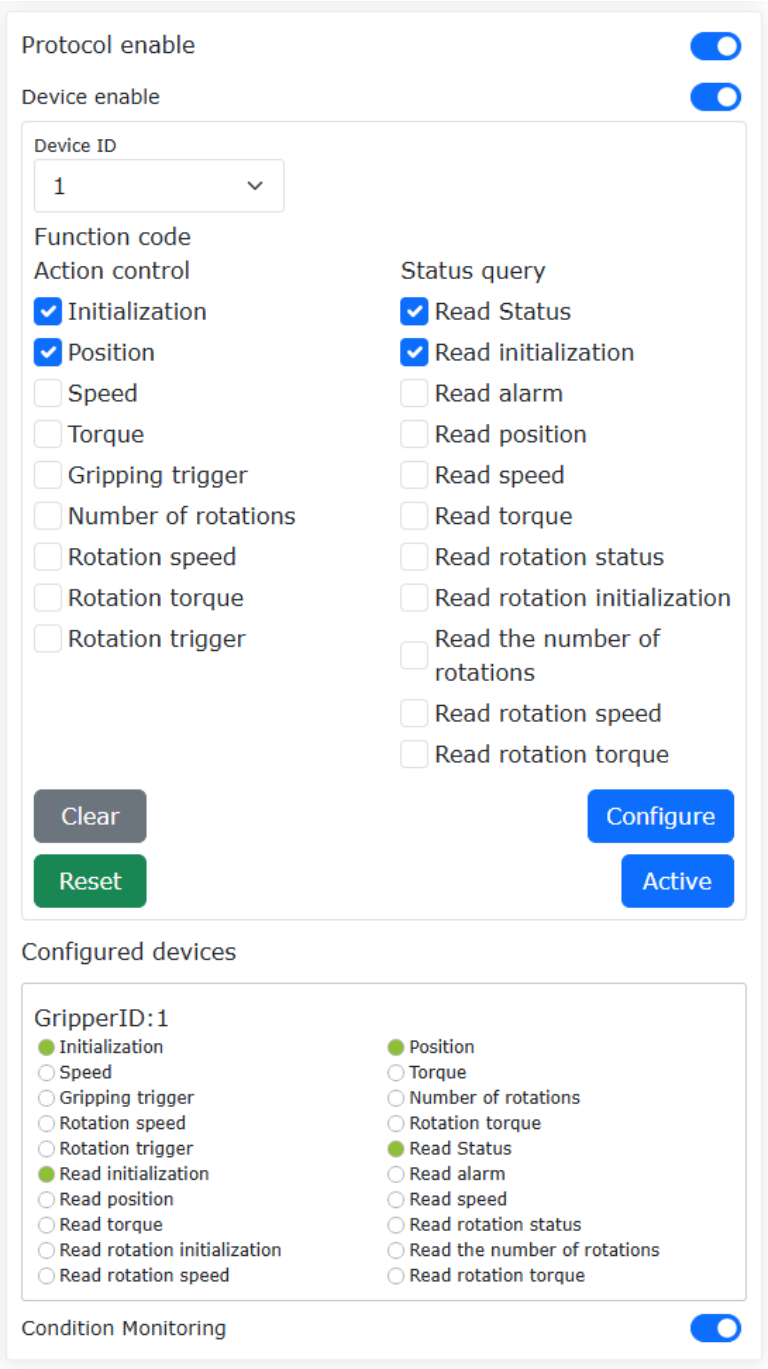
Note: The selected function codes should be queried from the product manual provided by the gripper manufacturer to match the gripper's adapted functions, and should correspond to the end-effector Lua function codes. For details, please refer to "FR05-End-Effector Full Peripheral Protocol-V2.5-20241101.xlsx".

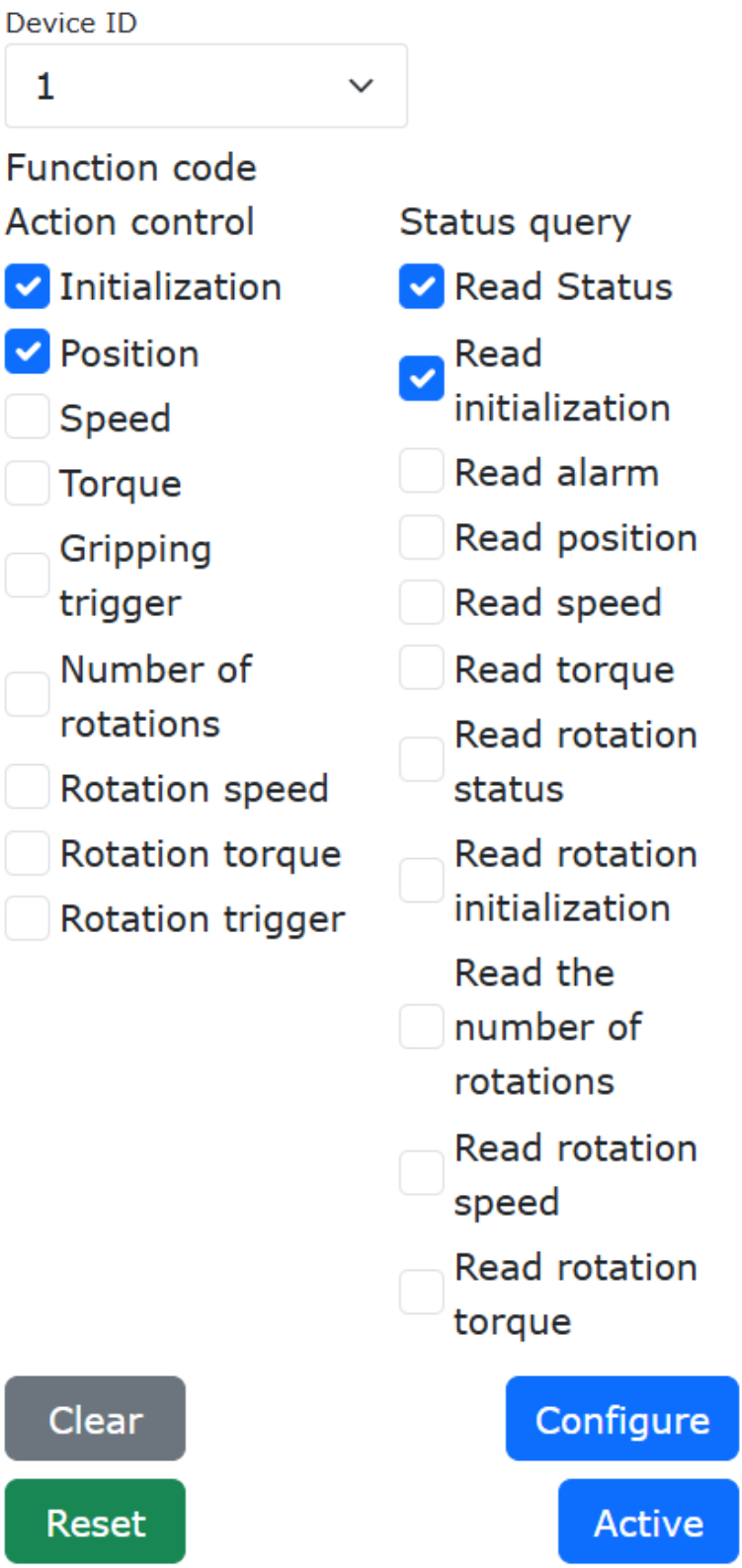
Step2: Select Gripper ID -> Reset -> Activate. The gripper performs an initialization. For specific initialization details, please refer to the product manual provided by the gripper manufacturer.

Figure 8.2-10 Activate Gripper

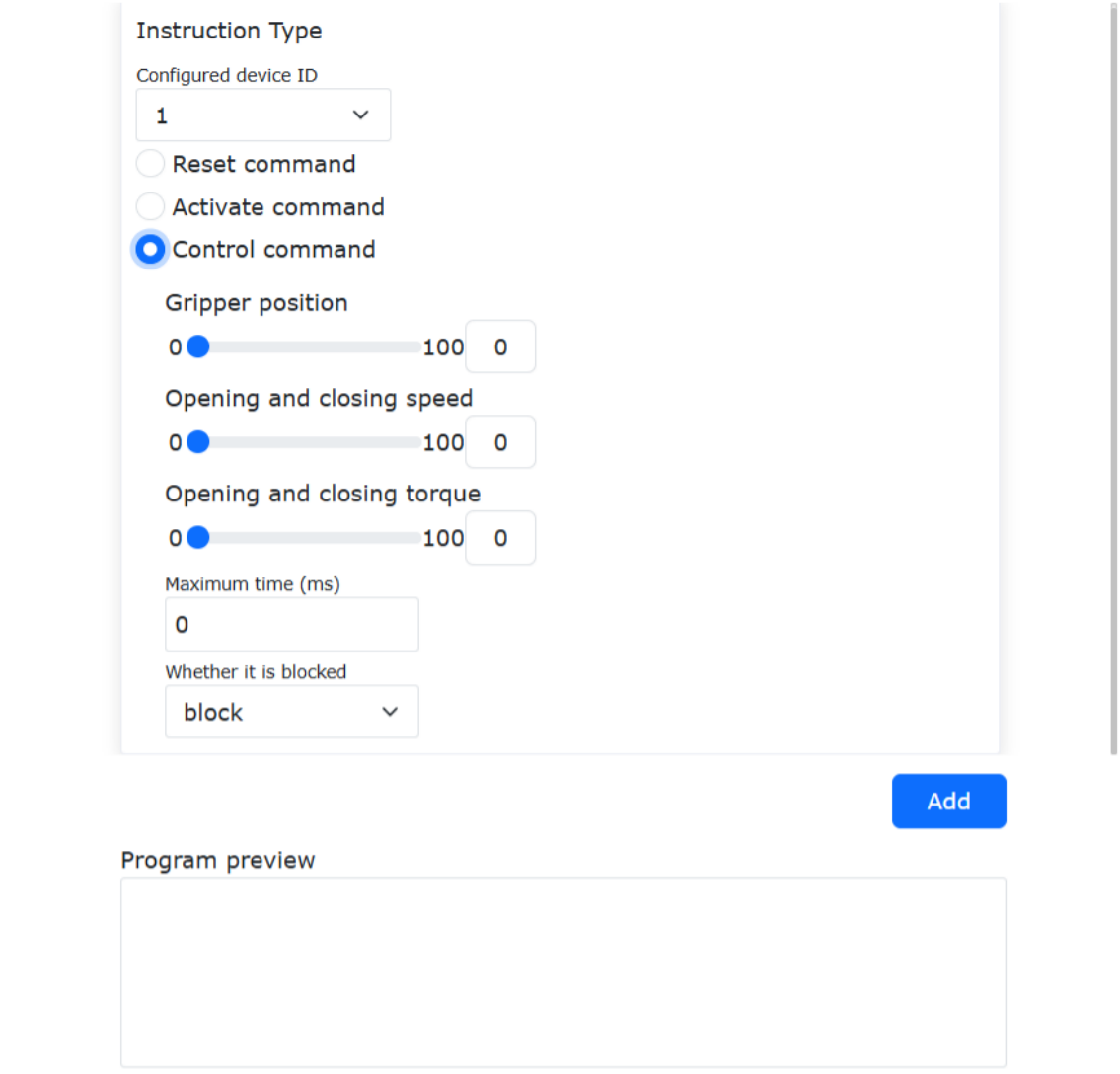
Step3: Enter Teach Program -> Program Programming -> Add Rotary Gripper Motion Command.

Figure 8.2-11 Add Rotary Gripper Motion Command

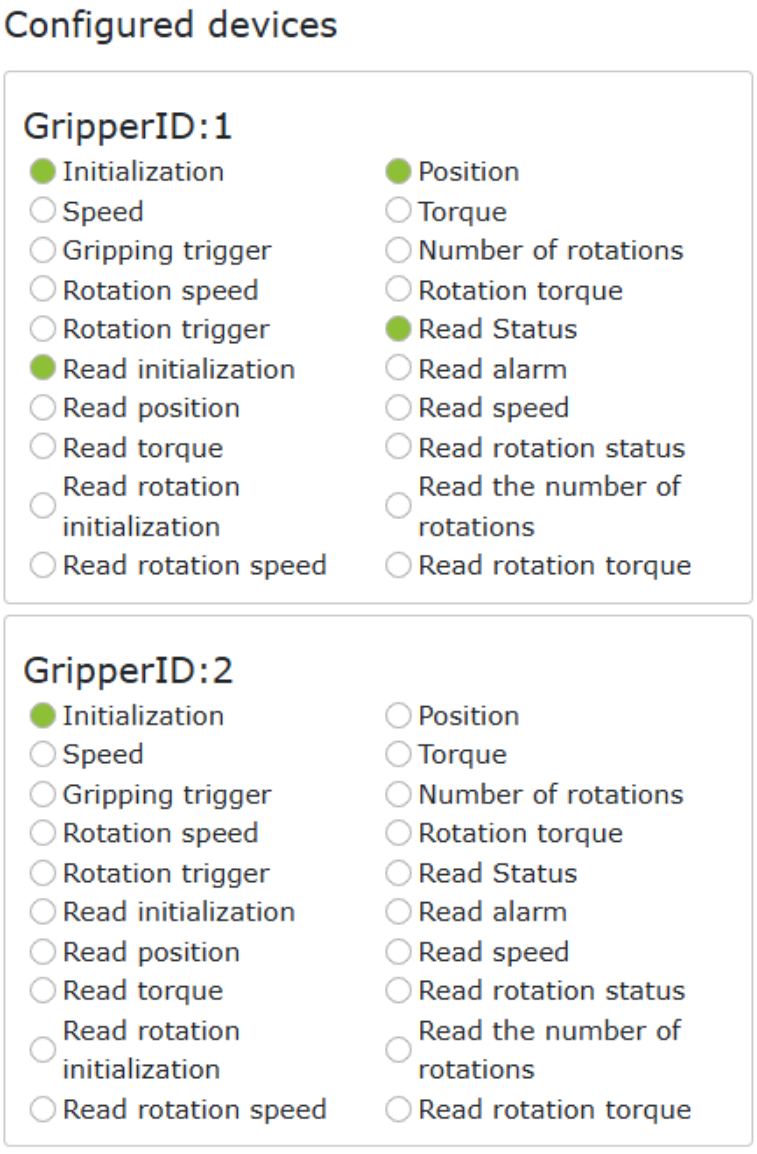

Protocol enable
Device enable
Device ID
1
Function code
Action control
Initialization
Position
Speed
Torque
Gripping trigger
Number of rotations
Rotation speed
Rotation torque
Rotation trigger
Status query
Read Status
Read initialization
Read alarm
Read position
Read speed
Read torque
Read rotation status
Read rotation initialization
Read the number of rotations
Read rotation speed
Read rotation torque
Clear
Configure
Reset
Active
Configured devices
GripperID:1
Initialization
Speed
Gripping trigger
Rotation speed
Rotation trigger
Read initialization
Read position
Read torque
Read rotation initialization
Read rotation speed
Position
Torque
Number of rotations
Rotation torque
Read Status
Read alarm
Read speed
Read rotation status
Read the number of rotations
Read rotation torque
Condition Monitoring

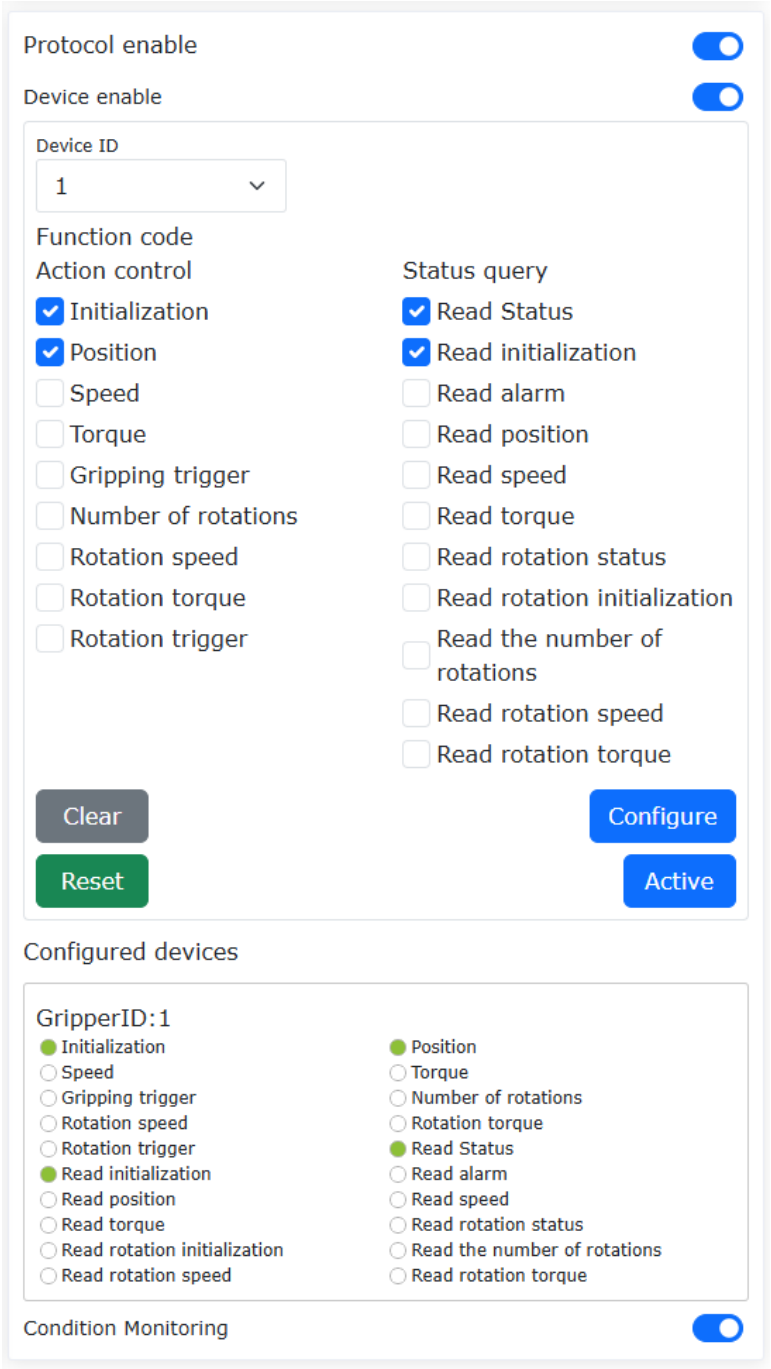

Device ID
1
Function code
Action control
Initialization
Position
Speed
Torque
Gripping trigger
Number of rotations
Rotation speed
Rotation torque
Rotation trigger
Status query
Read Status
Read initialization
Read alarm
Read position
Read speed
Read torque
Read rotation status
Read rotation initialization
Read the number of rotations
Read rotation speed
Read rotation torque
Clear
Configure
Reset
Active

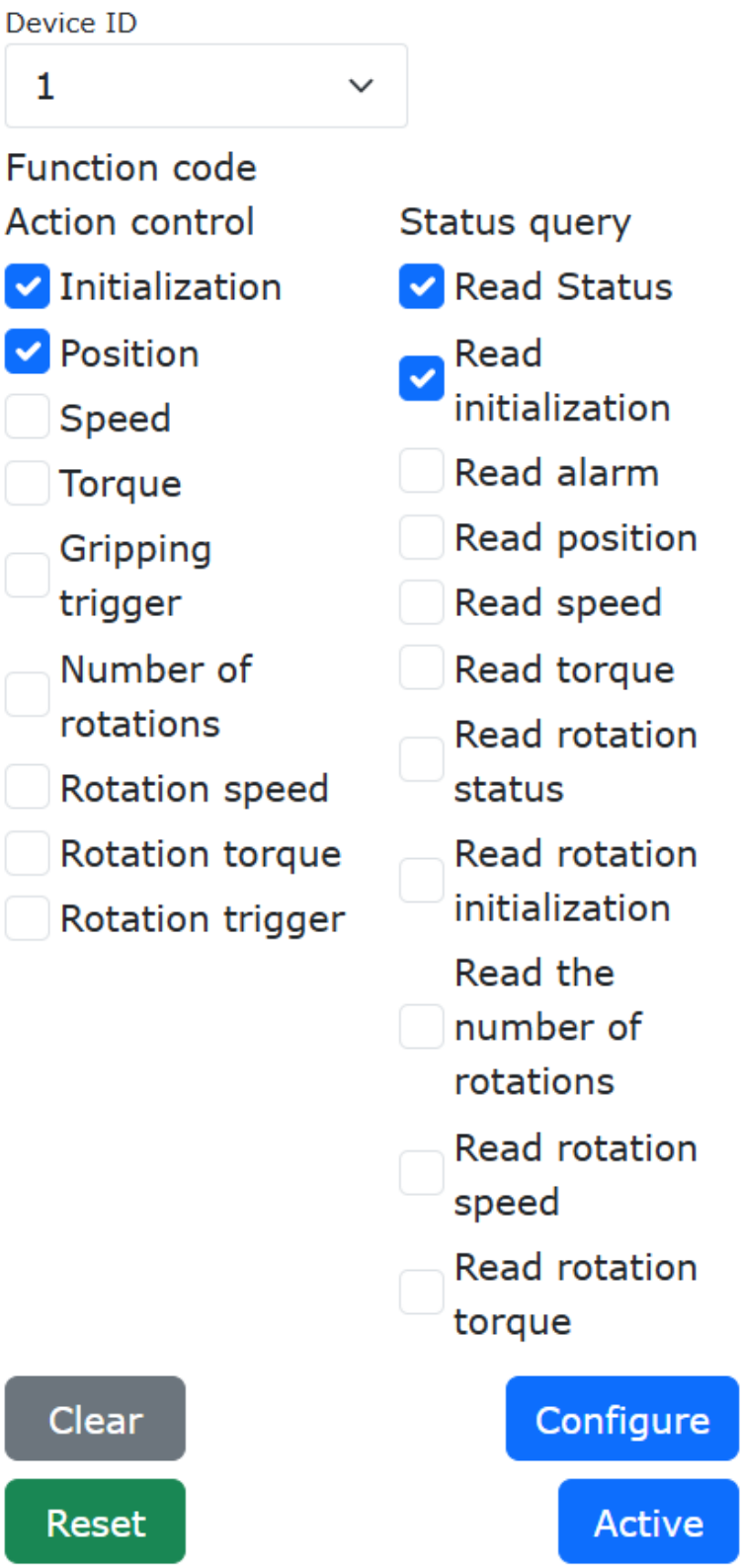
Gripper


Instruction Type
Configured device ID
1
Reset command
Activate command
Control command
Gripper position
0
100
0
Opening and closing speed
0
100
0
Opening and closing torque
0
100
0
Maximum time (ms)
0
Whether it is blocked
block
Add
Program preview

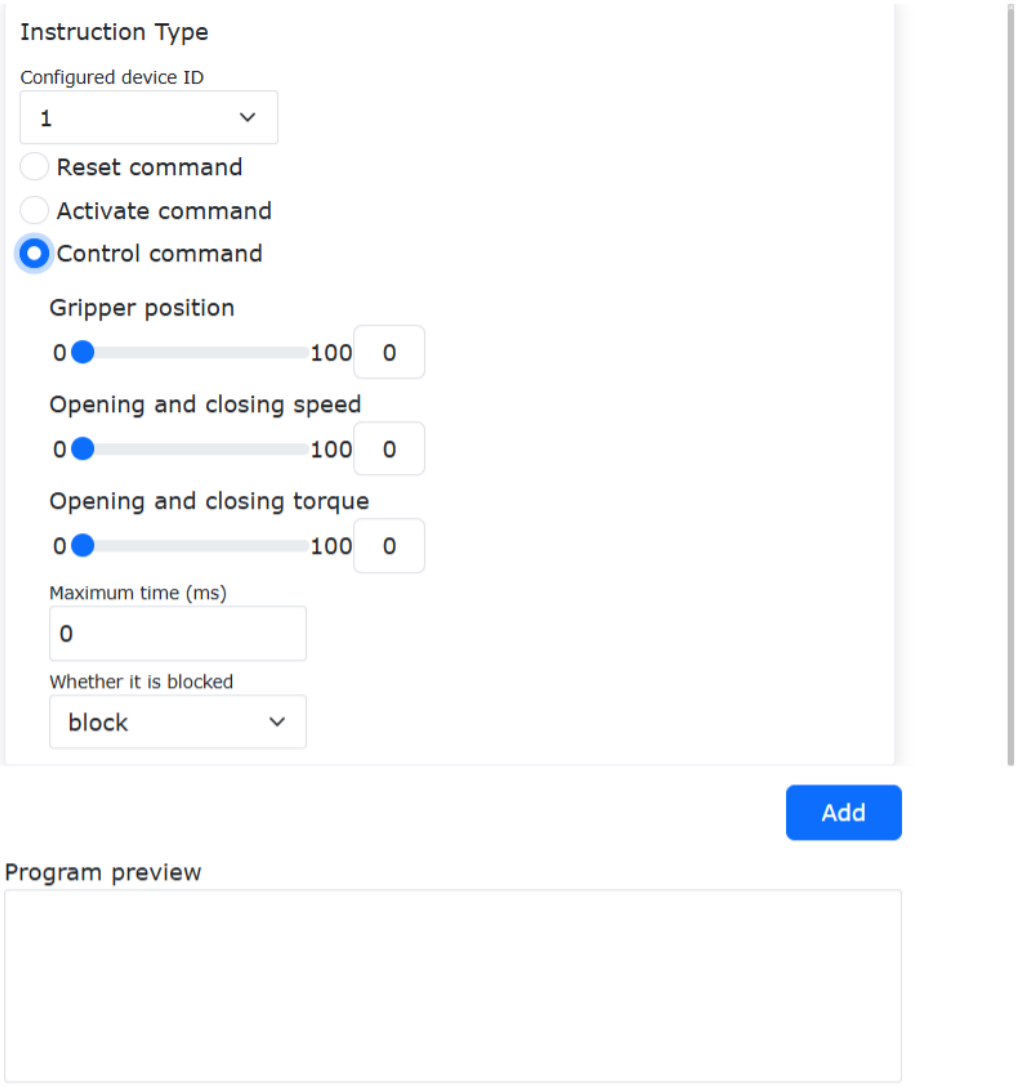

1↔ while (1) do
2↔ MoveGripper(1,40,87,42,2000,0,0,0,0,0)
3↔ MoveGripper(1,90,87,42,2000,0,0,0,0,0)
4↔ end


Configured devices
GripperID:1
Initialization
Speed
Gripping trigger
Rotation speed
Rotation trigger
Read initialization
Read position
Read torque
Read rotation initialization
Read rotation speed
Position
Torque
Number of rotations
Rotation torque
Read Status
Read alarm
Read speed
Read rotation status
Read the number of rotations
Read rotation torque
GripperID:2
Initialization
Speed
Gripping trigger
Rotation speed
Rotation trigger
Read initialization
Read position
Read torque
Read rotation initialization
Read rotation speed
Position
Torque
Number of rotations
Rotation torque
Read Status
Read alarm
Read speed
Read rotation status
Read the number of rotations
Read rotation torque


Protocol enable
Device enable
Device ID
1
Function code
Action control
Initialization
Position
Speed
Torque
Gripping trigger
Number of rotations
Rotation speed
Rotation torque
Rotation trigger
Status query
Read Status
Read initialization
Read alarm
Read position
Read speed
Read torque
Read rotation status
Read rotation initialization
Read the number of rotations
Read rotation speed
Read rotation torque
Clear
Configure
Reset
Active
Configured devices
GripperID:1
Initialization
Speed
Gripping trigger
Rotation speed
Rotation trigger
Read initialization
Read position
Read torque
Read rotation initialization
Read rotation speed
Position
Torque
Number of rotations
Rotation torque
Read Status
Read alarm
Read speed
Read rotation status
Read the number of rotations
Read rotation torque
Condition Monitoring


Device ID
1
Function code
Action control
Initialization
Position
Speed
Torque
Gripping trigger
Number of rotations
Rotation speed
Rotation torque
Rotation trigger
Status query
Read Status
Read initialization
Read alarm
Read position
Read speed
Read torque
Read rotation status
Read rotation initialization
Read the number of rotations
Read rotation speed
Read rotation torque
Clear
Configure
Reset
Active

Gripper



1↔ MoveGripper(1,100,60,60,2000,0,1,0,0,0)

2↔ MoveGripper(1,100,60,42,2000,0,1,20,60,60)

3↔ MoveGripper(1,0,60,60,2000,0,1,20,60,60)

4↔ MoveGripper(1,0,60,60,2000,0,1,0,60,60)

Figure 8.2-12 Rotary Gripper Motion Command Example

Note: The rotation turns are absolute rotation turns. The maximum forward rotation turns are 90, and the maximum reverse rotation turns are 90. A reset operation is required after rotation.

1.3.8.3 Force Sensor

In the "Initial" -> "Peripherals" -> "Force Sensor" interface, force sensors can currently be used via Pre-Adapted Devices and the End-Effector Lua Custom Open Protocol.

1.3.8.3.1 Pre-Adapted Devices

Step1: Click "Pre-Adapted Devices" to enter the end-effector peripheral configuration interface.

The force sensor configuration information includes manufacturer, type, software version, and mounting position. Users can configure the corresponding force sensor information according to specific production needs. If the user needs to change the configuration, they can first select the corresponding number, click the "Clear" button to clear the corresponding information, and reconfigure according to the requirements;

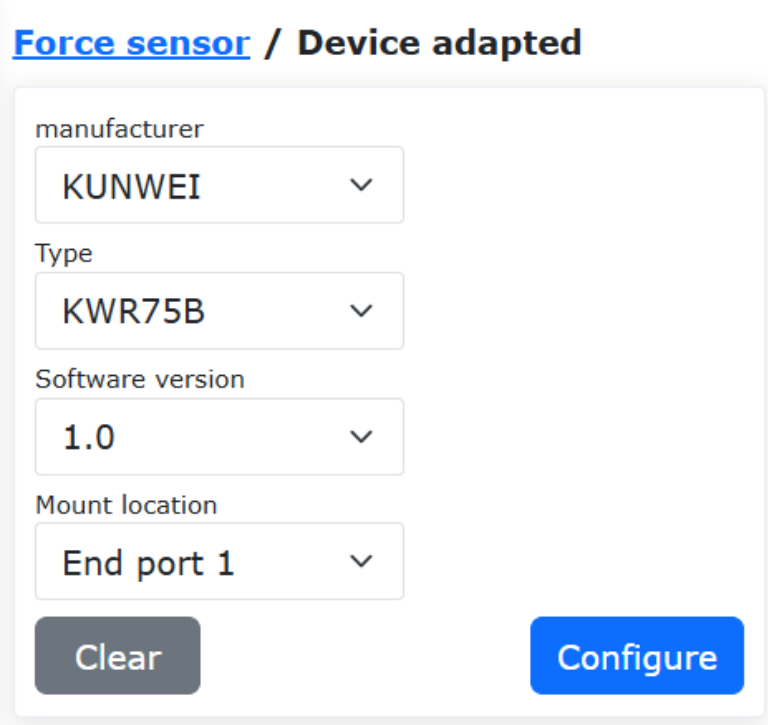


Figure 8.3-1 Force Sensor Configuration

Important: Before clicking to clear the configuration, the corresponding sensor should be in an inactive state.

Step2: After the force sensor configuration is completed, the user can view the corresponding force sensor information in the information table at the bottom of the page. If a configuration error is found, click the "Clear" button to reconfigure.

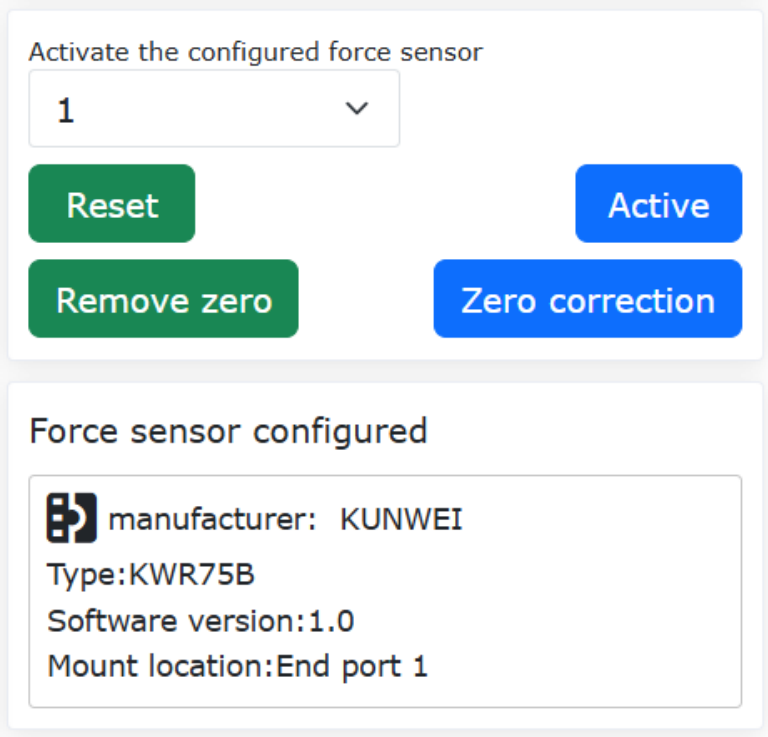


Figure 8.3-2 Force Sensor Configuration Information

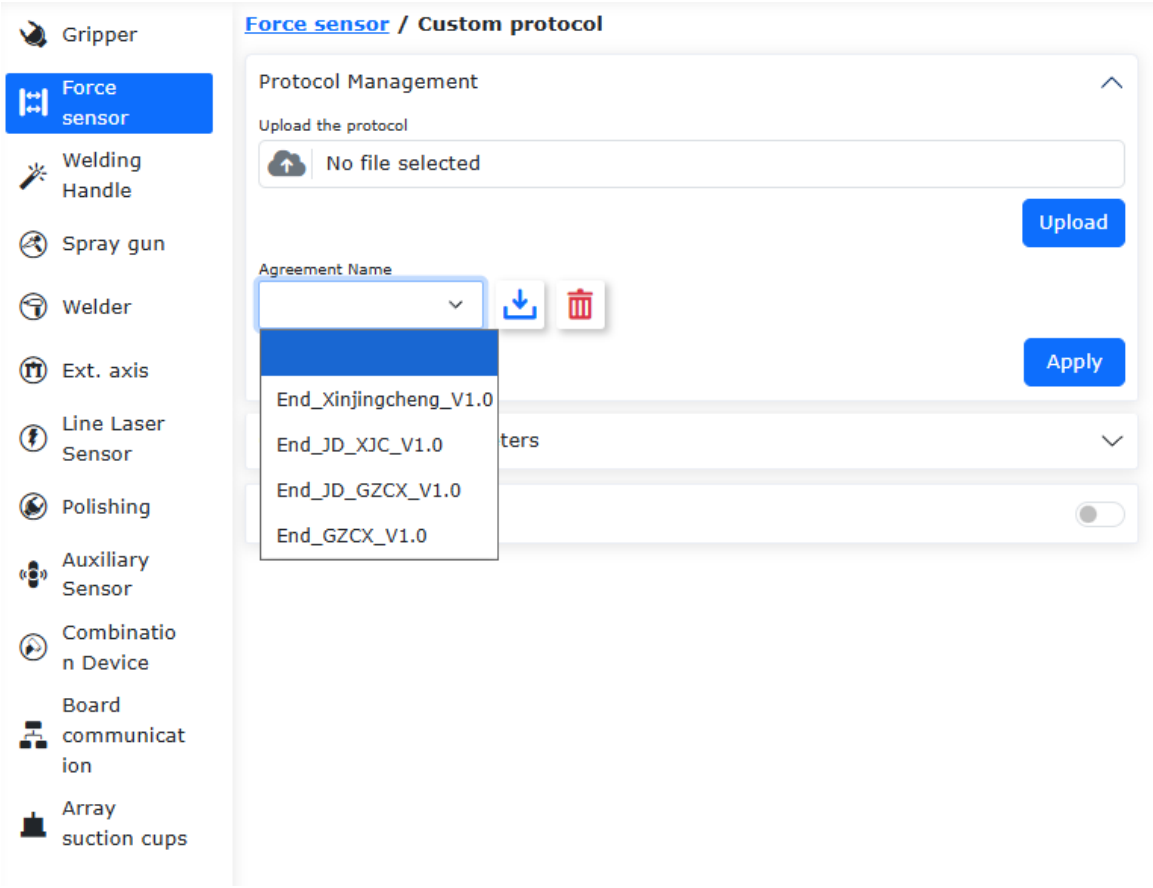
Step3: Select the configured force sensor number, click the "Reset" button, after the page pops up indicating the command was sent successfully, then click the "Activate" button. Check the activation status in the force sensor information table to determine whether the activation was successful; Additionally, the force sensor will have an initial value. The user can choose "Zero Calibration" and "Remove Zero" according to usage requirements. Force sensor zero calibration requires ensuring the force sensor is level and vertically downward, and the robot has no configured load.

Step4: After the force sensor configuration is completed, it is necessary to configure the sensor type tool coordinate system. The sensor tool coordinate system values can be directly input and applied based on the distance between the sensor and the end-effector tool center.

1.3.8.3.2 Force Sensor End-Effector Lua Protocol

Open the WebApp, click "Initial Setup", "Peripherals", "Force Sensor", "Custom Protocol" in sequence. Click "Protocol Management" to configure the end-effector protocol. Currently, the preset built-in protocols for the force sensor are shown in the figure below.Version 3.9.2 has added two embedded combination protocols for gripper + force sensor: End_JD_XJC_V1.0.lua and End_JD_GZCX_V1.0.lua.

Figure 8.3-2-2 Force Sensor Preset Built-in Protocol


Gripper
Force sensor
Welding Handle
Spray gun
Welder
Ext. axis
Line Laser Sensor
Polishing
Auxiliary Sensor
Combinatio n Device
Board communicat ion
Array suction cups
Force sensor / Custom protocol
Protocol Management
Upload the protocol
No file selected
Upload
Agreement Name
Apply
End_Xinjingcheng_V1.0
End_JD_XJC_V1.0
End_JD_GZCX_V1.0
End_GZCX_V1.0

1.3.8.3.3 Welding Handle End Lua Protocol

Open WebApp, then click “Initial Settings,” “Peripherals,” “Welding Handle,” “Custom Protocol” in sequence. Click “Protocol Management” to configure the end protocol. The currently preset embedded protocols for the welding handle are shown in the figure below. Version 3.9.2 added three new embedded combination protocols for SmartTool+gripper or force sensor: End_SM_JD_V1.3.lua, End_SM_GZCX_V1.3.lua, End_SM_XJC_V1.3.lua.

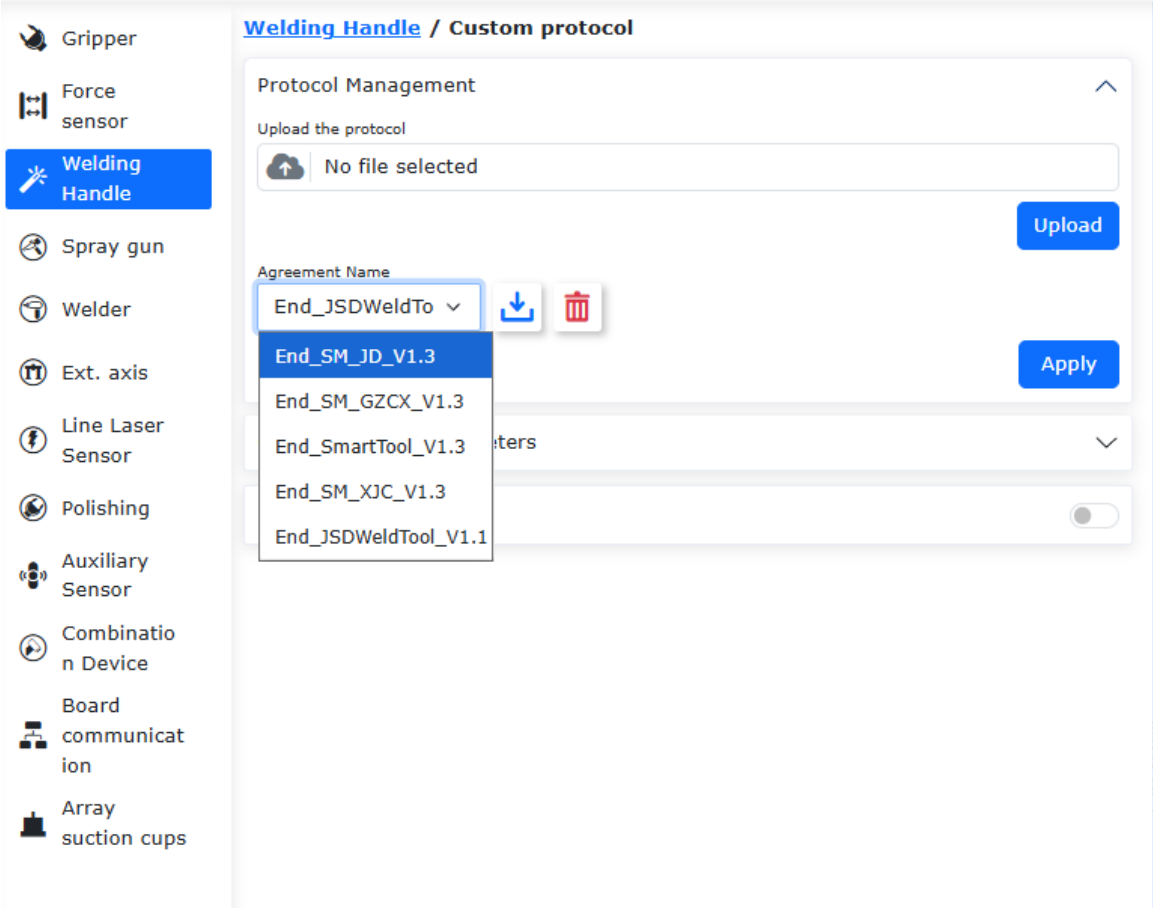


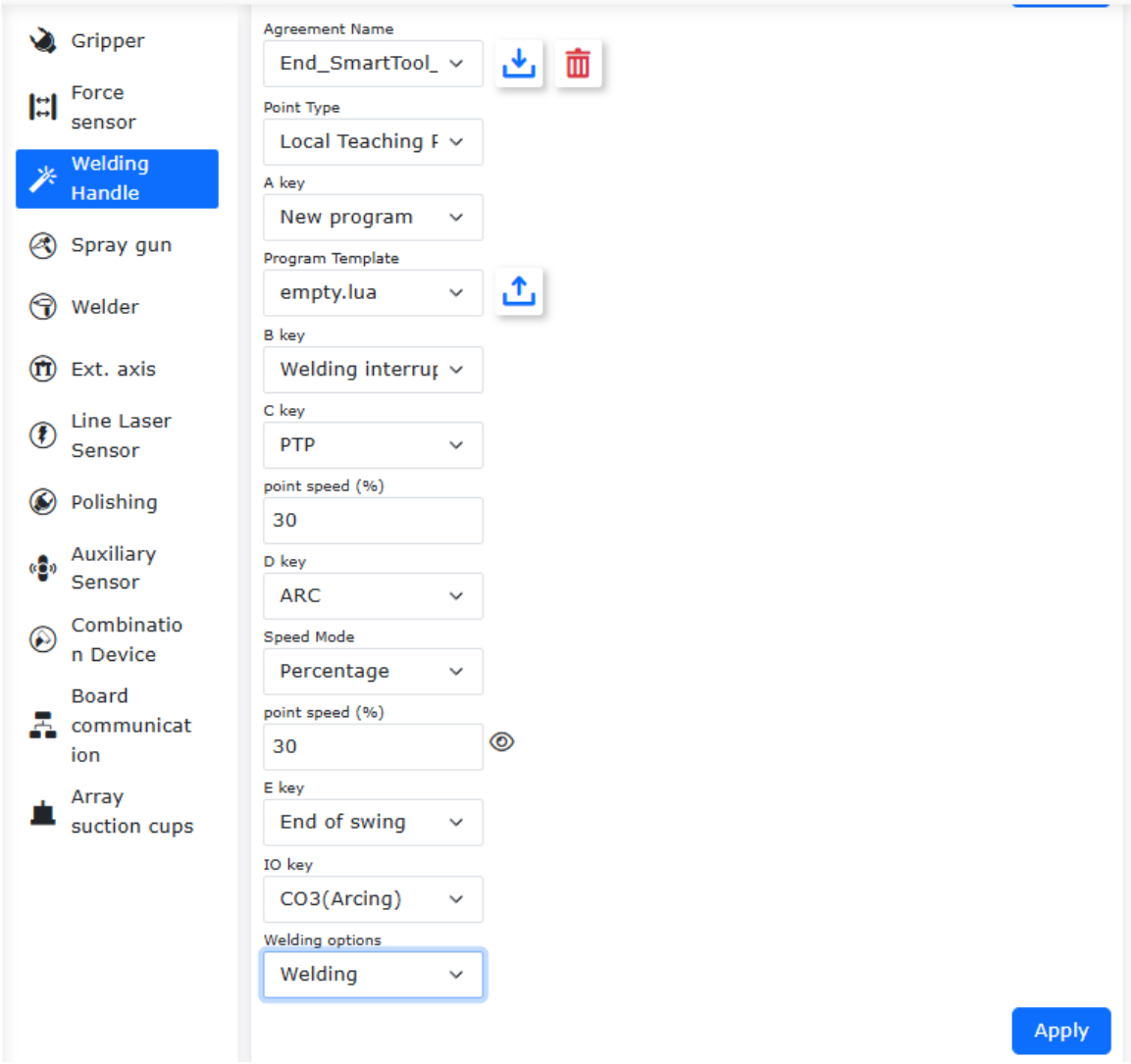
Figure 8.3-2-3 Preset Embedded Protocols for Welding Handle

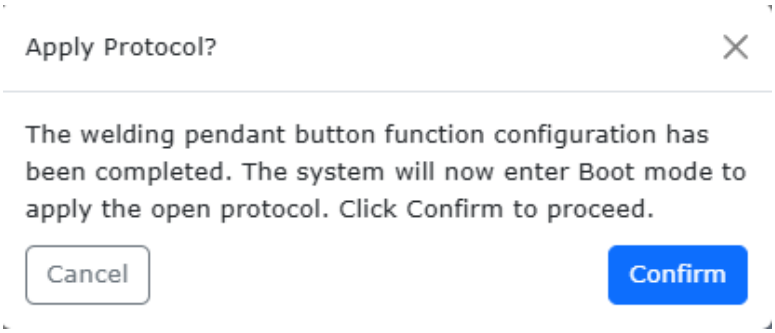
1.3.8.3.3.1 Automatic Generation of End Lua Protocol

This newly added feature allows for automatic generation of the end Lua protocol through web page configuration for protocols related to embedded SmartTool welding handle peripherals (currently only four protocols support automatic generation: End_SmartTool_V1.3.lua, End_SM_JD_V1.3.lua, End_SM_GZCX_V1.3.lua, End_SM_XJC_V1.3.lua). The generated protocol is uploaded and applied to the end without requiring the user to write it. Users configure the A, B, C, D, E, and IO keys of the SmartTool welding handle according to their needs. After configuration is complete, the robot must be disabled, and then click “Apply.” At this point, the page will prompt “Enter boot and apply open protocol?” Clicking Confirm will cause the robot to enter boot state and automatically upload the automatically generated end Lua protocol. After restarting the robot, the SmartTool can be used according to the configured keys.

Figure 8.3-2-4 Automatic Generation of SmartTool Welding Handle Configuration Protocol

Figure 8.3-2-5 Page Prompt “Enter boot and apply open protocol?”


Gripper
Force sensor
Welding Handle
Spray gun
Welder
Ext. axis
Line Laser Sensor
Polishing
Auxiliary Sensor
Combinatio n Device
Board communicat ion
Array suction cups
Agreement Name
End_SmartTool_
Point Type
Local Teaching F
A key
New program
Program Template
empty.lua
B key
Welding interrup
C key
PTP
point speed (%)
30
D key
ARC
Speed Mode
Percentage
point speed (%)
30
E key
End of swing
IO key
CO3(Arcing)
Welding options
Welding
Apply


Apply Protocol?
The welding pendant button function configuration has been completed. The system will now enter Boot mode to apply the open protocol. Click Confirm to proceed.
Cancel
Confirm

1.3.8.3.3.2 SmartTool Program Generation Template Import

If the SmartTool key is configured with the program generation function, based on the open protocol, two types of generated programs are provided: a blank Lua program is generated by default, or the user can choose to upload a template starting with "**template_**" as the template for the new program. When the new program selects the template program, the Lua file generated by triggering "New Program" on the SmartTool includes the content of the uploaded template file. Any subsequently added instructions are appended after the template content.

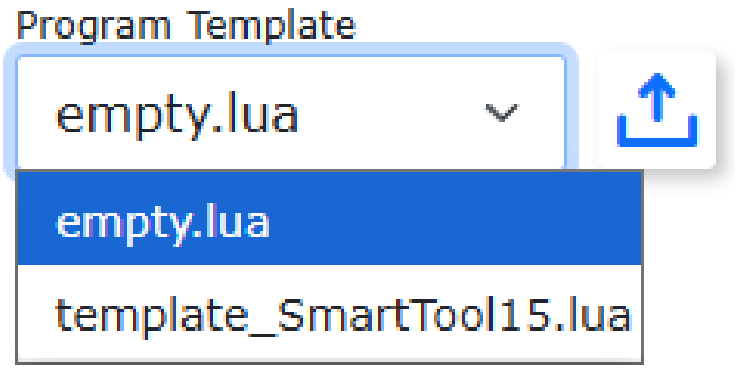


Figure 8.3-2-6 SmartTool Program Generation Template Import

1.3.8.3.3.3 SmartTool Motion Instruction Point Configuration

When configuring the "PTP," "LIN," and "ARC" instructions in SmartTool, you can choose the storage database for the generated instruction points to be "Global Teaching Points" or "Local Teaching Points." When "Global Teaching Points" is selected, the generated instruction points can be viewed through "Teaching Program," "Teaching Points." When "Local Teaching Points" is selected, the generated instruction points can be viewed through "Teaching Program," "Program Programming," "Local Teaching Points."

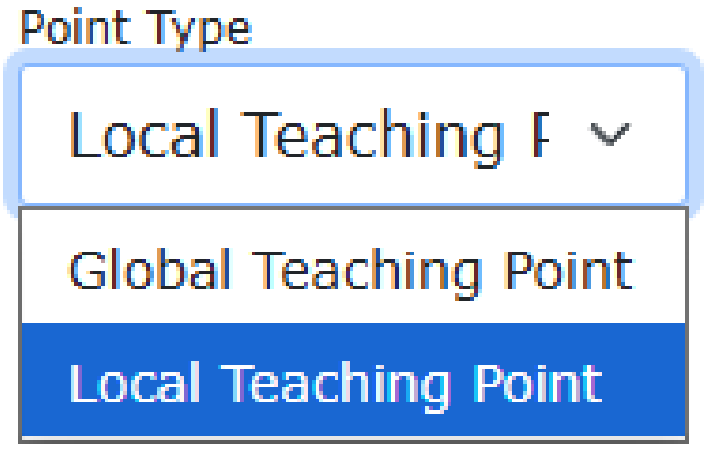


Figure 8.3-2-7 SmartTool Motion Instruction Point "Global Teaching Points" and "Local Teaching Points" Configuration

1.3.8.3.3.4 SmartTool Anti-Mistouch Mode

The SmartTool based on the open protocol adds an anti-mistouch mode. Click "Initial Settings," "Peripherals," "Welding Handle," "Custom Protocol" in sequence. After enabling the end protocol, you can see the switch for "Anti-Mistouch Mode." When this function is enabled, the two key functions "Undo Program" and "Clear Program" on the SmartTool need to be pressed twice to trigger.

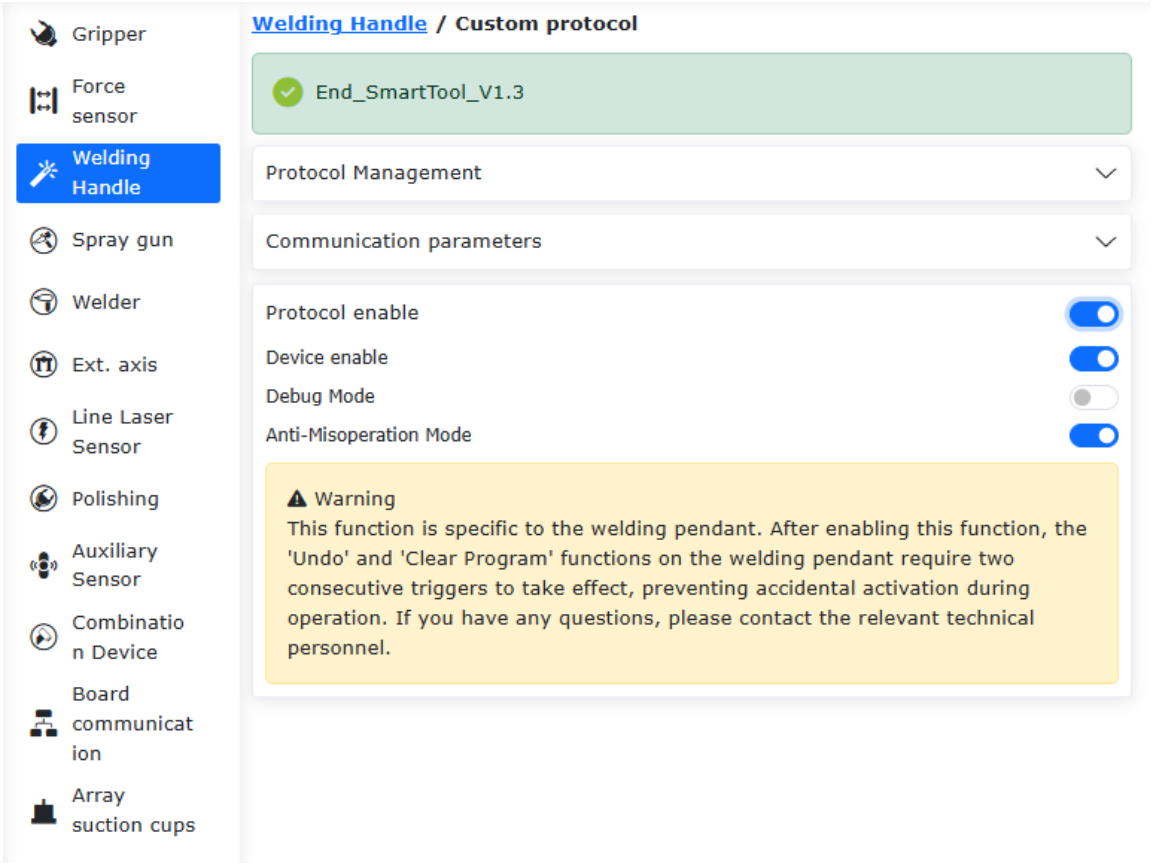


Figure 8.3-2-8 SmartTool "Anti-Mistouch Mode" Function

1.3.8.3.3.5 Example of Lua End Peripheral Protocol for Welding Handle

The functions of the six keys A, B, C, D, E, and IO can be modified and defined by changing the key value on line 31 of the code. Among them, K38=Getbit(R[7],1) and K0=Getbit(R[7],2) are for "Clear Program" and "Undo Key" respectively and cannot be modified. The subsequent five K values can be modified according to the definitions in the *End Full Peripheral Protocol* document. In this example (embedded SmartTool protocol), the corresponding key functions are: A: LIN, B: PTP, C: Create Program, D: Weld Interruption Recovery, E: Weld Interruption Exit, IO: LIN+Welding+Weaving.

Example of Lua End Peripheral Protocol for Welding Handle (SmartTool)

```
1 function Getbit(X,Bit)
2 return ((X&(1<<Bit))>>Bit)
3 end
4
5 if(Getbit(GetRobotState(),0)==1)then
```

(continues on next page)

(continued from previous page)

```
local SetParams={B6=3}-- B6 - Operating DO port number is DO3
SetWeldParams(SetParams)
while(1)
do
IwdgTaskHandle()
MainLoop()
UpDownLoadHandle()
SdoRwPara()
EndErrClear()
local BFlag=LuaBreak()
if(BFlag==1)then
break
end
local R={0}
local T={0x7D,0x01,0x30,0xC0,0x00,0x04,0x00,0x00,0x00,0x00}
DelayMs(100)
T[7],T[8],T[9],T[10]=GetIoCmd()
Dword=GetRobotState()
T[7]=Getbit(Dword,4)
T[12],T[11]=WeldToolCrcValue(T)
T[13]=0x0E
WeldToolSlaveSetCmd(T)
DelayMs(3)
Len=EndRxWeldData(R)
if((Len==13)and(R[1]==0x7D)and(R[2]==0x01)and(R[3]==0x30))then
local key={K38=Getbit(R[7],1),K0=Getbit(R[7],2),K3=Getbit(R[7],3),K25=Getbit(R[7],4),
 ↪K39=Getbit(R[7],5),K27=Getbit(R[7],6),K28=Getbit(R[7],7), K44=Getbit(R[8],0),
K6=Getbit(R[8],1),K7=Getbit(R[8],2)}--smarttool welding handle key settings, Undo key -␣
 ↪K38 Undo Program; Clear key - K0 Clear Program; A key - K3 LIN; B key - K25 PTP; C key␣
 ↪- K39 Create Program; D key - K27 Weld Interruption Recovery; E key - K28 Weld␣
 ↪Interruption Exit; IO key - K44 LIN+Welding+Weaving  Manual/Auto key - K6 Manual/Auto;␣
 ↪Run/Pause key - K7 Run/Pause
SetWeldToolKeys(key)
end
LuaGc()
end
end
```

1.3.8.3.4 Sensor Load Identification

Under the “Initial” -> “Basic” -> “Load” menu bar, click “Sensor Identification” to enter the sensor load identification interface.

Specific Pose Identification: Clear the end-effector load data, configure the force sensor, establish the sensor coordinate system, adjust the robot end-effector posture to vertical downward, perform “Zero Calibration” and then install the end-effector load. First, select the corresponding sensor tool coordinate system, adjust the robot so that the sensor and tool are vertical downward, record data, and calculate the mass. Then, adjust the robot to 3 different postures, record three sets of data respectively, calculate the center of mass, confirm it is correct and click Apply.

Dynamic Identification: Clear the end-effector load data, configure the force sensor, establish the sensor coordinate system, adjust the robot end-effector posture to vertical downward, perform “Zero Calibration” and then install the end-effector load. Click “Identification Start”, drag the robot to move, then click “Identification Stop”, and the load

result will be automatically applied to the robot.

Auto Zero: After the sensor records the initial position, it can automatically zero.

Figure 8.3-3 Sensor Load Identification

1.3.8.3.5 Force Sensor Assisted Dragging

After configuring the sensor, it can be paired with the sensor to provide better assistance for dragging the robot. For the first use, you can configure according to the data in the right figure. After applying, you do not need to enter the drag mode; directly drag the end-effector force sensor to control the robot to move in a fixed posture. (The data in the figure below is a reference standard)

Figure 8.3-4 Force/Torque Sensor Drag Lock

Note: Singularity Strategy is a function developed under force sensor assisted locking for singularity crossing and avoidance.

Singularity Avoidance Strategy is the default function option. After enabling assisted dragging, the avoidance function is enabled by default. Singularity avoidance is a function that applies virtual force to move the robot away from a singular configuration when the robot is in a singular pose.

Singular Configurations:

Elbow Singularity: Rotation axes 2, 3, and 4 are in the same plane. At this time, the elbow joint is fully extended or fully contracted. Due to FR robot mechanical limits, the fully contracted configuration cannot be reached.

Wrist Singularity: Rotation axes 4 and 6 are parallel. Due to FR robot mechanical limits, this configuration cannot be reached.

Shoulder Singularity: The wrist center point is located in the plane formed by rotation axes 1 and 2.

Singularity Crossing Function: Select "Singularity Strategy" as "Cross" and apply. When the robot detects that the current pose is in a singular configuration, it automatically switches to the current loop drag mode. When it detects exiting the singular configuration, the drag mode switches back to force sensor assisted dragging to continue motion.

Adaptive Selection: Enable when assembly is required; after enabling, dragging becomes heavier;

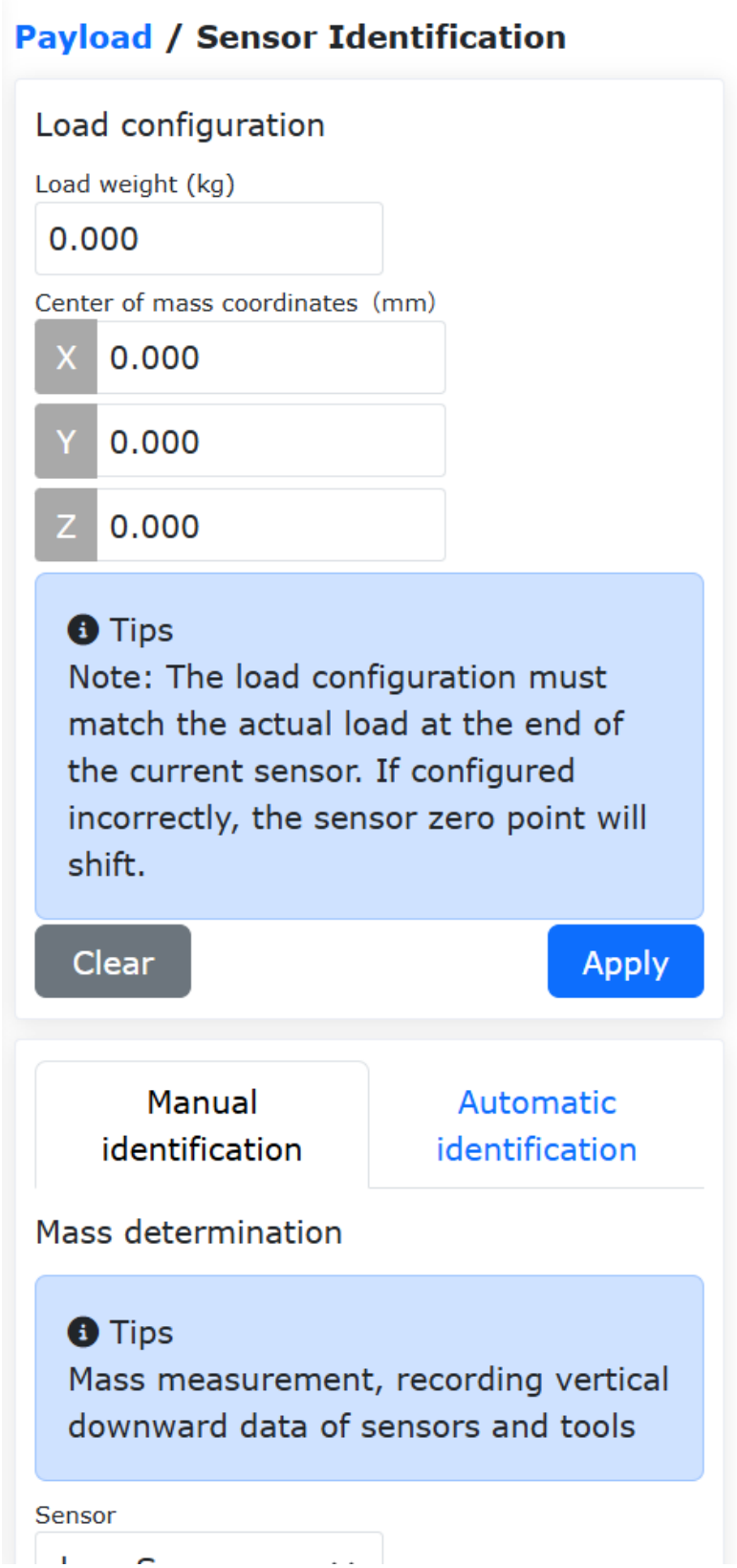
Inertia Parameters: Adjust the feel during the dragging process, operate with caution under the guidance of technical personnel.

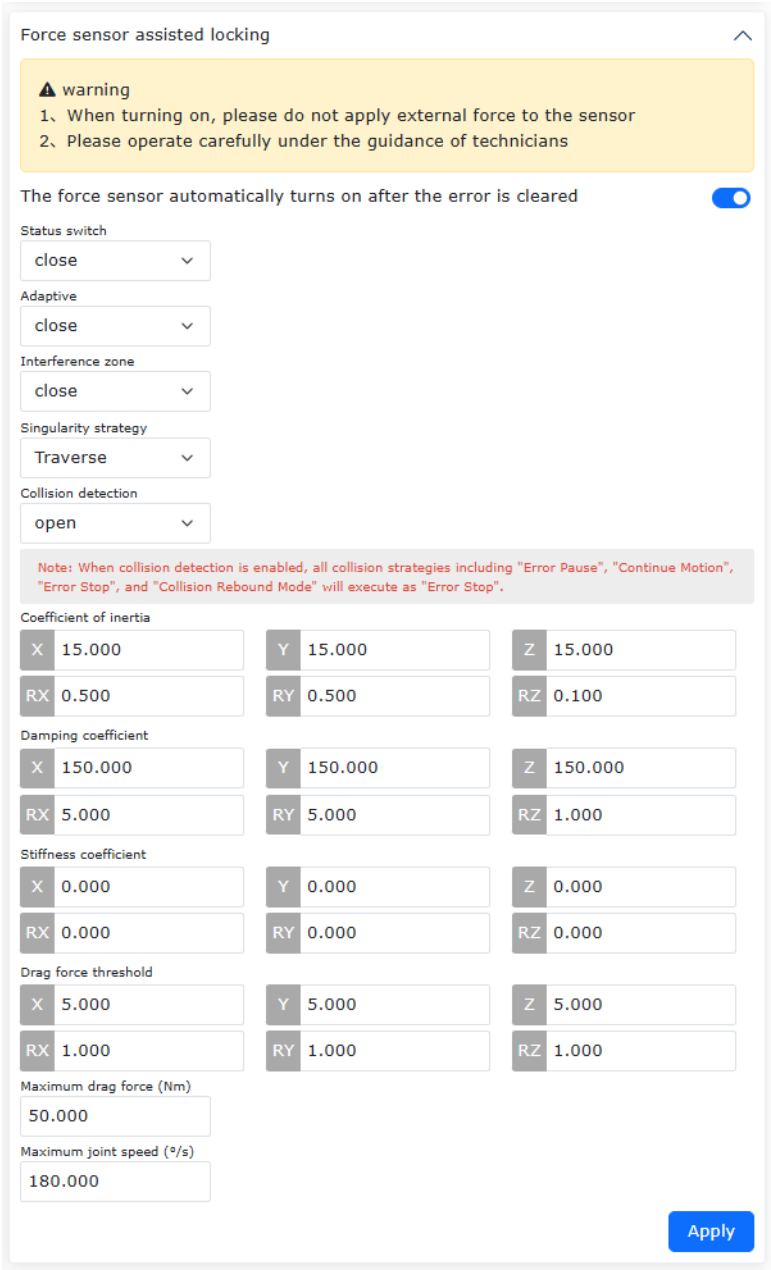
Damping Parameters:

- Translation Direction: Recommended parameter range [100-200];
- Rotation Direction: Recommended parameter range [3-10], with the RZ direction range [0.1-5];
- Effect: When dragging with the sensor, increasing damping makes dragging difficult, decreasing damping makes dragging the robot too easy (recommended not to set too small);
- Overall Damping Parameter Range: Translation XYZ: [100-1000]; Rotation RX, RY: [3-50], RZ: [2-10];
- Maximum Drag Force is 50, Maximum Drag Speed is 180.

Stiffness Parameters: All set to 0;

Drag Force Threshold: Translation XYZ: [5-10]; Rotation RX, RY, RZ: [0.5-5];


Payload / Sensor Identification
Load configuration
Load weight (kg)
0.000
Center of mass coordinates (mm)
X
0.000
Y
0.000
Z
0.000
Tips
Note: The load configuration must match the actual load at the end of the current sensor. If configured incorrectly, the sensor zero point will shift.
Clear
Apply
Manual identification
Automatic identification
Mass determination
Tips
Mass measurement, recording vertical downward data of sensors and tools
Sensor


Force sensor assisted locking
warning
1、When turning on, please do not apply external force to the sensor
2、Please operate carefully under the guidance of technicians
The force sensor automatically turns on after the error is cleared
Status switch
close
Adaptive
close
Interference zone
close
Singularity strategy
Traverse
Collision detection
open
Note: When collision detection is enabled, all collision strategies including "Error Pause", "Continue Motion", "Error Stop", and "Collision Rebound Mode" will execute as "Error Stop".
Coefficient of inertia
X 15.000
Y 15.000
Z 15.000
RX 0.500
RY 0.500
RZ 0.100
Damping coefficient
X 150.000
Y 150.000
Z 150.000
RX 5.000
RY 5.000
RZ 1.000
Stiffness coefficient
X 0.000
Y 0.000
Z 0.000
RX 0.000
RY 0.000
RZ 0.000
Drag force threshold
X 5.000
Y 5.000
Z 5.000
RX 1.000
RY 1.000
RZ 1.000
Maximum drag force (Nm)
50.000
Maximum joint speed (°/s)
180.000
Apply

Important: Locking is achieved by increasing the force threshold for translation directions XYZ or rotation directions RX, RY, RZ.

1.3.8.3.6 Force/Torque Sensor Collision Detection

Command Description: The "FT_Guard" command is the collision detection command. Select the corresponding sensor coordinate system, check the effective torque direction detection, set the current value, maximum collision threshold, and minimum collision threshold. The normal collision detection condition range is (Current Value - Min Threshold, Current Value + Max Threshold). Add the "Enable" and "Disable" commands to the program.

F/T

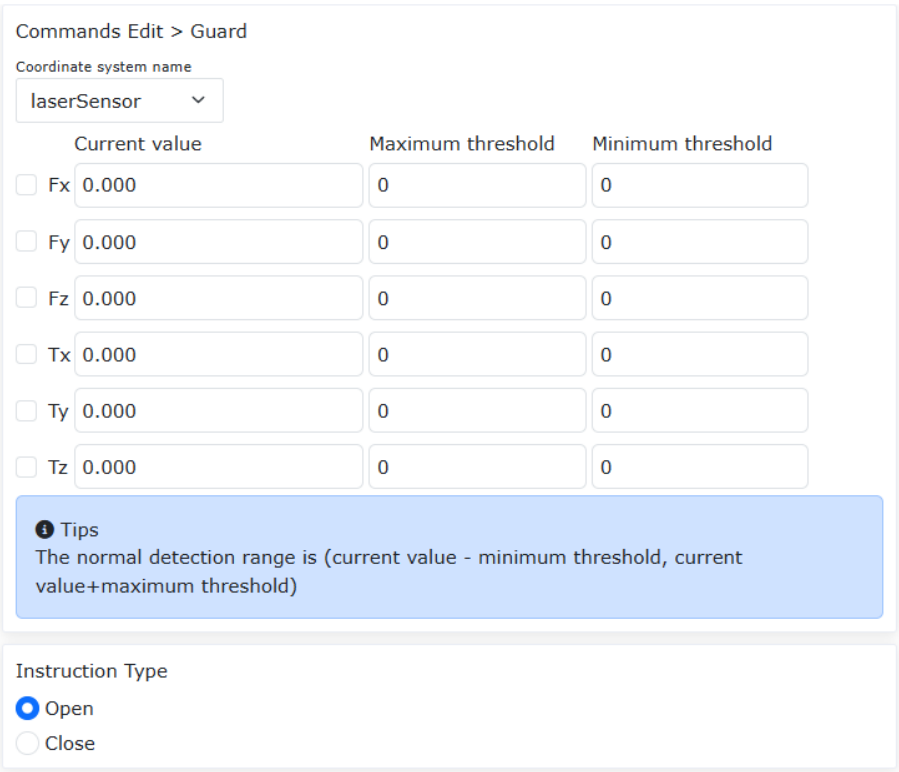


Figure 8.3-5 FT_Guard Command Editing

Program Example:

No.	Command Format	Comment
1	FT_Guard(1,1,1,1,1,0,0,0,5,0,0,0,0,0,10,0,(	# Force/Torque Collision Detection Enable
2	PTP(template1,100,-1,0)	# Motion Command
3	FT_Guard(0,1,1,1,1,0,0,0,5,0,0,0,0,0,10,0,(	# Force/Torque Collision Detection Disable

1.3.8.3.7 Force/Torque Sensor Force Control Motion

Command Description: The "FT_Control" command is the force control motion command, which allows the robot to move near the set force, commonly used in grinding scenarios. Select the corresponding sensor coordinate system, check the effective torque direction detection, set the detection threshold, and the PID proportional coefficients in each direction (generally set p to 0.001), set the maximum adjustment distance (for X,Y,Z) and maximum adjustment angle (for RX,RY,RZ). Add the "Enable" and "Disable" commands to the program.

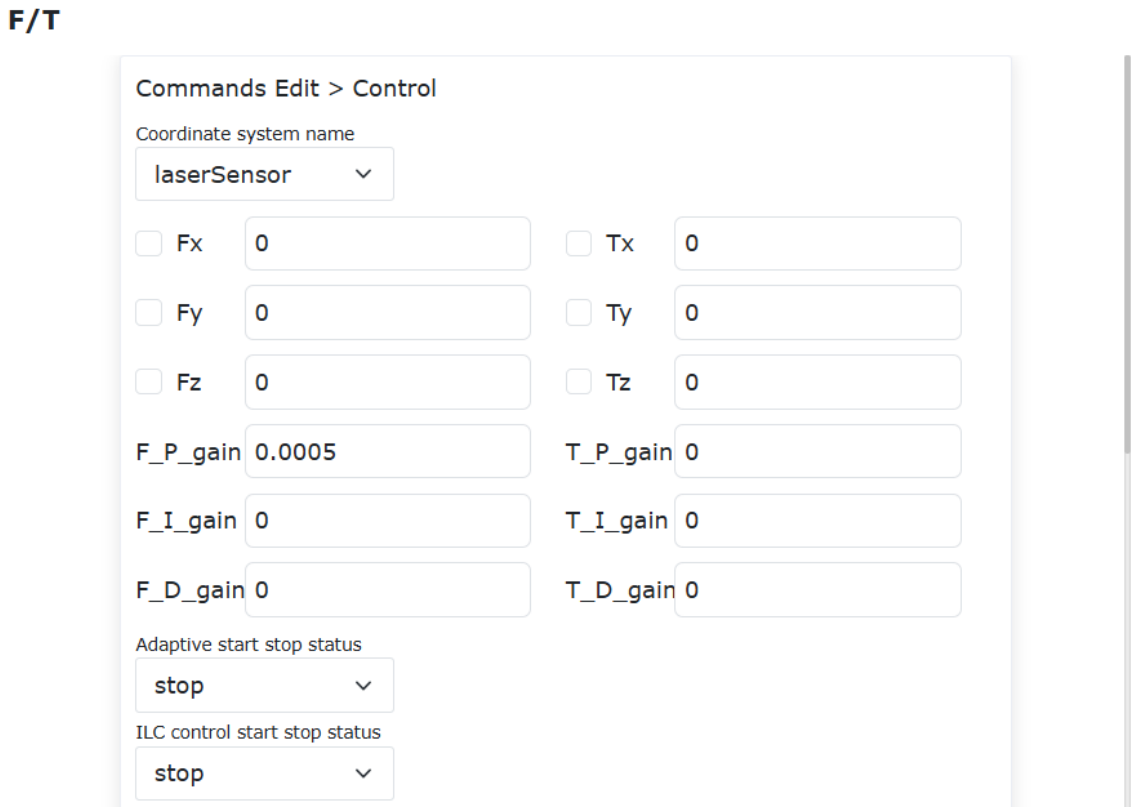
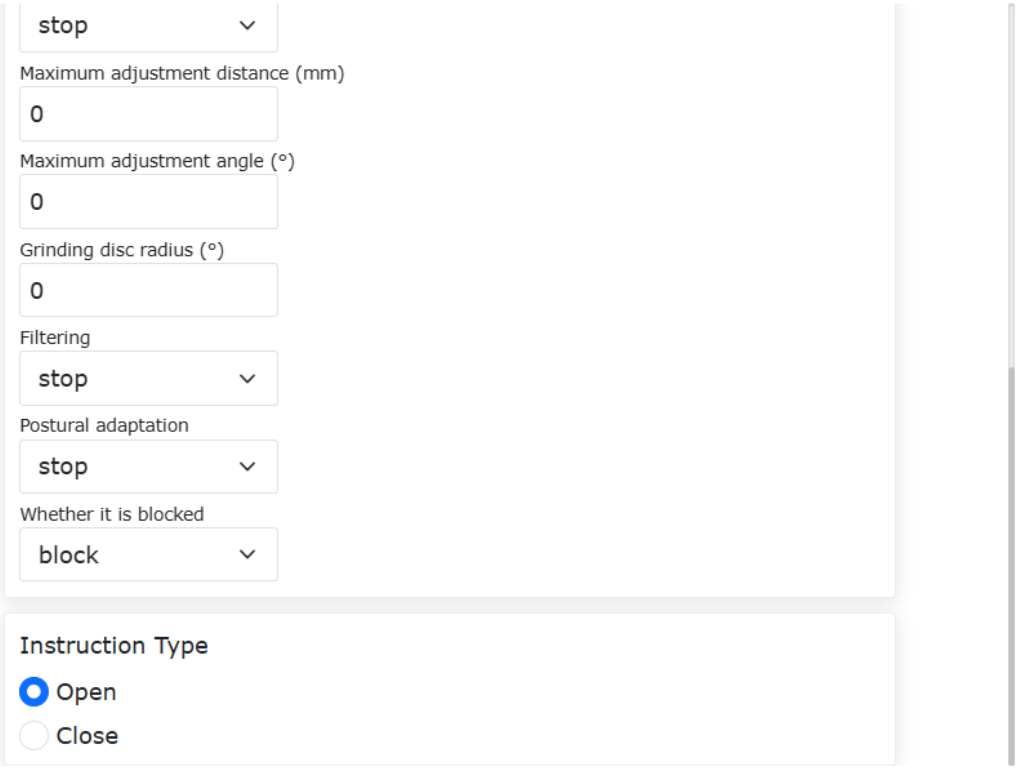


Figure 8.3-6 FT_Control Command Editing

Program Example:

No.	Command Format	Comment
1	FT_Control(1,11,1,0,1,0,0,0,10,0,5,0,0,0,0.	# Force/Torque Motion Control Enable
2	Lin(template3,100,-1,0,0)	# Motion Command
3	FT_Control(0,11,1,0,1,0,0,0,10,0,5,0,0,0,0.	# Force/Torque Motion Control Disable

F/T



1.3.8.3.8 Force/Torque Sensor Spiral Insertion

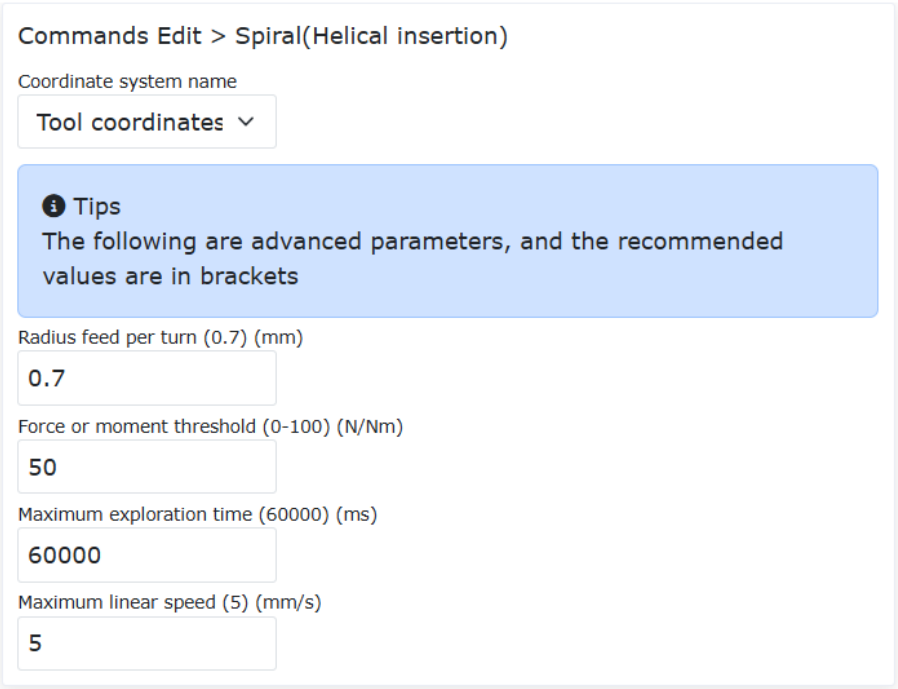
Command Description: The "FT_Spiral" command is for spiral search insertion, generally used for shaft-hole assembly actions of cylindrical shafts. Before running the action, the robot end-effector needs to be dragged to the approximate position of the hole. According to the current scene, set the command parameters, add them to the program. After running, the robot will explore with a spiral motion.

Figure 8.3-7 FT_Spiral Command Editing

Program Example:

No.	Command Format	Comment
1	FT_Control(1,10,0,0,1,0,0,0,0,0,5,0,0,0,0,0	# Force/Torque Motion Control Enable
2	FT_SpiralSearch(0,0.7,0,60000,5)	# Spiral Insertion
3	FT_Control(0,10,0,0,1,0,0,0,0,0,5,0,0,0,0,0	# Force/Torque Motion Control Disable

F/T



1.3.8.3.9 Force/Torque Sensor Rotation Insertion

Command Description: The "FT_Rot" command is for rotation search insertion, generally used to follow the spiral insertion action, for keyed shaft-hole assembly. Before running the action, the robot end-effector needs to be moved to the hole position found by the spiral search or a taught hole position that is completely aligned. According to the current scene, set the command parameters, add them to the program. After running, the robot will explore with a slow rotation.

Figure 8.3-8 FT_Rot Command Editing

Program Example:

No.	Command Format	Comment
1	FT_Control(1,10,0,0,1,0,0,0,0,0,5,0,0,0,0,0.0	# Force/Torque Motion Control Enable
2	FT_RotInsertion(0,3,0,5,1,0,1)	# Rotation Insertion
3	FT_Control(0,10,0,0,1,0,0,0,0,0,5,0,0,0,0,0.0	# Force/Torque Motion Control Disable

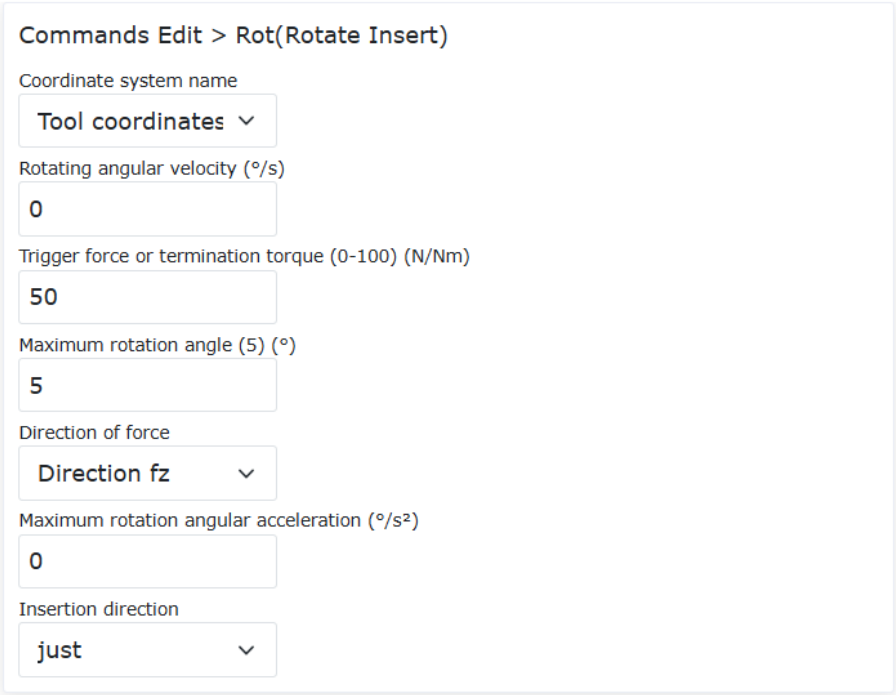
1.3.8.3.10 Force/Torque Sensor Linear Insertion

Command Description: The "FT_Lin" command is for linear search insertion, generally used to follow the spiral insertion action or rotation insertion action, for keyed shaft-hole assembly. Before running the action, the robot end-effector needs to be moved to the hole position found by the spiral search, the end position of the rotation insertion action, or a taught hole position that is completely aligned. According to the current scene, set the command parameters, add them to the program. After running, the robot will perform linear motion in the set direction.

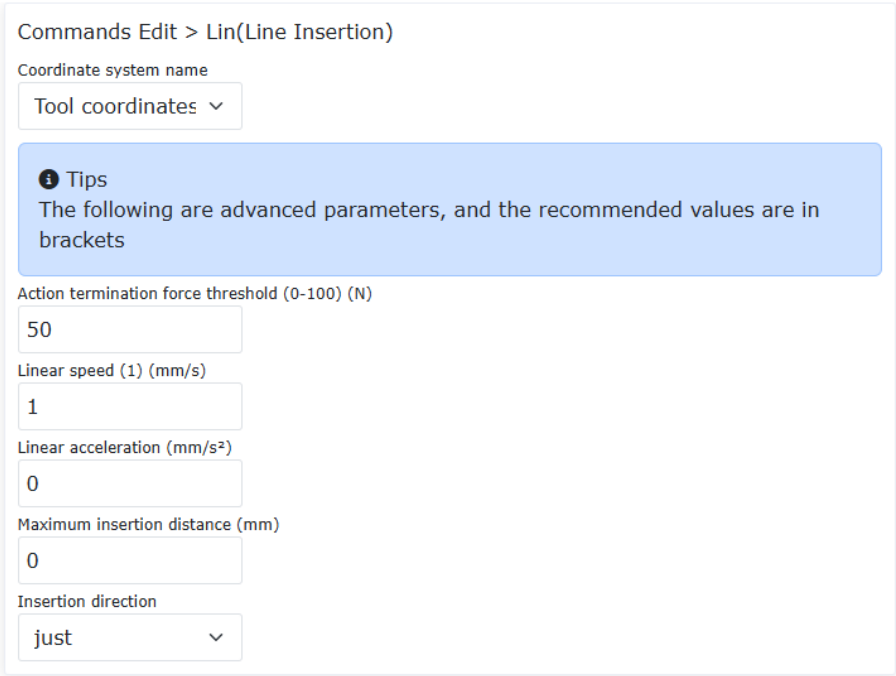
Figure 8.3-9 FT_Lin Command Editing

Program Example:

F/T


Commands Edit > Rot(Rotate Insert)
Coordinate system name
Tool coordinates
Rotating angular velocity (°/s)
0
Trigger force or termination torque (0-100) (N/Nm)
50
Maximum rotation angle (5) (°)
5
Direction of force
Direction fz
Maximum rotation angular acceleration (°/s²)
0
Insertion direction
just

F/T


Commands Edit > Lin(Line Insertion)
Coordinate system name
Tool coordinates
Tips
The following are advanced parameters, and the recommended values are in brackets
Action termination force threshold (0-100) (N)
50
Linear speed (1) (mm/s)
1
Linear acceleration (mm/s²)
0
Maximum insertion distance (mm)
0
Insertion direction
just

No.	Command Format	Comment
1	FT_Control(1,10,0,0,1,0,0,0,0,0,5,0,0,0,0,0.0	# Force/Torque Motion Control Enable
2	FT_LinInsertion(0,50,1,0,100,1)	# Linear Insertion
3	FT_Control(0,10,0,0,1,0,0,0,0,0,5,0,0,0,0,0.0	# Force/Torque Motion Control Disable

1.3.8.3.11 Force/Torque Sensor Surface Finding

Command Description: The "FT_FindSurface" command is for surface finding, generally used to find the surface of an object. According to the current scene, set the corresponding coordinate system, movement direction, movement axis, search linear speed, search linear acceleration, maximum search distance, action termination force threshold and other parameters, add them to the program. Run the program, the action starts executing, and the robot end-effector begins to slowly move towards the direction of the surface.

F/T

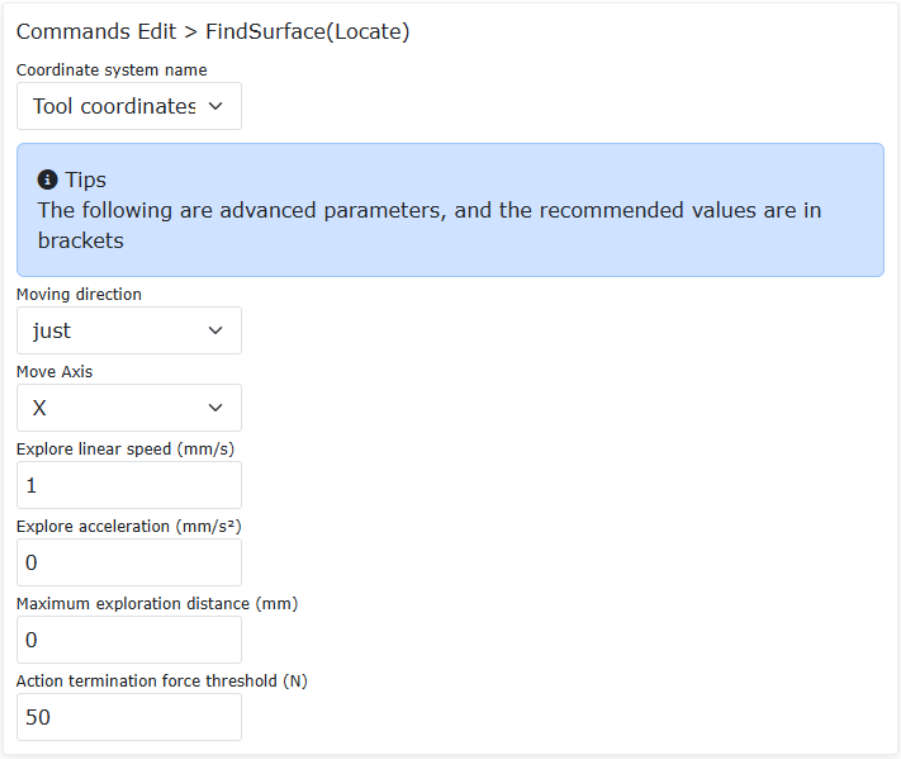


Figure 8.3-10 FT_FindSurface Command Editing

Program Example:

No.	Command Format	Comment
1	PTP(1,30,-1,0)	# Initial Position
2	FT_FindSurface(0,1,3,1,0,100,5)	# Surface Finding

1.3.8.3.12 Force/Torque Sensor Center Finding

Command Description: The "FT_CalCenter" command is for center finding, generally used to find the middle plane surface between two surfaces. According to the current scene, set the corresponding coordinate system, movement direction, movement axis, search linear speed, search linear acceleration, maximum search distance, action termination force threshold and other parameters, find surface A and surface B respectively, add them to the program. Run the program, the action starts executing, the robot slowly moves towards the direction of surface A, after locating surface A, the robot slowly moves towards the direction of surface B, after locating surface B, the center plane position can be calculated.

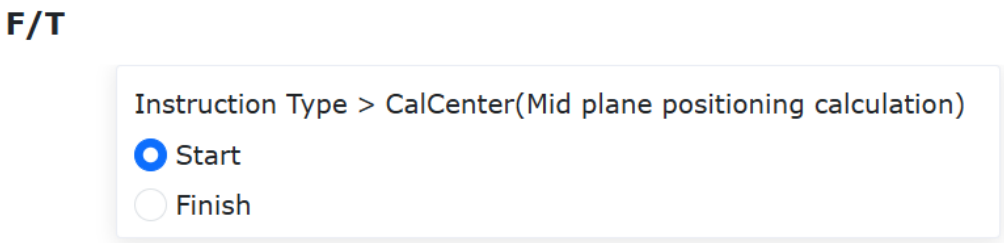


Figure 8.3-11 FT_CalCenter Command Editing

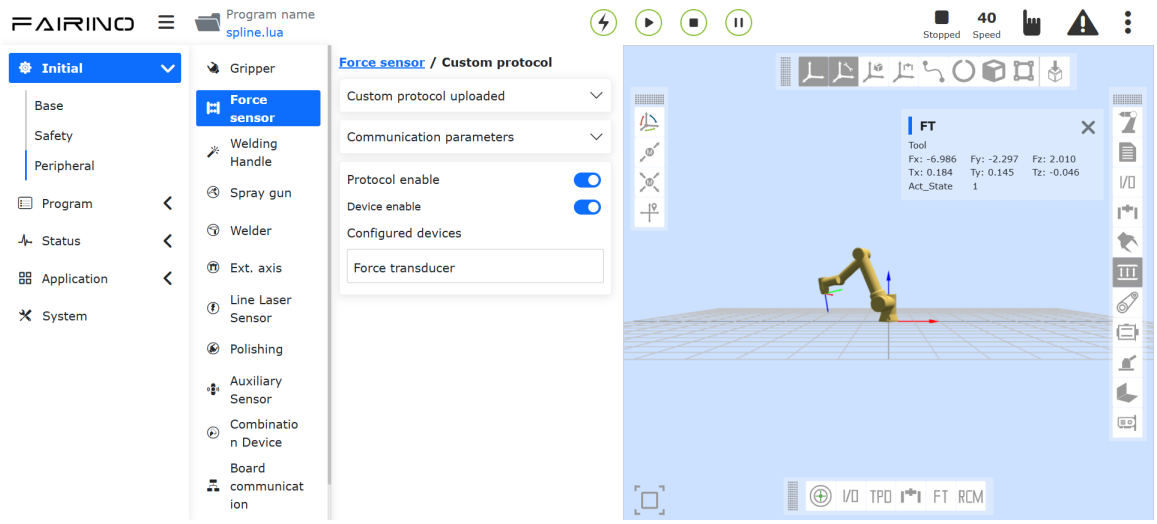
Program Example:

No.	Command Format	Comment
1	PTP(1,30,-1,0)	# Initial Position
2	FT_CalCenterStart()	# Surface Finding Start
3	FT_Control(1,10,0,0,1,0,0,0,0,0,-10,0,0,0,0.00001,0,0,0,0,0,0,100,0)	# Force/Torque Motion Control Enable
4	FT_FindSurface(1,2,2,10,0,200,5)	# Locate Plane A
5	FT_Control(0,10,0,0,1,0,0,0,0,0,-10,0,0,0,0.00001,0,0,0,0,0,0,100,0)	# Force/Torque Motion Control Disable
6	PTP(1,30,-1,0)	# Initial Position
7	FT_Control(1,10,0,0,1,0,0,0,0,0,-10,0,0,0,0.00001,0,0,0,0,0,0,100,0)	# Force/Torque Motion Control Enable
8	FT_FindSurface(1,1,2,20,0,200,5)	# Locate Plane B
9	FT_Control(0,10,0,0,1,0,0,0,0,0,10,0,0,0,	# Force/Torque Motion Control Disable
10	pos={}	# Define array pos
11	pos = FT_CalCenterEnd()	# Get located center Cartesian pose
12	MoveCart(pos,GetActualTCPNum(),Get/ 1,0)	# Move to the located center position

1.3.8.3.13 Custom Open Protocol

Click the "Custom Protocol" card to enter the interface, enable the force sensor. The configured device displays the force sensor. Click to enter the FT interface to query force sensor data.

Figure 8.3-12 Enable Force Sensor



1.3.8.4 Welding Pendant

In the “Initial” -> “Peripherals” -> “Welding Pendant” interface, the welding pendant can currently be used through adapted devices and the end Lua custom open protocol.

1.3.8.4.1 Adapted Devices

1.3.8.4.1.1 Configuration Steps

Step1: Click the “Adapted Devices” card to enter the Adapted Devices interface. The configuration information is divided into Manufacturer, Type, Software Version, and Mounting Position. Users can configure the corresponding information according to specific production needs. If users need to change the configuration, they can first select the corresponding manufacturer, click the “Clear” button to clear the corresponding information, and reconfigure according to requirements.

Figure 8.4-1 Welding Pendant Adapted Device Configuration

Important: The corresponding device should be in an inactive state before clicking to clear the configuration.

Step2: Configure key positions A-E and the IO key sequentially. After the Smart Tool configuration is completed, the task manager internally maintains the function corresponding to each button. When it detects that a button is pressed, it automatically executes the function item corresponding to that button.

A-E Key Functions:

- **Motion Command: When selecting PTP, LIN, or ARC motion commands, you need to input the corresponding point speed. For LIN and ARC commands, you can choose “Percentage” or “Physical Speed”:**
  - **Percentage:** Input a debugging speed percentage. The robot moves at a percentage of its maximum speed. The actual robot movement speed is calculated as: V = Robot Maximum Speed × Global Speed Percentage × Point Speed Percentage. Hover the mouse over the small eye icon to the right of the “Point Speed” input box to display the actual physical speed (in mm/s) of the robot in both manual and automatic modes under the current settings.

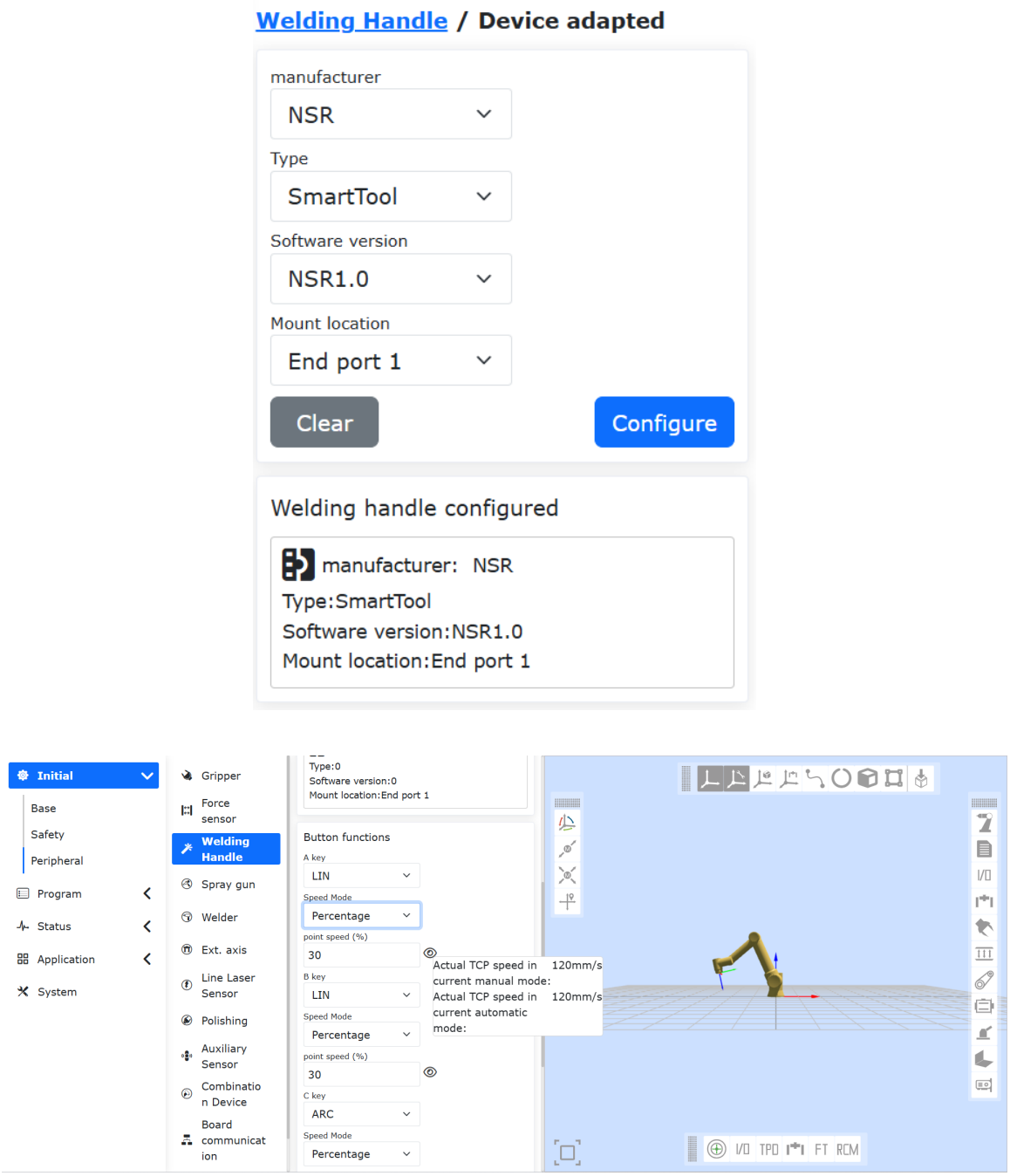


Figure 8.4-2-1 Display Actual Physical Speed Value When Inputting Percentage

- **Physical Speed:** The input speed is the actual operating speed of the robot, in mm/s. The input acceleration is typically set to twice the speed. (The maximum physical speed of the LIN command is limited by the global speed percentage. If the robot's maximum operating speed is 1000 mm/s and the global speed is 50%, then the maximum physical speed for the LIN command is 1000 × 50% = 500 mm/s).

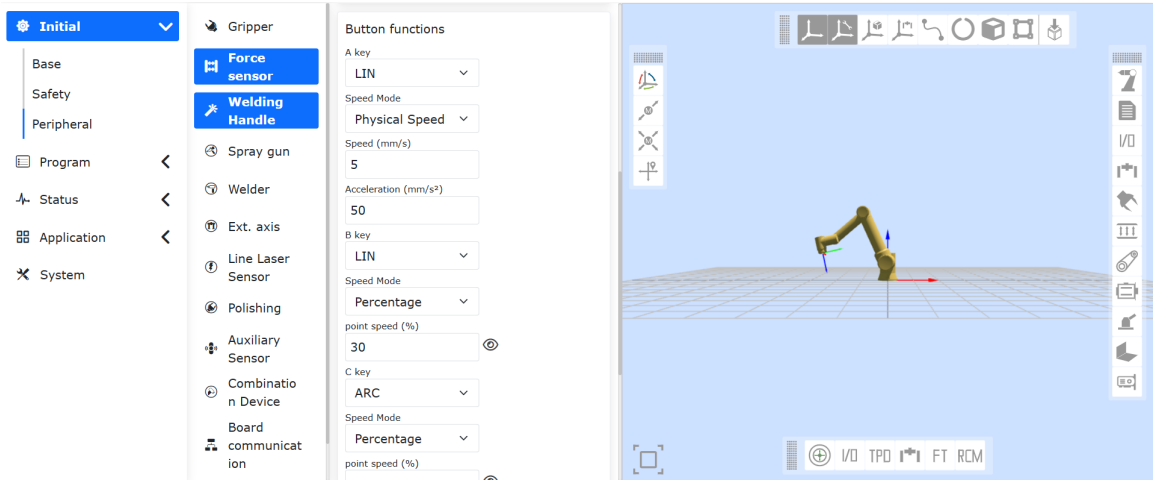


Figure 8.4-2-2 Input Actual Physical Speed

After successful configuration, a related motion command is added to the teaching program. When configuring the ARC motion command, you must first configure a PTP or LIN command.

- **DO Output:** When selecting "DO Output," a drop-down menu appears, allowing you to select DO0DO7 options.

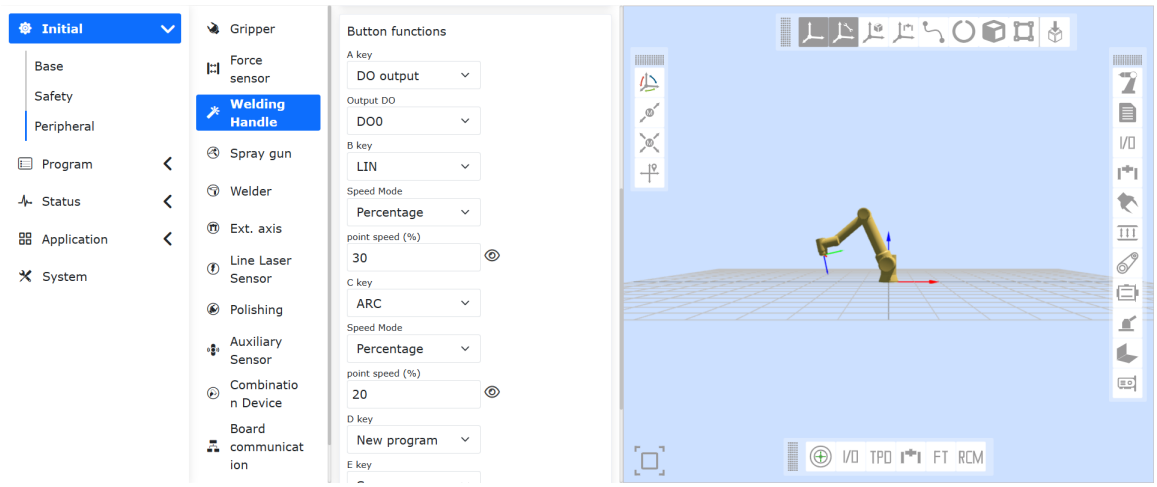


Figure 8.4-2-3 Smart Tool Configuration (A-E Keys)

IO Key Functions:

- **IO Signal Configuration**: The dropdown menu allows selection of DO0DO7 options, CO0CO7 options, End-DO0, End-DO1, and extended IO (Aux-DO0Aux-DO127);
- **Combined Command**: After selecting "IO Signal", under specific conditions, the "Welder Selection" and "Point Speed" configuration items are displayed, generating different program commands.

Important:

- When the IO signal is configured as DO0~DO7 or CO0~CO7 (without "Arc Start" configured), the program adds SetDO; "Welder Selection" and "Point Speed" are hidden at this time.
- When the IO signal is configured as End-DO0 or End-DO1, the program adds SetToolDO; "Welder Selection" and "Point Speed" are hidden at this time.
- When the IO signal is configured as extended IO (without "Welder Arc Start" configured), the program adds SetAuxDO; "Welder Selection" and "Point Speed" are hidden at this time.

- When the IO signal is configured as CO0~CO7 (with “Arc Start” configured) and “Welder Selection” is “None”, the program adds SetDO; “Welder Selection” and “Point Speed” are hidden at this time.
- When the IO signal configuration item is extended IO (with “Welder Arc Start” configured) and “Welder Selection” is “None”, the program adds SetAuxDO; “Welder Selection” and “Point Speed” are hidden at this time.
- When the IO signal is configured as CO0~CO7 (with “Arc Start” configured) or extended IO (with “Welder Arc Start” configured), and “Welder Selection” is “Welding”, the first press adds ARCStart, the second adds ARCEnd, the third adds ArcStart, the fourth adds ARCStart, alternating and repeating the above operations; “Welder Selection” and “Point Speed” are hidden at this time.
- When the IO signal is configured as CO0~CO7 (with “Arc Start” configured) or extended IO (with “Welder Arc Start” configured), and “Welder Selection” is “LIN+Welding”, the first press adds LIN and ARCStart, the second adds LIN and ARCEnd, the third adds LIN and ARCStart, the fourth adds LIN and ARCEnd, alternating and repeating the above operations; “Welder Selection” and “Point Speed” are displayed at this time.
- When the IO signal is configured as CO0~CO7 (with “Arc Start” configured) or extended IO (with “Welder Arc Start” configured), and “Welder Selection” is “LIN+Welding+Weaving”, the first press adds LIN, ARCStart, and WeaveStart, the second adds LIN, ARCEnd, and WeaveEnd, the third adds LIN, ARCStart, and WeaveStart, the fourth adds LIN, ARCEnd, and WeaveEnd, alternating and repeating the above operations; “Welder Selection” and “Point Speed” are hidden at this time.

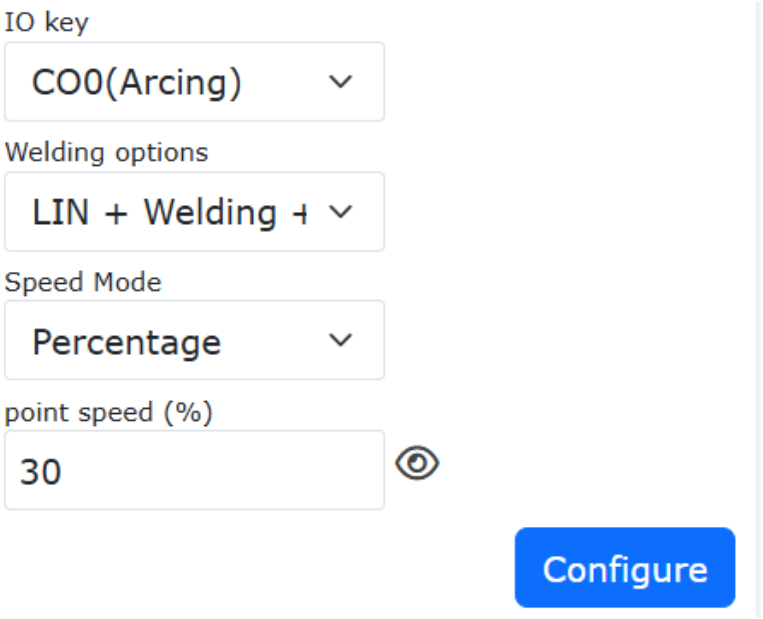


Figure 8.4-3 IO Key

1.3.8.4.2 Welding Handle End-Effector Lua Protocol

Click "Custom Protocol" to enter the welding handle function interface for adapting the end-effector Lua open protocol.

1.3.8.4.2.1 Protocol Management

Open the WebApp, click "Initial Setup", "Peripherals", "Welding Handle", "Custom Protocol" in sequence. Click "Protocol Management" to configure the end-effector protocol. Currently, the preset built-in protocols for the welding handle are shown in the figure below.

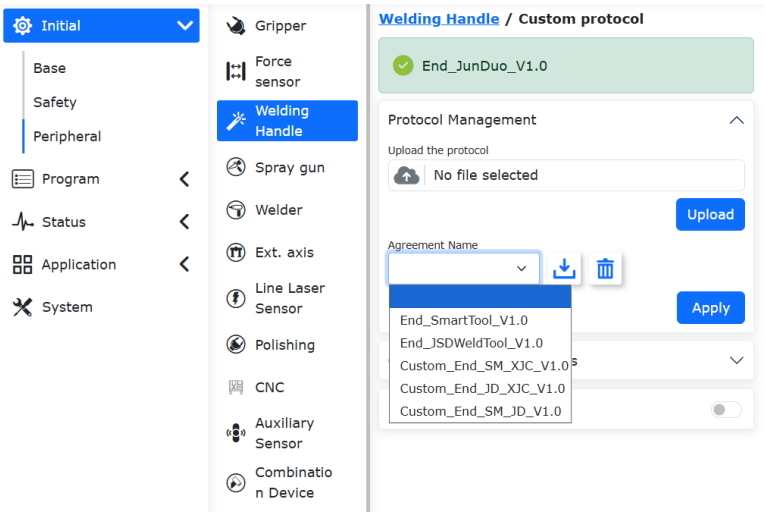


Figure 8.4-4 Welding Handle Preset Built-in Protocol

Turn on the "End-Effector Protocol Enable" slider to adapt the welding handle. The parameters are retained after a power restart once enabled.

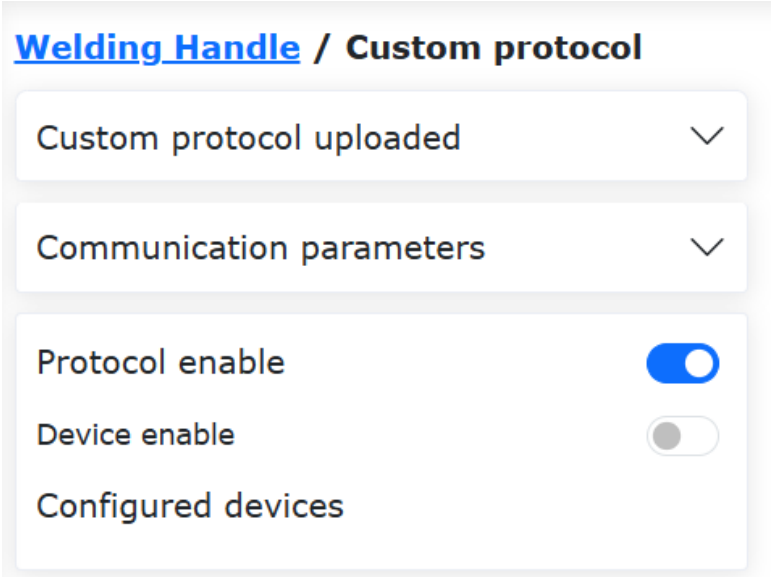


Figure 8.4-5 End-Effector Open Protocol Enable

1.3.8.4.2.2 Example of Combined Device Lua End-Effector Peripheral Protocol

The functions of the five buttons A, B, C, D, E can be modified and defined through the key values in line 30 of the code. Among them, *K38=Getbit(R[7],1)*, *K0=Getbit(R[7],2)* are for "Clear Program" and "Undo Button" and cannot be modified. The subsequent 5 K values can be modified according to the definitions in the "End-Effector Full Peripheral Protocol" document.

In this example (embedded SmartTool protocol), the corresponding button functions are: A: MoveL, B: ArcStart, C: ArcEnd, D: Rewelding start, E: Rewelding quit.

```
function Getbit(X,Bit)
return ((X&(1<<Bit))>>Bit)
end

if(Getbit(GetRobotState(),0)==1)then
local SetParams={A3=2000,B6=3}--Set welding parameters, A3-Arc start/end timeout is␣
↪2000ms, B6-Operate DO port number is 3. To configure welding parameters, please refer␣
↪to "RD36-Welding Handle Custom Parameter Table-V0.2-20250903"
SetWeldParams(SetParams)
while(1)
do
IwdgTaskHandle()
MainLoop()
UpDownLoadHandle()
SdoRwPara()
EndErrClear()
local BFlag=LuaBreak()
if(BFlag==1)then
break
end
local R={0}
local T={0x7D,0x01,0x30,0xC0,0x00,0x04,0x00,0x00,0x00,0x00}
DelayMs(100)
T[7],T[8],T[9],T[10]=GetIoCmd()
T[7]=Getbit(T[7],3)
T[12],T[11]=WeldToolCrcValue(T)
T[13]=0x0E
WeldToolSlaveSetCmd(T)
DelayMs(3)
Len=EndRxWeldData(R)
if((Len==13)and(R[1]==0x7D)and(R[2]==0x01)and(R[3]==0x30))then
local key={K38=Getbit(R[7],1),K0=Getbit(R[7],2),K3=Getbit(R[7],3),K32=Getbit(R[7],4),
↪K33=Getbit(R[7],5),K27=Getbit(R[7],6),K28=Getbit(R[7],7),
K6=Getbit(R[8],1),K7=Getbit(R[8],2)}--SmartTool welding handle button settings, Undo␣
↪button - K38 Undo program; Clear button - K0 Clear program; A button - K3 Linear Move;␣
↪B button - K32 ArcStart; C button - K33 ArcEnd; D button - K27 Rewelding start; E␣
↪button - K28 Rewelding quit; Manual/Auto button - K6 Manual/Auto; Run/Pause button -␣
↪K7 Run/Pause
SetWeldToolKeys(key)
end
LuaGc()
end
end
```

1.3.8.4.2.3 Open Protocol Template

Taking the Jiashida adaptation open protocol as an example:

```
function Getbit(X,Bit)                    -- Extract the corresponding bit of X
return ((X&(1<<Bit))>>Bit)
end
while(1)
do
IwdgTaskHandle()
MainLoop()
UpDownLoadHandle()
SdoRwPara()
EndErrClear()
local BFlag=LuaBreak()
if(BFlag==1)then
break
end
RxData={}
T0={0x7D,0x08,0x22,0xB3,0x01,0x00}
T1={0x7D,0x08,0x22,0xB4,0x03,0x00}
T2={0x7D,0X08,0X22,0XB5,0x1E,0x00}
DelayMs(5)
RxLen=WeldToolMasterGetCmd(RxData)                                          -- The␣
↪WeldToolMasterGetCmd() function is used to get commands sent by the welding pendant␣
↪(for when the welding pendant acts as the master station). Requires pushing an empty␣
↪table onto the stack (X={}) when used.
if (RxData[1]==0x7D)and(RxData[2]==0x08)and(RxData[3]==0x22) then
if(RxData[4] == 0xB3)then
   -- Taking Jiashida welding pendant function code as an example, here it is 0xB3 (Set␣
↪Welding Parameters).
local SetParams={A2=RxData[7],A1=RxData[8],A6=(ByteToDwFloat(RxData[9],RxData[10],
↪RxData[11],RxData[12]))*1000,
A8=(ByteToDwFloat(RxData[13],RxData[14],RxData[15],RxData[16])),
↪A7=(ByteToDwFloat(RxData[17],RxData[18],RxData[19],RxData[20])),
A4=(ByteToDwFloat(RxData[21],RxData[22],RxData[23],RxData[24]))*1000,
↪A5=(ByteToDwFloat(RxData[25],RxData[26],RxData[27],RxData[28]))*1000}
SetWeldParams(SetParams)                                                    -- The␣
↪SetWeldParams() function is used to set the controller's welding parameters. Need to␣
↪refer to the welding pendant custom parameter table to determine the welding␣
↪parameters that need to be modified (divided into 3 areas A, B, C in total).
Dword=GetRobotState()                                                       -- The␣
↪GetRobotState() function is used to get the robot's related status. Currently bit0 is␣
↪the robot enable status, bit1 is the robot fault status, bit2 is the robot moving␣
↪status, bit3 is the arc start/end command signal. Refer to End Full Peripherals␣
↪Protocol V2.7.
T0[7]=((Dword)&(1<<1))
T0[8],T0[9]=WeldToolCrcValue(T0)                                            --␣
↪WeldToolCrcValue() Faro custom protocol CRC check
T0[10]=0x0E
EndTxWeldData(T0)                                                           -- The␣
↪EndTxWeldData() function is used to send packaged data (here it is responding to the␣
↪welding pendant's set welding parameters command).
```

(continues on next page)

(continued from previous page)

```
DelayMs(5)
end
if(RxData[4] == 0xB4)then                                                    -- 0xB4 Real-
↪time Control Command
local key={K0=Getbit(RxData[7],0),K1=Getbit(RxData[7],1),K2=Getbit(RxData[7],2),
↪K3=Getbit(RxData[7],3),
K4=Getbit(RxData[7],4),K5=Getbit(RxData[7],5),K6=Getbit(RxData[7],6),K7=Getbit(RxData[7],
↪7),
K8=Getbit(RxData[8],0),K9=Getbit(RxData[8],1),K10=Getbit(RxData[8],2),
↪K11=Getbit(RxData[8],3),
K12=Getbit(RxData[8],4),K13=Getbit(RxData[8],5),K14=Getbit(RxData[8],6),
↪K15=Getbit(RxData[9],0),
K16=Getbit(RxData[9],1),K17=Getbit(RxData[9],2),K18=Getbit(RxData[9],3),
↪K19=Getbit(RxData[9],4),
K20=Getbit(RxData[9],5),K21=Getbit(RxData[9],6),K22=Getbit(RxData[9],7),
↪K23=Getbit(RxData[10],0),
K24=Getbit(RxData[10],1)}                                                    -- Key values␣
↪need to refer to End Full Peripherals Protocol V2.7 Table 26. K0-K31 correspond to␣
↪DWordInput10's bit0-bit31, K32-K63 correspond to DWordInput9's bit0-bit31.
SetWeldToolKeys(key)                                                         -- The␣
↪SetWeldToolKeys() function is used to upload the welding pendant button status. Can␣
↪adjust the key values filled in the table according to the actual situation of the␣
↪welding pendant.
Dword=GetRobotState()
T1[7]=(Dword)&(0x1)
T1[8]=(Dword>>1)&(0x1)
T1[9]=(Dword>>2)&(0x1)
T1[10],T1[11]=WeldToolCrcValue(T1)
T1[12]=0X0E
EndTxWeldData(T1)
DelayMs(5)
end
if(RxData[4] == 0xB5)then
-- Read Welding Parameters (Get from the controller, give to the welding pendant)
local wldpams={"A2","A1","A6","A8","A7","A4","A5"}
-- Fill according to the welding parameters actually needed by the welding pendant.␣
↪Jiashada needs these here, refer to End Full Peripherals Protocol V2.7 Table 26.
GetWeldParams(wldpams)                                                       --␣
↪GetWeldParams() gets the corresponding welding parameters and replaces their values in␣
↪the table (assuming A2=100, then after calling the function, wldpams[1]=100).
T2[7]=wldpams[1]
T2[8]=wldpams[2]
wldpams[3]=wldpams[3]/1000
wldpams[6]=wldpams[6]/1000
wldpams[7]=wldpams[7]/1000
for i=0,4 do
T2[9+(i*4)+3],T2[9+(i*4)+2],T2[9+(i*4)+1],T2[9+(i*4)+0]=DwFloatToByte(wldpams[3+i])
end
for i=0,7 do
T2[29+i]=0
end
T2[37],T2[38]=WeldToolCrcValue(T2)
```

(continues on next page)

(continued from previous page)

```
T2[39]=0x0E
EndTxWeldData(T2)
DelayMs(5)
end
end
LuaGc()
end
```

1.3.8.4.2.4 Instructions Supported by Open Protocol

The following instructions can be configured in the open protocol, while bits 39-63 are reserved for future expansion.

Table 8.4-1 Instructions Supported by Open Protocol

Bit	Description
0	Clear Program
1	Save Program
2	Generate Safe Point (LIN Command)
3	Generate Linear Motion Point (LIN Command)
4	Add Arc Transition Point
5	Add Arc End Point and Generate ARC Command
6	Switch Mode, default is Manual Mode
7	Toggle Robot Run Status
8	Toggle Robot Drag Status
9	Start Spot Weld
10	Add Start Weave Arc Command
11	Add End Weave Arc Command
12	Jog in Positive X Direction
13	Jog in Negative X Direction
14	Jog in Positive Y Direction
15	Jog in Negative Y Direction
16	Jog in Positive Z Direction
17	Jog in Negative Z Direction
18	Jog in Positive RX Direction
19	Jog in Negative RX Direction
20	Jog in Positive RY Direction
21	Jog in Negative RY Direction
22	Jog in Positive RZ Direction
23	Jog in Negative RZ Direction
24	Generate Start Point
25	PTP
26	Fixed Pose Drag
27	Weld Interruption Recovery
28	Weld Interruption Exit
29	SetDO
30	Offline
31	Configuration Parameter Update
32	ArcStart
33	ArcEnd
34	Lin+ArcStart+WeaveStart

continues on next page

Table 3 – continued from previous page

35	Lin+ArcEnd+WeaveEnd
36	Lin+ArcStart
37	Lin+ArcEnd
38	Undo Program
39	Reserved
…	Reserved
63	Reserved

1.3.8.4.2.5 Configurable Parameters in Open Protocol

The following parameters can be configured in the open protocol.

Table 8.4-2 Configurable Parameters in Open Protocol

Index	Data Content	Data Type	Range
0	Welding Speed	float	0-100%
1	Air Move Speed	float	0-100%
2	Arc Start/End Timeout	float	0-65535(ms)
3	Weave Left Dwell Time	float	0-99999 (ms)
4	Weave Right Dwell Time	float	0-99999 (ms)
5	Spot Weld Time	float	0-99999 (ms)
6	Weave Width	float	0-1000 (0.1mm)
7	Weave Frequency	float	0-100(0.1Hz)
8	Welder Control Type; 0-Control Box IO; 1-Digital Communication Protocol (UDP)	float	0-255
9	Welding Process Number (0-99)	float	0-99
10	Weave Type	float	0-255
11	Current Control Output Analog Output Port	float	0-1
12	Voltage Control Output Analog Output Port	float	0-1
13	Operation DO Port Number	float	0-15
14	Weave Parameter Number	float	0-255
15	Manual Mode Global Speed	float	0-100%
16	Auto Mode Global Speed	float	0-100%
17	Welding Current	float	0-999990 (0.1A)
18	Welding Voltage	float	0-999990 (0.1V)
19	Single Jog Maximum Distance	float	0-1000 (0.1mm)
20	Welder Ready Extended DI Port	float	0-127
21	Arc Success Extended DI Port	float	0-127
22	Weld Interruption Recovery Extended DI Port	float	0-127
23	Weld Interruption Exit Extended DI Port	float	0-127
24	Welder Arc Start Extended DO Port	float	0-127
25	Gas Detection Extended D0 Port	float	0-127
26	Forward Wire Feed Extended D0 Port	float	0-127
27	Reverse Wire Feed Extended D0 Port	float	0-127
28	Weld Interruption Recovery Enable	float	0-1
29	Go to Recovery Point Speed	float	0-100%
30	Motion Mode	float	0-1
31	Welding Arc Interruption Detection Enable	float	0-1
32	Whether to Include Wait Time(ms)	float	0-1

continues on next page

Table 4 – continued from previous page

33	Weave Callback Ratio	float	0-100%
34	Weave Position Wait Type	float	0-255
35	Arc Start Time	float	0-65535 (ms)
36	Arc End Time	float	0-65535 (ms)
37	Welding Arc Interruption Confirmation Duration	float	0-65535 (ms)
38	Overlap Distance	float	0-1000(0.1mm)
39	Arc Start Current	float	0-999990(0.1A)
40	Arc Start Voltage	float	0-999990(0.1V)
41	Arc End Current	float	0-999990(0.1A)
42	Arc End Voltage	float	0-999990(0.1V)
43	Minimum Welding Current	float	0-999990(0.1A)
44	Maximum Welding Current	float	0-999990(0.1A)
45	Analog Output Corresponding to Minimum Welding Current	float	0-100(0.1A)
46	Analog Output Corresponding to Maximum Welding Current	float	0-100(0.1A)
47	Minimum Welding Voltage	float	0-2000(0.1V)
48	Maximum Welding Voltage	float	0-2000(0.1V)
49	Analog Output Corresponding to Minimum Welding Voltage	float	0-100(0.1V)
50	Analog Output Corresponding to Maximum Welding Voltage	float	0-100(0.1V)
51	Vertical Triangle Weave Left Chord Length	float	0-1000(0.1mm)
52	Vertical Triangle Weave Right Chord Length	float	0-1000(0.1mm)
53	Weave Direction Azimuth Angle	float	-1800-1800(0.1°)
54	Weave Direction Tilt Angle	float	-1800-1800(0.1°)
55	Vertical Triangle Weave Apex Wait Time	float	0-99999(ms)

1.3.8.5 Spray Gun

1.3.8.5.1 Spray Gun Peripheral Configuration Steps

Step1: In the “Initial” -> “Peripherals” menu bar, click “Spray Gun” to enter the spray gun configuration interface.

The spraying function allows one-click configuration of keys for quick setup of the required DOs for spraying (by default, DO10 is configured for spray start/stop, DO11 for gun cleaning).

Users can also customize DO configuration according to their needs in “Initial” -> “Basic” -> “I/O Settings”.

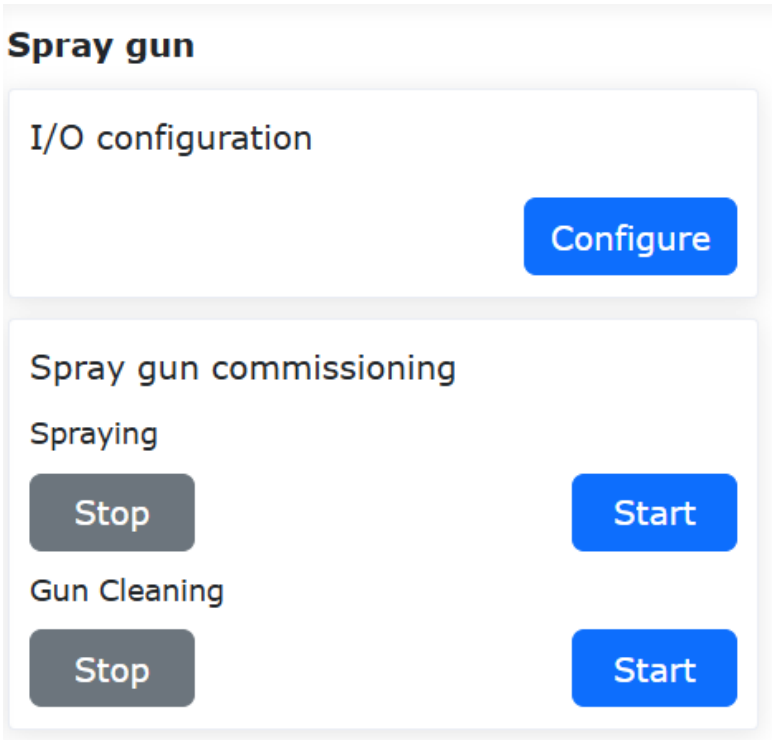
Important: Before using the spraying function, the corresponding tool coordinate system must be established and applied during program teaching.

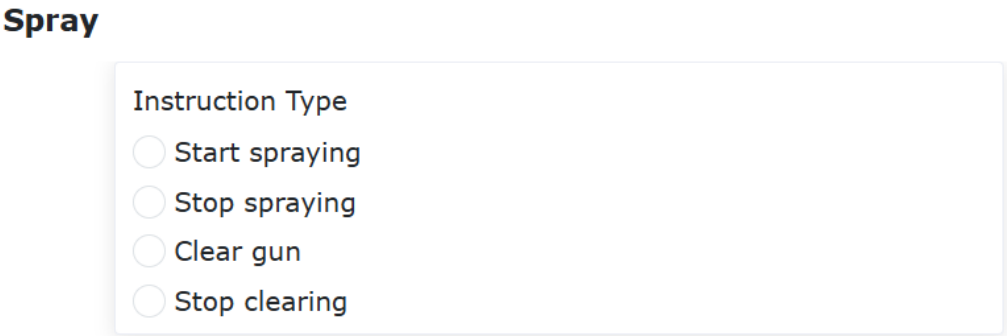
Step2: After configuration is complete, click the four buttons “Start Spraying”, “Stop Spraying”, “Start Cleaning”, and “Stop Cleaning” to debug the spray gun.

Figure 8.5-1 Spray Gun Configuration

Step3: Select the “Spray Gun” command in the program command interface. According to the specific program teaching requirements, add the four instructions “Start Spraying”, “Stop Spraying”, “Start Cleaning”, and “Stop Cleaning” at the appropriate locations.

Figure 8.5-2 Spray Gun Commands


Spray gun
I/O configuration
Configure
Spray gun commissioning
Spraying
Stop
Start
Gun Cleaning
Stop
Start


Spray
Instruction Type
Start spraying
Stop spraying
Clear gun
Stop clearing

1.3.8.5.2 Spraying Program Teaching

No.	Command Format	Comment
1	Lin(template1,100,-1,0,0)	#Start spraying point
2	SprayStart()	#Start spraying
3	Lin(template2,100,-1,0,0)	#Spraying path
4	Lin(template3,100,-1,0,0)	#Stop spraying point
5	SprayStop()	#Stop spraying
6	Lin(template4,100,-1,0,0)	#Cleaning point
7	PowerCleanStart()	#Start cleaning
8	WaitTime(5000)	#Cleaning time ms
9	PowerCleanStop()	#Stop cleaning

1.3.8.6 Welding Machine

Collaborative robots carrying welding torches for welding operations can significantly improve welding efficiency and quality. Faro collaborative robots can control welding through three methods: “Controller IO”, “Digital Communication Protocol (UDP)”, or “Digital Communication Protocol (Modbus TCP)”:

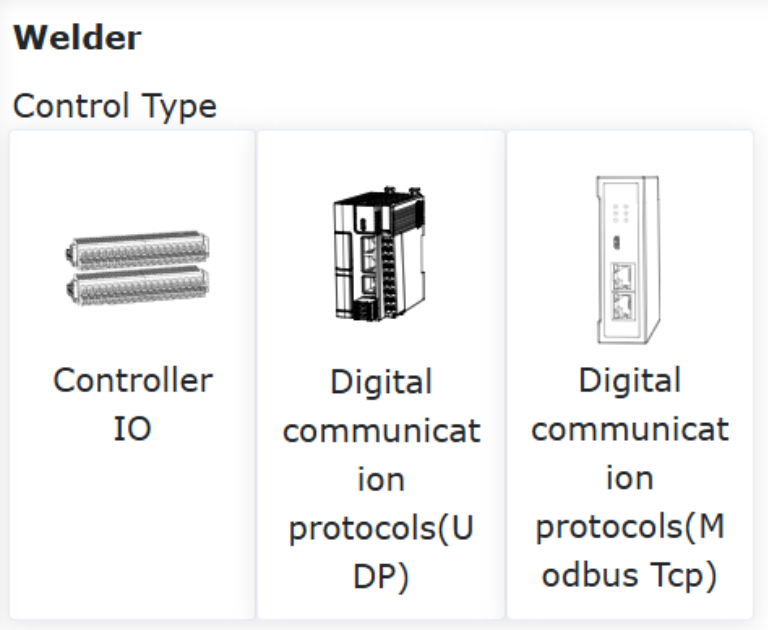
Controller IO: The robot controls the welding current and voltage by setting the control box analog output (0-10V), controls welding arc initiation, wire feeding, and gas supply through the control box digital output, and collects signals such as welder ready and arc success through the control box digital input.

Digital Communication Protocol (UDP): The robot communicates with a PLC via UDP, and the PLC then communicates with the welding machine via CANOpen bus or other protocols to control welding voltage, current, and operations like arc initiation, wire feeding, and gas supply (Refer to Appendix 1 for the robot UDP communication protocol content).

Digital Communication Protocol (Modbus TCP): This refers to the controller peripheral open protocol, typically a runnable LUA program. The program includes commands for creating communication, cyclically writing control data to the slave device, and reading real-time status data. When this LUA program is executed, the robot establishes communication with the device and performs data exchange. The IP address, port number, cycle, and other communication parameters can be customized in the controller peripheral open protocol LUA program. Users need to modify the protocol content according to the actual device situation when using it. Devices supported by the controller peripheral open protocol include grinding heads, laser sensors, CNC, welding machines, etc. The controller peripheral open protocol file name must start with *CtrlDev_*, such as “CtrlDev_Welding.lua”. A maximum of 4 open protocols can run simultaneously.

Figure 8.6-1 Welding Machine

Welding control via “Controller IO” or “Digital Communication Protocol (UDP)” mainly includes the following steps: Welding torch installation and signal wiring; Welding machine parameter configuration; Writing welding control programs.



1.3.8.6.1 Welding Torch Installation

The welding torch is installed at the robot end via an adapter plate, and the welding torch cable must be fixed to the robotic arm.

Figure 8.6-2 Welding Torch Installed at Robot End

After the welding torch is fixed and installed, calibrate the welding torch tool coordinate system using the six-point method and apply it as the current tool coordinate system. The calibration accuracy of the welding torch tool coordinate system will affect the actual welding accuracy.

Figure 8.6-3 Robot Tool Coordinate System Calibration and Application

1.3.8.6.2 Welding Machine Parameter Configuration

Collaborative robots can control the welding process through “Controller IO” signals or “Digital Communication Protocol”. The configuration operations for these two methods mainly differ in the following two points:

When using “Controller IO”, it is necessary to set the correspondence between the actual controlled welding current/voltage and the control box analog output values.

When using “Digital Communication Protocol”, communication parameters need to be configured.

1.3.8.6.2.1 “Controller IO” Welding Control Configuration

In the “Initial” -> “Peripherals” -> “Welding Machine” menu bar, click the “Controller I/O” card to enter the interface.

Figure 8.6-4 Controller I/O



1.3.8.6.2.2 Welding IO Signal Configuration

As shown in the figure below, select the DI input ports for the welder status signals and the DO output ports for the welder control signals, then click the “Configure” button. The meanings of each signal are as follows:

Figure 8.6-5 Set Welding Machine Signal Ports

Welder Ready: This signal is output from the welding machine to the robot when the welding machine is ready to perform welding operations.

When the welding machine is not ready due to faults or other reasons, this signal is not input to the robot. At this time, the robot WebApp prompt in the upper right corner “Welder not ready”. If your welding machine does not have a welder ready signal, you can set this port to “None”.

Figure 8.6-6 Welder Not Ready Error

Figure 8.6-7 Welder Ready Set to “None”

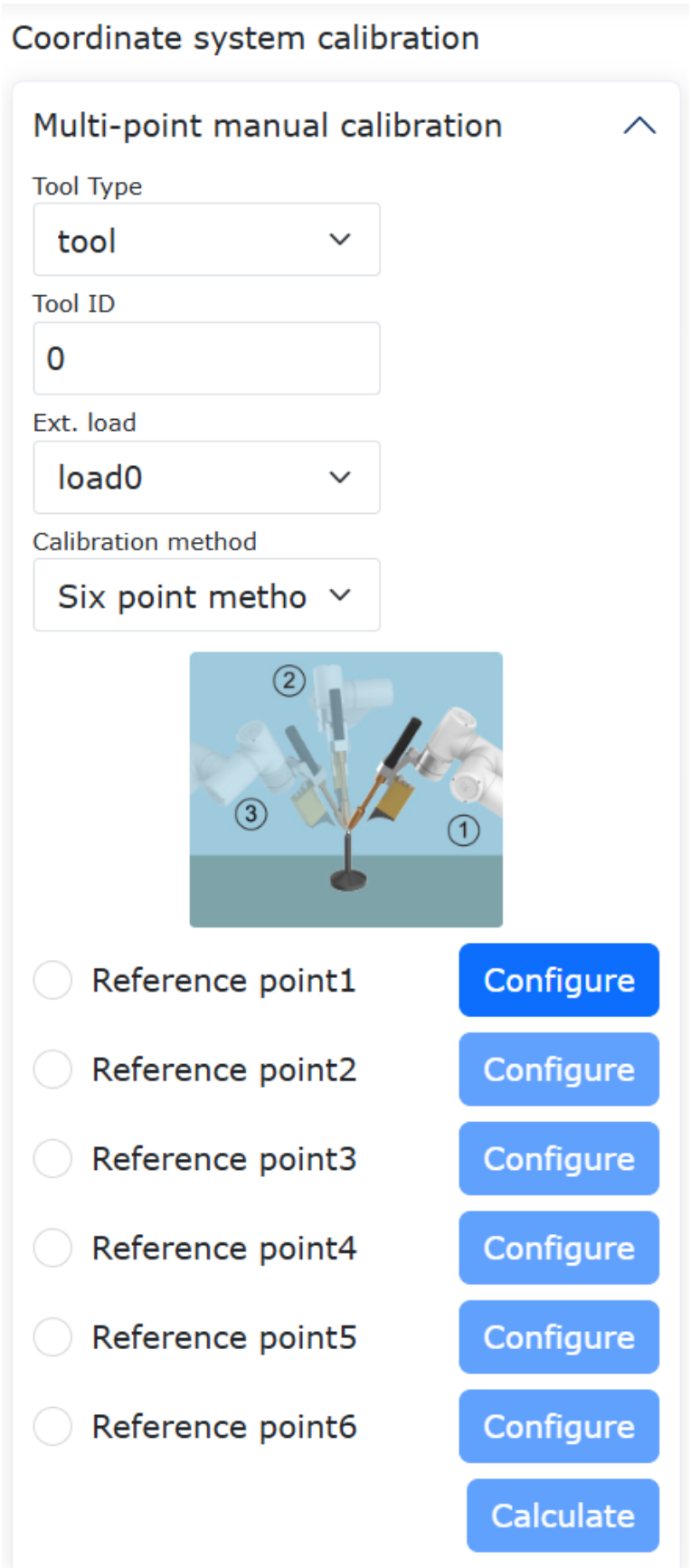
Arc Success: The arc has been successfully initiated. After the robot outputs the arc initiation signal to the welding machine, it waits for the arc success feedback signal from the welding machine. If the robot does not detect the arc success signal from the welding machine within the set timeout period, the robot reports an “Arc initiation timeout” error.

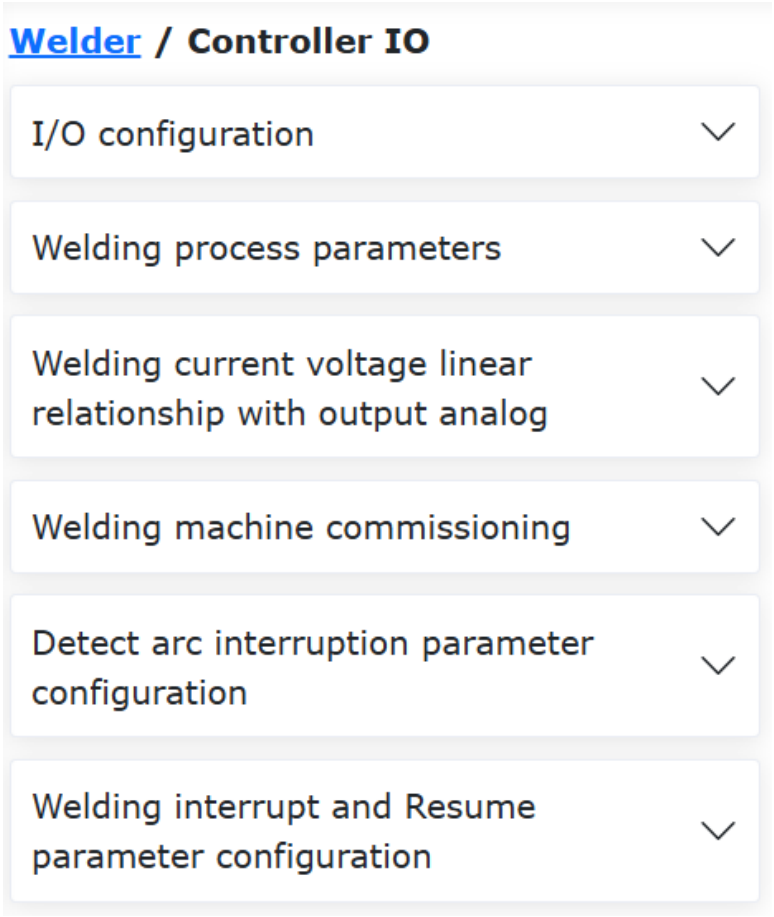
If the arc success signal is not configured, welding can still be performed using the robot welding function, but the robot will report a “Arc success DI not configured” warning. If your welding machine has an arc success signal output, we recommend configuring this signal for safer welding.

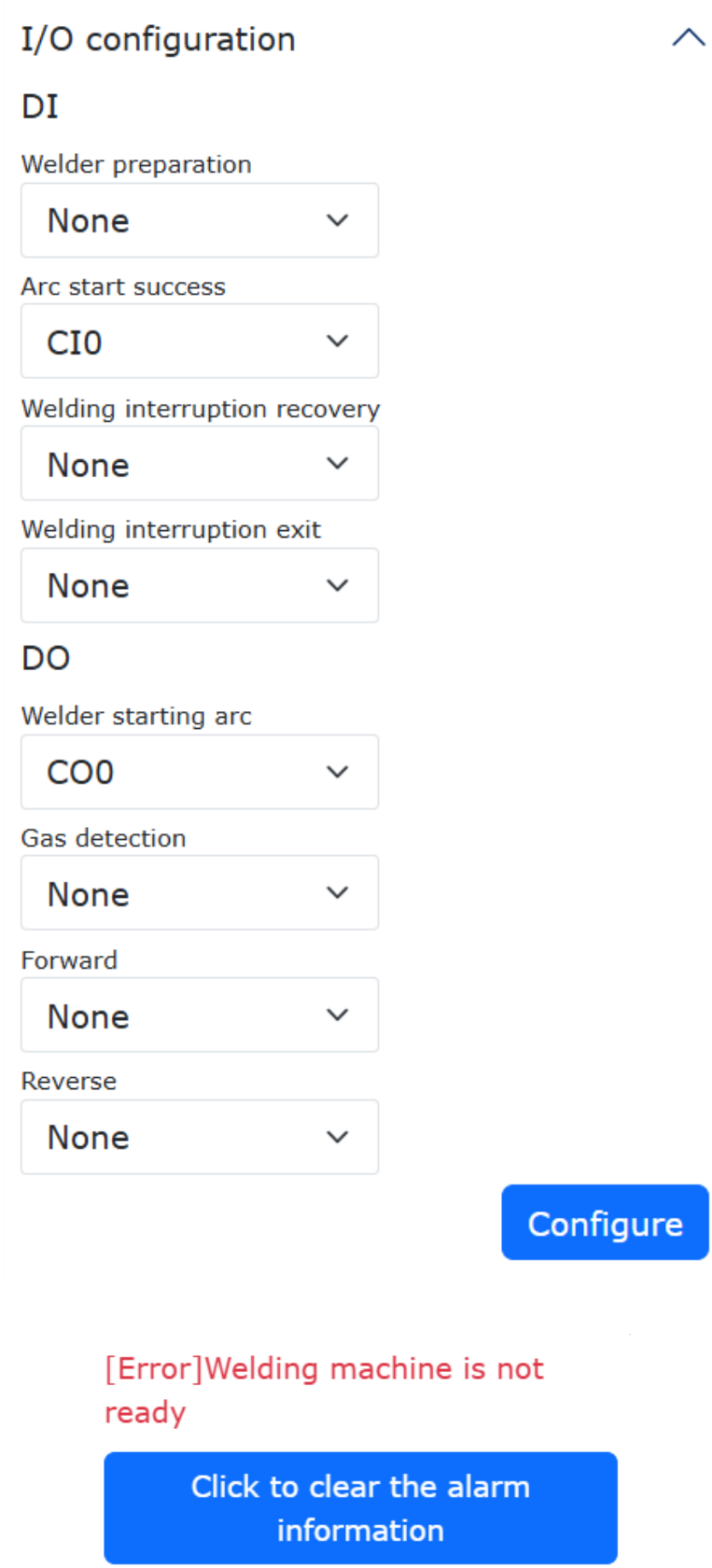
Figure 8.6-8 Arc Initiation Timeout Error

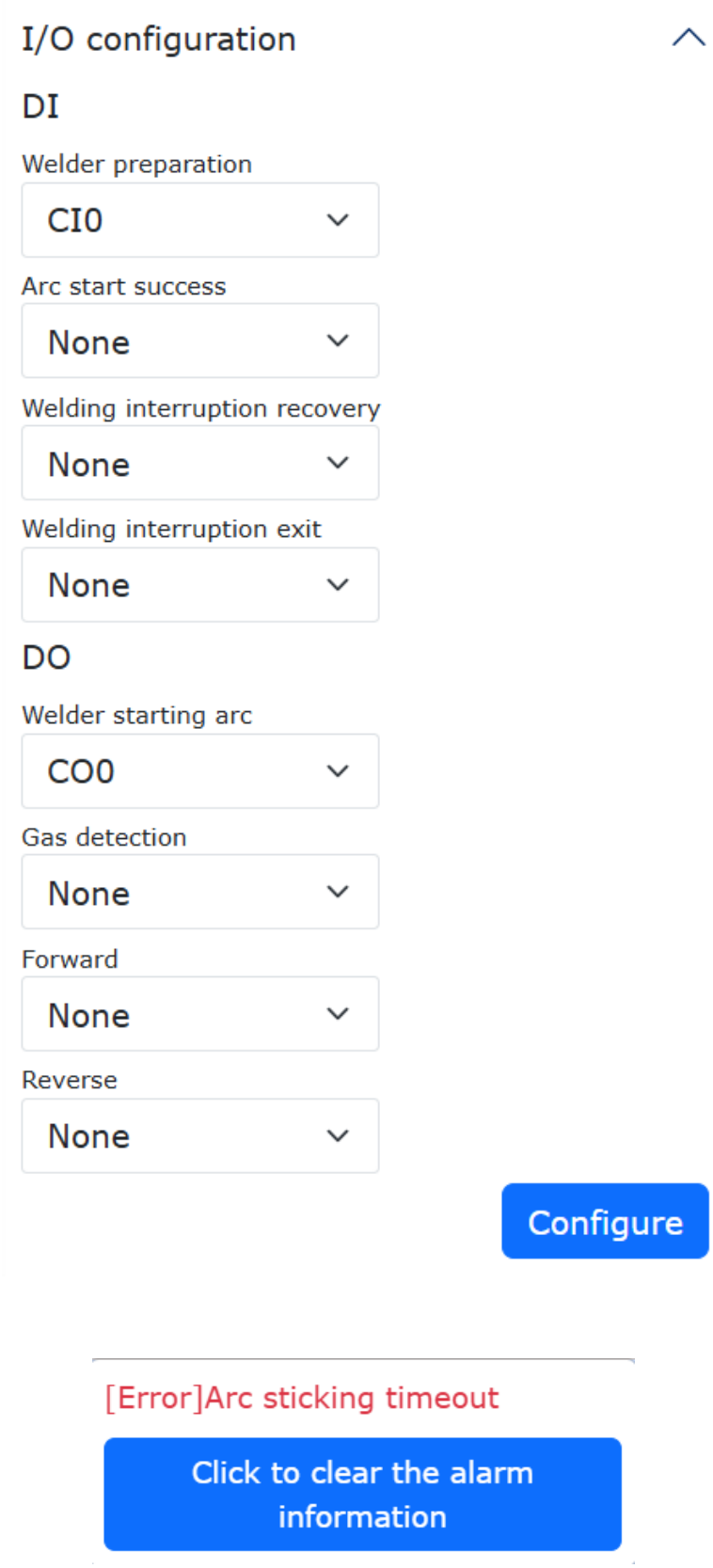
Figure 8.6-9 Arc Success DI Not Configured Warning

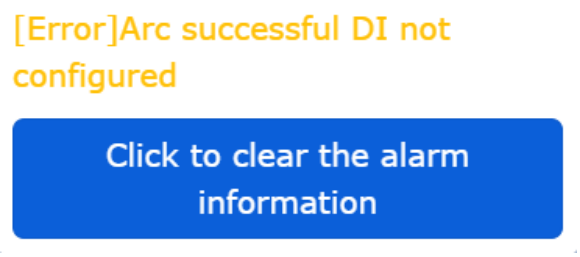
Welding Interruption Recovery: Triggered when the arc is unexpectedly interrupted during robot welding or the operator actively pauses welding. When this signal input to the robot changes from invalid to valid after a welding interruption, the robot automatically resumes welding from the original interruption position.


Coordinate system calibration
Multi-point manual calibration
Tool Type
tool
Tool ID
0
Ext. load
load0
Calibration method
Six point metho
2
3
1
Reference point1
Configure
Reference point2
Configure
Reference point3
Configure
Reference point4
Configure
Reference point5
Configure
Reference point6
Configure
Calculate


Welder / Controller IO
I/O configuration
Welding process parameters
Welding current voltage linear relationship with output analog
Welding machine commissioning
Detect arc interruption parameter configuration
Welding interrupt and Resume parameter configuration


I/O configuration
DI
Welder preparation
None
Arc start success
CI0
Welding interruption recovery
None
Welding interruption exit
None
DO
Welder starting arc
CO0
Gas detection
None
Forward
None
Reverse
None
Configure
[Error]Welding machine is not ready
Click to clear the alarm information


I/O configuration
DI
Welder preparation
CI0
Arc start success
None
Welding interruption recovery
None
Welding interruption exit
None
DO
Welder starting arc
CO0
Gas detection
None
Forward
None
Reverse
None
Configure
[Error]Arc sticking timeout
Click to clear the alarm information



Welding Interruption Exit: Triggered when the arc is unexpectedly interrupted during robot welding or the operator actively pauses welding. When this signal input to the robot changes from invalid to valid after a welding interruption, the robot terminates welding. After termination, welding cannot be resumed.

Welding Arc Initiation: The DO output port through which the robot controls the welding machine to initiate the arc. When the robot program executes the arc initiation command, the corresponding DO output port for arc initiation automatically outputs a valid signal.

Gas Detection: The DO output port through which the robot controls the welding machine to supply gas. When the robot executes the welding gas supply command, the corresponding DO output port for gas supply automatically outputs a valid signal.

Forward Wire Feed: The DO output port through which the robot controls the welding machine for forward wire feeding. When the robot executes the forward wire feed command, the corresponding DO output port for forward wire feed automatically outputs a valid signal.

Reverse Wire Feed: The DO output port through which the robot controls the welding machine for reverse wire feeding. When the robot executes the reverse wire feed command, the corresponding DO output port for reverse wire feed automatically outputs a valid signal.

1.3.8.6.2.3 Welding Process Parameter Configuration

As shown in the figure below, find the "Welding Process Parameters" section on the welding configuration page. The collaborative robot provides 100 sets of welding process parameters, numbered 0 to 99. Process number 0 indicates not using the welding process curve, while process numbers 1-99 use the welding process curve.

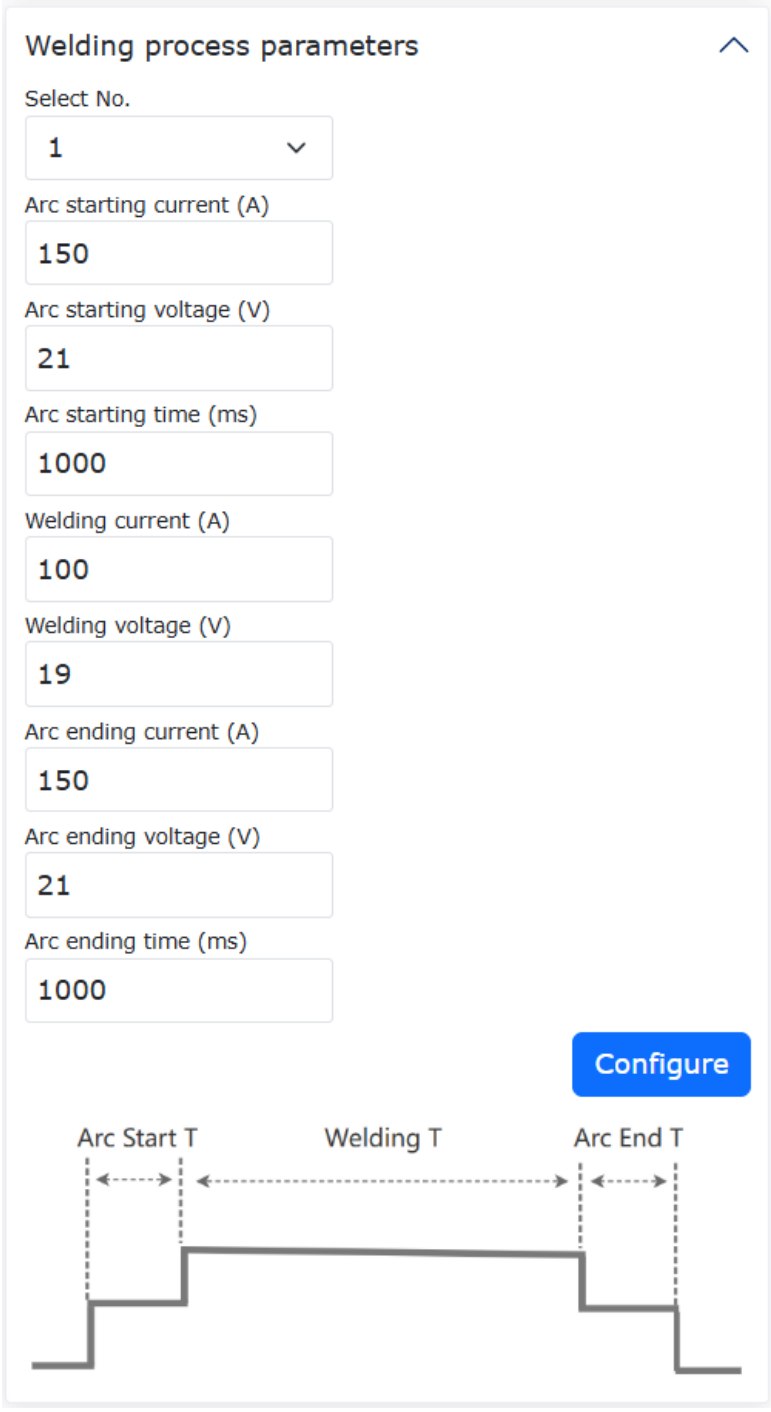
Figure 8.6-10 Welding Process Parameter Configuration

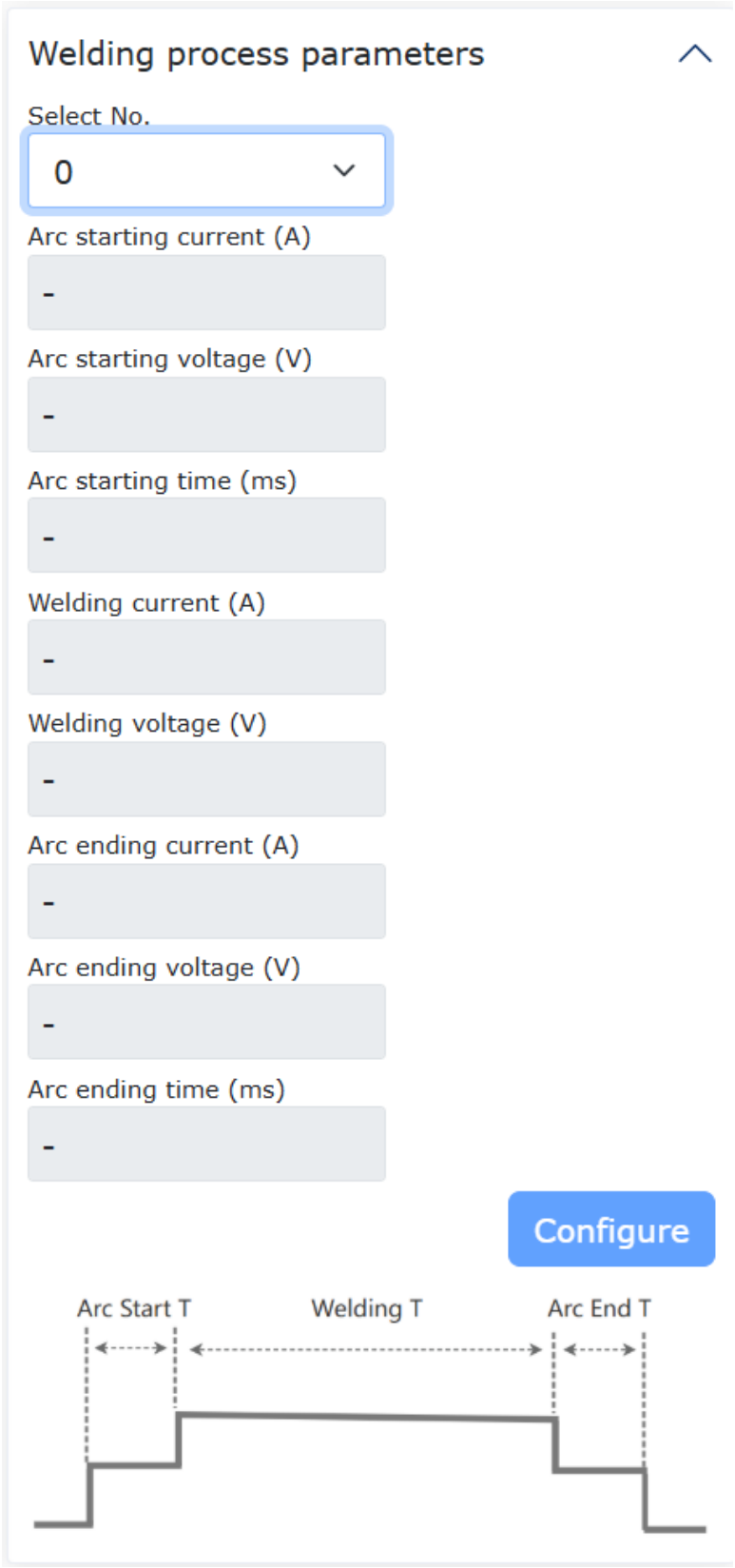
When using the welding process curve, take selecting welding process number 1 as an example. Input the parameters from Arc Initiation Current to Arc Closing Time as shown in Figure 8, then click the "Configure" button. The actual welding process represented by these parameters is as follows:

Set welding current 200A, voltage 23V; Execute arc initiation, wait for arc success; After arc success, maintain the arc for 500ms (Arc initiation time, robot does not move); Set welding current 150A, welding voltage 21V, then the robot starts moving and performs welding; After welding to the end point, set welding current to 100A, welding voltage to 19V (Arc closing current, Arc closing voltage); After setting the arc closing current and voltage, maintain arc burning for 500ms (robot does not move), finally extinguish the arc.

When not using the welding process curve, i.e., selecting welding process parameter number 0, as shown below, the welding process is: Set welding current and welding voltage; The robot controls the welding machine to initiate the arc and waits for arc success; After arc success, the robot starts moving and performs welding; The robot extinguishes the arc immediately after welding to the end point.

Figure 8.6-11 Not Using Welding Process Curve


Welding process parameters
Select No.
1
Arc starting current (A)
150
Arc starting voltage (V)
21
Arc starting time (ms)
1000
Welding current (A)
100
Welding voltage (V)
19
Arc ending current (A)
150
Arc ending voltage (V)
21
Arc ending time (ms)
1000
Configure
Arc Start T
Welding T
Arc End T


Welding process parameters
Select No.
0
Arc starting current (A)
-
Arc starting voltage (V)
-
Arc starting time (ms)
-
Welding current (A)
-
Welding voltage (V)
-
Arc ending current (A)
-
Arc ending voltage (V)
-
Arc ending time (ms)
-
Configure
Arc Start T
Welding T
Arc End T

1.3.8.6.2.4 Setting the Relationship Diagram Between Welding Current/Voltage and Analog Output

When the collaborative robot welding control type is selected as "Controller IO", the welding current and voltage values are controlled by the magnitude of the control box analog output (control box analog output voltage range is 0 ~ 10V). At this time, it is necessary to configure the linear correspondence between the control box analog output value and the actual welding current and welding voltage values.

As shown in Figure 12, find the "Analog Current Voltage Relationship Diagram" on the welding machine configuration page. "A-V" represents the correspondence between welding current and the control box output analog voltage, and "V-V" represents the correspondence between welding voltage and the control box output analog voltage.

Select "A-V", input the welding current range 0-1000A, analog output voltage 0-10V, output AO as "Ctrl-AO0" (the analog output port for welding current control is AO0), click the "Configure" button; Under these parameters, when the control box outputs an analog voltage of 1.5V, it corresponds to a welding current of 150A.

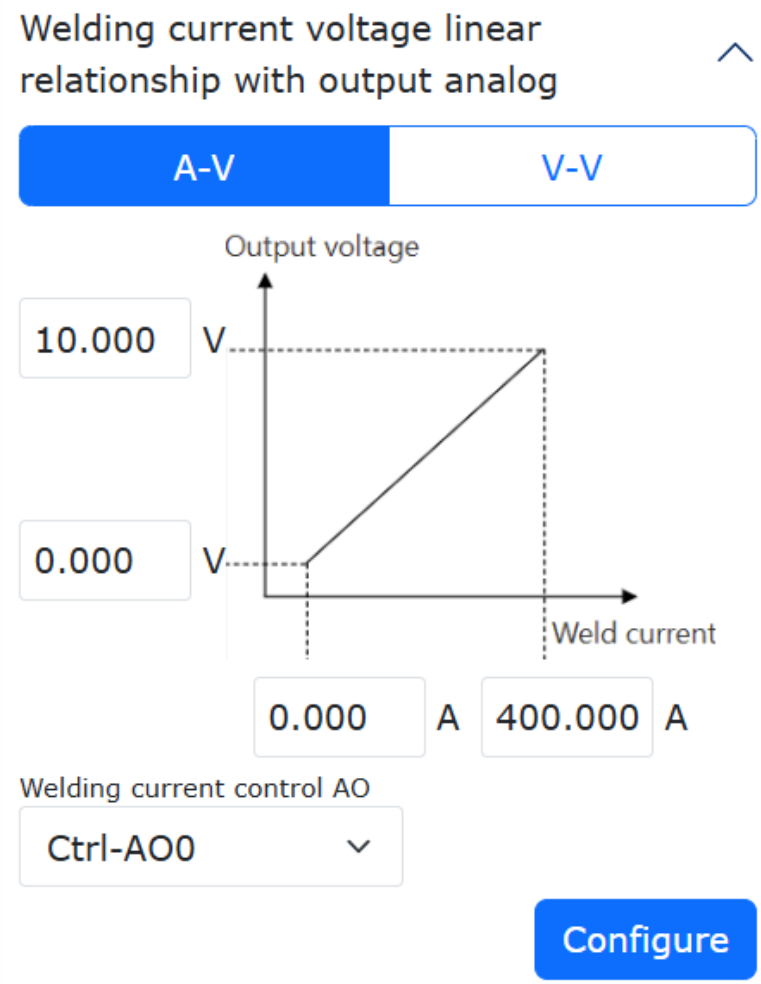
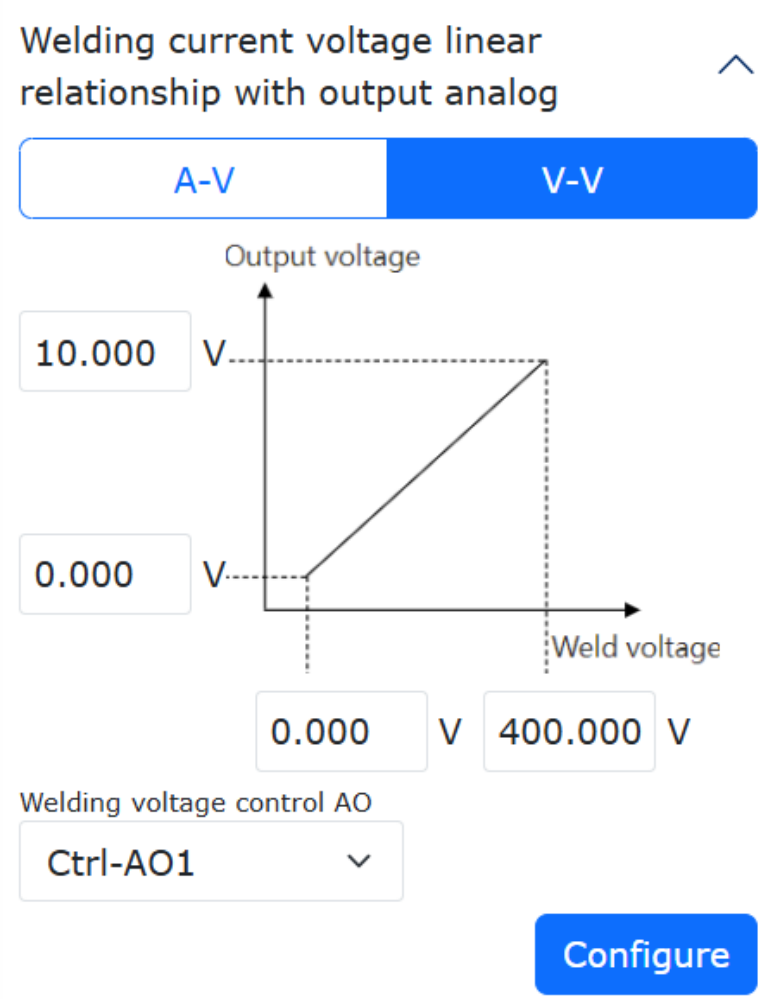


Figure 8.6-12 Welding Current vs Output Analog Correspondence Configuration

As shown in Figure 13, click "V-V" to set the correspondence between welding voltage and the control box analog output voltage. Input the welding voltage range as 0-60V, the analog output voltage value as 0-10V, and the output AO as "Ctrl-AO1" (the analog output port for welding voltage control is AO1), then click the "Configure" button. In this case, if the control box AO1 analog output is 3.5V, it actually controls the welding voltage to be 21V.

Figure 8.6-13 Welding Voltage vs Output Analog Correspondence Configuration


Welding current voltage linear relationship with output analog
A-V
V-V
Output voltage
10.000
V
0.000
V
Weld voltage
0.000
V
400.000
V
Welding voltage control AO
Ctrl-AO1
Configure

1.3.8.6.2.5 Welding Machine Debugging

As shown in Figure 14, find "Welding Machine Debugging" on the welding machine configuration page. Select process number 1, input the timeout time as 1000ms, click "Gas On", and the robot will control the welding machine to start supplying shielding gas. Click the "Gas Off" button, and the robot will control the welding machine to stop supplying shielding gas. The operation methods for other buttons like "Arc Start", "Forward Wire Feed", "Reverse Wire Feed", etc., are the same and will not be repeated.

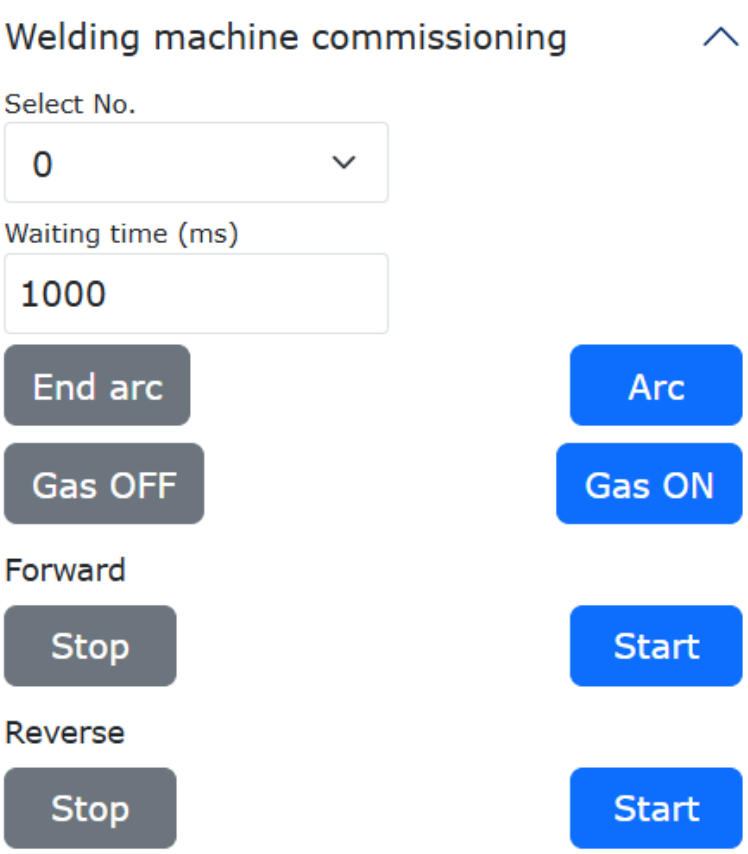


Figure 8.6-14 Welding Machine Debugging

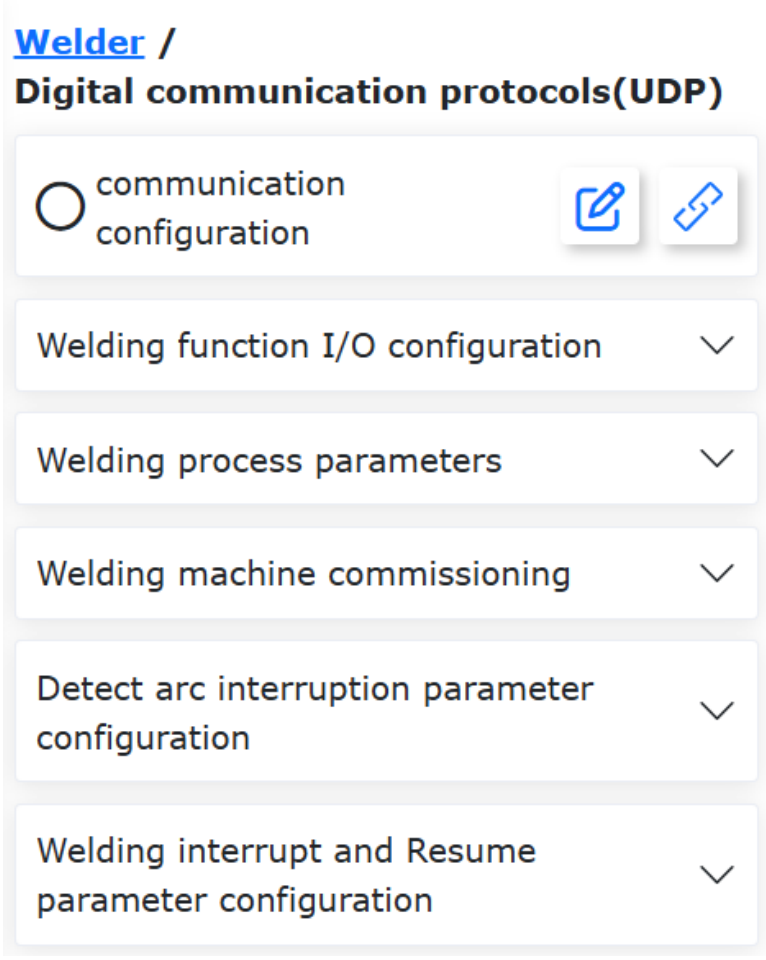
1.3.8.6.2.6 "Digital Communication Protocol (UDP)" Welding Control Configuration

When the robot uses the "Digital Communication Protocol" for welding control, it essentially communicates with a PLC via UDP. The robot transmits control data such as arc initiation, wire feeding, gas supply, current, and voltage to the PLC via UDP communication. The PLC then further controls the welding machine via CANOpen bus (or other methods). Simultaneously, the PLC collects actual welding current, voltage, and arc success signals and feeds them back to the robot. (Refer to Appendix 1 for the robot UDP communication protocol content).

In the "Initial" -> "Peripherals" menu bar, click "Welding Machine" to enter the welding machine configuration interface. As shown below:

Figure 8.6-15 Digital Communication Protocol (UDP)

Since the robot communicates with the PLC via UDP, UDP communication parameters need to be configured. The meanings of these parameters are as follows:


Welder /
Digital communication protocols(UDP)
communication configuration
Welding function I/O configuration
Welding process parameters
Welding machine commissioning
Detect arc interruption parameter configuration
Welding interrupt and Resume parameter configuration

IP Address: The IP address of the PLC end for UDP communication. **Port Number**: The UDP communication port number on the PLC end. **Communication Cycle**: The cycle for UDP communication between the robot and the PLC, default is 2ms. **Packet Loss Detection Cycle, Packet Loss Count**: When the number of lost packets within the packet loss detection cycle exceeds the set value, the robot reports a "UDP communication packet loss exception" error and automatically cuts off the communication. **Communication Interruption Confirmation Duration**: If the robot does not receive a complete frame of PLC feedback data within this duration, it reports a "UDP communication interruption" error alarm and cuts off the UDP communication. **Auto-reconnect after Power Restart**: Whether the robot automatically attempts to reconnect and recover after detecting a robot power restart. **Auto-reconnect after Communication Interruption**: Whether the robot automatically attempts to reconnect and recover after detecting a UDP communication interruption. **Reconnection Cycle, Reconnection Count**: When auto-reconnect after UDP communication interruption is enabled and a interruption is detected, the robot attempts reconnection at the set cycle. If the reconnection count reaches the maximum set value and the connection is still not successful, the robot reports a "UDP communication interruption" error alarm and cuts off the UDP communication.

After configuring the above parameters, click the "Configure" button. After successful configuration, click the "Load" button.

Figure 8.6-16 UDP Communication Configuration



Note:

Name: **Edit Button**

Function: Open/Close UDP communication parameter configuration

---



Note:

Name: **Load Button**

Function: Load UDP communication

1.3.8.6.2.7 Welding IO Signal Configuration

Select the DI input ports for the welder status signals and the DO output ports for the welder control signals, then click the "Configure" button. The meanings of each signal are as follows:

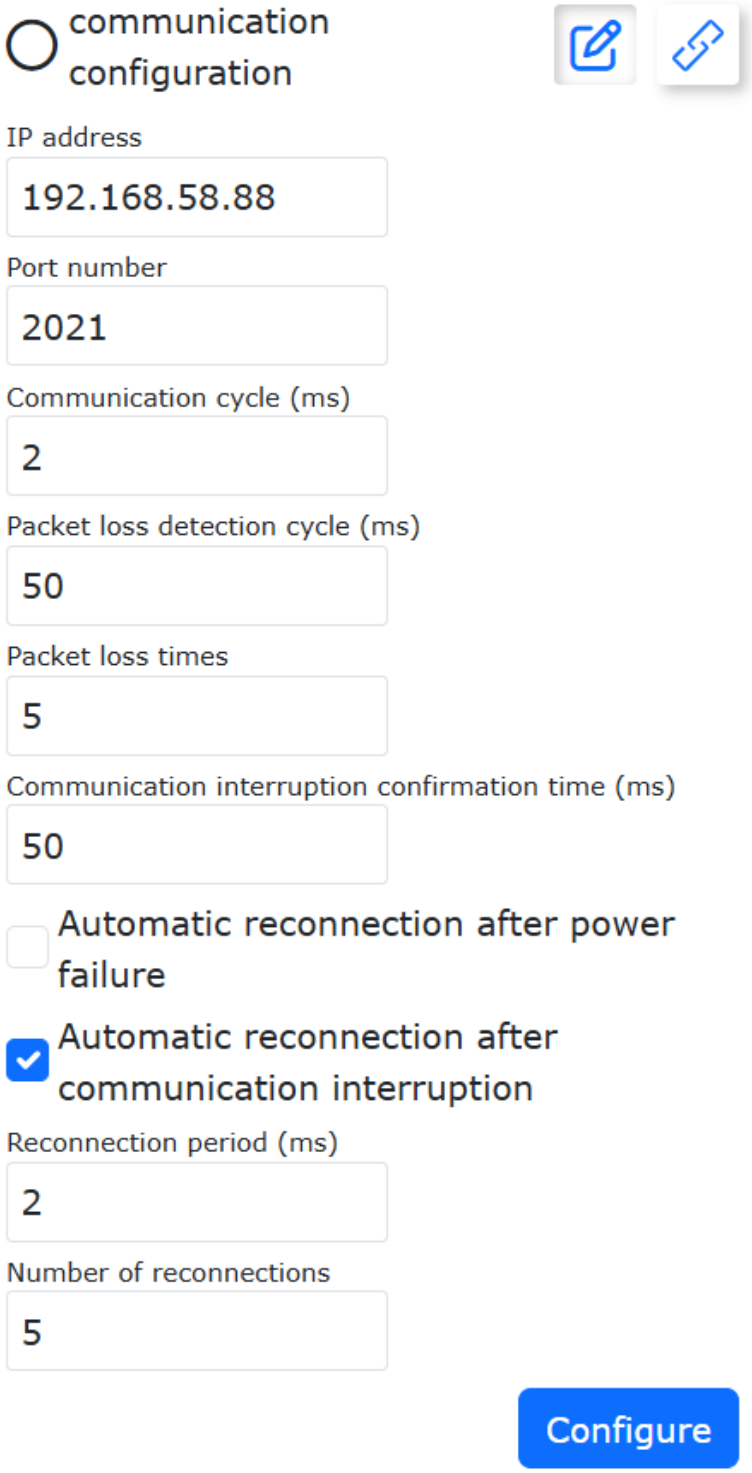
Figure 8.6-17 Set Welding Machine Signal Ports

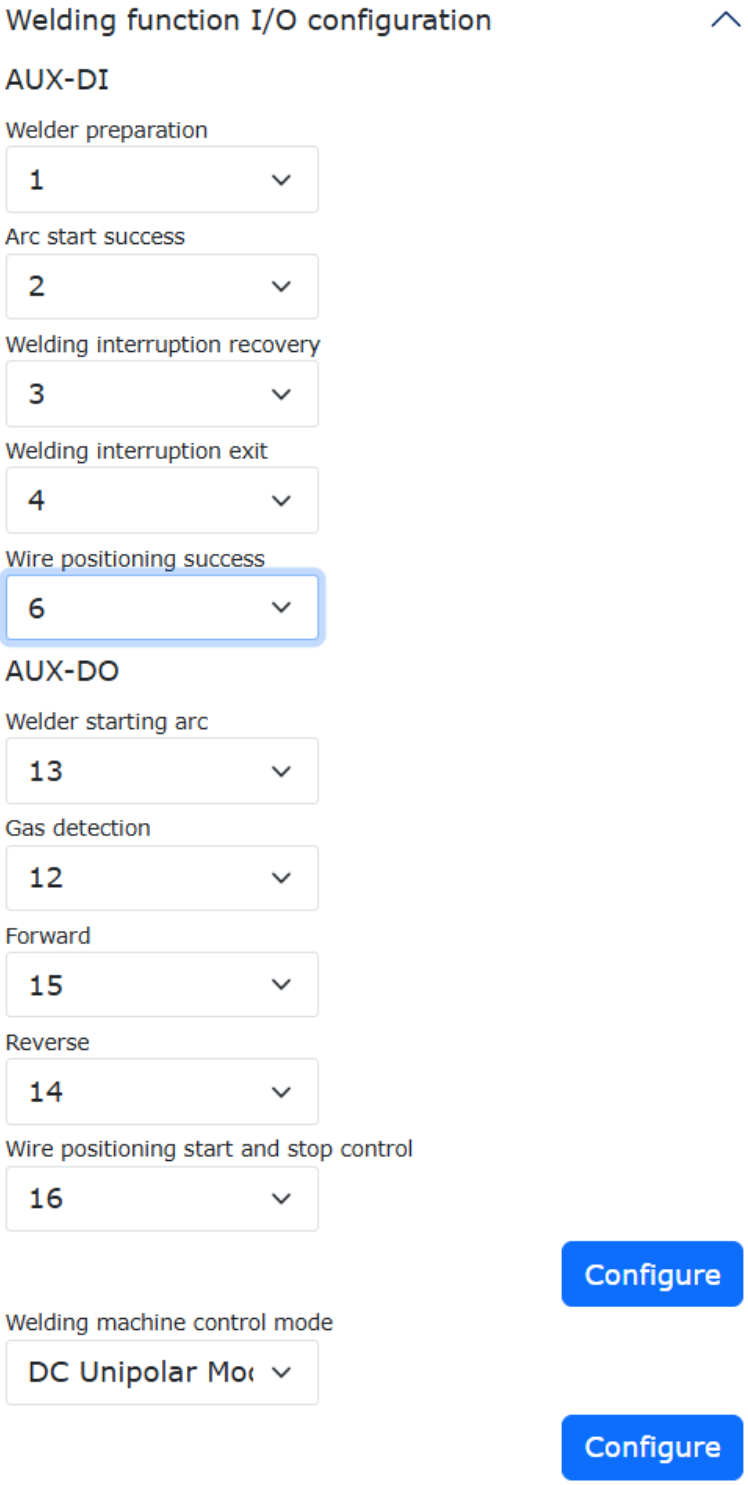
Welder Ready: This signal is output from the welding machine to the robot when the welding machine is ready to perform welding operations.

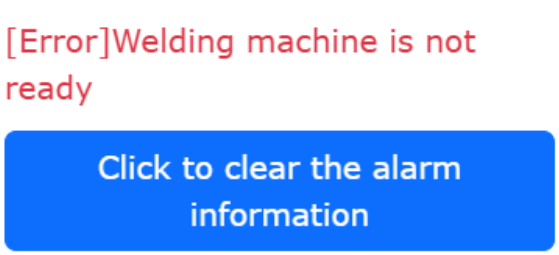
When the welding machine is not ready due to faults or other reasons, this signal is not input to the robot. At this time, the robot WebApp prompt in the upper right corner "Welder not ready". If your welding machine does not have a welder ready signal, you can set this port to "-1".

Figure 8.6-18 Welder Not Ready Error

Figure 8.6-19 Welder Ready Set to "-1"


communication configuration
IP address
192.168.58.88
Port number
2021
Communication cycle (ms)
2
Packet loss detection cycle (ms)
50
Packet loss times
5
Communication interruption confirmation time (ms)
50
Automatic reconnection after power failure
Automatic reconnection after communication interruption
Reconnection period (ms)
2
Number of reconnections
5
Configure


Welding function I/O configuration
AUX-DI
Welder preparation
1
Arc start success
2
Welding interruption recovery
3
Welding interruption exit
4
Wire positioning success
6
AUX-DO
Welder starting arc
13
Gas detection
12
Forward
15
Reverse
14
Wire positioning start and stop control
16
Configure
Welding machine control mode
DC Unipolar Mo
Configure



Arc Success: The arc has been successfully initiated. After the robot outputs the arc initiation signal to the welding machine, it waits for the arc success feedback signal from the welding machine. If the robot does not detect the arc success signal from the welding machine within the set timeout period, the robot reports an "Arc initiation timeout" error.

If the arc success signal is not configured, welding can still be performed using the robot welding function, but the robot will report a "Arc success DI not configured" warning. If your welding machine has an arc success signal output, we recommend configuring this signal for safer welding.

Figure 8.6-20 Arc Initiation Timeout Error

Figure 8.6-21 Arc Success DI Not Configured Warning

Welding Interruption Recovery: Triggered when the arc is unexpectedly interrupted during robot welding or the operator actively pauses welding. When this signal input to the robot changes from invalid to valid after a welding interruption, the robot automatically resumes welding from the original interruption position.

Welding Interruption Exit: Triggered when the arc is unexpectedly interrupted during robot welding or the operator actively pauses welding. When this signal input to the robot changes from invalid to valid after a welding interruption, the robot terminates welding. After termination, welding cannot be resumed.

Welding Arc Initiation: The DO output port through which the robot controls the welding machine to initiate the arc. When the robot program executes the arc initiation command, the corresponding DO output port for arc initiation automatically outputs a valid signal.

Gas Detection: The DO output port through which the robot controls the welding machine to supply gas. When the robot executes the welding gas supply command, the corresponding DO output port for gas supply automatically outputs a valid signal.

Forward Wire Feed: The DO output port through which the robot controls the welding machine for forward wire feeding. When the robot executes the forward wire feed command, the corresponding DO output port for forward wire feed automatically outputs a valid signal.

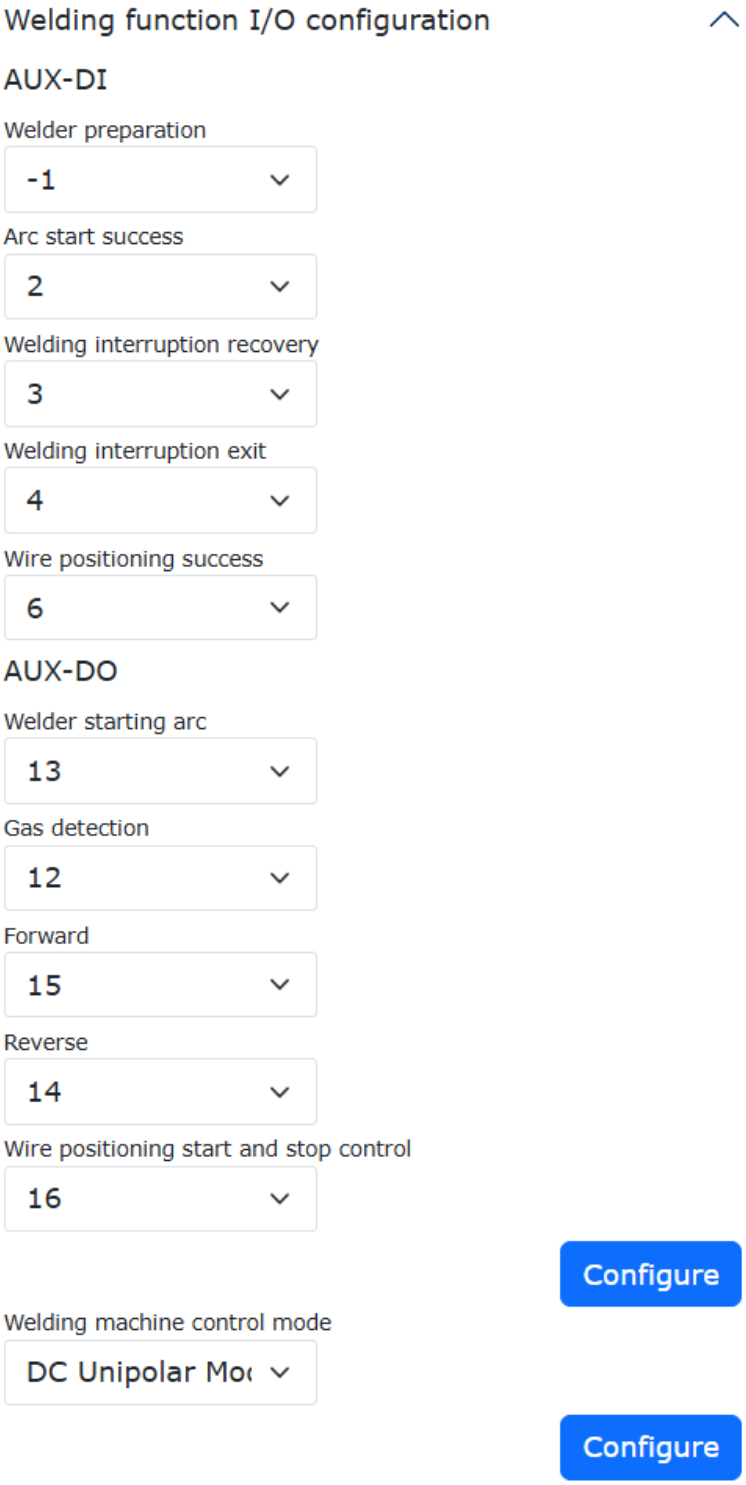
Reverse Wire Feed: The DO output port through which the robot controls the welding machine for reverse wire feeding. When the robot executes the reverse wire feed command, the corresponding DO output port for reverse wire feed automatically outputs a valid signal.

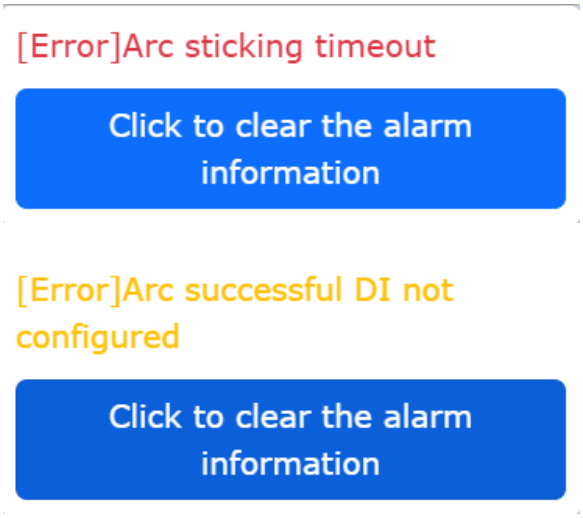
1.3.8.6.2.8 Welding Process Parameter Configuration

As shown in Figure 22, find the "Welding Process Parameters" section on the welding configuration page. The collaborative robot provides 100 sets of welding process parameters, numbered 0 to 99. Process number 0 indicates not using the welding process curve, while process numbers 1-99 use the welding process curve.

Figure 8.6-22 Welding Process Parameter Configuration

When using the welding process curve, take selecting welding process number 1 as an example. Input the parameters from Arc Initiation Current to Arc Closing Time as shown in Figure 8, then click the "Configure" button. The actual welding process represented by these parameters is as follows:


Welding function I/O configuration
AUX-DI
Welder preparation
-1
Arc start success
2
Welding interruption recovery
3
Welding interruption exit
4
Wire positioning success
6
AUX-DO
Welder starting arc
13
Gas detection
12
Forward
15
Reverse
14
Wire positioning start and stop control
16
Configure
Welding machine control mode
DC Unipolar Mo
Configure



Set welding current 200A, voltage 23V; Execute arc initiation, wait for arc success; After arc success, maintain the arc for 500ms (Arc initiation time, robot does not move); Set welding current 150A, welding voltage 21V, then the robot starts moving and performs welding; After welding to the end point, set welding current to 100A, welding voltage to 19V (Arc closing current, Arc closing voltage); After setting the arc closing current and voltage, maintain arc burning for 500ms (robot does not move), finally extinguish the arc.

When not using the welding process parameters, i.e., selecting welding process parameter number 0, the welding process is: Set the corresponding welding current and welding voltage via the set current/voltage interface; The robot controls the welding machine to initiate the arc and waits for arc success; After arc success, the robot starts moving and performs welding; The robot extinguishes the arc immediately after welding to the end point.

Figure 8.6-23 Not Using Welding Process Curve

1.3.8.6.2.9 Welding Machine Debugging

Find “Welding Machine Debugging” on the welding machine configuration page. Select process number 1, input the timeout time as 1000ms, click “Gas On”, and the robot will control the welding machine to start supplying shielding gas. Click the “Gas Off” button, and the robot will control the welding machine to stop supplying shielding gas. The operation methods for other buttons like “Arc Start”, “Forward Wire Feed”, “Reverse Wire Feed”, etc., are the same and will not be repeated.

Figure 8.5-24 Welding Machine Debugging

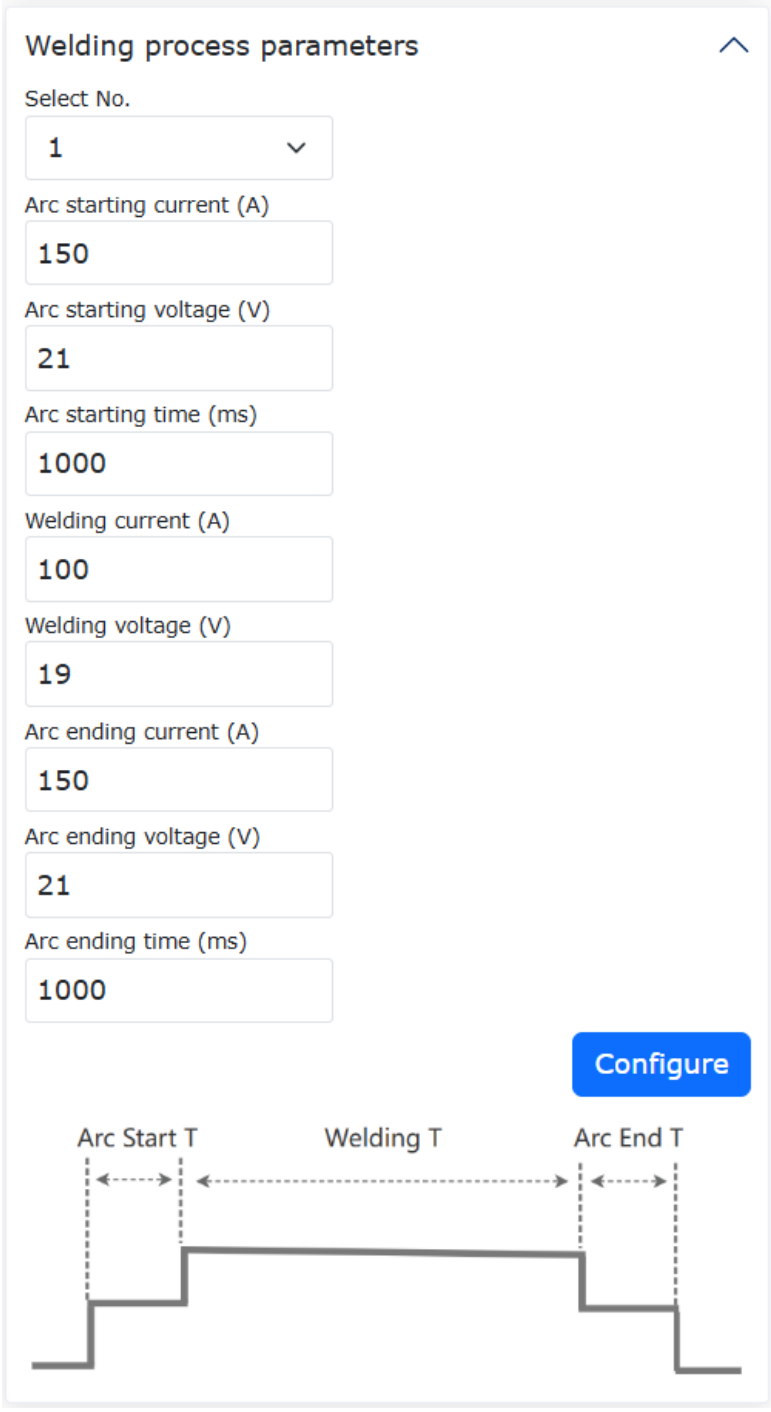
1.3.8.6.3 Welding Program Writing

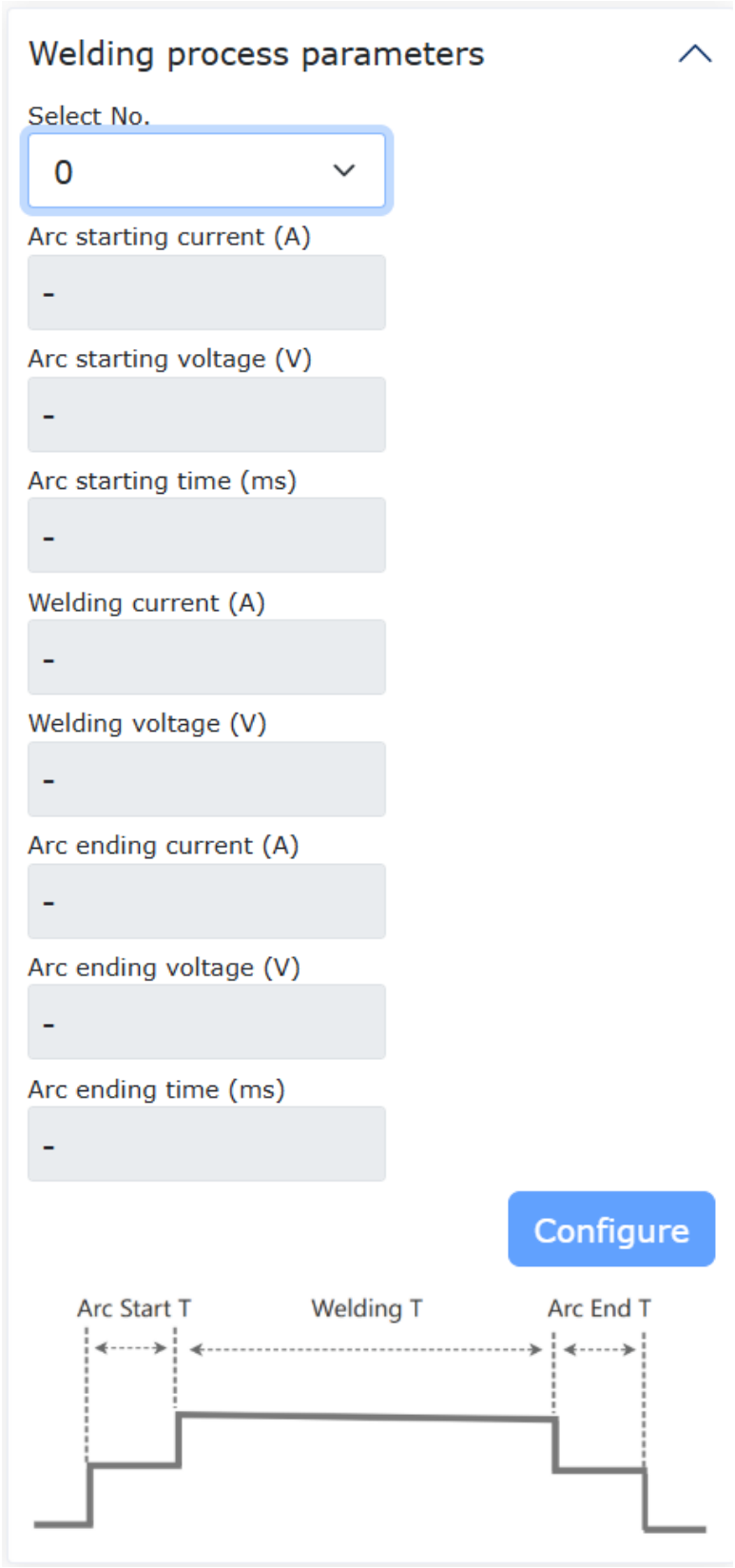
1.3.8.6.3.1 Writing Programs Using Welding Process Curves

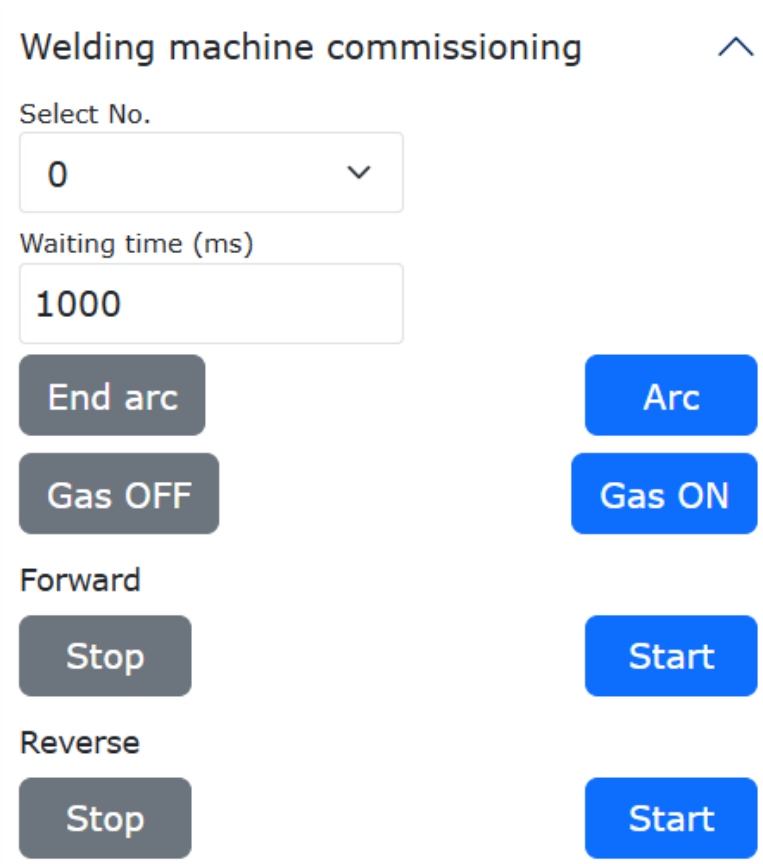
When using the welding process curve (i.e., selecting welding process parameter numbers 1 ~ 99), the voltage and current control during the welding process follows the curve parameters set for a specific process number, and there is no need to separately add instructions to set welding voltage and current. As shown in Figure 25, click “Teach” -> “Program Programming”, and create a new user program “testWeld.lua”.

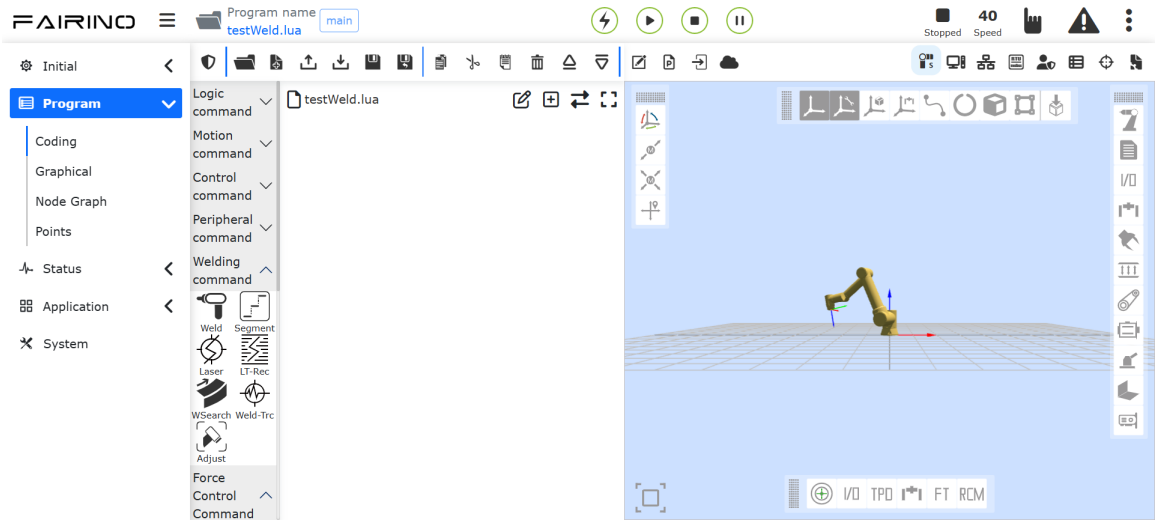
Figure 8.6-25 Create “testWeld.lua” Program

In the opened welding instruction addition page, select the control type as “Controller I/O” (select based on the actual configured welding control method), select the welding process number as 1 (Process number 0 does not use the welding


Welding process parameters
Select No.
1
Arc starting current (A)
150
Arc starting voltage (V)
21
Arc starting time (ms)
1000
Welding current (A)
100
Welding voltage (V)
19
Arc ending current (A)
150
Arc ending voltage (V)
21
Arc ending time (ms)
1000
Configure
Arc Start T
Welding T
Arc End T


Welding process parameters
Select No.
0
Arc starting current (A)
-
Arc starting voltage (V)
-
Arc starting time (ms)
-
Welding current (A)
-
Welding voltage (V)
-
Arc ending current (A)
-
Arc ending voltage (V)
-
Arc ending time (ms)
-
Configure
Arc Start T
Welding T
Arc End T


Welding machine commissioning
Select No.
0
Waiting time (ms)
1000
End arc
Arc
Gas OFF
Gas ON
Forward
Stop
Start
Reverse
Stop
Start


FAIRINO
Program name
testWeld.lua
main
Stopped
40
Speed
Initial
Program
Coding
Graphical
Node Graph
Points
Status
Application
System
Logic command
Motion command
Control command
Peripheral command
Welding command
Weld
Segment
Laser
LT-Rec
WSearch
Weld-Trc
Adjust
Force Control Command
testWeld.lua
I/O
TPD
FT
RCM

process curve, numbers 1-99 do), set the maximum wait time as 10000ms, click the “Arc Start” button and then the “Arc Close” button sequentially, and finally click “Apply”.

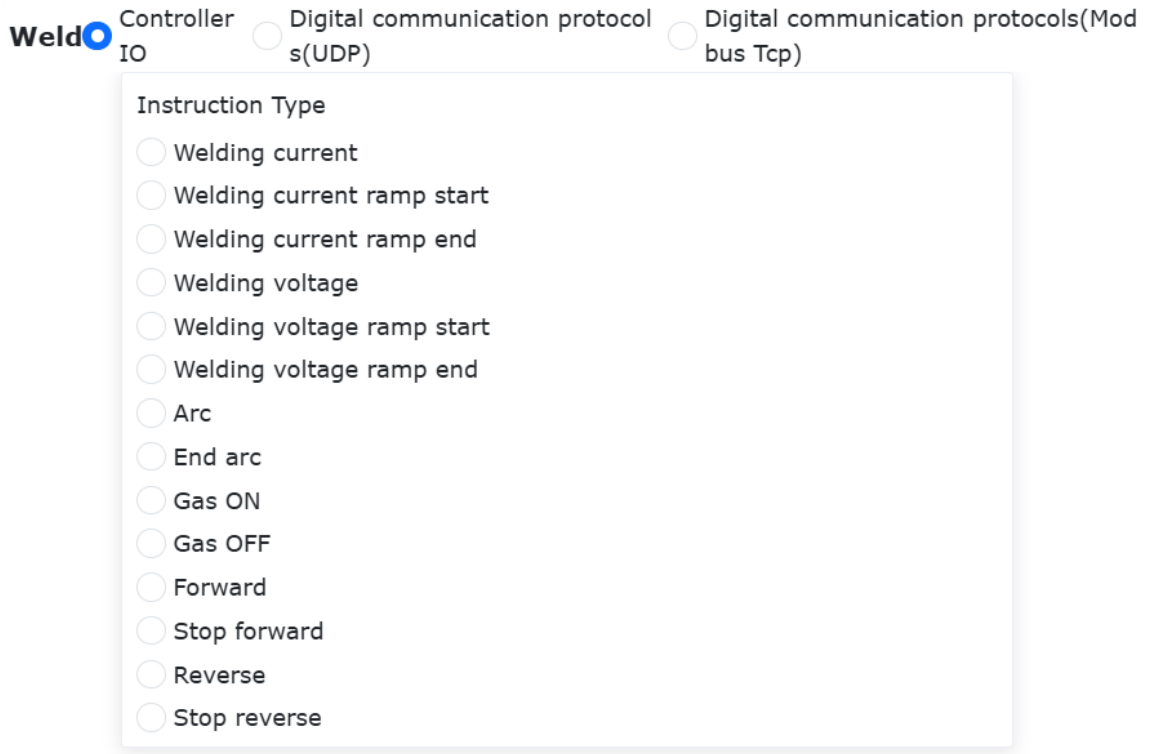


Figure 8.6-26 Welding Instruction Addition

Now the “testWeld.lua” program has added the welding arc start instruction and the welding arc close instruction. Since the arc start and close operations selected welding process curve number 1, the voltage and current control during the welding process follows the curve parameters set for process number 1, and there is no need to separately add instructions to set welding voltage and current.



Figure 8.6-27 Arc Start and Close Program

Add two linear motion instructions and adjust the instruction order so that the robot first moves to point “P1”, executes arc start, then moves to point “P2”, and executes arc close, achieving welding from point “P1” to point “P2”.

Figure 8.6-28 Robot Welding from Point P1 to P2



1.3.8.6.3.2 Writing Programs Without Using Welding Process Curves

When not using the welding process curve (i.e., selecting welding process parameter number 0), instructions to set welding voltage and current must be added to the welding program to control the actual welding parameters. Click "Teach" -> "Program Teaching", and create a new user program "testWeld.lua".

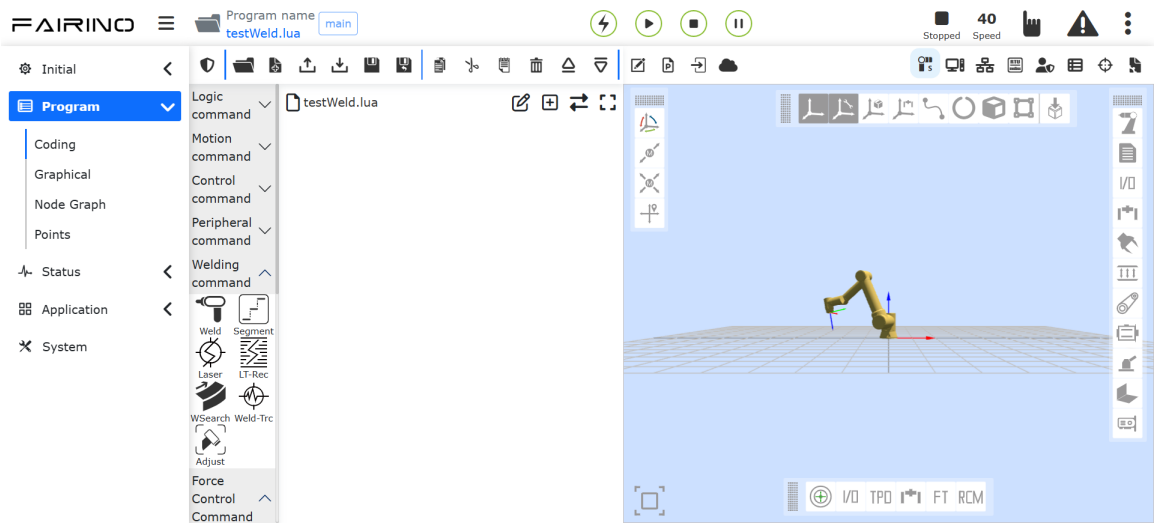
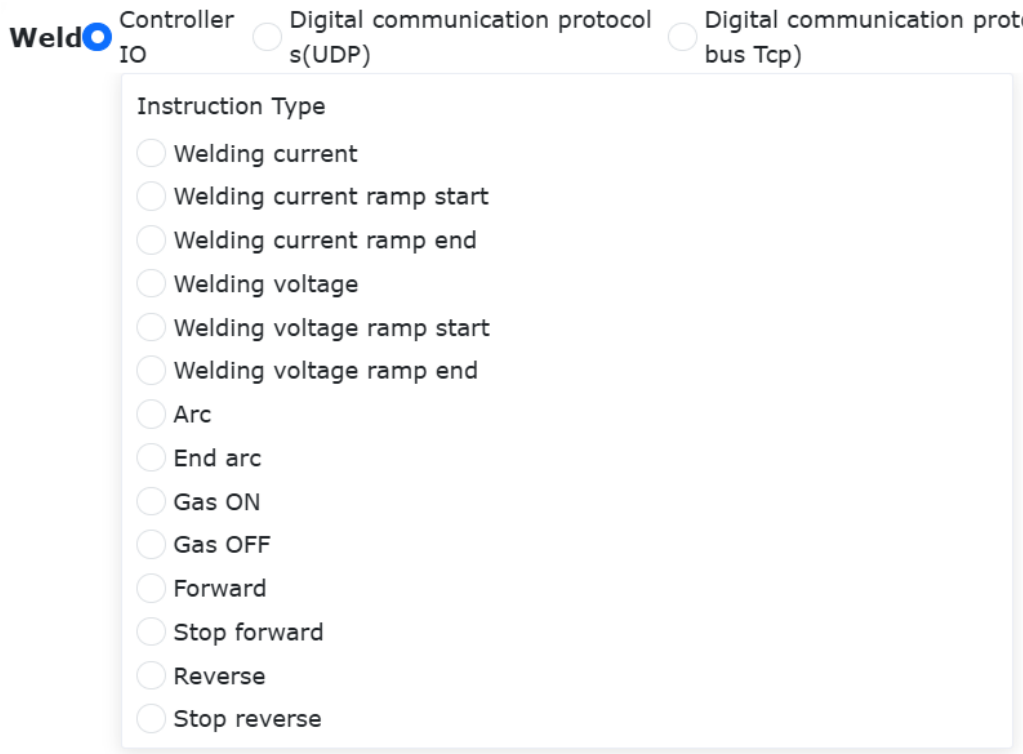


Figure 8.6-29 Create "testWeld.lua" Program

In the opened welding instruction addition page, select the control type as "Controller I/O" (select based on the actual configured welding control method), select the welding process number as 0 (Process number 0 does not use the welding process curve, numbers 1-99 do), set the welding current control AO as "Ctrl-AO0", welding current as 150A, click the "Add" button; set the welding voltage control AO as "Ctrl-AO1", welding voltage as 21V, click the "Add" button; set the maximum wait time as 10000ms, click the "Arc Start" button and then the "Arc Close" button sequentially, and finally click "Apply".

Figure 8.6-30 Welding Instruction Addition



Now the "testWeld.lua" program has added the welding arc start instruction and the welding arc close instruction. Since the arc start and close instructions selected welding process number 0, when the program executes the set welding voltage and current instructions, the robot will automatically output the corresponding control box analog signal based on the set welding voltage and current values and the "Welding voltage, current vs output analog correspondence" set in the welding machine configuration page.

Figure 8.6-31 Set Welding Voltage, Current, Arc Start, Arc Close Program

Add two linear motion instructions and adjust the instruction order so that the robot first moves to point "P1", executes arc start, then moves to point "P2", and executes arc close, achieving welding from point "P1" to point "P2".

Figure 8.6-32 Robot Welding from Point P1 to P2

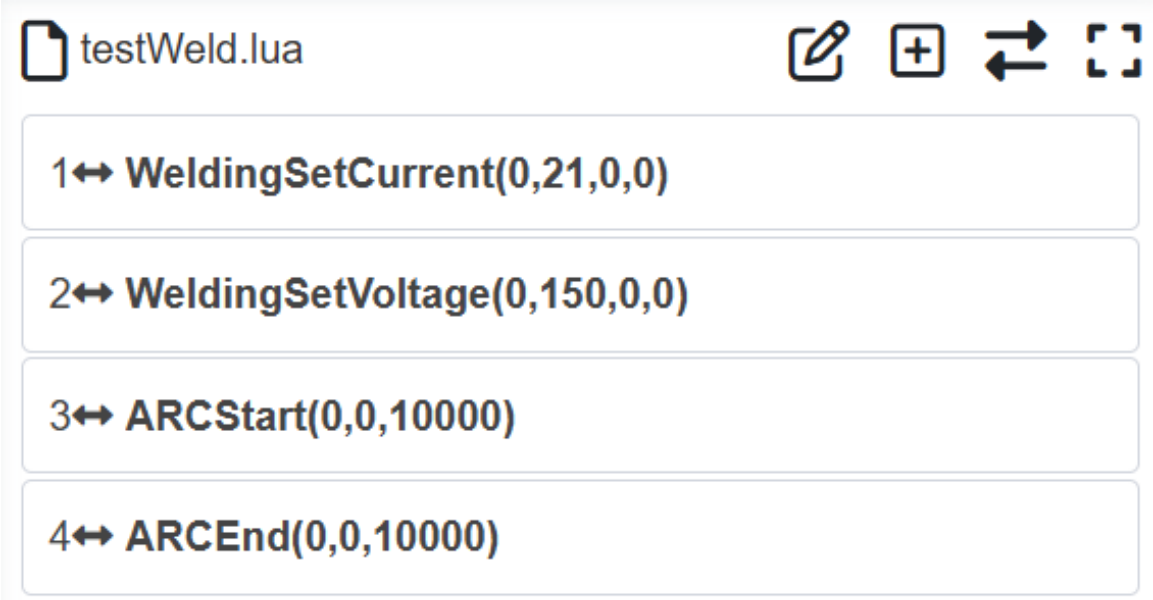
Running the above program will achieve welding along a straight line P1 ~ P2. Before running the program, please check: Whether the welding torch is correctly installed, whether the welding torch tool coordinate system has been calibrated, and applied as the current tool coordinate system; Whether the welding power supply, gas path, and wire feed path are working normally; Whether the signal line connections between the robot and the welding machine are normal.

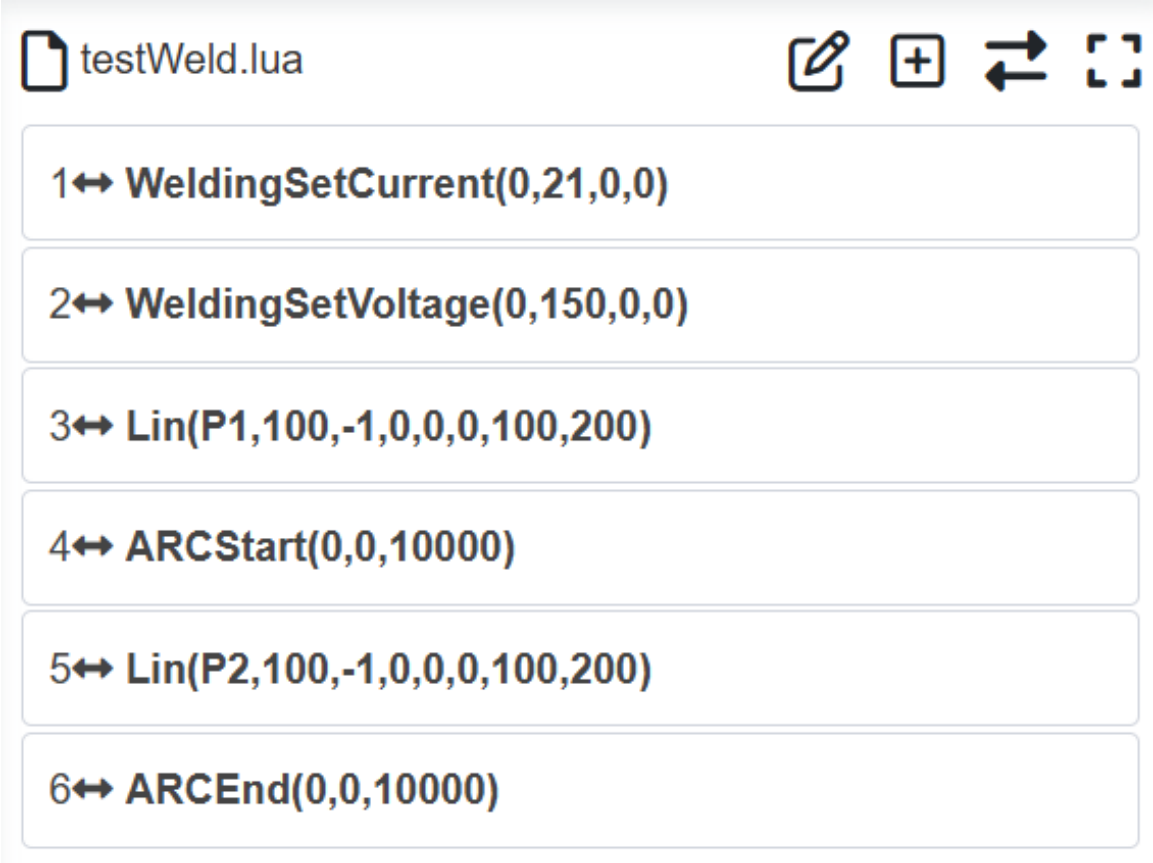
1.3.8.6.4 Welding Interruption and Recovery

Welding interruption may occur during robot welding under the following circumstances: The operator actively pauses welding to observe the actual welding condition or perform operations like cleaning the nozzle; Unexpected welding arc interruption; The robot collides, causing welding to pause;

After a welding interruption occurs during robot welding, the operator can switch the robot to manual mode, drag the robot to a safe position, and address the cause of the interruption.

After the issue is resolved, the collaborative robot can automatically move from the current position back to the position where the welding interruption occurred, re-ignite the arc, and resume welding. The specific operation process is:


testWeld.lua
1 WeldingSetCurrent(0,21,0,0)
2 WeldingSetVoltage(0,150,0,0)
3 ARCStart(0,0,10000)
4 ARCEnd(0,0,10000)


testWeld.lua
1 WeldingSetCurrent(0,21,0,0)
2 WeldingSetVoltage(0,150,0,0)
3 Lin(P1,100,-1,0,0,0,100,200)
4 ARCStart(0,0,10000)
5 Lin(P2,100,-1,0,0,0,100,200)
6 ARCEnd(0,0,10000)

Configure welding interruption recovery parameters; Execute the welding program, pause welding during the process to cause an interruption; Switch the robot to manual mode, handle the relevant issues, then switch the robot back to automatic mode after completion; Click the "Resume Welding" button, and the robot automatically resumes welding.

1.3.8.6.4.1 Welding Interruption Recovery Parameter Configuration

In the "Initial" -> "Peripherals" menu bar, click "Welding Machine" to enter the welding machine configuration interface. Find the "Detect Arc Interruption Parameter Configuration" section, turn on "Function Enable", input "Confirmation Duration" as 20ms, click the "Configure" button. This means that if the arc success signal remains invalid for more than 20ms during welding, the robot will report a "Welding arc interruption" error.

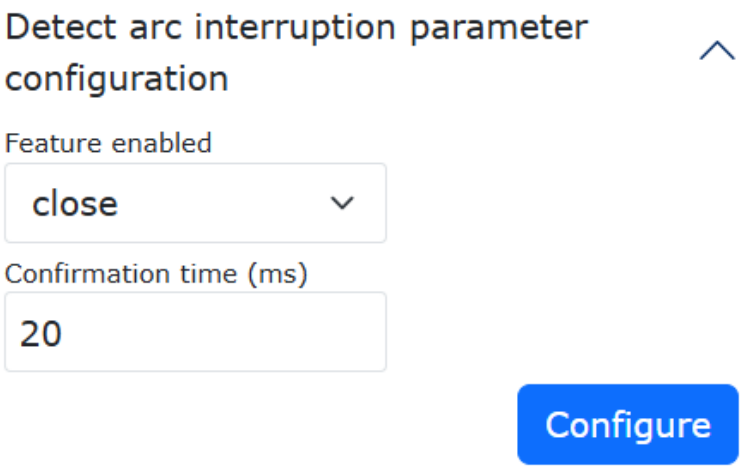


Figure 8.6-33 Detect Arc Interruption Parameter Configuration

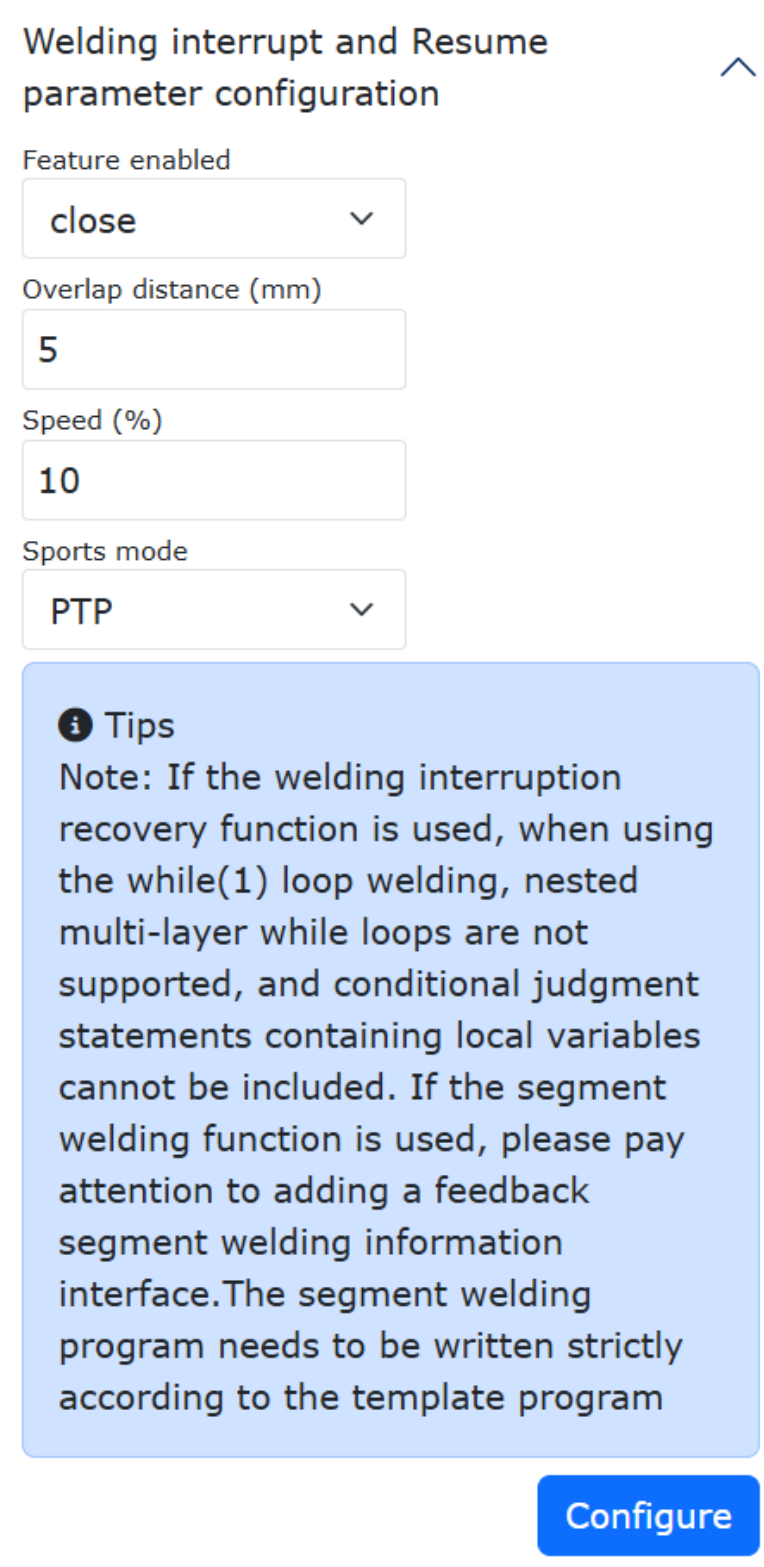
Find the "Welding Interruption Recovery Parameter Configuration" section, turn on "Function Enable", input "Overlap Distance" as 5mm, "Speed" as 10%, "Motion Mode" as "PTP", click the "Configure" button. The explanations for these three parameters are as follows:

Overlap Distance: To ensure the continuity between the resumed weld and the weld before the interruption, the arc restart point for welding recovery needs to have a certain overlap distance with the original weld.

Speed: After a welding interruption, the robot often needs to be moved to a safe position and the weld needs to be treated. After treatment is completed and welding recovery is executed, the robot will move from the current position to the welding restart point. This "Speed" indicates the speed at which the robot moves to the restart point.

Motion Mode: After a welding interruption, the robot often needs to be moved to a safe position and the weld needs to be treated. After treatment is completed and welding recovery is executed, the robot will move from the current position to the welding restart point. This "Motion Mode" indicates the motion mode used by the robot to move to the restart point, with "LIN" and "PTP" available for selection.

Figure 8.6-34 Welding Interruption Recovery Parameter Configuration


Welding interrupt and Resume parameter configuration
Feature enabled
close
Overlap distance (mm)
5
Speed (%)
10
Sports mode
PTP
Tips
Note: If the welding interruption recovery function is used, when using the while(1) loop welding, nested multi-layer while loops are not supported, and conditional judgment statements containing local variables cannot be included. If the segment welding function is used, please pay attention to adding a feedback segment welding information interface.The segment welding program needs to be written strictly according to the template program
Configure

1.3.8.6.4.2 Welding Interruption Recovery Application

Take the "testWeld" program as an example. Switch the robot to automatic mode, click the start button, and the robot begins welding. During welding, click the pause button. At this point, welding is interrupted, and a welding interruption recovery prompt box pops up in the WebApp right corner. Click the "Resume Welding" button, and the robot automatically moves to the restart point and executes the subsequent welding tasks.

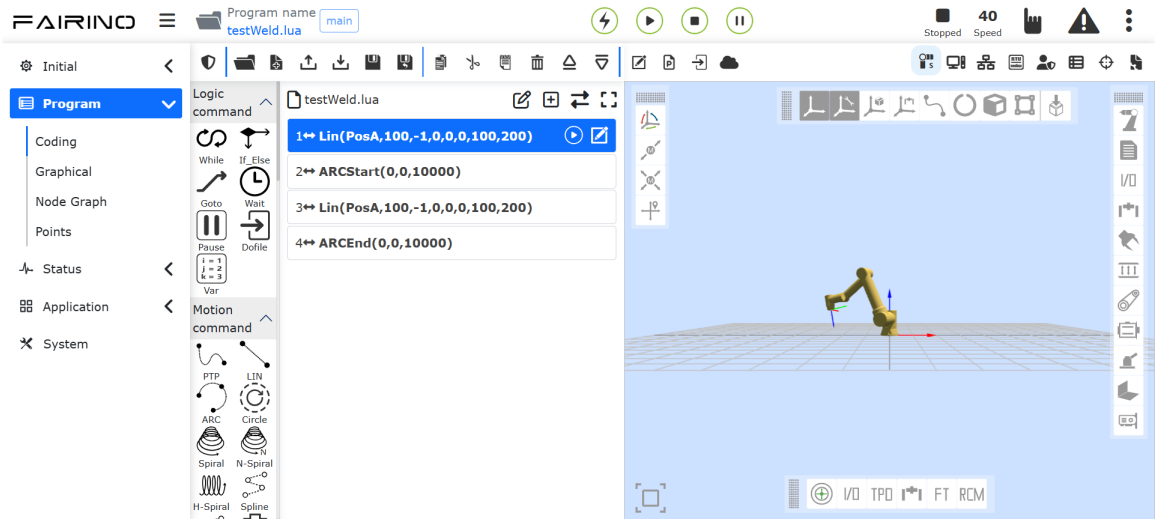


Figure 8.6-35 Execute Welding Program

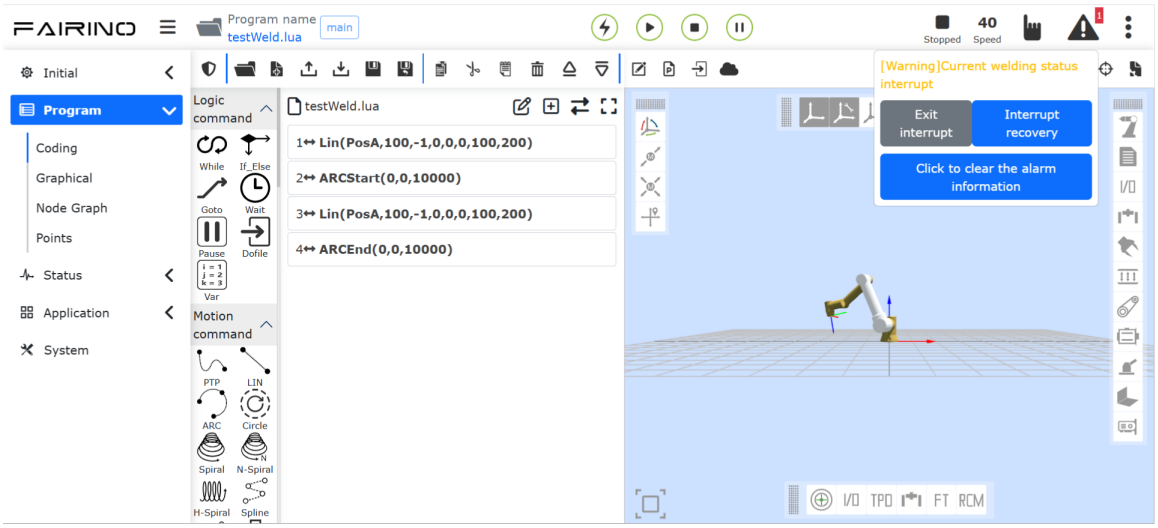


Figure 8.6-36 Welding Recovery

Warning: The collaborative robot welding interruption recovery function can only be used for linear welds or circular arc welds. When using a while (1) loop for welding, nested multi-layer while loops are not supported, and conditional judgment statements containing local variables cannot be included. If using stitch welding function, please pay attention to adding the interface for feedback stitch welding information.

1.3.8.6.5 Attachment 1: Robot UDP Communication Protocol

Warning:

1) CRC check method: Uses modbus 16 check but only takes the lower 8 bits for verification. The data areas for verification are D100-D176, D200-D273.

2) Arc Tracking: The actual current feedback converts the actual current obtained by the PLC from the welder into an analog value of 0-4095 and transmits it to the analog channel 0 of the UDP data protocol, i.e., D168.

3) Speed conversion logic: Robot issued speed (unit mm/s) V ÷ lead × 60 = V'; PLC converts the robot issued speed V' × encoder resolution ÷ 60 = V" (unit pulse/s).

1.3.8.6.5.1 Robot Controller -> PLC

No.	Register Address	Data Type	Data Value	Variable Name
1	D199	INT	0x5A5A	Frame Header
2	D200	INT		1# Motor Control Word
3	D201	DINT		1# Target Position Input
4	D202	DINT		1# Target Position Input
5	D203	INT		1# Homing Control Word
6	D204	DINT		1# Homing High Speed Input
7	D205	DINT		1# Homing High Speed Input
8	D206	DINT		1# Homing Low Speed Input
9	D207	DINT		1# Homing Low Speed Input
10	D208	DINT		1# Position Offset (Reserved)
11	D209	DINT		1# Position Offset (Reserved)
12	D210	DINT		1# Speed Offset (Reserved)
13	D211	DINT		1# Speed Offset (Reserved)
14	D212	DINT		1# Torque Offset (Reserved)
15	D213	DINT		1# Torque Offset (Reserved)
16	D214	INT		2# Motor Control Word
17	D215	DINT		2# Target Position Input
18	D216	DINT		2# Target Position Input
19	D217	INT		2# Homing Control Word
20	D218	DINT		2# Homing High Speed Input
21	D219	DINT		2# Homing High Speed Input
22	D220	DINT		2# Homing Low Speed Input
23	D221	DINT		2# Homing Low Speed Input
24	D222	DINT		2# Position Offset (Reserved)
25	D223	DINT		2# Position Offset (Reserved)
26	D224	DINT		2# Speed Offset (Reserved)
27	D225	DINT		2# Speed Offset (Reserved)
28	D226	DINT		2# Torque Offset (Reserved)
29	D227	DINT		2# Torque Offset (Reserved)
30	D228	INT		3# Motor Control Word
31	D229	DINT		3# Target Position Input
32	D230	DINT		3# Target Position Input
33	D231	INT		3# Homing Control Word

continues on next page

Table 5 – continued from previous page

No.	Register Address	Data Type	Data Value	Variable Name
34	D232	DINT		3# Homing High Speed Input
35	D233	DINT		3# Homing High Speed Input
36	D234	DINT		3# Homing Low Speed Input
37	D235	DINT		3# Homing Low Speed Input
38	D236	DINT		3# Position Offset (Reserved)
39	D237	DINT		3# Position Offset (Reserved)
40	D238	DINT		3# Speed Offset (Reserved)
41	D239	DINT		3# Speed Offset (Reserved)
42	D240	DINT		3# Torque Offset (Reserved)
43	D241	DINT		3# Torque Offset (Reserved)
44	D242	INT		4# Motor Control Word
45	D243	DINT		4# Target Position Input
46	D244	DINT		4# Target Position Input
47	D245	INT		4# Homing Control Word
48	D246	DINT		4# Homing High Speed Input
49	D247	DINT		4# Homing High Speed Input
50	D248	DINT		4# Homing Low Speed Input
51	D249	DINT		4# Homing Low Speed Input
52	D250	DINT		4# Position Offset (Reserved)
53	D251	DINT		4# Position Offset (Reserved)
54	D252	DINT		4# Speed Offset (Reserved)
55	D253	DINT		4# Speed Offset (Reserved)
56	D254	INT		Reserved
57	D255	INT		Welding Mode Setting (0-DC Mono, 1-Pulse Mono, 2-JOB Mode, 3-Local Control Mode, 4-Separate Mode, 5-CC/CV, 6-TIG, 7-CMT Mode)
58	D256	INT		General Output DO(0-15)
59	D257	INT		General Output DO(16-31)
60	D258	INT		General Output DO(32-47)
61	D259	INT		General Output DO(48-63)
62	D260	INT		General Output DO(64-79)
63	D261	INT		General Output DO(80-95)
64	D262	INT		High-Speed Output DO(96-111)
65	D263	INT		High-Speed Output DO(112-127)
66	D264	INT		Analog Output AO0
67	D265	INT		Analog Output AO1
68	D266	INT		Analog Output AO2
69	D267	INT		Analog Output AO3
70	D268	REAL		Issued Welding Voltage
71	D269	REAL		Issued Welding Voltage
72	D270	REAL		Issued Welding Current
73	D271	REAL		Issued Welding Current
74	D272	REAL		Packet Loss Detection Cycle
75	D273	INT		Number of Lost Packets
76	D274	INT		Frame Count (0-255)
77	D275	INT		CRC Check Code

1.3.8.6.5.2 PLC -> Robot Controller

No.	Register Address	Data Type	Data Value	Variable Name
1	D99	INT	0x5A5A	Frame Header
2	D100	INT		1# Motor Status Word
3	D101	DINT		1# Current Position
4	D102	DINT		1# Current Position
5	D103	INT		1# Homing Status Word
6	D104	DINT		1# Homing High Speed Feedback
7	D105	DINT		1# Homing High Speed Feedback
8	D106	DINT		1# Homing Low Speed Feedback
9	D107	DINT		1# Homing Low Speed Feedback
10	D108	INT		1# Fault Code
11	D109	DINT		1# Following Deviation (Reserved)
12	D110	DINT		1# Following Deviation (Reserved)
13	D111	DINT		1# Speed Feedback (Reserved)
14	D112	DINT		1# Speed Feedback (Reserved)
15	D113	DINT		1# Real-time Torque (Reserved)
16	D114	DINT		1# Real-time Torque (Reserved)
17	D115	INT		2# Motor Status Word
18	D116	DINT		2# Current Position
19	D117	DINT		2# Current Position
20	D118	INT		2# Homing Status Word
21	D119	DINT		2# Homing High Speed Feedback
22	D120	DINT		2# Homing High Speed Feedback
23	D121	DINT		2# Homing Low Speed Feedback
24	D122	DINT		2# Homing Low Speed Feedback
25	D123	INT		2# Fault Code
26	D124	DINT		2# Following Deviation (Reserved)
27	D125	DINT		2# Following Deviation (Reserved)
28	D126	DINT		2# Speed Feedback (Reserved)
29	D127	DINT		2# Speed Feedback (Reserved)
30	D128	DINT		2# Real-time Torque (Reserved)
31	D129	DINT		2# Real-time Torque (Reserved)
32	D130	INT		3# Motor Status Word
33	D131	DINT		3# Current Position
34	D132	DINT		3# Current Position
35	D133	INT		3# Homing Status Word
36	D134	DINT		3# Homing High Speed Feedback
37	D135	DINT		3# Homing High Speed Feedback
38	D136	DINT		3# Homing Low Speed Feedback
39	D137	DINT		3# Homing Low Speed Feedback
40	D138	DINT		3# Fault Code
41	D139	DINT		3# Following Deviation (Reserved)
42	D140	DINT		3# Following Deviation (Reserved)
43	D141	DINT		3# Speed Feedback (Reserved)
44	D142	DINT		3# Speed Feedback (Reserved)
45	D143	DINT		3# Real-time Torque (Reserved)
46	D144	DINT		3# Real-time Torque (Reserved)
47	D145	INT		4# Motor Status Word

continues on next page

Table 6 – continued from previous page

No.	Register Address	Data Type	Data Value	Variable Name
48	D146	DINT		4# Current Position
49	D147	DINT		4# Current Position
50	D148	INT		4# Homing Status Word
51	D149	DINT		4# Homing High Speed Feedback
52	D150	DINT		4# Homing High Speed Feedback
53	D151	DINT		4# Homing Low Speed Feedback
54	D152	DINT		4# Homing Low Speed Feedback
55	D153	DINT		4# Fault Code
56	D154	DINT		4# Following Deviation (Reserved)
57	D155	DINT		4# Following Deviation (Reserved)
58	D156	DINT		4# Speed Feedback (Reserved)
59	D157	DINT		4# Speed Feedback (Reserved)
60	D158	DINT		Real-time Torque (Reserved)
61	D159	DINT		Real-time Torque (Reserved)
62	D160	INT		General Input DI(0-15)
63	D161	INT		General Input DI(16-31)
64	D162	INT		General Input DI(32-47)
65	D163	INT		General Input DI(48-63)
66	D164	INT		General Input DI(64-79)
67	D165	INT		General Input DI(80-95)
68	D166	INT		High-Speed Input DI(96-111)
69	D167	INT		High-Speed Input DI(112-127)
70	D168	INT		Analog Input AI0
71	D169	INT		Analog Input AI1
72	D170	INT		Analog Input AI2
73	D171	INT		Analog Input AI3
74	D172	REAL		Actual Current Feedback
75	D173	REAL		Actual Current Feedback
76	D174	REAL		Actual Voltage Feedback
77	D175	REAL		Actual Voltage Feedback
78	D176	INT		Fault Code 0-No Fault, 1-Data Packet Loss
79	D177	INT		Frame Count
80	D178	INT		CRC Check Code

1.3.8.6.6 Digital Communication Protocol (Modbus TCP)

Click “Initial” -> “Peripherals” -> “Welder” to enter the welder interface, then click the “Digital Communication Protocol (Modbus TCP)” card to enter the Welder Open Protocol interface.

1.3.8.6.6.1 Protocol Configuration

In the Open Protocol Configuration, click the "Upload" button to upload the completed Open Protocol LUA program file to the controller. Select an Open Protocol ID and Open Protocol Name, then click the "Configure" button (the selected Protocol ID must match the ID written in the Open Protocol file) to assign an ID to each open protocol.

Chart 8.6-37 Controller Peripheral Open Protocol Upload and Configuration

In the configured protocol list, click the "Load" button. The running status indicator lights up, indicating that the open protocol has been loaded successfully.

Chart 8.6-38 Controller Peripheral Open Protocol Loading and Running Indication

1.3.8.6.6.2 Welder Open Protocol

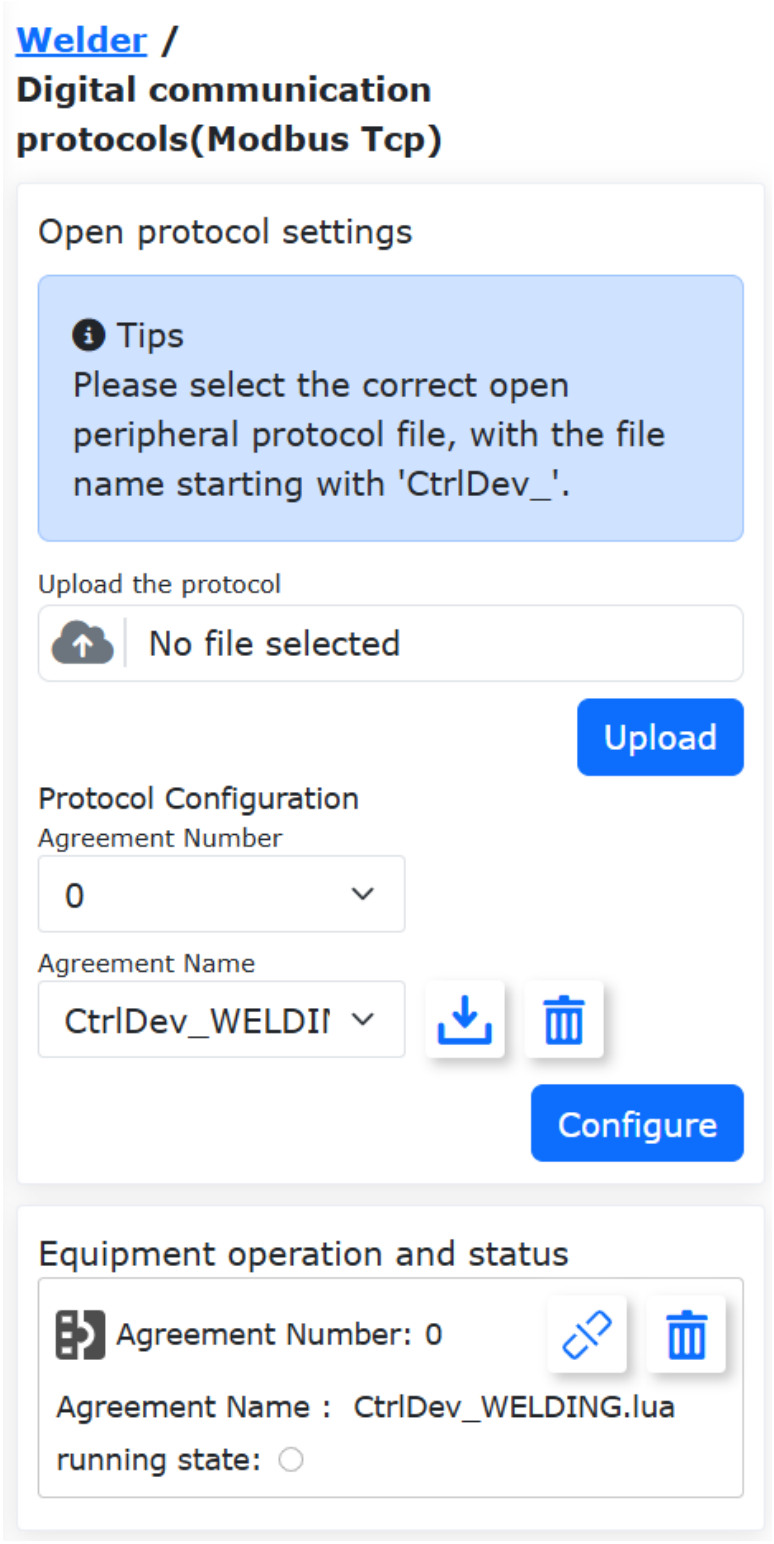
The robot communicates with the welder via the controller peripheral open protocol using ModbusTCP. Write the corresponding communication protocol LUA file according to the welder slave register definitions. Configure communication parameters such as the welder IP address, port number, and register addresses for arc start control, wire feed control, etc., in this file. Upload this protocol to the robot controller and load it to enable communication between the robot and the welder.

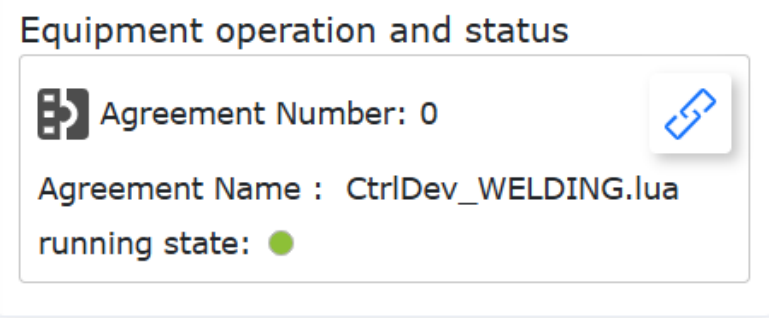
1.3.8.6.6.3 Welder Open Protocol Example

```
local id = 1 --Protocol number, must match the protocol number configured in WebApp
local ctrlValues = {0, 0, 0, 0, 0, 0}
local realTimeState = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
ModbusTCPMasterClose(id)
ModbusTCPMasterCreate('192.168.58.45', 502, 1, id)
while(1) do
setArcStart, setWireForward, setWireReverse, setShieldingGas, setTouchEnable,␣
↪setRobotError,setRobotEnableState,default1,default2, default3, default4, setCurrent,␣
↪setVoltage, SetMode = WeldingGetCtrlState()
local ctrlWord = 0
ctrlWord = SetBitWithIndex(ctrlWord, 0, setArcStart)
ctrlWord = SetBitWithIndex(ctrlWord, 1, setWireForward)
ctrlWord = SetBitWithIndex(ctrlWord, 2, setWireReverse)
ctrlWord = SetBitWithIndex(ctrlWord, 3, setShieldingGas)
ctrlWord = SetBitWithIndex(ctrlWord, 4, setTouchEnable)
ctrlWord = SetBitWithIndex(ctrlWord, 7, setRobotError)
ctrlValues[1] = setRobotEnableState
ctrlValues[2] = ctrlWord
ctrlValues[3] = 0
ctrlValues[4] = setCurrent
ctrlValues[5] = setVoltage
ctrlValues[6] = 0
ModbusTCPMasterSetHoldRegs(id, 201, 6, ctrlValues, "U16")
localtmpCtrlMode={0,0,0,0}
tmpCtrlMode[1]=SetMode
ModbusTCPMasterSetHoldRegs(id,0x1000,1,tmpCtrlMode,"U16")
sleep_ms(10)

getWeldState, getCurrent, getVoltage,default1, default2, getWelderErrorCode =␣
```

(continues on next page)


Welder /
Digital communication protocols(Modbus Tcp)
Open protocol settings
Tips
Please select the correct open peripheral protocol file, with the file name starting with 'CtrlDev_'.
Upload the protocol
No file selected
Upload
Protocol Configuration
Agreement Number
0
Agreement Name
CtrlDev_WELDII
Configure
Equipment operation and status
Agreement Number: 0
Agreement Name : CtrlDev_WELDING.lua
running state:



(continued from previous page)

```
↪ModbusTCPMasterGetHoldRegs(id, 211, 6, "U16")
realTimeState[1] = GetBitWithIndex(getWeldState, 0) + GetBitWithIndex(getWeldState, 1) *␣
↪2  --welderType
realTimeState[2] = GetBitWithIndex(getWeldState, 5) --arc state(WCR)
realTimeState[3] = GetBitWithIndex(getWeldState, 4) --touch state
realTimeState[4] = GetBitWithIndex(getWeldState, 7) --welder error state
realTimeState[12] = getCurrent                      --current
realTimeState[13] = getVoltage                      --voltage
realTimeState[14] = getWelderErrorCode              --welder error code
realTimeState[15] = getWeldState / 255             --heart jump
WeldingSetRealtimeState(realTimeState)

local stopFlag = GetOpenLUAStopFlag(id)
if(stopFlag ~= 0) then
ModbusTCPMasterClose(id)
break
end

sleep_ms(10)
end
```

1.3.8.6.6.4 Welder Open Protocol Analysis

The welder open protocol mainly consists of three parts:

Establish Communication Connection: Specify the protocol number id (must match the protocol number set when loading the open protocol), welder IP address, port number, and other parameters. Use the "ModbusTCPMasterCreate()" command to establish a ModbusTCP connection between the robot and the welder.

Cyclically Write Control Data to Welder: When the welder open protocol executes, it first reads the current welder control data from the robot controller's internal memory, then writes this data to the welder to control its actions. The return values of the instruction "WeldingGetCtrlState()", which reads robot welding control data in the protocol, are defined in Table 2-1. The control data can be decomposed according to the actual welder control register definitions and then written to the welder via ModbusTCP.

Table 8.19-1 WeldingGetCtrlState() Return Values

No.	Type	Name	Description
1	uint16_t	setArcStart	Arc Start Signal; 0-Arc Extinguish; 1-Arc Start
2	uint16_t	setWireForward	Forward Wire Feed: 0-Stop Wire Feed; 1-Forward Wire Feed
3	uint16_t	setWireReverse	Reverse Wire Feed: 0-Stop Wire Feed; 1-Reverse Wire Feed
4	uint16_t	setShieldingGas	Shielding Gas Control: 0-Stop Gas; 1-Start Gas
5	uint16_t	setTouchEnable	Wire Touch Sensing Enable: 0-Disable; 1-Enable
6	uint16_t	setRobotError	Robot Fault: 0-No Fault; 1-Fault
7	uint16_t	setRobotEnableState	Robot Enable State: 0-Not Enabled; 1-Enabled
8	uint16_t	default1	Reserved
9	uint16_t	default2	Reserved
10	uint16_t	default3	Reserved
11	uint16_t	default4	Reserved
12	uint16_t	setCurrent	Set Welding Current (0.1A)
13	uint16_t	setVoltage	Set Welding Voltage (0.01V)
14	uint16_t	SetMode	Set Welding Mode: 0-DC Mono, 1-Pulse Mono, 2-JOB Mode, 3-Local Control Mode, 4-Separate Mode, 5-CC/CV, 6-TIG, 7-CMT Mode
15	uint16_t	default6	Reserved
16	uint16_t	default7	Reserved
17	uint16_t	default8	Reserved
18	uint16_t	default9	Reserved
19	uint16_t	default10	Reserved
20	uint16_t	default11	Reserved

Cyclically Read Status Data from Welder: The welder open protocol first reads real-time status data from the welder via ModbusTCP, then writes the relevant data to the robot controller, allowing the robot to monitor the welder's real-time action status. The parameter for the protocol's interface "WeldingSetRealtimeState()", which sets the welder status in the robot, is an array containing all welder statuses (note: in the open protocol LUA, array indexing starts from 1) as shown in Table 2-2. The welder status data read via ModbusTCP according to the actual welder status register definitions can be combined into the welder status array and written to the robot controller.

Table 8.19-2 WeldingSetRealtimeState() Detailed Parameters

Type	Name	Array Index	Description
uint16_t[2	realTimeState	1	Welder Model
uint16_t[2	realTimeState	2	Arc State: 0-Arc Off; 1-Arc On
uint16_t[2	realTimeState	3	Wire Contact State: 0-Not Contacted; 1-Contacted
uint16_t[2	realTimeState	4	Welder Fault State: 0-No Fault; 1-Welder Fault
uint16_t[2	realTimeState	5	Reserved
uint16_t[2	realTimeState	6	Reserved
uint16_t[2	realTimeState	7	Reserved
uint16_t[2	realTimeState	8	Reserved
uint16_t[2	realTimeState	9	Reserved
uint16_t[2	realTimeState	10	Reserved
uint16_t[2	realTimeState	11	Reserved
uint16_t[2	realTimeState	12	Real-time Welding Current (0.1A)
uint16_t[2	realTimeState	13	Real-time Welding Voltage (0.01V)
uint16_t[2	realTimeState	14	Welder Fault Code
uint16_t[2	realTimeState	15	Welder Communication Heartbeat Data
uint16_t[2	realTimeState	16	Reserved
uint16_t[2	realTimeState	17	Reserved
uint16_t[2	realTimeState	18	Reserved
uint16_t[2	realTimeState	19	Reserved
uint16_t[2	realTimeState	20	Reserved

1.3.8.6.6.5 Welding Machine Open Protocol Upload and Load

Successively click “Initial”, “Peripherals”, “Control Box”, “Peripheral Open Protocol”, then click the “Upload” button to upload the welding machine open protocol file “CtrlDev_WELDING.lua” (The protocol file name must start with *CtrlDev_* and have the extension *.lua*).

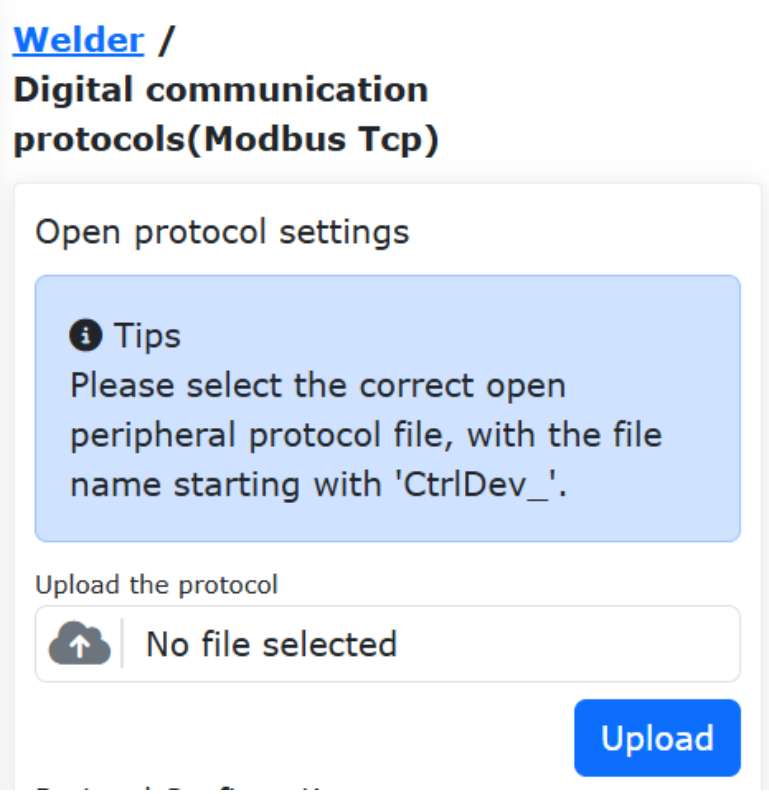
Figure 8.6-39 Upload Welding Machine Open Protocol

In “Protocol Configuration”, select a “Protocol Number” (needs to match the protocol number in the open protocol file), here using number 1 as an example, and select the “Protocol Name” as the welding machine open protocol “CtrlDev_WELDING.lua”. Click the “Configure” button. At this point, the configured welding machine open protocol is displayed in “Device Operation and Status”.

Figure 8.6-40 Configure Welding Machine Open Protocol

Click the “Connect” button to load the welding machine open protocol. The running status indicator lights up, indicating that the robot and the welding machine are communicating.

Figure 8.6-41 Load Welding Machine Open Protocol



1.3.8.6.6.6 Welding Machine Debugging

Before debugging the welding machine, please ensure that the welding machine open protocol has been loaded normally and the relevant register address configurations are correct.

Successively click “Initial”, “Peripherals”, “Welding Machine”, and select “Digital Communication Protocol (ModbusTcp)”.

Figure 8.6-42 Select “Digital Communication Protocol (ModbusTcp)”

Click buttons such as “Arc Start”, “Arc Stop”, “Gas On”, “Gas Off”, etc., and observe whether the actual welding machine actions are consistent with the settings. If the welding machine does not perform the set actions, check if the register configuration in the welding machine open protocol is incorrect and perform further debugging.

Figure 8.6-43 Welding Machine Debugging

1.3.8.6.6.7 Welding Program Writing

Click “Initial”, “Teach Program”, “Program Programming”, and create a new program “testWeld.lua”.

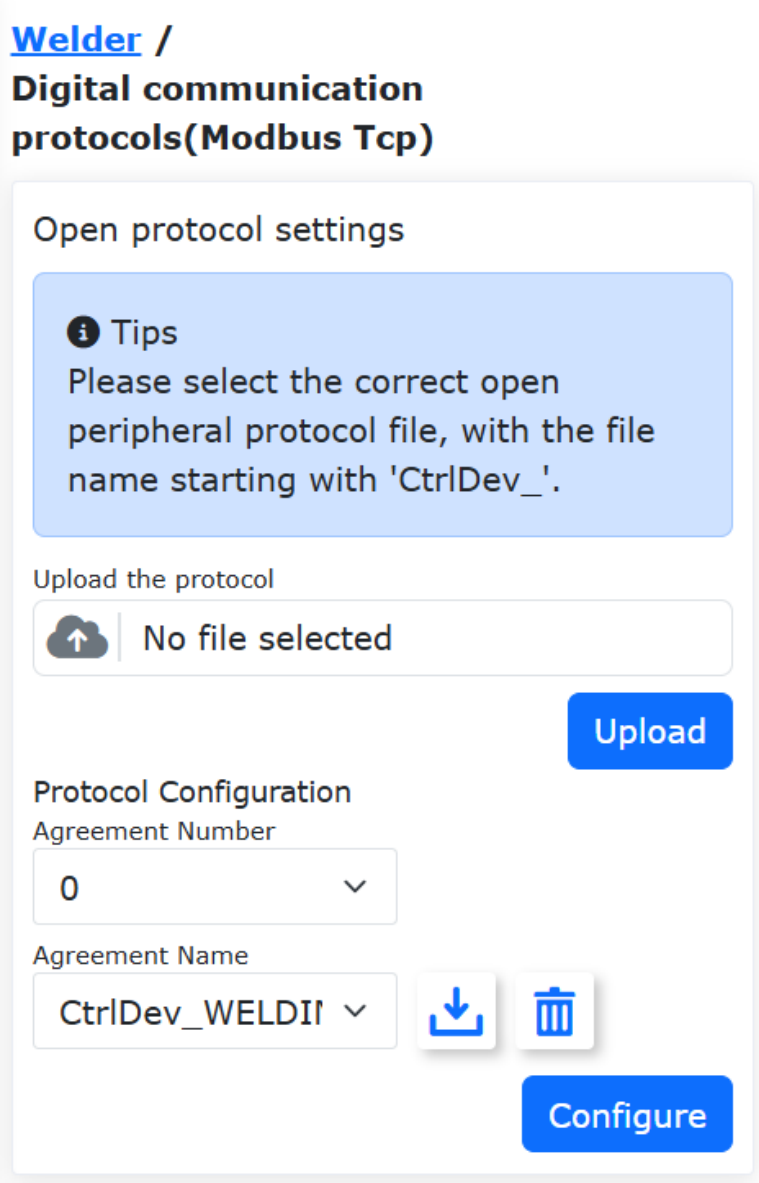
Figure 8.6-44 Create Welding LUA Program

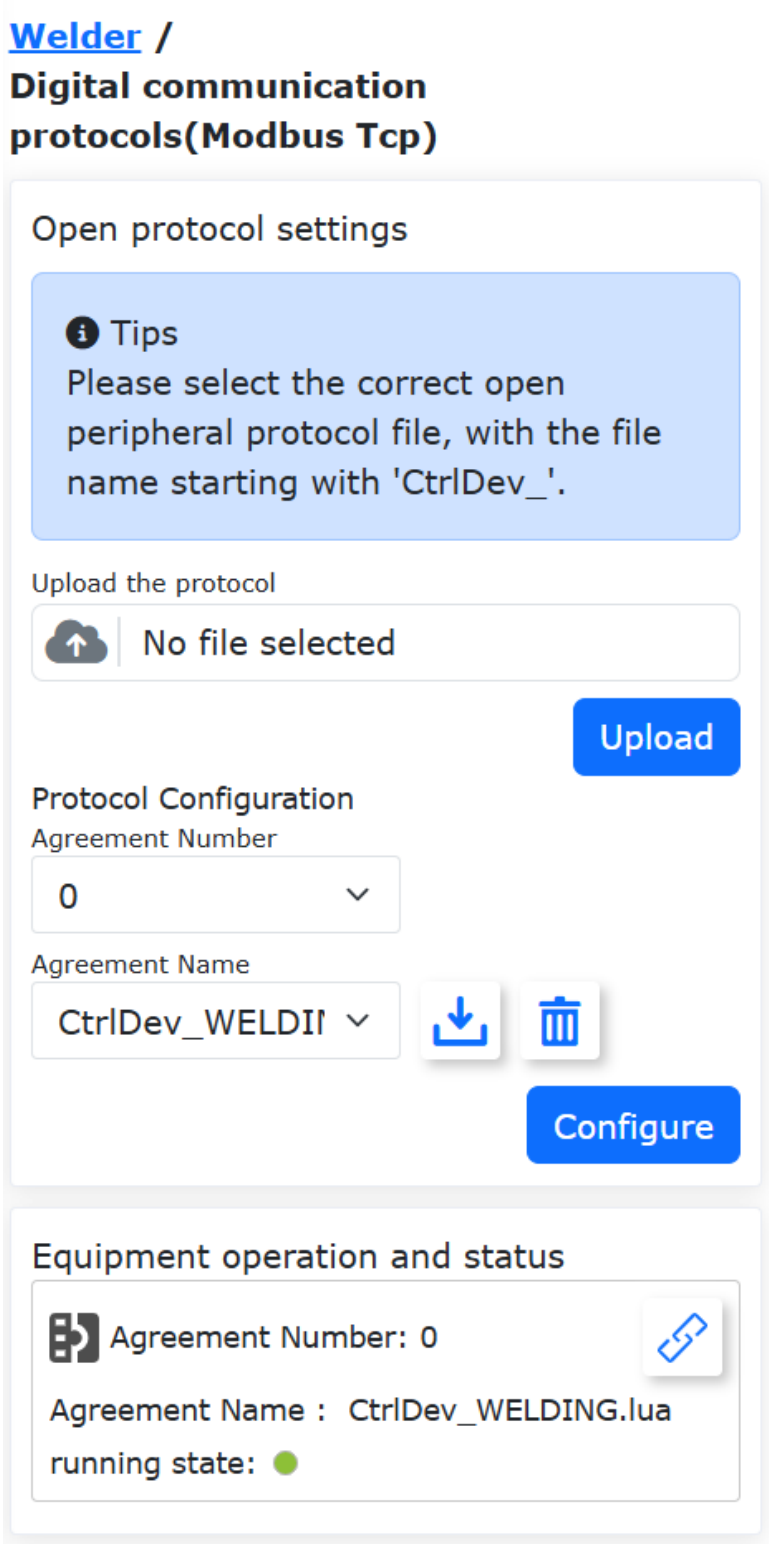
Click the “Welding” button. In the pop-up welding command addition page, select “Digital Communication Protocol (Modbus Tcp)”. Then successively select “Arc Start”, click “Add”, select “Arc Stop”, click “Add”, and finally click the “Apply” button.

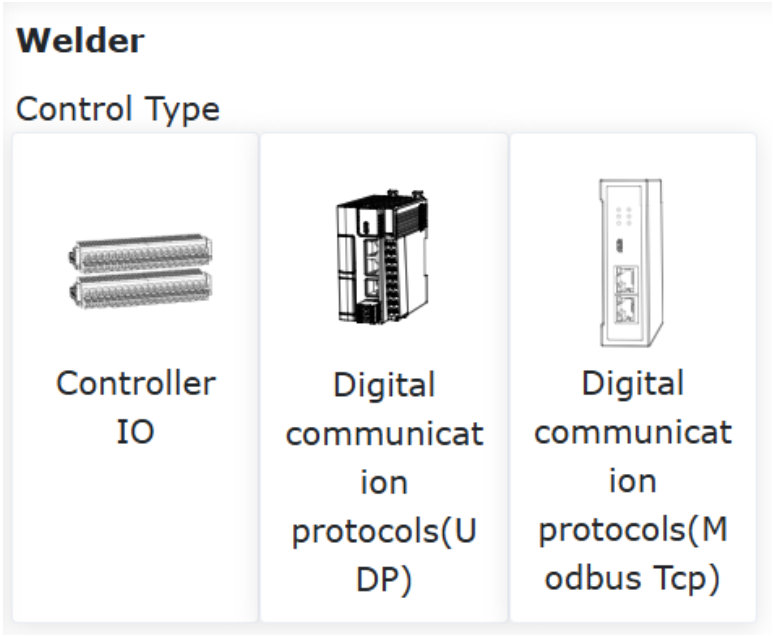
Figure 8.6-45 Add Arc Start and Arc Stop Commands

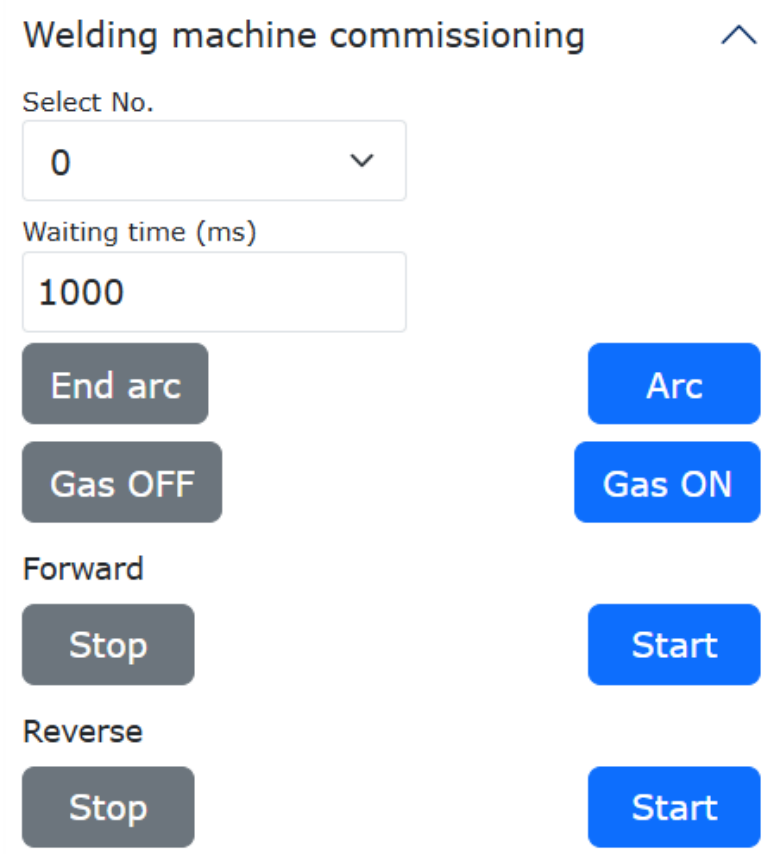
At this point, the Arc Start and Arc Stop commands have been successfully added to “testWeld.lua”.

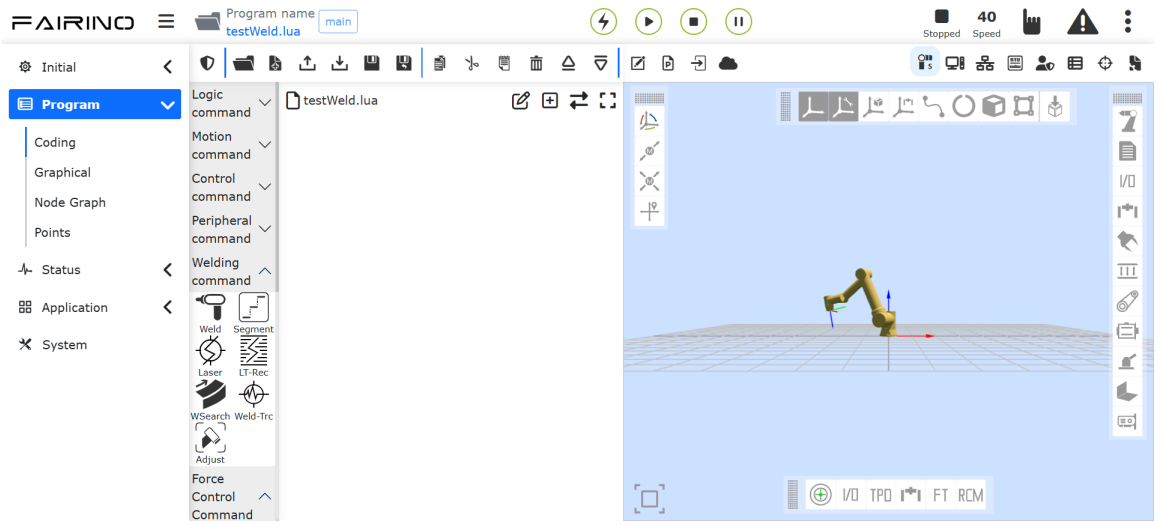
Figure 8.6-46 Added Arc Start and Arc Stop Commands

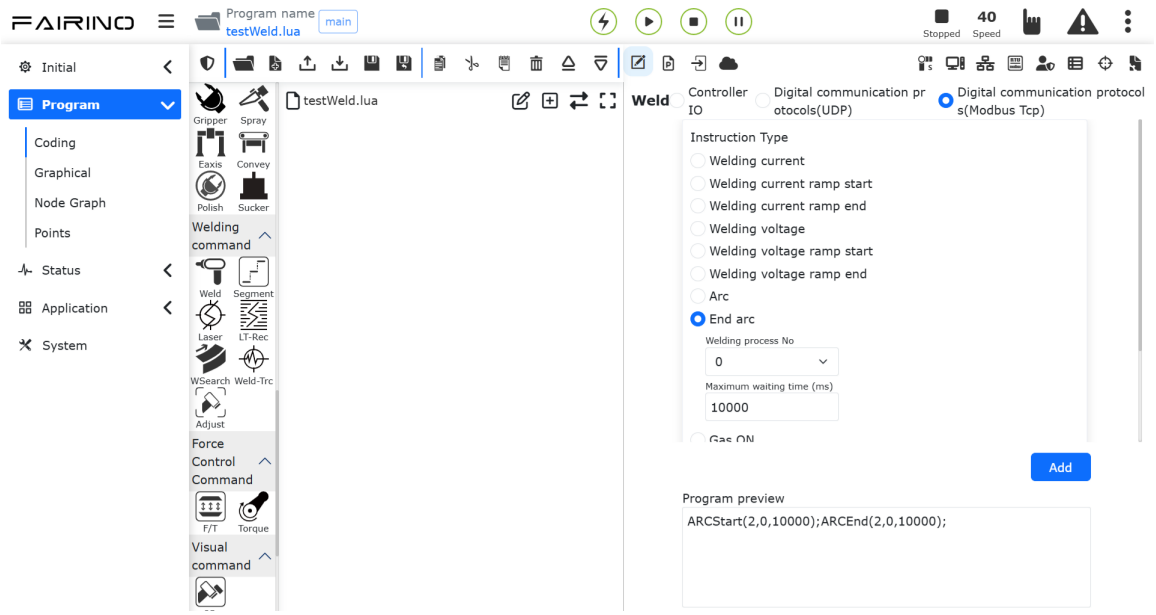

Welder /
Digital communication protocols(Modbus Tcp)
Open protocol settings
Tips
Please select the correct open peripheral protocol file, with the file name starting with 'CtrlDev_'.
Upload the protocol
No file selected
Upload
Protocol Configuration
Agreement Number
0
Agreement Name
CtrlDev_WELDI
Configure


Welder /
Digital communication protocols(Modbus Tcp)
Open protocol settings
Tips
Please select the correct open peripheral protocol file, with the file name starting with 'CtrlDev_'.
Upload the protocol
No file selected
Upload
Protocol Configuration
Agreement Number
0
Agreement Name
CtrlDev_WELDI
Configure
Equipment operation and status
Agreement Number: 0
Agreement Name : CtrlDev_WELDING.lua
running state:


Welder
Control Type
Controller IO
Digital communication protocols(UDP)
Digital communication protocols(Modbus Tcp)


Welding machine commissioning
Select No.
0
Waiting time (ms)
1000
End arc
Arc
Gas OFF
Gas ON
Forward
Stop
Start
Reverse
Stop
Start


FAIRINO
Program name
testWeld.lua
main
Stopped
40
Speed
Initial
Program
Coding
Graphical
Node Graph
Points
Status
Application
System
Logic command
Motion command
Control command
Peripheral command
Welding command
Weld
Segment
Laser
LT-Rec
WSearch
Weld-Trc
Adjust
Force Control Command
testWeld.lua
I/O
TPD
FT
RCM


FAIRINO
Program name
testWeld.lua
main
Stopped
40
Speed
Initial
Program
Coding
Graphical
Node Graph
Points
Status
Application
System
Gripper
Spray
Eaxis
Convey
Polish
Sucker
Welding command
Weld
Segment
Laser
LT-Rec
WSearch
Weld-Trc
Adjust
Force Control Command
F/T
Torque
Visual command
testWeld.lua
Weld
Controller IO
Digital communication protocols(UDP)
Digital communication protocols(Modbus Tcp)
Instruction Type
Welding current
Welding current ramp start
Welding current ramp end
Welding voltage
Welding voltage ramp start
Welding voltage ramp end
Arc
End arc
Welding process No
0
Maximum waiting time (ms)
10000
Gas ON
Add
Program preview
ARCStart(2,0,10000);ARCEnd(2,0,10000);


testWeld.lua
1 ARCStart(0,0,10000)
2 ARCEnd(0,0,10000)

Successively add the welding start point and welding end point. Switch the robot to automatic mode, and under safe conditions, start the program. The robot will then control the welding machine to perform welding along one weld seam.



Figure 8.6-47 Welding Program

1.3.8.6.6.8 Welding Machine Open Protocol Unload

Successively click “Initial”, “Peripherals”, “Control Box”, “Peripheral Open Protocol”. In the “Device Operation and Status” section, click the “Unload” button.

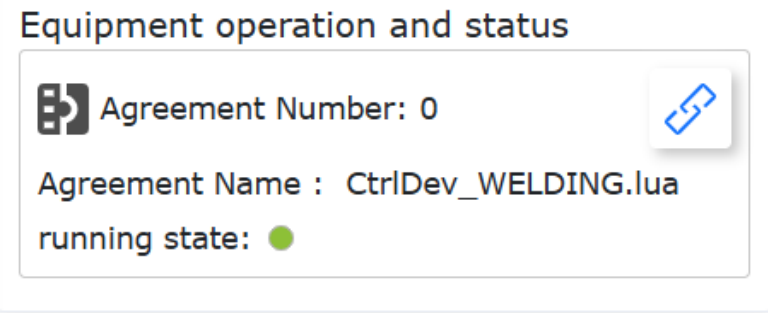


Figure 8.6-48 Unload Open Protocol

At this point, the protocol running status indicator turns off.

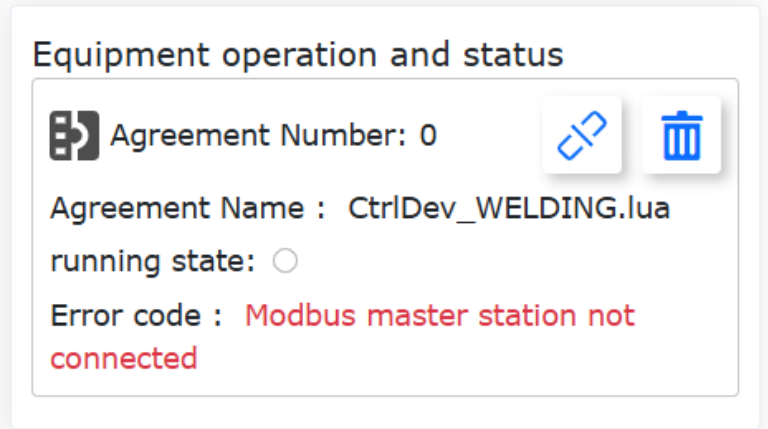


Figure 8.6-49 Open Protocol Unloaded

If welding debugging is performed or a welding program is executed at this time, the robot will report a "Protocol Not Loaded Error" in the lower left corner of the WebApp.

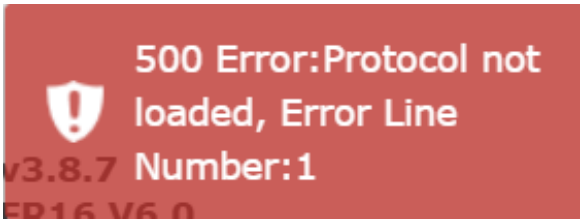


Figure 8.6-50 Protocol Not Loaded Error

1.3.8.7 Extended Axis Configuration

In "Initial" -> "Peripherals", click "Extended Axis" to enter the Extended Axis configuration interface, which includes Extended Axis Coordinate System configuration and Extended Axis Peripheral configuration. The interface when first entering Extended Axis configuration is as follows:

Figure 8.7-1 Initial Extended Axis Configuration Interface

Currently, Extended Axis Peripheral configuration is divided into the following two types based on communication method:

- Controller + PLC (UDP Communication).
- Controller + Servo Drive (485 Communication).

1.3.8.7.1 Extended Axis Coordinate System

The Extended Axis Coordinate System settings interface allows for the application, clearing, and configuration of the extended axis coordinate system.



Note:

Name: **Apply**

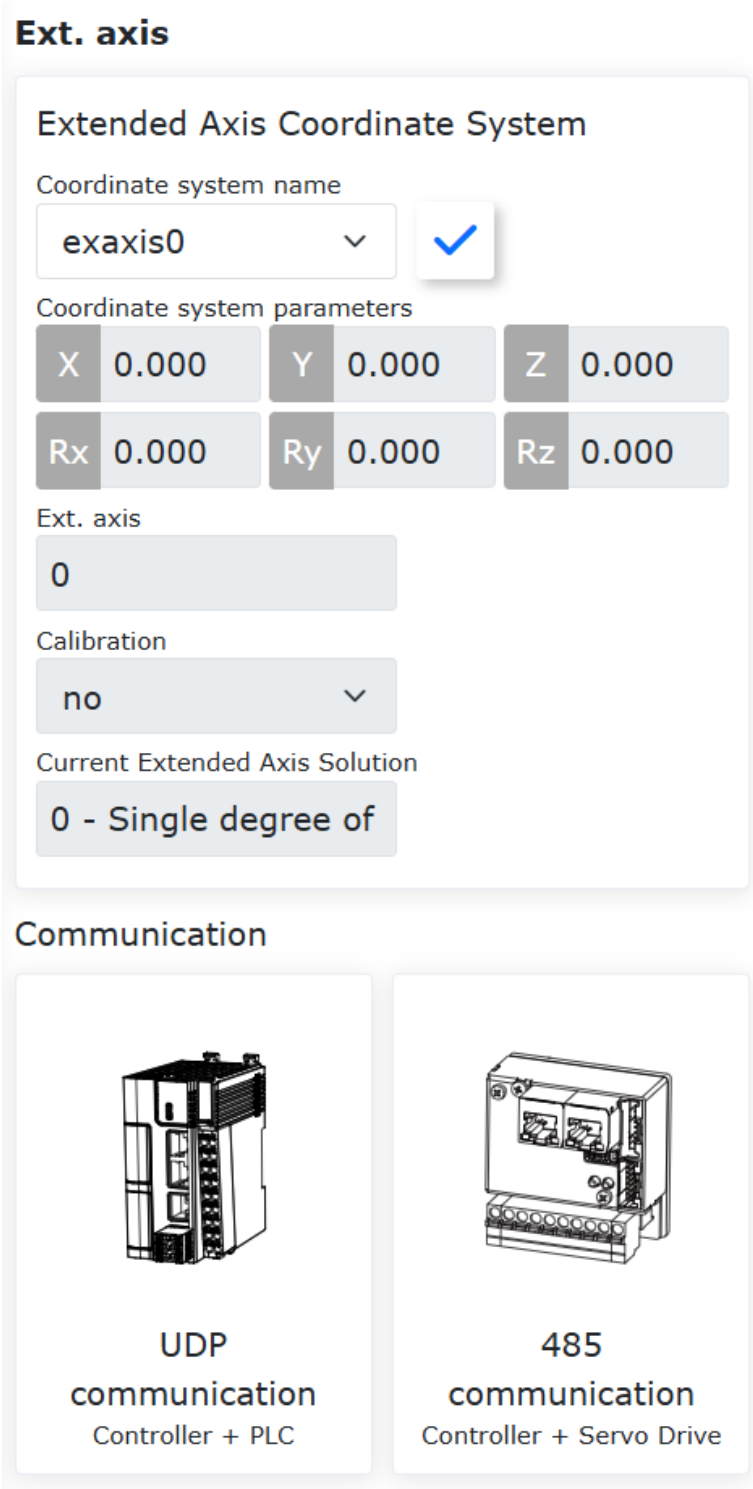
Function: Apply the extended axis coordinate system



Note:

Name: **Clear**

Function: Clear extended axis coordinate system data


Ext. axis
Extended Axis Coordinate System
Coordinate system name
exaxis0
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Ext. axis
0
Calibration
no
Current Extended Axis Solution
0 - Single degree of
Communication
UDP
communication
Controller + PLC
485
communication
Controller + Servo Drive

There are 5 numbers in the drop-down list for the Extended Axis Coordinate System, from exaxis0 to exaxis4. Selecting a corresponding coordinate system will display its coordinate values below. After selecting a coordinate system, click the "Apply" button, and the currently used extended axis coordinate system changes to the selected one, as shown below.

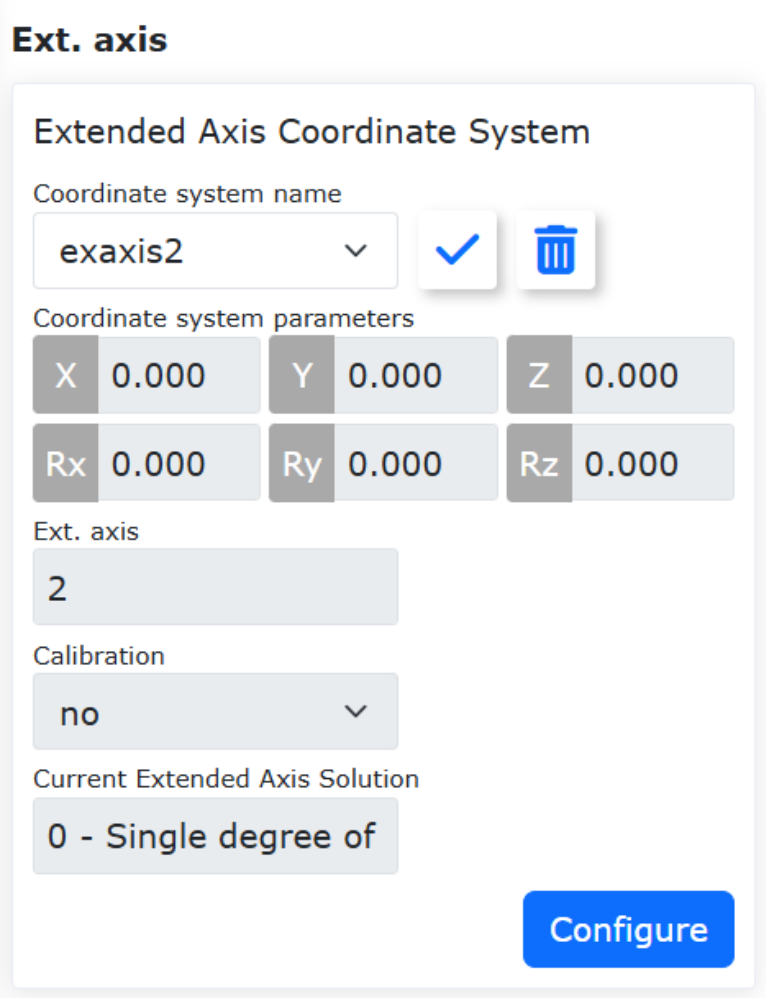


Figure 8.7-2 Extended Axis Coordinate System

Select an extended axis coordinate system other than "exaxis0", click "Configure" to enter the Extended Axis Coordinate System configuration interface to reset the extended axis coordinate system for that number. As shown below:

Important:

- Before calibration, first clear the extended axis coordinate system to be calibrated, and apply this extended axis coordinate system.
- Select the extended axis number. "Get Info" can retrieve the drive information for the corresponding extended axis, which can be used for parameter configuration.

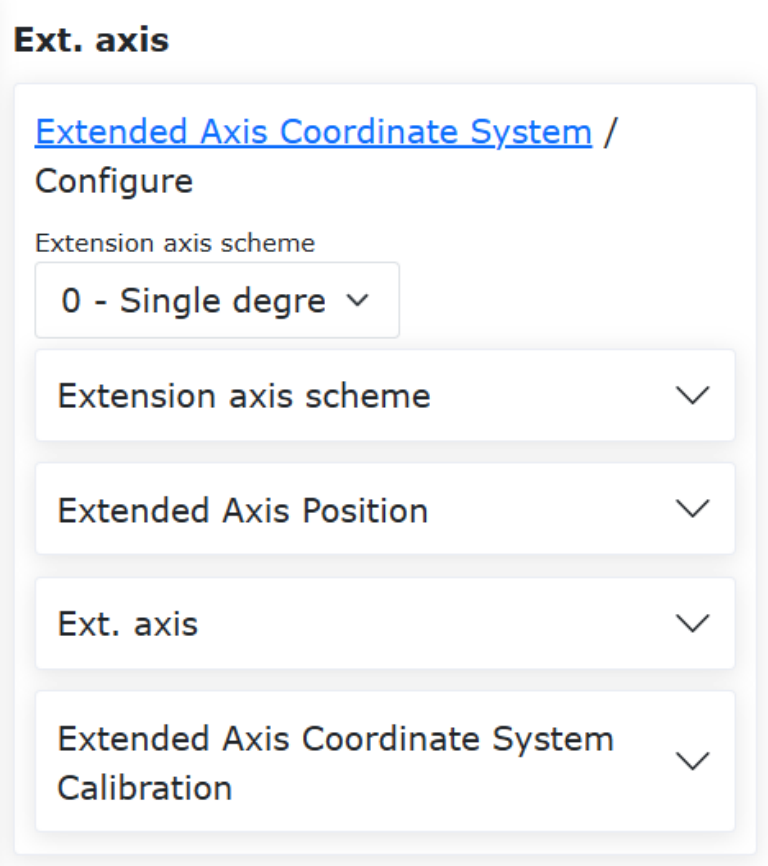


Figure 8.7-3 Extended Axis Coordinate System Calibration

The current extended axis solutions are as follows:

- 0 - Single Degree of Freedom Linear Slide
- 1 - Two Degree of Freedom L-type Positioner
- 2 - Three Degree of Freedom (Temporarily unavailable)
- 3 - Four Degree of Freedom (Temporarily unavailable)
- 4 - Single Degree of Freedom Positioner
- 5 - Two Degree of Freedom AGV

Single Degree of Freedom Linear Slide: First set the DH parameters, then set the robot's position relative to the extended axis (the linear slide is considered *on* the extended axis). If not calibrating, just click Save; at this point, the extended axis can only move asynchronously.

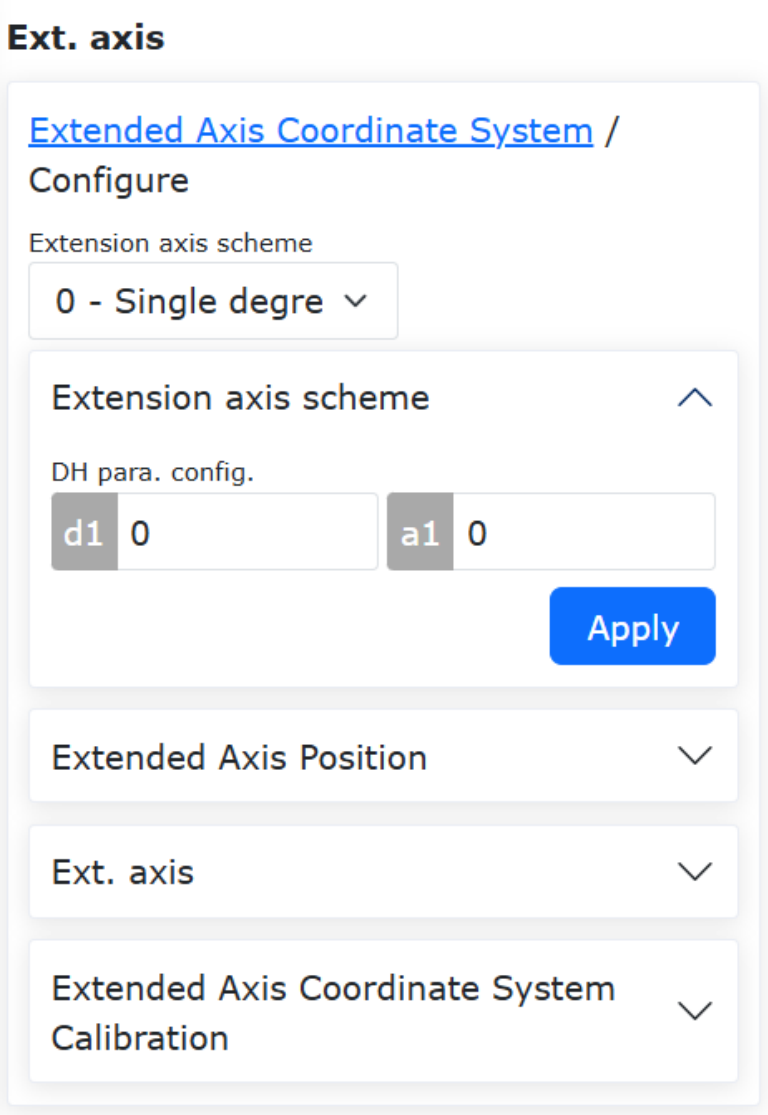


Figure 8.7-4 Linear Slide DH Parameter Configuration

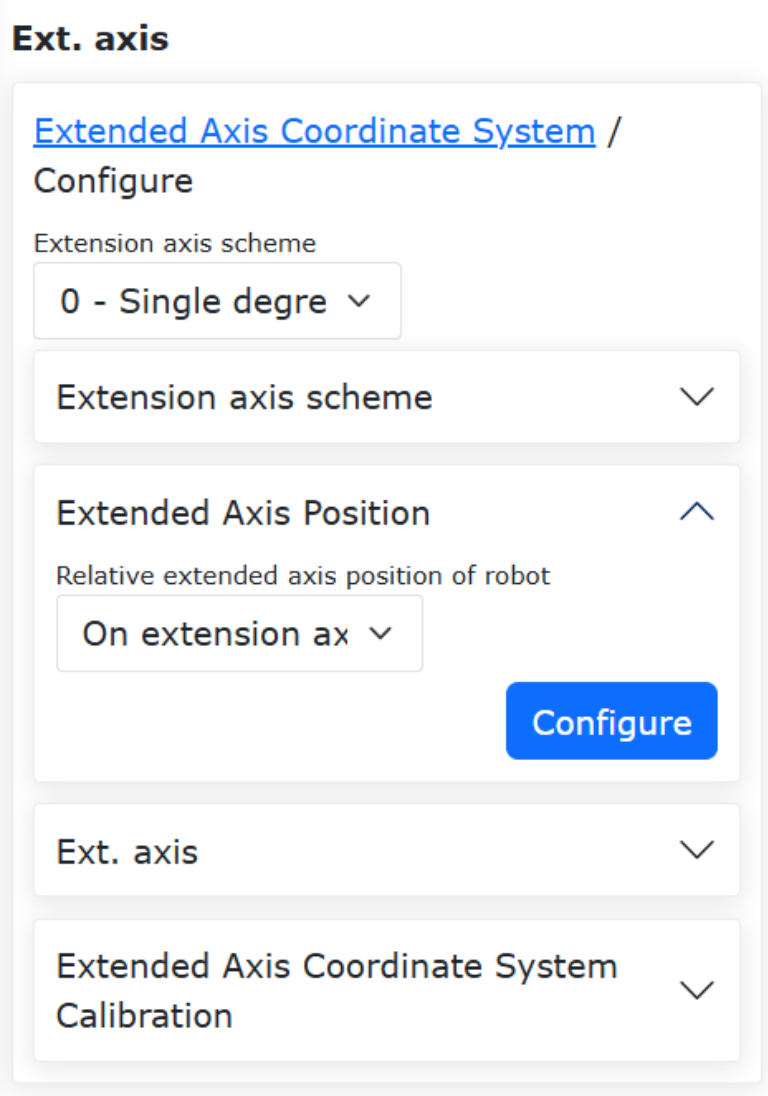


Figure 8.7-5 Linear Slide – Robot Position Relative to Extended Axis Configuration

If synchronous movement with the robot is required, at the extended axis zero point, click the operation area 'Eaxis' to enable the extended axis. Align the robot end center point (using the tool end point under the applied tool coordinate system) to a fixed point on the extended axis with two different postures, setting Point 1 and Point 2 respectively.

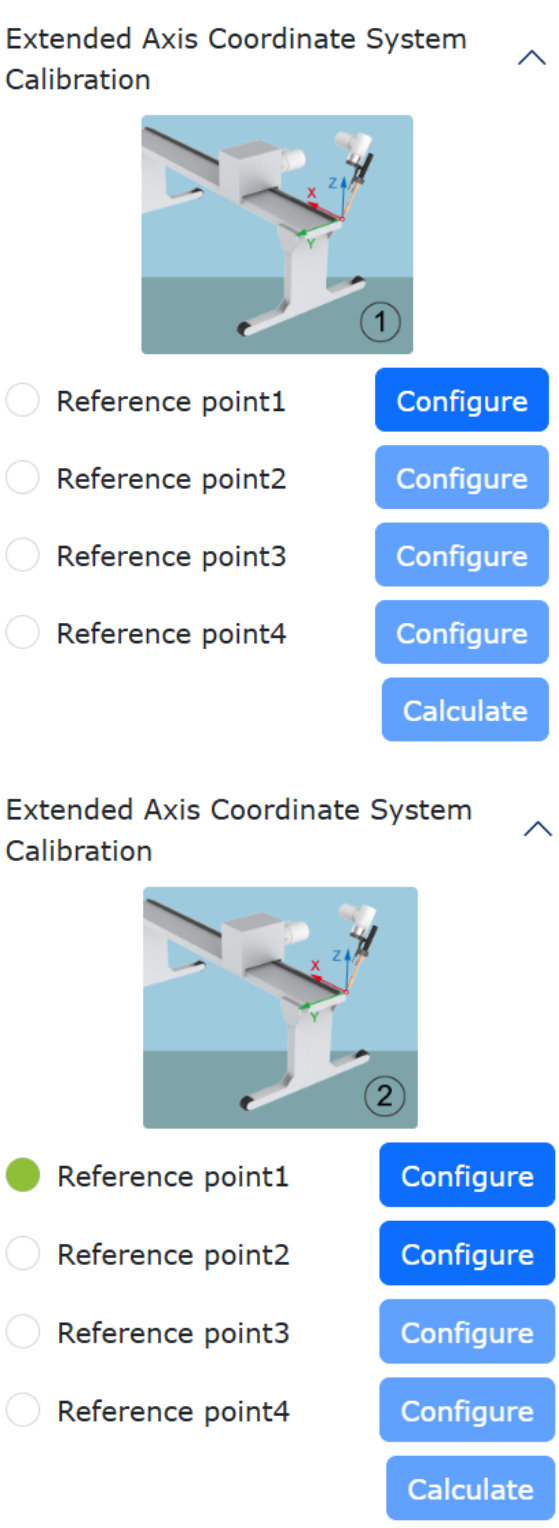


Figure 8.7-6 Linear Slide Calibration Points 1 and 2

Disable the axis, move the extended axis a certain distance, enable it again, and similarly align the robot end center point to the previous fixed point, setting Point 3. Disable the axis, move the extended axis back to the zero point, and enable the extended axis. Move the robot end center point to a point in space directly above the fixed point, setting Point 4. Calculate the coordinate system and save.

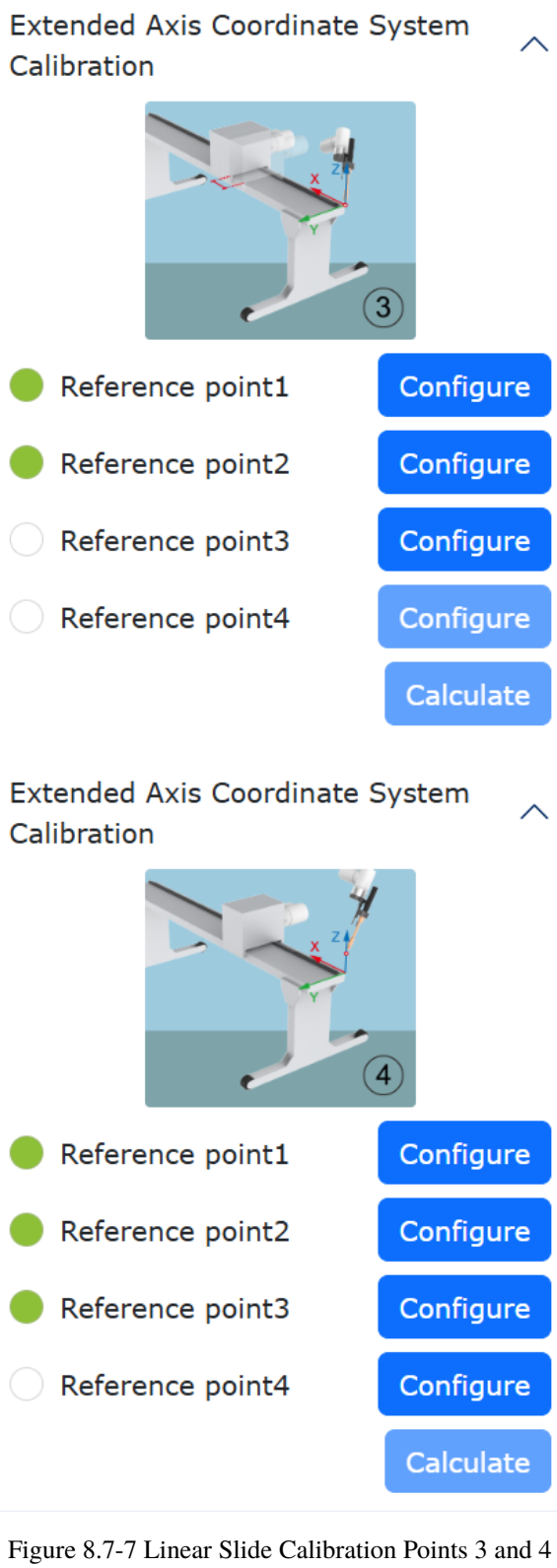


Figure 8.7-7 Linear Slide Calibration Points 3 and 4

Two Degree of Freedom L-type Positioner: The positioner consists of two extended axes. First set the DH parameters. Measure the DH parameters of the positioner according to the diagram and input them into the boxes. Set the robot's position relative to the extended axis (the positioner is considered *outside* the extended axis). If not calibrating, just

click Save; at this point, the extended axes can only move asynchronously.

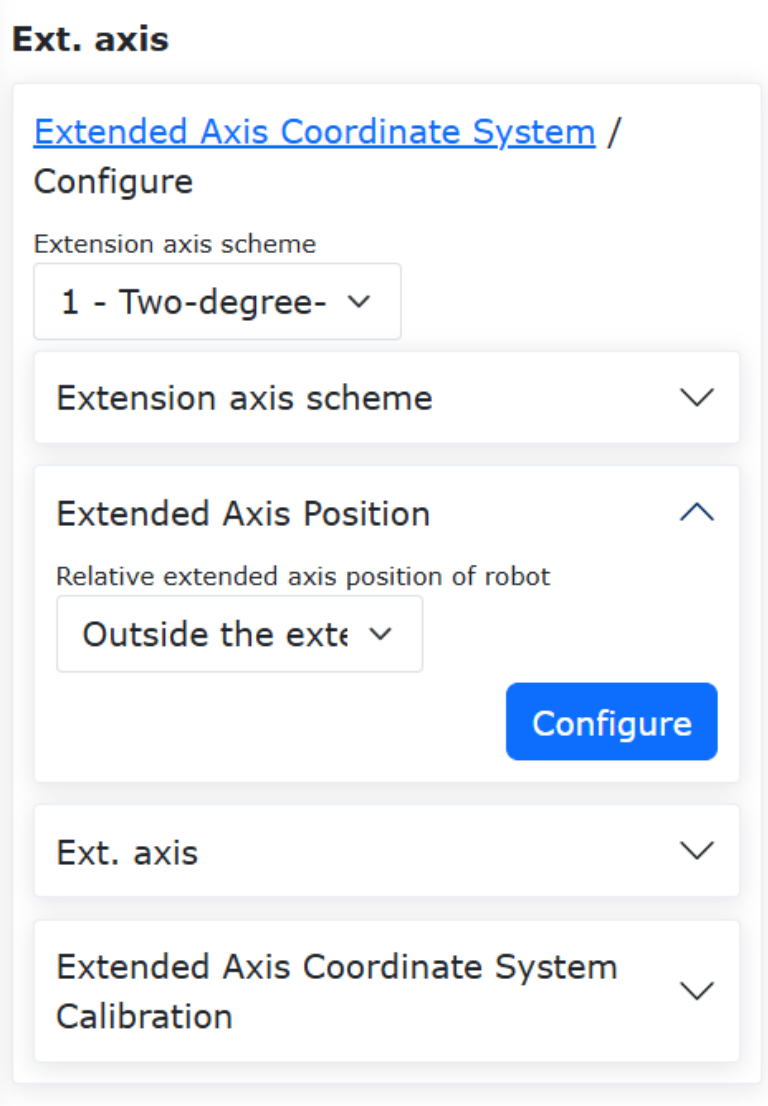


Figure 8.7-8 Two DOF L-type Positioner DH Parameter Configuration

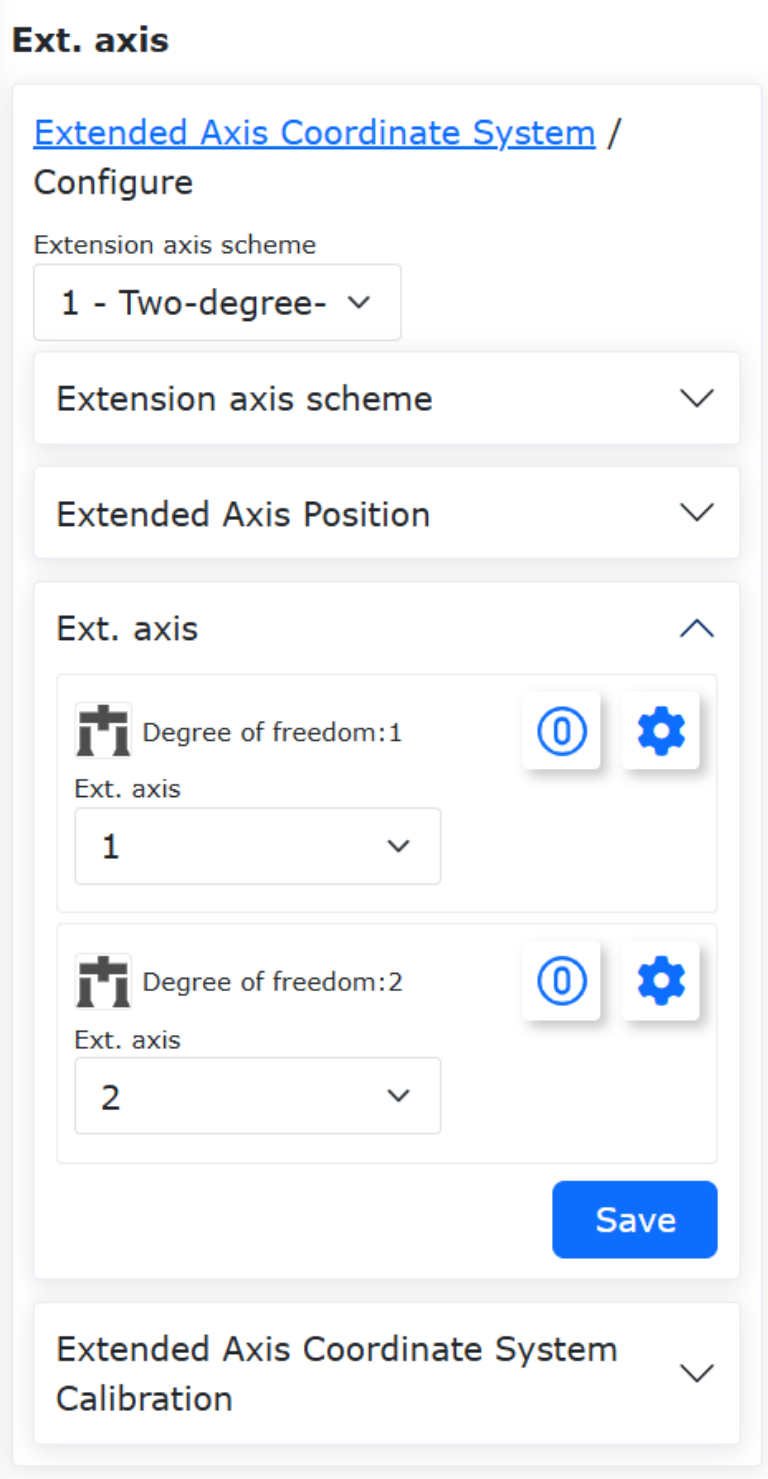


Figure 8.7-9 Two DOF L-type Positioner – Robot Position Relative to Extended Axis

If synchronous movement with the robot is required, at the extended axes zero point, click the operation area 'Eaxis' to enable the extended axes. Establish a coordinate system on the positioner. Select a point and input its Cartesian pose in that coordinate system. For example, selecting a point in the positive Y direction, measuring Y as 100mm, input the values as shown in the figure. Click "Reference Point" to set the reference point. The subsequent four calibration points all require aligning the robot end center point (using the tool end point under the applied tool coordinate system)

to this reference point.

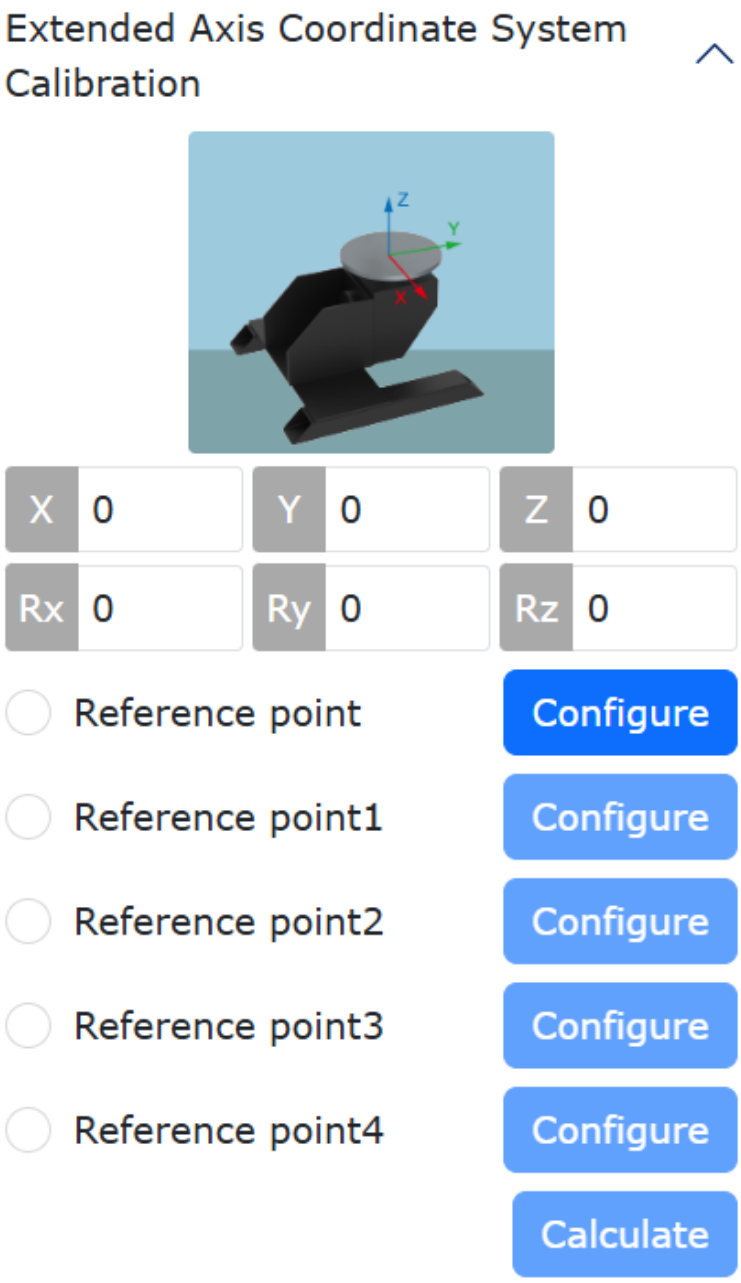
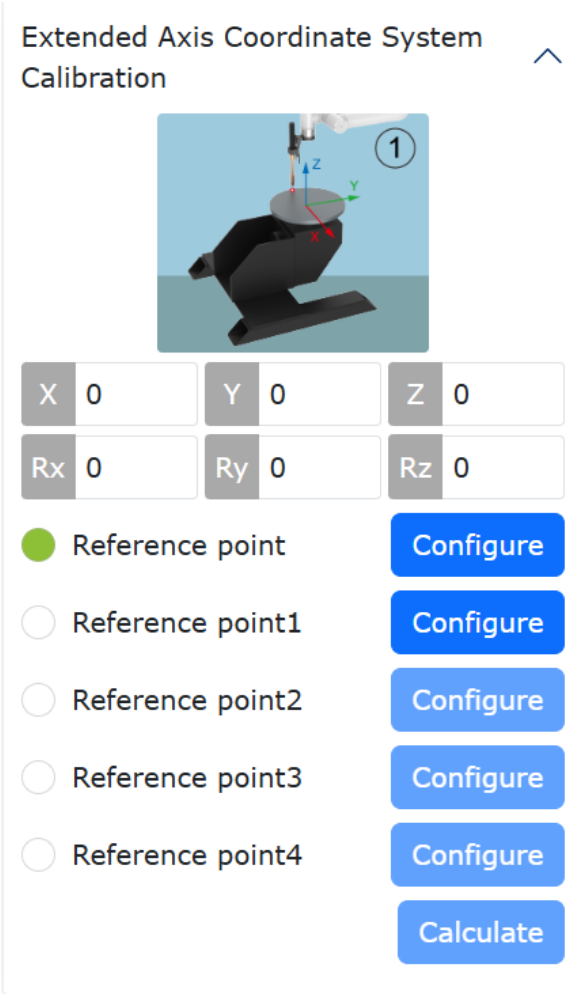
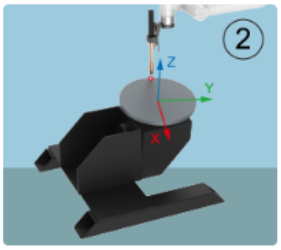


Figure 8.7-10 Two DOF L-type Positioner – Reference Point Configuration

Align the robot end center point (using the tool end point under the applied tool coordinate system) to this reference point and set Point 1. Click the operation area 'Eaxis' to jog the two axes a small distance, align the robot end center point to the reference point, and set Point 2. Continue jogging the two axes, align the robot end center point to the reference point, and set Point 3. Finally, continue jogging the two axes, align the robot end center point to the reference point, and set Point 4. Click "Calculate" to get the coordinate system result, then click "Save" and "Apply".


Extended Axis Coordinate System Calibration
1
Z
Y
X
X 0
Y 0
Z 0
Rx 0
Ry 0
Rz 0
Reference point
Configure
Reference point1
Configure
Reference point2
Configure
Reference point3
Configure
Reference point4
Configure
Calculate

Extended Axis Coordinate System Calibration



X	0	Y	0	Z	0
Rx	0	Ry	0	Rz	0

Reference point — Configure

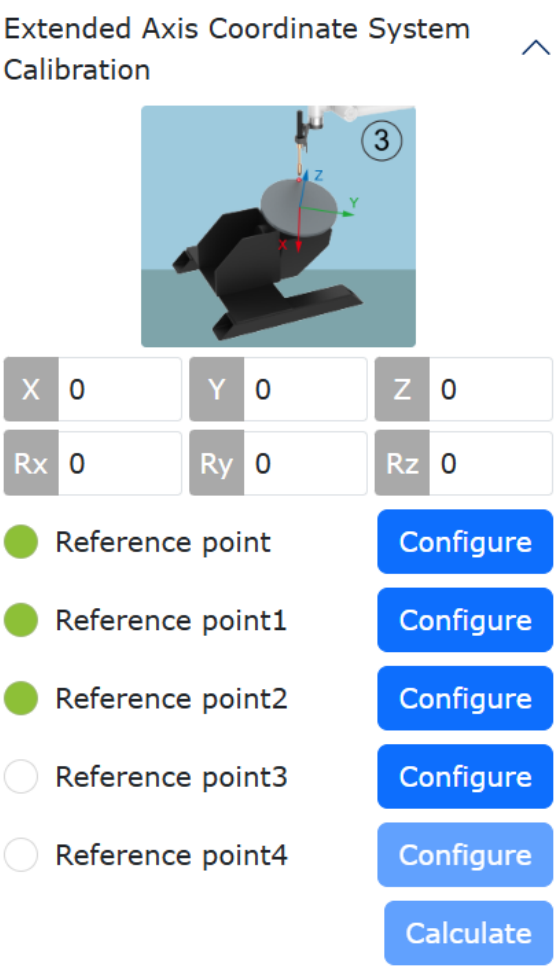
Reference point1 — Configure

Reference point2 — Configure

Reference point3 — Configure

Reference point4 — Configure

Calculate


Extended Axis Coordinate System Calibration
3
Z
Y
X
X 0
Y 0
Z 0
Rx 0
Ry 0
Rz 0
Reference point
Configure
Reference point1
Configure
Reference point2
Configure
Reference point3
Configure
Reference point4
Configure
Calculate

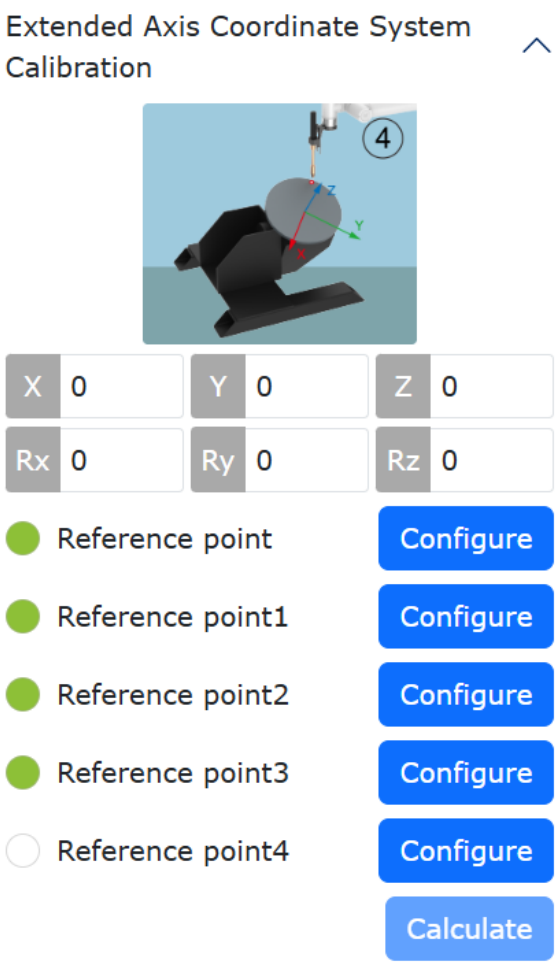


Figure 8.7-11 Two DOF L-type Positioner Calibration

Single Degree of Freedom Positioner: Consists of one rotating extended axis. Set the DH parameters to 0. Set the robot's position relative to the extended axis as *outside* the extended axis. If not calibrating, just click Save; at this point, the extended axis can only move asynchronously.

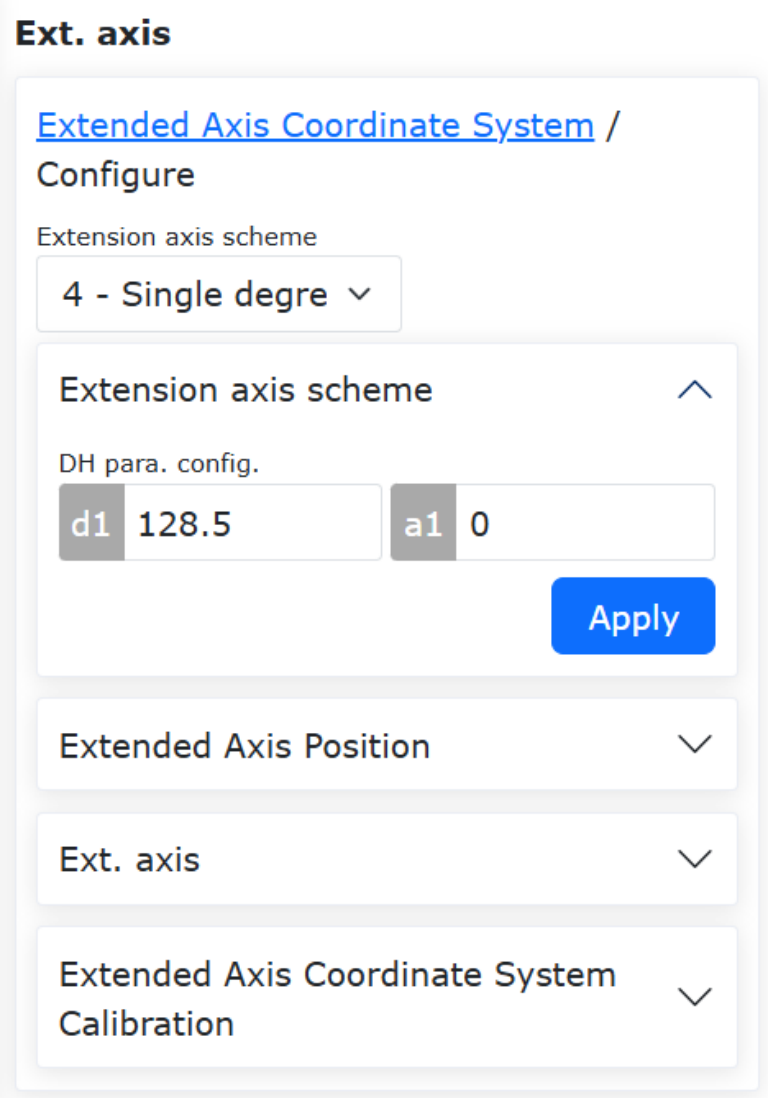


Figure 8.7-12 Single DOF Positioner DH Parameter Configuration

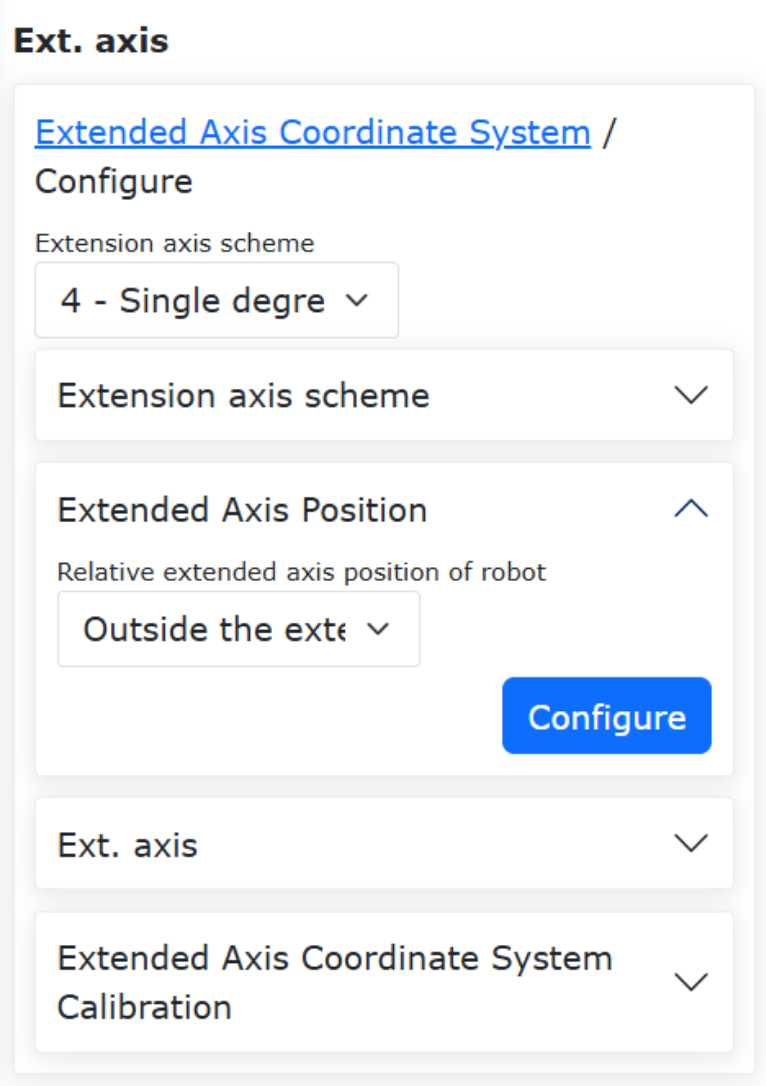


Figure 8.7-13 Single DOF Positioner – Robot Position Relative to Extended Axis

If synchronous movement with the robot is required, at the extended axis zero point, click the operation area 'Eaxis' to enable the extended axis. Establish a coordinate system on the positioner. Select a point and input its Cartesian pose in that coordinate system. Click "Reference Point" to set the reference point.

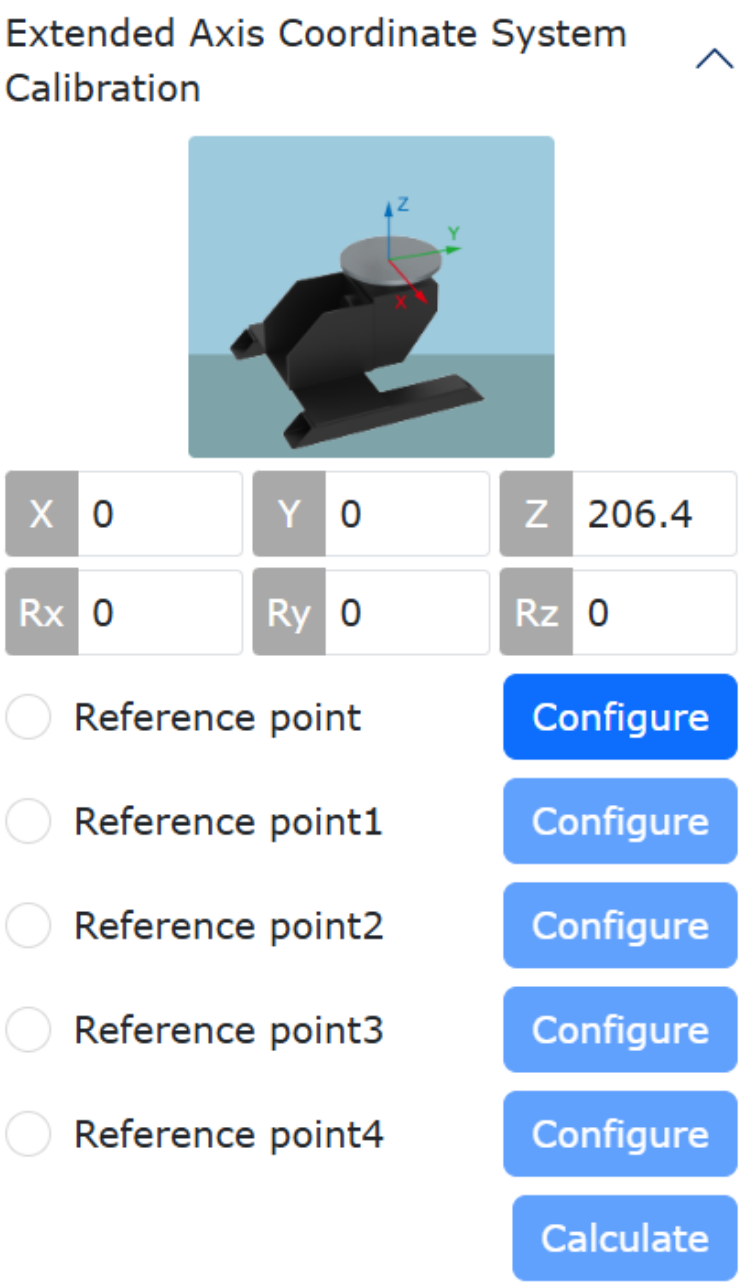
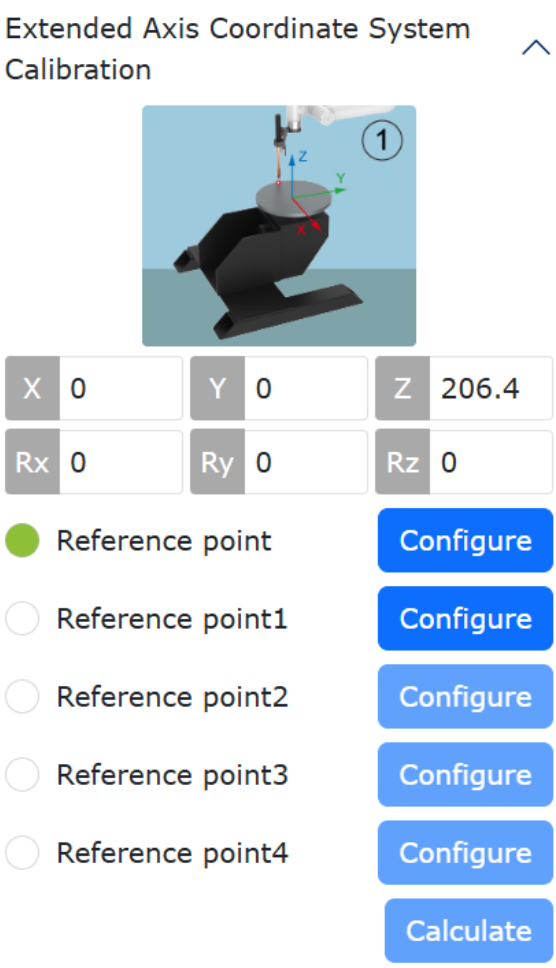
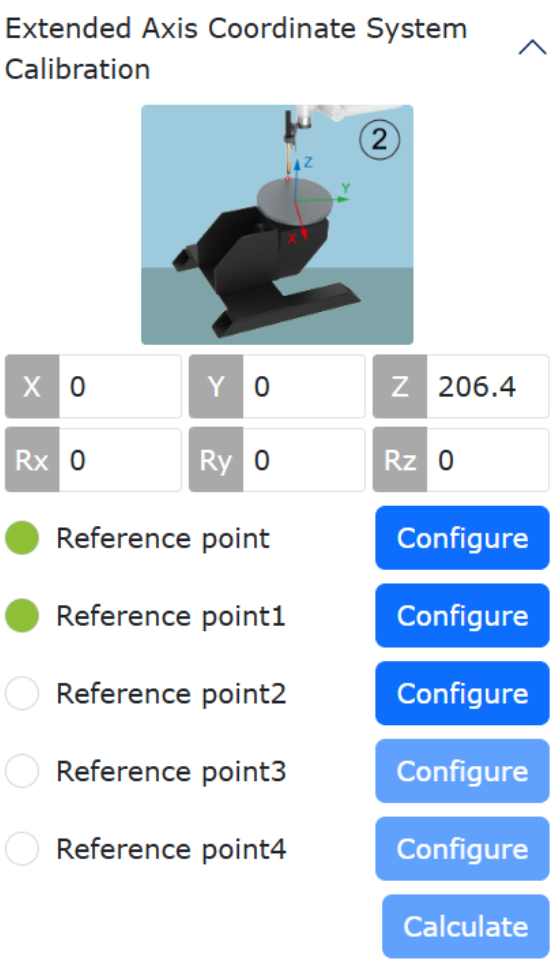
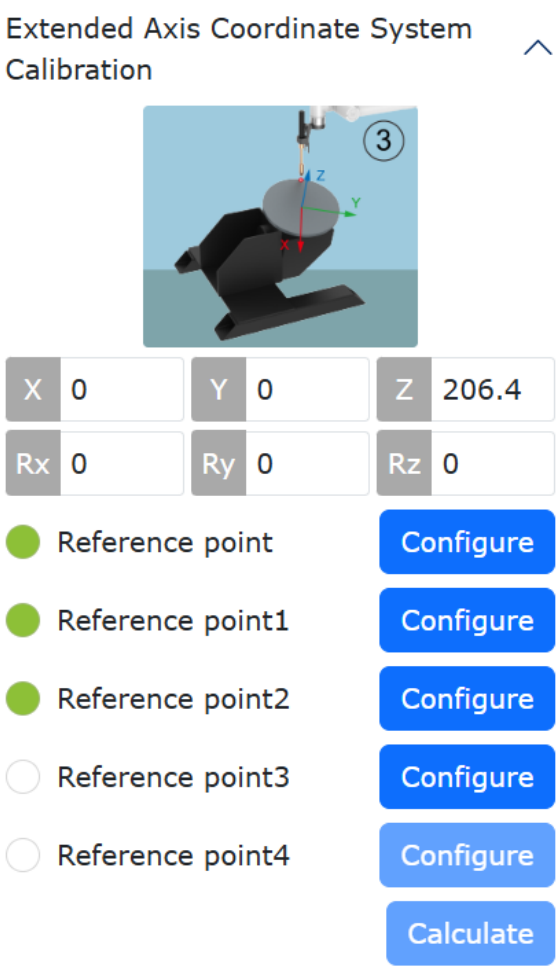


Figure 8.7-14 Single DOF Positioner Reference Point Configuration

The subsequent four calibration points all require aligning the robot end center point (using the tool end point under the applied tool coordinate system) to this reference point. Align the robot end center point to this reference point and set Point 1. Click the operation area 'Eaxis' to jog the rotation axis a small distance, align the robot end center point to the reference point, and set Point 2. Continue jogging the rotation axis, align the robot end center point to the reference point, and set Point 3. Finally, continue jogging the rotation axis, align the robot end center point to the reference point, and set Point 4. Click "Calculate" to get the coordinate system result, then click "Save" and "Apply".


Extended Axis Coordinate System Calibration
1
Z
Y
X
X 0
Y 0
Z 206.4
Rx 0
Ry 0
Rz 0
Reference point
Configure
Reference point1
Configure
Reference point2
Configure
Reference point3
Configure
Reference point4
Configure
Calculate


Extended Axis Coordinate System Calibration
2
Z
Y
X
X 0
Y 0
Z 206.4
Rx 0
Ry 0
Rz 0
Reference point
Configure
Reference point1
Configure
Reference point2
Configure
Reference point3
Configure
Reference point4
Configure
Calculate


Extended Axis Coordinate System Calibration
3
Z
Y
X
X 0
Y 0
Z 206.4
Rx 0
Ry 0
Rz 0
Reference point
Configure
Reference point1
Configure
Reference point2
Configure
Reference point3
Configure
Reference point4
Configure
Calculate

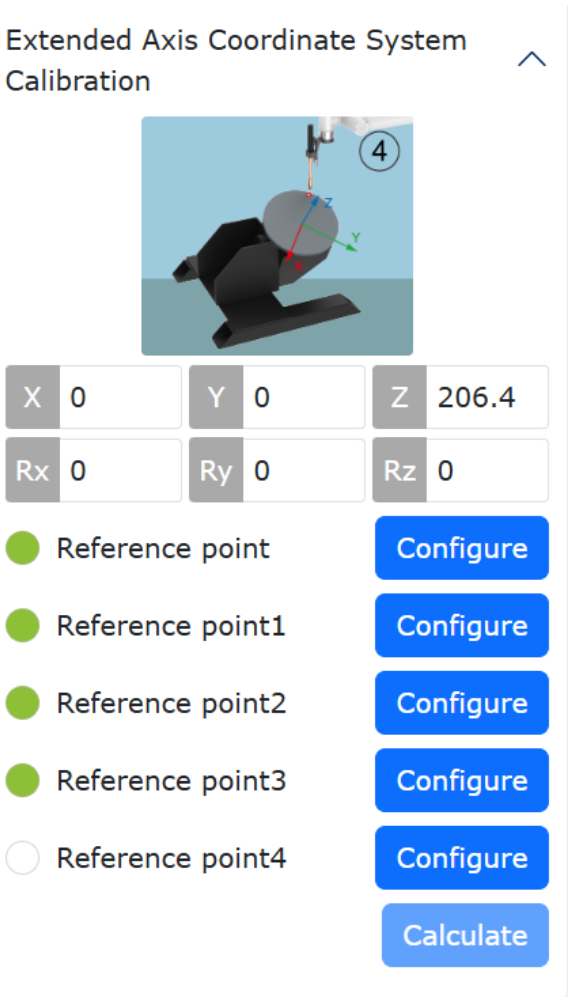


Figure 8.7-15 Single DOF Positioner Calibration

Important:

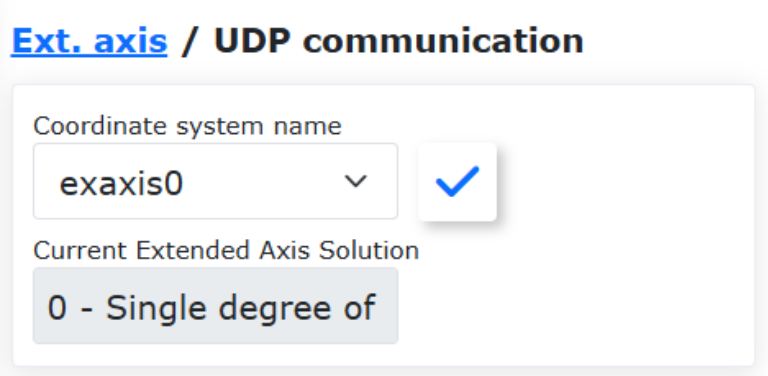
1. The extended axis coordinate system is calibrated based on the tool coordinate system and needs to be established on the basis of an already created tool coordinate system.
2. Extended axis systems generally use exaxis1~exaxis4. Applying exaxis0 means no extended axis coordinate system is used. When calibrating an extended axis coordinate system, first apply the extended axis coordinate system to exaxis0, then select another extended axis coordinate system for calibration and application.

1.3.8.7.2 Controller + PLC (UDP Communication)

Before using the extended axis UDP communication method, it is necessary to first establish the corresponding extended axis coordinate system, configure the corresponding extended axis solution under that coordinate system, and apply the established tool coordinate system during program teaching. The extended axis function is primarily used in conjunction with the welding machine function and the laser tracking sensor function.

Figure 8.7-16 Extended Axis Coordinate System Application and Current Extended Axis Solution Display

When only needing to modify the current extended axis coordinate system, select the coordinate system in the Peripheral Extended Axis configuration interface to apply it. When needing to change the extended axis solution, enter the



Extended Axis Coordinate System configuration interface to modify it.

When the extended axis solution is "0 - Single Degree of Freedom Linear Slide", "1 - Two Degree of Freedom L-type Positioner", "2 - Three Degree of Freedom", "3 - Four Degree of Freedom", or "4 - Single Degree of Freedom Positioner", after successful UDP communication configuration, the interface displays "UDP Extended Axis" and "Positioning Completion Time Setting" content. When the extended axis solution is "5 - Two Degree of Freedom AGV", the interface displays "Two Degree of Freedom AGV Test" content.

1.3.8.7.2.1 UDP Communication Configuration

Note: 

Name: **Edit Button**

Function: UDP communication parameter configuration

Note: 

Name: **Load Button**

Function: Load UDP communication

Step1: Configure extended axis UDP communication parameters: Set IP Address, Port Number, Communication Cycle, Packet Loss Detection Cycle, Packet Loss Count, etc. The Reconnection Cycle and Reconnection Count can only be configured after the Communication Interruption Auto-reconnect switch is turned on.

- IP Address: Custom IP address;
- Port Number: Define according to the actual situation;
- Communication Cycle: Define according to the actual situation, unit ms;
- Packet Loss Detection Communication Cycle: 10 ~ 1000 ms;

- Packet Loss Count: 1 ~ 100;
- Communication Interruption Confirmation Duration: 0 ~ 500 ms;
- Power-off Restart Auto-reconnect: On/Off;
- Communication Interruption Auto-reconnect: On/Off;
- Reconnection Cycle: 1 ~ 1000 ms;
- Reconnection Count: 1 ~ 100;

Figure 8.7-17 Extended Axis UDP Communication Parameter Configuration

---

Important:

1. After setting the Communication Disconnect Confirmation Duration, communication disconnect is confirmed and an error is reported only when the communication abnormality exceeds this duration.
2. After UDP communication is disconnected, a UDP disconnect error is triggered (can be reset). Click the Clear Warning Information button, and UDP communication will be re-established.

Step2: After successful communication parameter configuration, click the "Load" button to establish UDP communication. After successful communication, the button in front of "UDP Communication Configuration" turns green. Check the extended axis status in the robot's various status views; the extended axis should be servoed and in position.

Figure 8.7-18 Extended Axis UDP Communication Established

Figure 8.7-19 Extended Axis Servo Ready

Important:

1. When UDP communication is not established, UDP extended axis number information cannot be configured or viewed.
2. Before loading extended axis UDP communication, be sure to first configure and apply an extended axis coordinate system other than number 0.

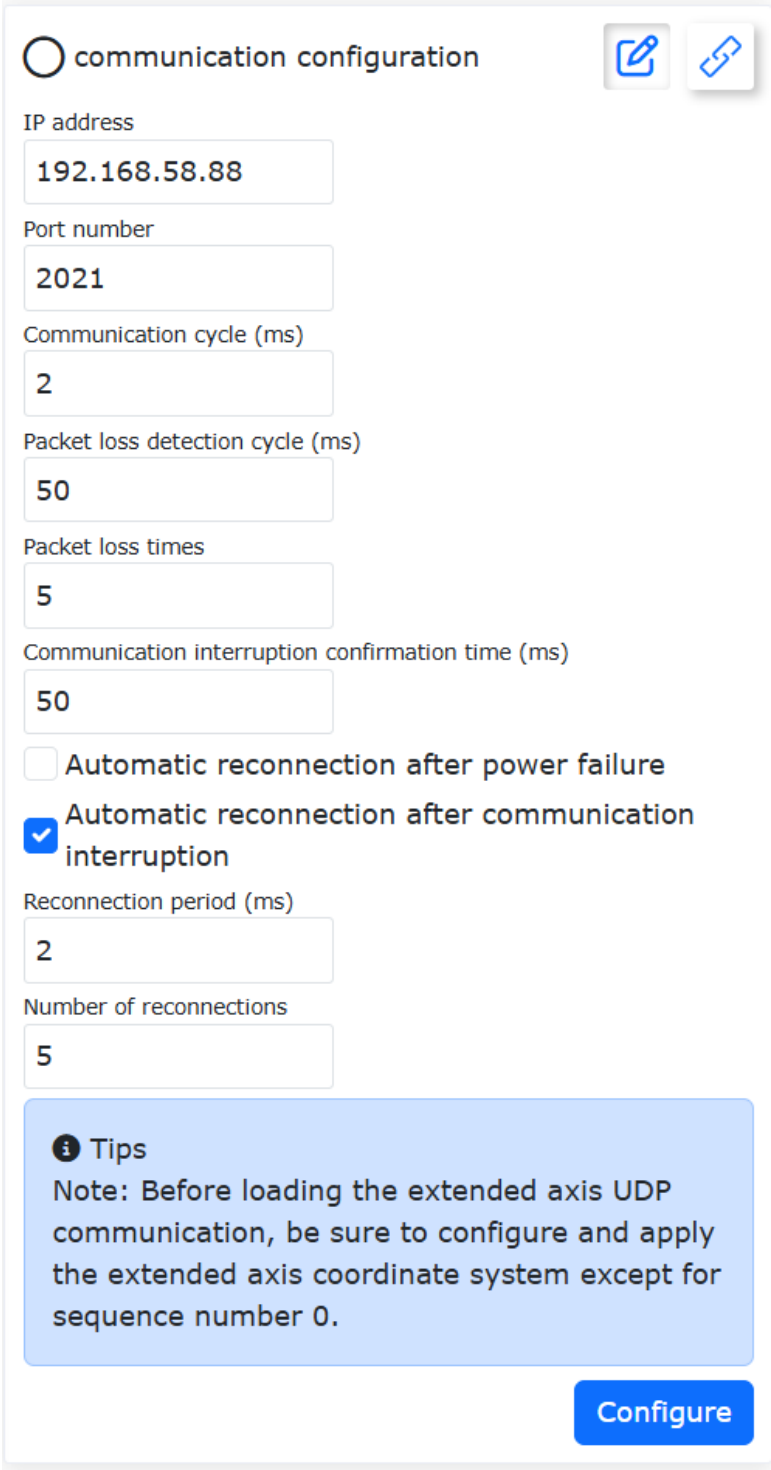
1.3.8.7.2.2 UDP Extended Axis

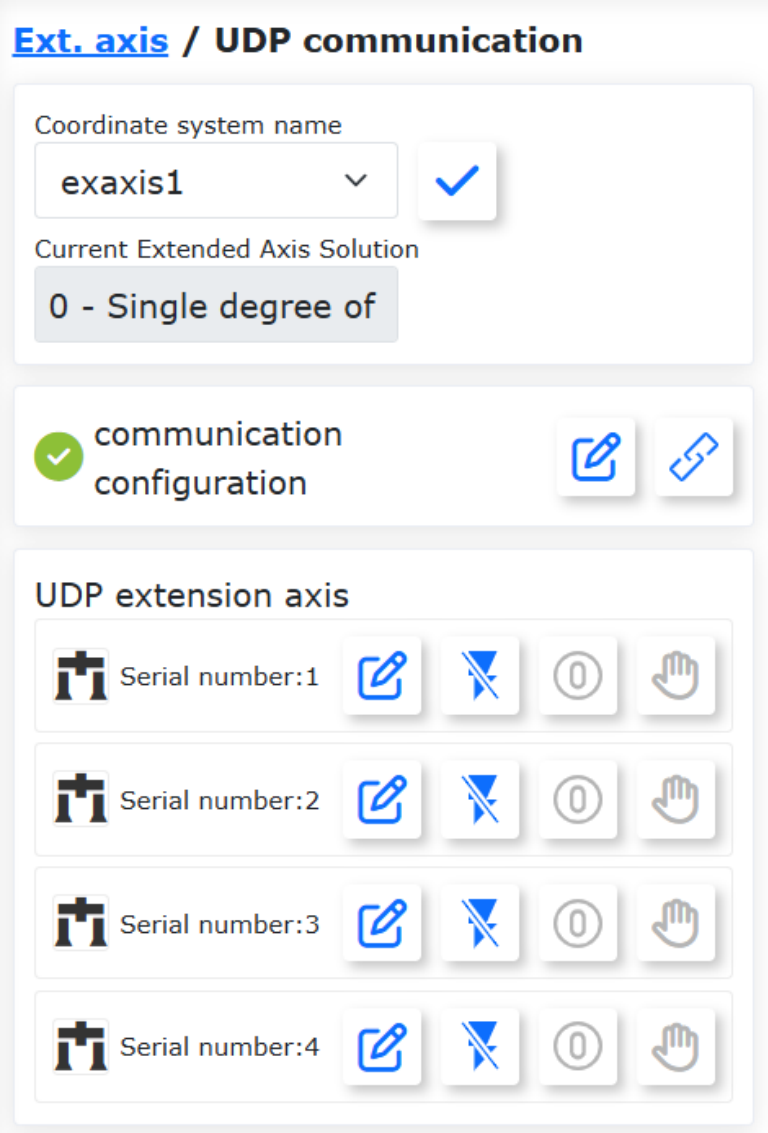
Note: 

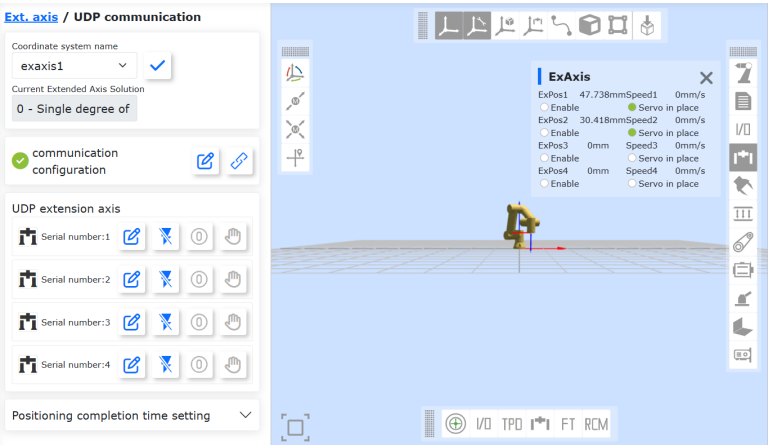
Name: **Edit Button**

Function: Extended axis parameter configuration

Note: 


communication configuration
IP address
192.168.58.88
Port number
2021
Communication cycle (ms)
2
Packet loss detection cycle (ms)
50
Packet loss times
5
Communication interruption confirmation time (ms)
50
Automatic reconnection after power failure
Automatic reconnection after communication interruption
Reconnection period (ms)
2
Number of reconnections
5
Tips
Note: Before loading the extended axis UDP communication, be sure to configure and apply the extended axis coordinate system except for sequence number 0.
Configure


Ext. axis / UDP communication
Coordinate system name
exaxis1
Current Extended Axis Solution
0 - Single degree of
communication configuration
UDP extension axis
Serial number:1
Serial number:2
Serial number:3
Serial number:4


Ext. axis / UDP communication
Coordinate system name
exaxis1
Current Extended Axis Solution
0 - Single degree of
communication configuration
UDP extension axis
Serial number:1
Serial number:2
Serial number:3
Serial number:4
Positioning completion time setting
ExAxis
I/O TPD FT RCM

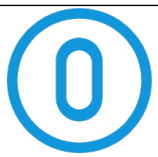
Name: **Enable Button**

Function: Extended axis enabled state. Clicking the button disables the extended axis.

Note: 

Name: **Disable Button**

Function: Extended axis disabled state. Clicking the button enables the extended axis.

Note: 

Name: **Homing Button**

Function: Set extended axis homing method

Note: 

Name: **Test Button**

Function: Extended axis function test

Step1: Select any extended axis number (currently only numbers 1, 2, 3, 4 are available). Click the "Edit" button behind the extended axis number to enter the detailed configuration interface. Set Axis Type, Axis Direction, Operating Speed, Acceleration, Positive Limit, Negative Limit, Lead, Encoder Resolution, Start Offset, Manufacturer, Model, and Mode. Click "Configure" to complete the configuration.

- Axis Type: Linear Slide, Rotary Axis, Infinite Rotary Axis;
- Axis Direction: Positive / Negative;
- Operating Speed: 0~2000 mm/s;
- Acceleration: 0 ~ 2000 mm/s^2;
- Positive Limit: 0 ~ 50000;
- Negative Limit: -50000 ~ 0;
- Lead: 0~1000;
- Encoder Resolution: 0 ~ 10000000;
- Start Offset: 0 ~ 10000 mm;
- Manufacturer: Hechuan, Huichuan, Panasonic;

- Model: Automatically matched model list based on the manufacturer;
- Mode: Incremental System and Absolute Position System;

Figure 8.7-20 Extended Axis Parameter Configuration

Step2: After completing the extended axis parameter configuration, click the "Disable" button to enable the corresponding extended axis number. After successful enabling, the homing method can be set and the extended axis can be tested. When the extended axis is not enabled, the homing method setting and extended axis test cannot be performed.

Figure 8.7-21 Extended Axis Enable/Disable

Step3: If the extended axis is not successfully enabled, the button to enter the settings interface is grayed out. After the extended axis is successfully enabled, click the "Homing" button to enter the homing method setting interface. Set the Homing Method, Homing Speed, and Zero Latch Speed. Click the "Set" button, and the extended axis will start homing. The homing status will be displayed in the blank area below the Axis Direction. When the "Homing Completed" prompt appears, it indicates successful extended axis zero point setting.

- Homing Method: Current Position Homing, Negative Limit Homing, Positive Limit Homing;
- Homing Speed: 0~2000 mm/s;
- Zero Latch Speed: 0~2000 mm/s;

Figure 8.7-22 Homing Method Setting

Step4: If the extended axis is not successfully enabled, the button to enter the settings interface is grayed out. After the extended axis is successfully enabled and the homing method is set, click the "Test" button to enter the extended axis test interface. Set the Running Speed, Acceleration, and Maximum Distance to test the extended axis with forward rotation and reverse rotation. During rotation, you can click the "Stop" button to test if the extended axis can stop normally.

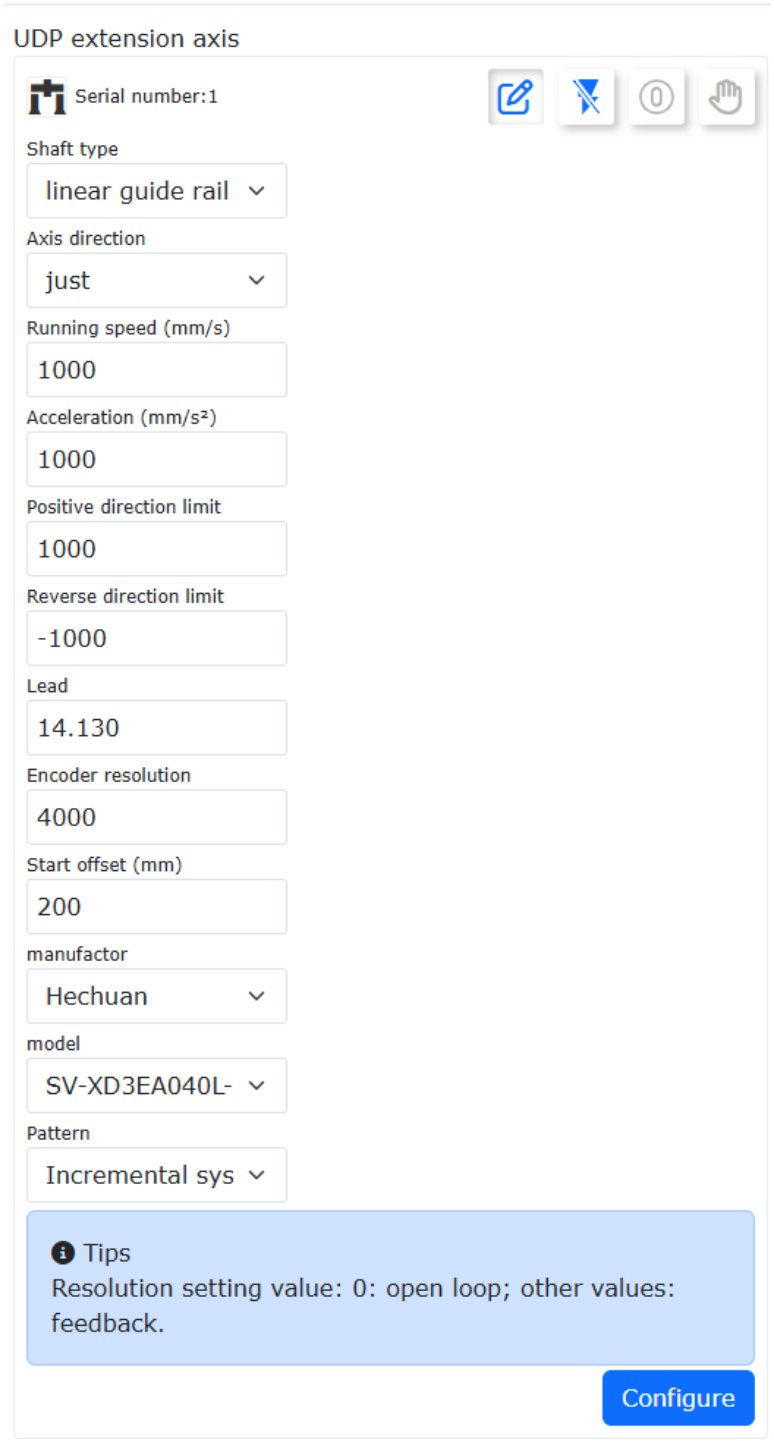
Figure 8.7-23 Extended Axis Test

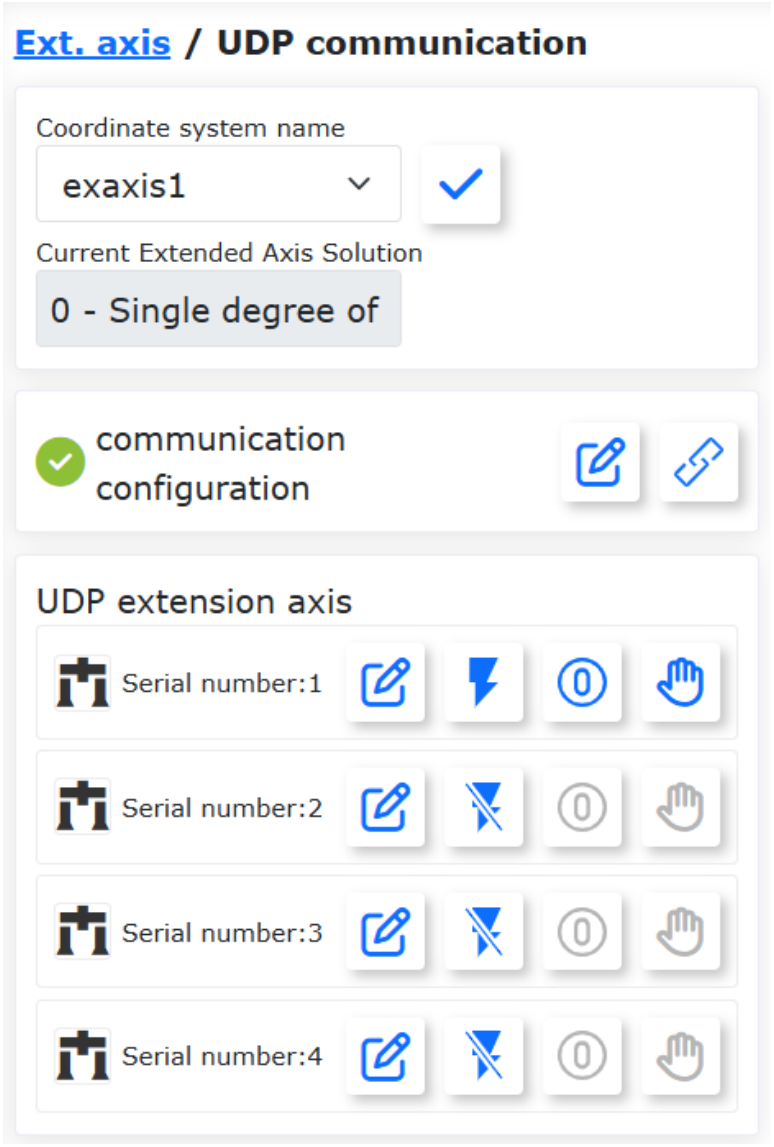
Step5: The extended axis is often used in conjunction with a laser sensor. In this case, the laser sensor is usually installed externally. The sensor reference point configuration needs to use the three-point calibration method instead of the previously used six-point calibration method. Align the tool center point to the bottom middle point of the right cross-section (the side closer to the camera) and set Point 1. Align the tool center point to the bottom middle point of the other cross-section (the left cross-section) and set Point 2. Move the tool center point to the top middle point of the right cross-section of the sensor and set Point 3. Calculate and save, then click Apply to complete the three-point calibration.

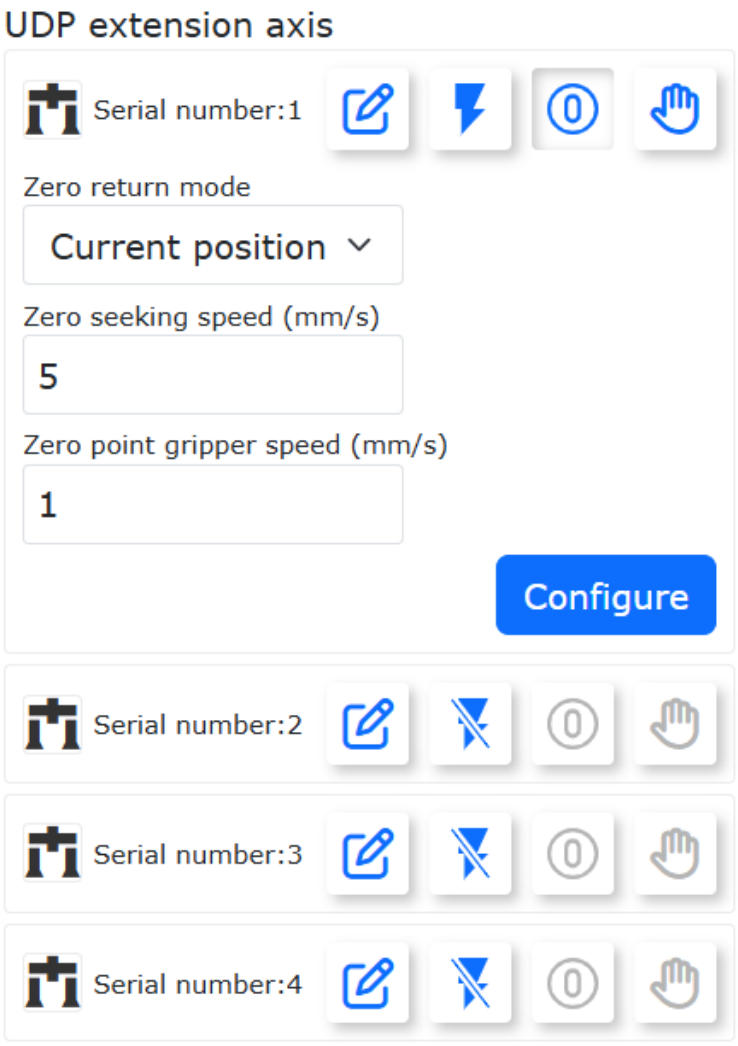
Figure 8.7-24 Sensor Three-Point Calibration

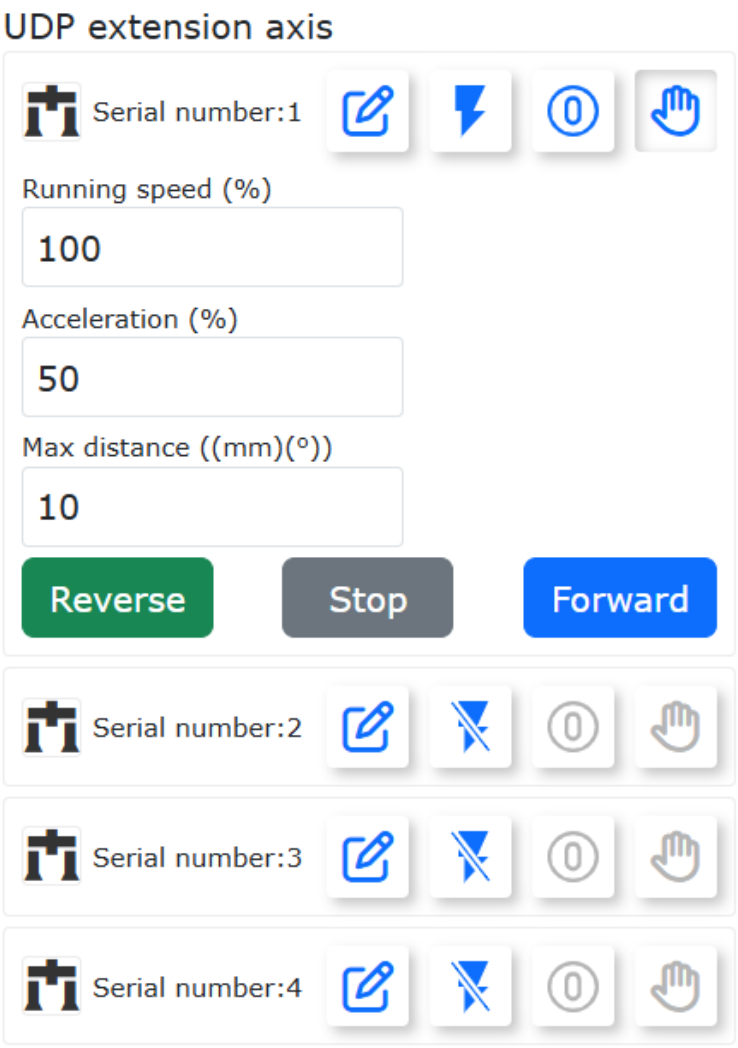
Step6: In the "Teach Program" -> "Program Programming" interface, select the "Extended Axis" command from the peripheral instructions. Add instructions in the appropriate places according to the specific program teaching requirements.

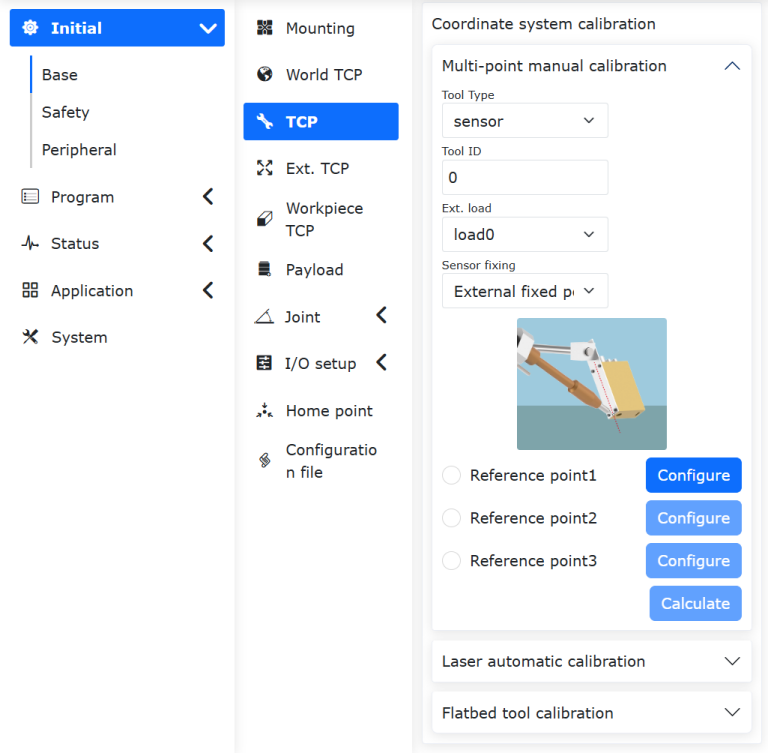
Figure 8.7-25 Extended Axis Command Editing


UDP extension axis
Serial number:1
Shaft type
linear guide rail
Axis direction
just
Running speed (mm/s)
1000
Acceleration (mm/s²)
1000
Positive direction limit
1000
Reverse direction limit
-1000
Lead
14.130
Encoder resolution
4000
Start offset (mm)
200
manufactor
Hechuan
model
SV-XD3EA040L-
Pattern
Incremental sys
Tips
Resolution setting value: 0: open loop; other values: feedback.
Configure


Ext. axis / UDP communication
Coordinate system name
exaxis1
Current Extended Axis Solution
0 - Single degree of
communication configuration
UDP extension axis
Serial number:1
Serial number:2
Serial number:3
Serial number:4

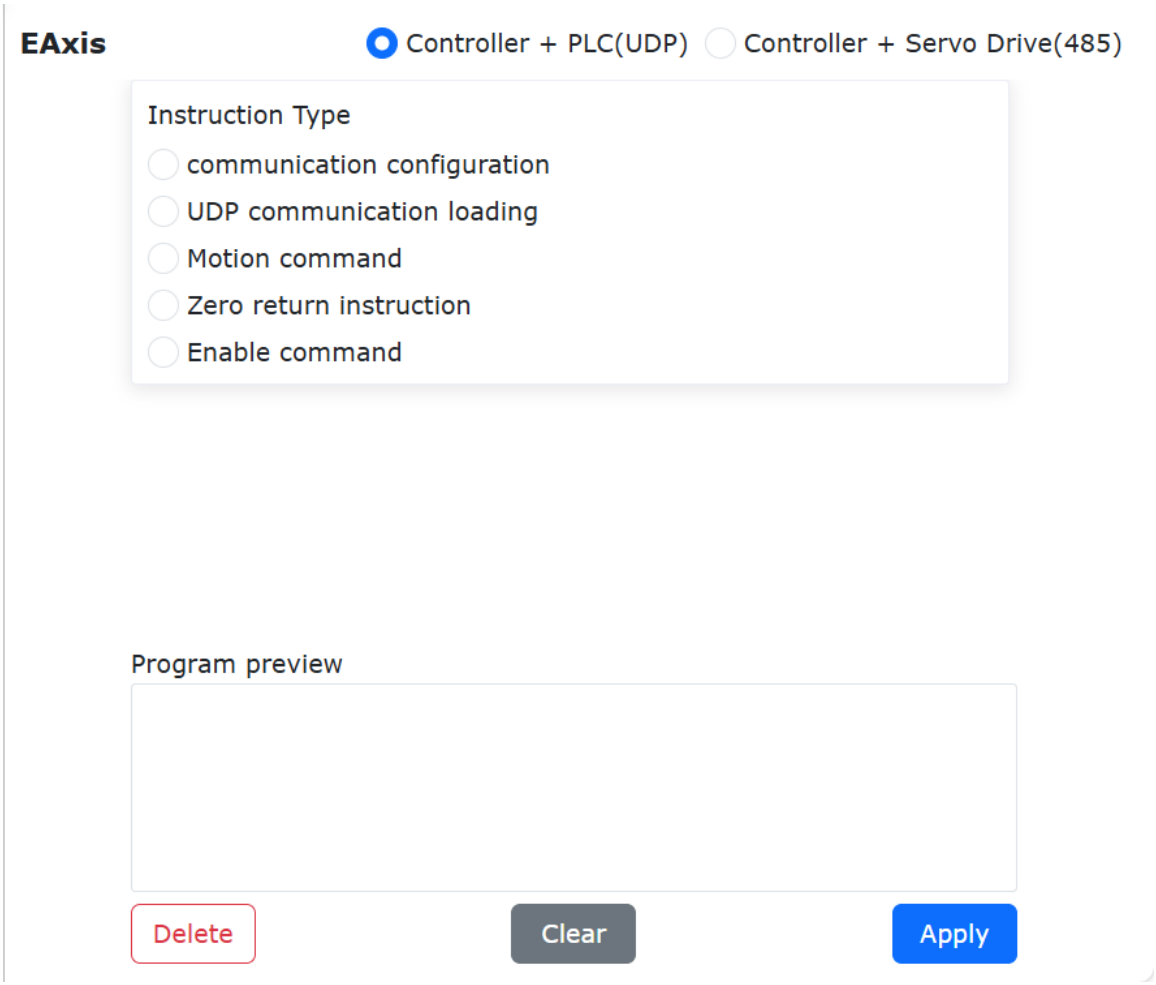

UDP extension axis
Serial number:1
Zero return mode
Current position
Zero seeking speed (mm/s)
5
Zero point gripper speed (mm/s)
1
Configure
Serial number:2
Serial number:3
Serial number:4


UDP extension axis
Serial number:1
Running speed (%)
100
Acceleration (%)
50
Max distance ((mm)(°))
10
Reverse
Stop
Forward
Serial number:2
Serial number:3
Serial number:4



1.3.8.7.2.3 Extended Axes with Laser Tracking Welding Teaching Program

No.	Command Format	Comment
1	EXT_AXIS_PTP(1,1laserstart)	#External axis moves to laser sensor start point
2	PTP(laserstart,10,-1,0)	#Robot moves to laser sensor start point
3	LTSearchStart(3,20,10,10000)	#Start searching
4	LTSearchStop()	#Stop searching
5	EXT_AXIS_PTP(1,1,seamPos)	#External axis moves to weld seam start point
6	Lin(seamPos,20,-1,00,0)	#Robot moves to weld seam start point
7	LTTrackOn()	#Laser tracking on
8	ARCStart(0,10000)	#Welder arc start
9	EXT_AXIS_PTP(1,1,laserend)	#External axis moves to weld seam end point
10	Lin(laserend,10,-1,0,0)	#Robot moves to weld seam end point
11	ARCEnd(0,10000)	#Welder arc end
12	LTTrackOff	#Laser tracking off


EAxis
Controller + PLC(UDP)
Controller + Servo Drive(485)
Instruction Type
communication configuration
UDP communication loading
Motion command
Zero return instruction
Enable command
Program preview
Delete
Clear
Apply

1.3.8.7.2.4 Positioning Completion Time

After the extended axis establishes UDP communication, input the time and click the “Configure” button to complete the setting. This configuration item is used to monitor the time when the extended axis motion stops.

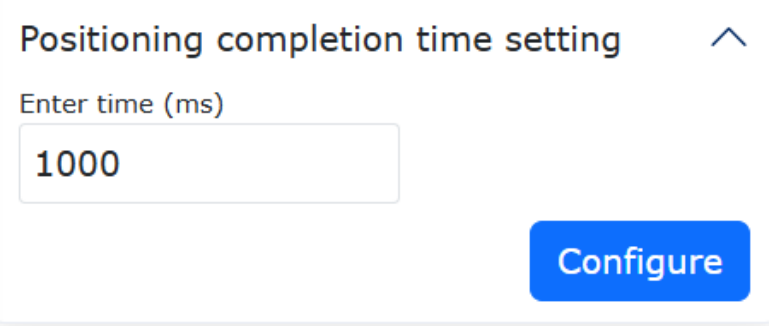


Figure 8.7-26 Positioning Completion Time Configuration

1.3.8.7.3 Two-DOF Trolley Test

This content is displayed in the UDP communication interface only when the extended axis scheme is set to “5-Two-DOF Trolley” in the extended axis coordinate system configuration; otherwise, it cannot be viewed.



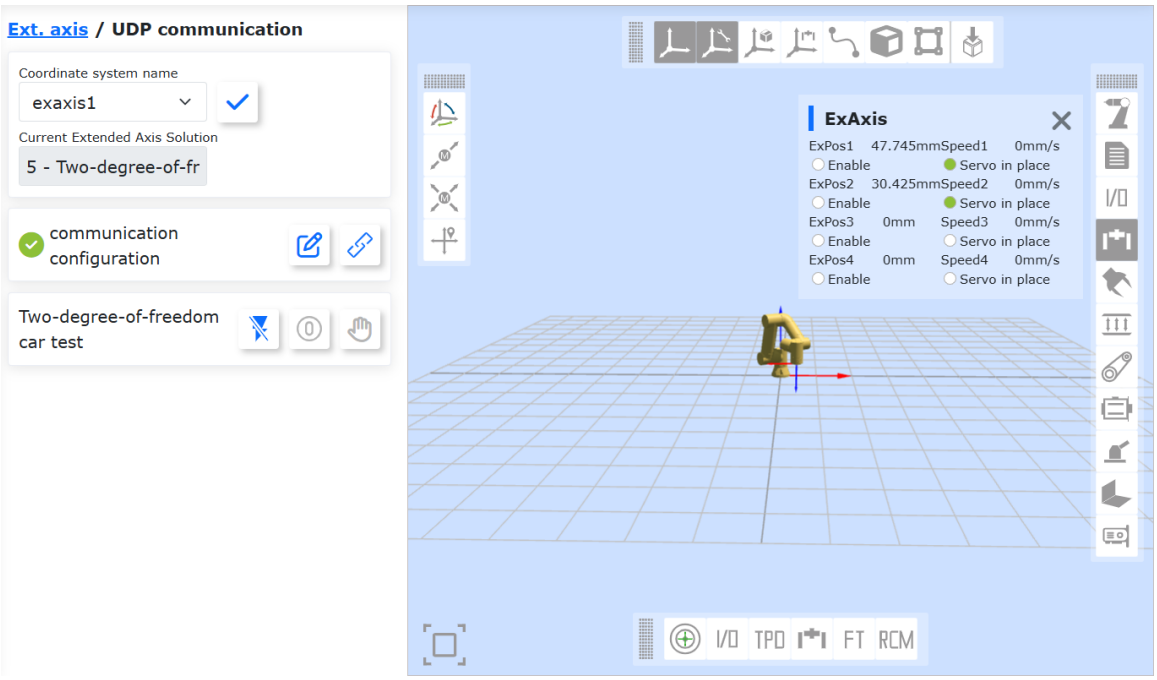
Figure 8.7-27 Interface for Extended Axis Scheme “5-Two-DOF Trolley”

Important: The two-DOF trolley by default uses extended axis numbers 1 and 2. After UDP communication is successful, check the servo ready status for extended axes 1 and 2 via the extended axis status in the robot’s various status monitors.

Figure 8.7-28 Two-DOF Trolley Extended Axis Servo Ready

Note:

Name: **Enable Button**



Function: Extended axis enabled state. Click the button to disable the extended axis.

Note: 

Name: **Disable Button**

Function: Extended axis disabled state. Click the button to enable the extended axis.

Note: 

Name: **Homing Button**

Function: Home the extended axis at the current position.

Note: 

Name: **Test Button**

Function: Test the two-DOF trolley functionality.

Step1: After UDP communication is successful, click the "Disable" button to enable the extended axes corresponding to the two-DOF trolley. Check that extended axes 1 and 2 are servo enabled via the extended axis status in the robot's various status monitors.

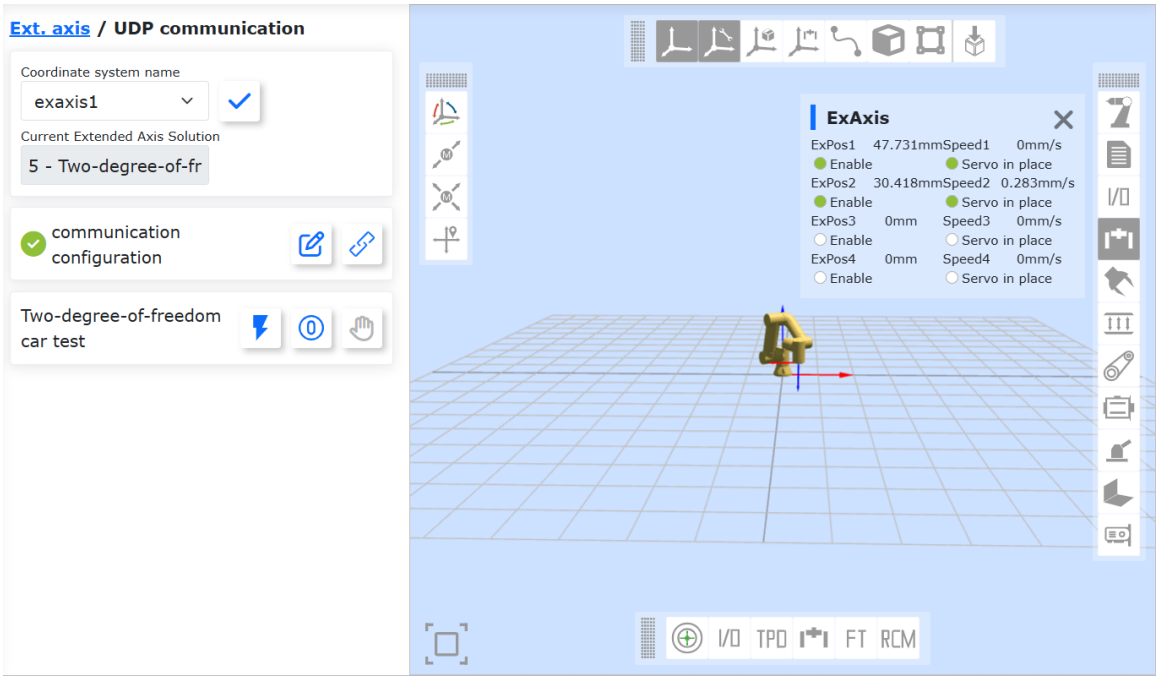


Figure 8.7-29 Two-DOF Trolley Extended Axes Enabled

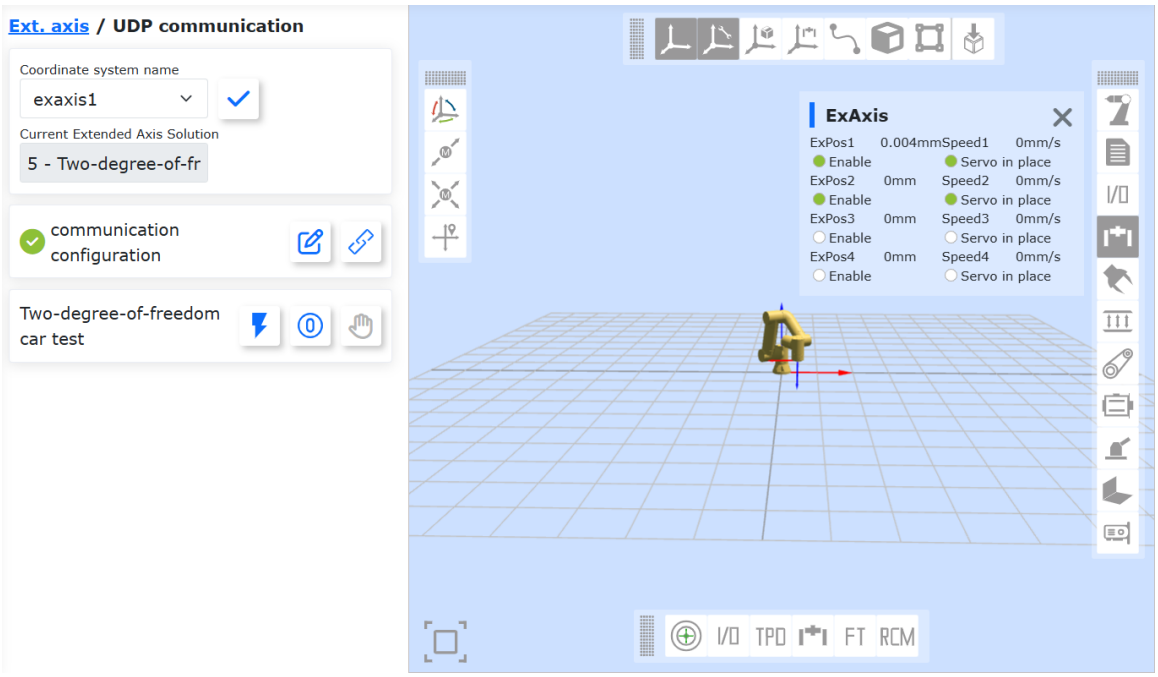
Step2: After the extended axes are successfully enabled, click the "Homing" button to set the current position of the extended axes as home. If homing is successful, the test button becomes highlighted; otherwise, it remains grayed out.

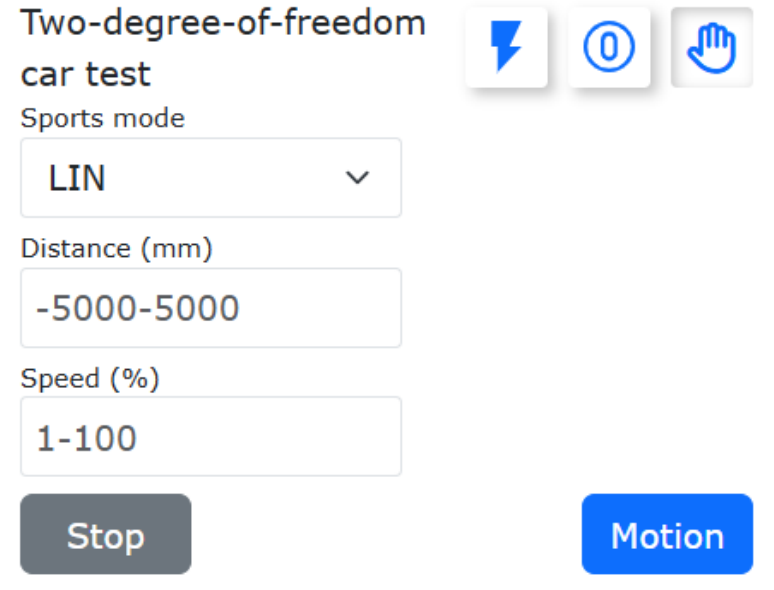
Figure 8.7-30 Two-DOF Trolley Current Position Homing Successful

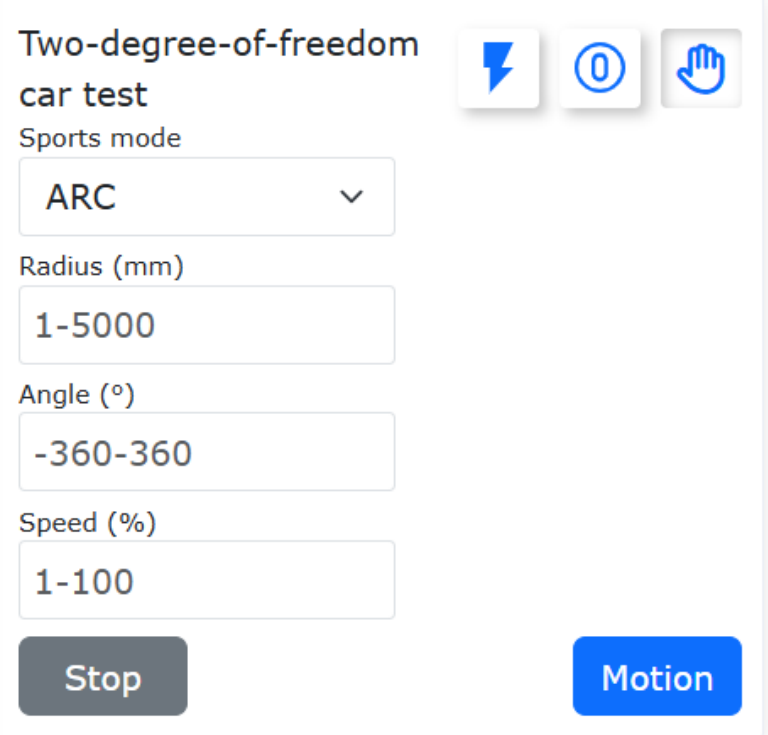
Step3: After the two-DOF trolley's current position homing is successful, click the "Test" button to enter the interface. Select the motion mode, input parameters for motion testing, and click the "Stop" button during motion to test the stop function.

- Motion Mode: Linear / Arc;
- Distance: -5000~5000mm (Linear motion mode);
- Radius: 1~5000mm (Linear motion mode);
- Angle: -360~360° (Arc motion mode);
- Speed: 1~100%

Figure 8.7-31 Two-DOF Trolley Test


Ext. axis / UDP communication
Coordinate system name
exaxis1
Current Extended Axis Solution
5 - Two-degree-of-fr
communication configuration
Two-degree-of-freedom car test
ExAxis
ExPos1 0.004mm Speed1 0mm/s
Enable
Servo in place
ExPos2 0mm Speed2 0mm/s
Enable
Servo in place
ExPos3 0mm Speed3 0mm/s
Enable
Servo in place
ExPos4 0mm Speed4 0mm/s
Enable
Servo in place
I/O
TPD
FT
RCM


Two-degree-of-freedom car test
Sports mode
LIN
Distance (mm)
-5000-5000
Speed (%)
1-100
Stop
Motion



1.3.8.7.4 Controller + Servo Drive (485 Communication)

1.3.8.7.4.1 Hardware Wiring

Before using RS485 communication to control the servo extended axis, please first connect the RS485 communication interface of the servo drive to the RS485 communication interface on the robot control box. The electrical interface diagram for the Faro Robot Yizhizao control box is as follows:

Figure 8.7-32 Faro Robot Mini Control Box Electrical Interface Diagram

Taking the Dynatek servo drive model FD100-750C as an example, refer to the drive panel terminal diagram and the X3A-IN terminal definition of the FD100-750C. When configuring the robot to communicate with the FD100-750C servo extended axis, it is necessary to connect the 485-A0 terminal and 485-B0 terminal on the control box to pins 4 and 5 of the drive's X3A-IN terminal, respectively. (Please note: You might see a terminal marked "485" on the servo drive panel. This terminal is currently not available for user use. Do not connect your RS485 communication cable to this terminal). Additionally, if connecting multiple servo drives and this drive is the last in the chain, it is necessary to turn on the RS485 communication termination resistor dip switch (No. 2 dip switch) on the panel.

Figure 8.7-33 FD100-750C Drive Panel

Figure 8.7-34 FD100-750C X3A-IN Terminal Definition

FAIR INNOVATION

Product Name: Controller
Product Model: FRC100-AC
Service Temperature: 0-45℃
Protection Class: IP54
Input : 220VAC/10A/Single Phase/50HZ
Output : 6000W/48VDC/41A

Manufacturer: FAIR INNOVATION (Suzhou) Robot System Co.,LTD
Address: 209 Zhuyuan oad,High-tech Zone,Suzou City,Jiangsu Province
5888 Zunxian Road,High-tech Zone,Zibo City,Shandong Province
Telephone: 0512-68562005
Serial Number:
Date Of Production:

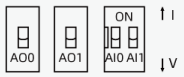
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
E-24V	E-0V	EI0-1	EI1-1	SI0-1	SI1-1	EDM-	EST0-1	EST1-1	485-B0	485-B1	5 V	B1+	B1-	B2+	B2-	GNDA	A00	A01
24V	0V	EI0-2	EI1-2	SI0-2	SI1-2	EDM+	EST0-2	EST1-2	485-A0	485-A1	GND	A1+	A1-	A2+	A2-	GNDA	AI 0	AI 1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
D00	D01	D02	D03	E-24V	D04	D05	D06	D07	E-24V	C00	C01	C02	C03	E-24V	C04	C05	C06	C07
DI 0	DI 1	DI 2	DI 3	E- 0V	DI 4	DI 5	DI 6	DI 7	E- 0V	CI 0	CI 1	CI 2	CI 3	E- 0V	CI 4	CI 5	CI 6	CI 7

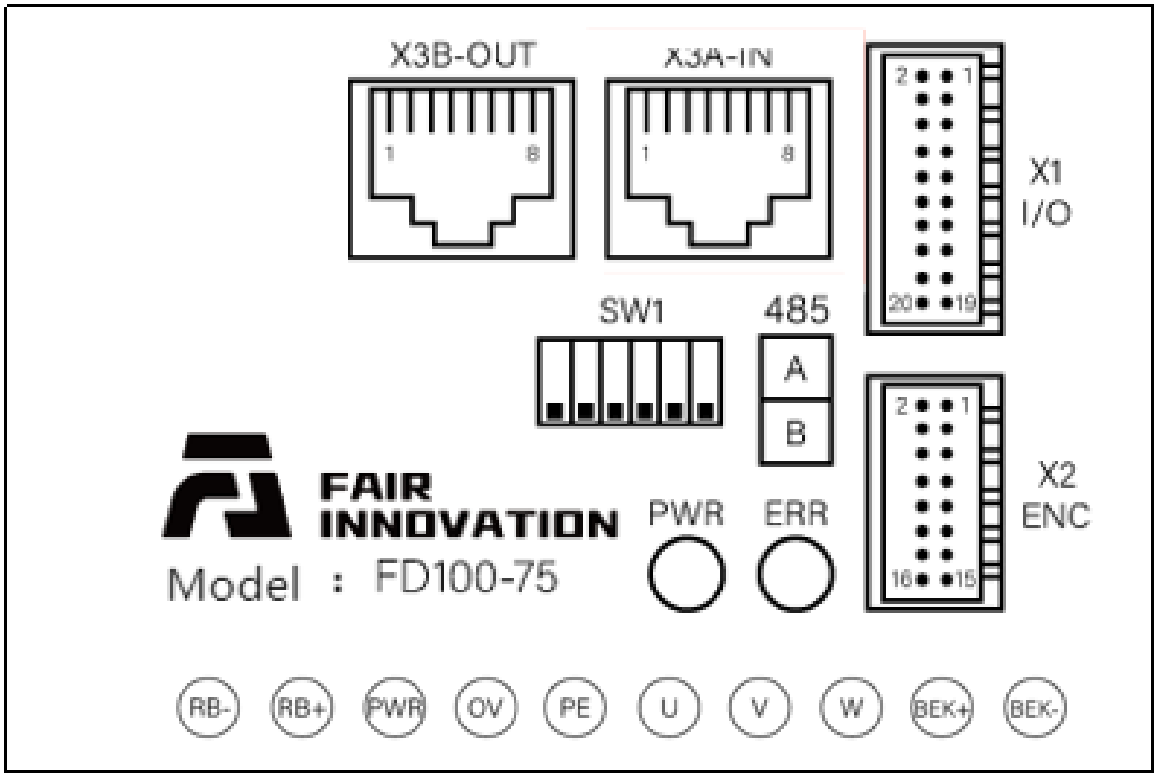
I/O Layout of user peripheral interfaces

I/O Interface

Instructions for use of dial switch in the chassis:
The switch is divided into AO/AI tow channels,
and the analog is set through the dial switch,
as shown on the right.



AO: I : Analog current output (0-20mA)
V : Analog voltage output (0-10V)
A I: I : Analog current input (0-20mA)
V : Analog voltage input (0-10V)



RJ45 Pin number	RJ45 Pin Description (FD100-750C)	RJ45 Pin Description (FD100-750E)	Dip switch number	Dip Description
1	CAN signal +	TXB_N	1	CAN Terminal resistance (120Ω)
2	CAN signal -	TXB_P	2	RS485 Terminal resistance (120Ω)
3	GND	RXB_N	3	ON=low, OFF=high
4	RS485 signal +	-	4	ON=low, OFF=high
5	RS485 signal -	-	5	ON=low, OFF=high
6	-	RXB_P	6	ON=low, OFF=high
7	-	-	-	-
8	GND	-	-	-

1.3.8.7.4.2 Communication Configuration

After ensuring your RS485 communication cable is correctly connected and both the robot and servo extended axis are powered on normally, open the robot WebApp.

Click on the image with the combination "Controller + Servo Drive" to enter the detailed configuration interface. In the servo drive configuration, select the number as "1" (Please note: When multiple servos are connected, this number is used to distinguish different servos, we will mention this number multiple times later), the manufacturer as "Dynatek", select the corresponding servo drive model, here the model is "FD00-750C", the software version is V1.0, fill in the resolution corresponding to the servo drive, here it is 131072, fill in the mechanical transmission ratio according to your mechanism model, here it is 15.45, and click the "Configure" button.

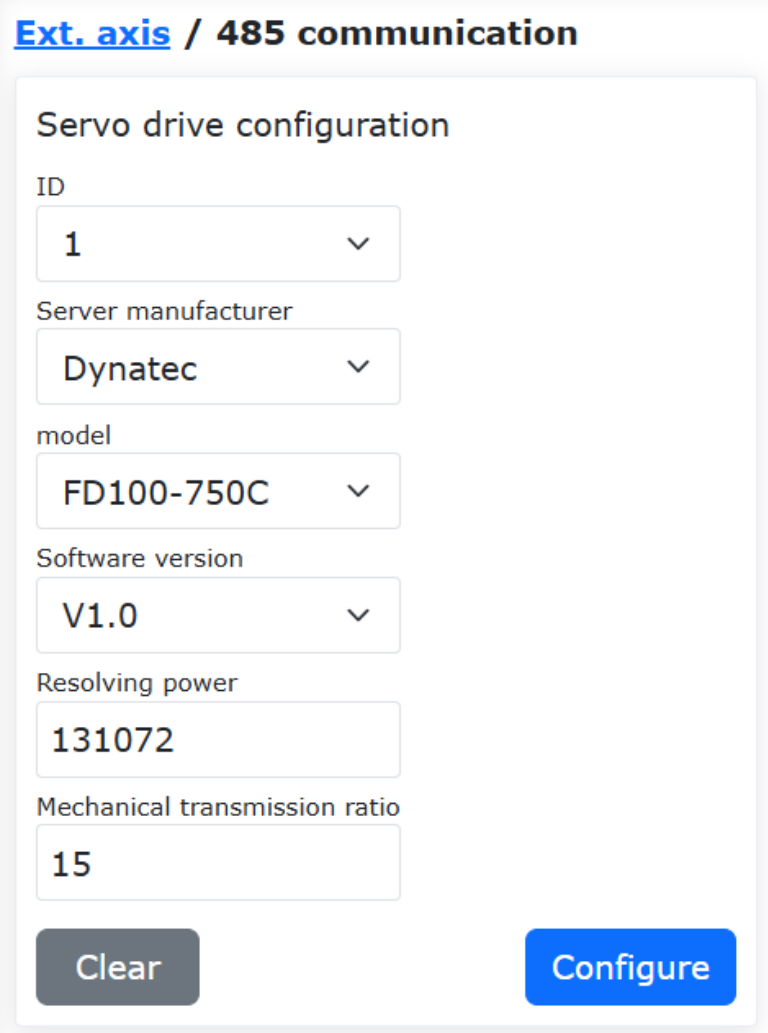
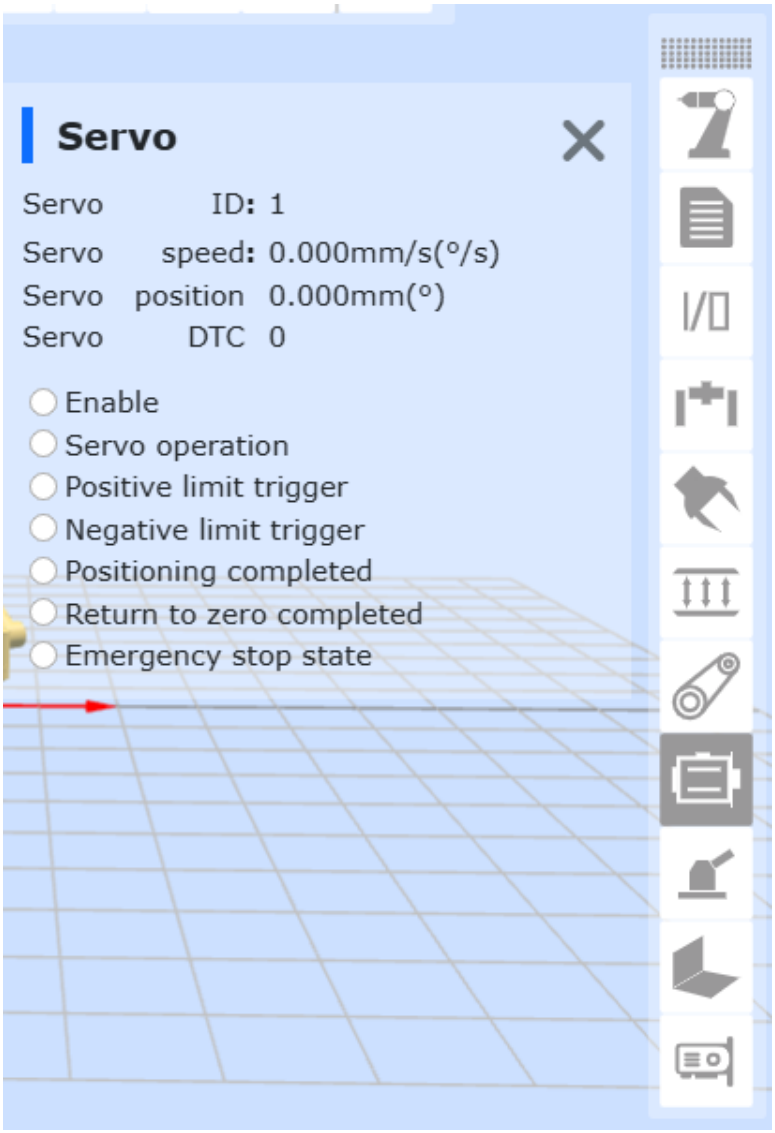


Figure 8.7-35 Servo Drive Configuration

So far, we have completed the 485 communication configuration between the robot and the servo drive. You can view the real-time status information of the servo in the "Servo Status Bar" on the right side of the WebApp, as shown below:

Figure 8.7-36 Servo Status Bar

Now you need to set the enable and homing method for the extended axis device in order, and then you can perform certain motion tests. Please follow this manual for the following test operations under safe conditions.


Servo
Servo ID: 1
Servo speed: 0.000mm/s(°/s)
Servo position 0.000mm(°)
Servo DTC 0
Enable
Servo operation
Positive limit trigger
Negative limit trigger
Positioning completed
Return to zero completed
Emergency stop state

1.3.8.7.4.3 Configured Servo Drives



Note:

Name: **View Button**

Function: Click to view servo drive configuration information.



Note:

Name: **Enable Button**

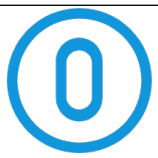
Function: Servo drive enabled state. Click the button to disable the servo drive.



Note:

Name: **Disable Button**

Function: Servo drive disabled state. Click the button to enable the servo drive.



Note:

Name: **Homing Button**

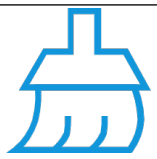
Function: Set the homing method for the servo drive.



Note:

Name: **Test Button**

Function: Test the servo drive.

Note: 

Name: **Servo Error Clear Button**

Function: When the servo drive indicates an error, click to clear it.

1.3.8.7.4.4 Servo Control Mode and Enable

In “Configured Servo Drives”, select the control mode as “Position Mode”, select the corresponding servo number, and click the “Disable” button. This will first set the servo drive number. After successful setting, it sets the control mode. After the control mode is set successfully, the servo drive is enabled (Please note: After switching the control mode, you need to first disable the servo drive, then enable the servo drive again for the servo control mode switch to take effect. After the servo is enabled, switching the control mode will be disabled).

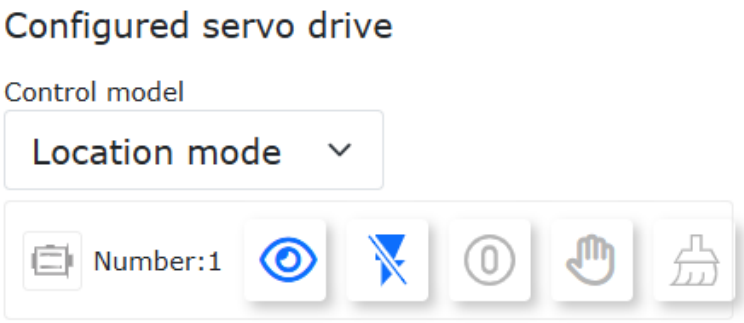


Figure 8.7-37 Servo Control Mode and Enable

After the servo is successfully enabled, check the “Servo” section in the robot’s various status bars to observe the “Servo Enable” status light is on, indicating the servo drive is enabled. Click the “Enable” status button to disable the servo drive, and the “Servo Enable” status light turns off.

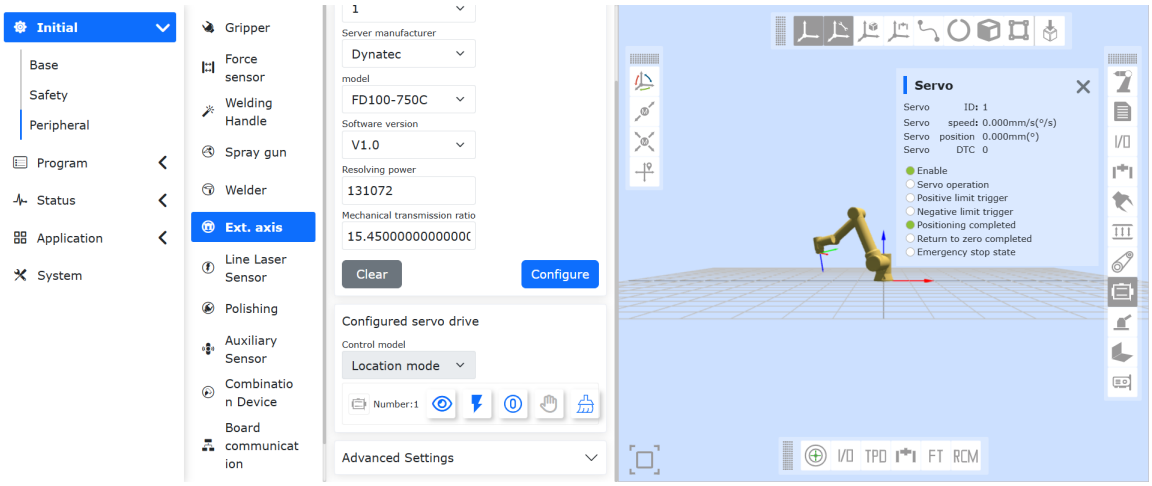


Figure 8.7-38 Servo Drive Status Bar

1.3.8.7.4.5 Servo Homing

After the servo drive is successfully enabled, the “Homing” button becomes highlighted. Click the button to enter the setup interface. Select the homing mode as “Current Position Homing”, homing speed as 5mm/s, and zero search speed as 1mm/s; click the “Set” button to complete the servo current position homing operation. In the “Servo” section of the robot’s various status bars, you can observe that the current “Servo Position” is 0; (Please perform homing tests with “Negative Limit Homing” or “Positive Limit Homing” only after fully reading this manual).

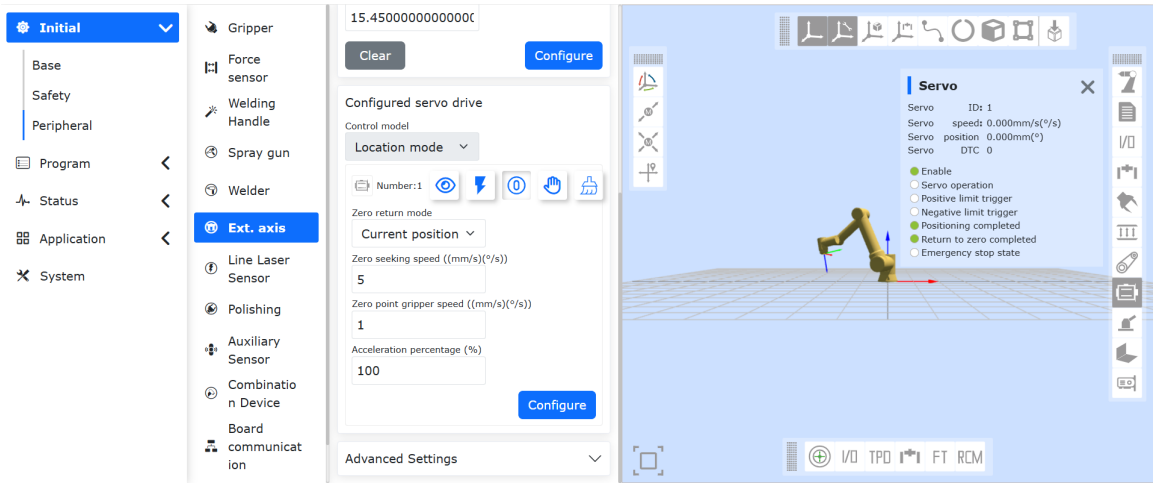


Figure 8.7-39 Servo Homing

1.3.8.7.4.6 Servo Motion

Before actually controlling the servo motor motion, please first understand the “Position Mode” and “Speed Mode” of the servo motor. We remind you again:

Position Mode: You can input a certain motion speed and target position parameter. The servo will move to the target position at the set speed and stop after reaching the target position.

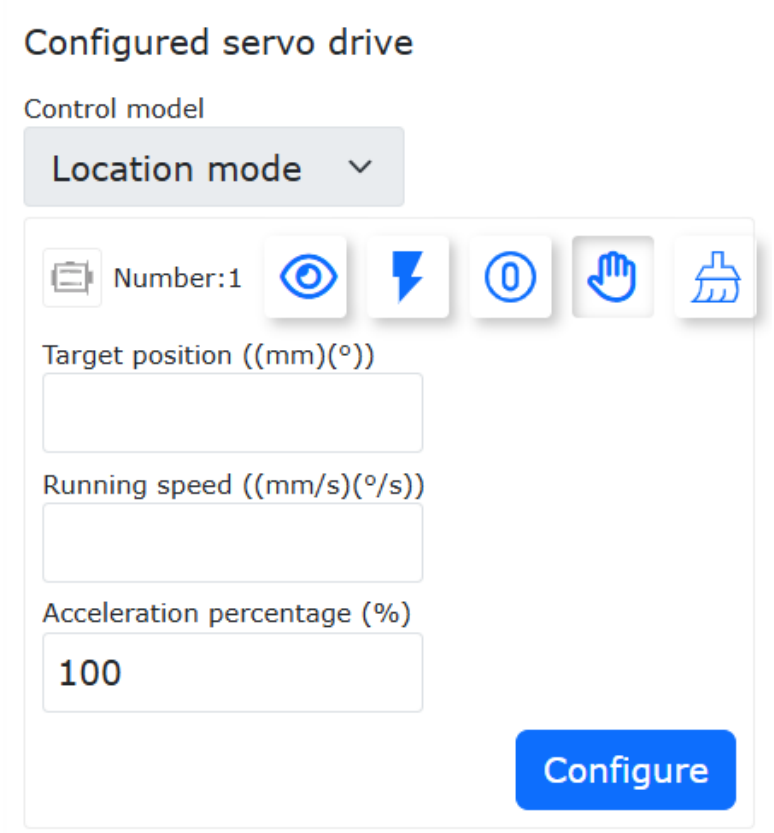
Speed Mode: You can input a certain target speed. The servo will continue to move at the target speed you set until you set the target speed to 0 or disable the servo motor;

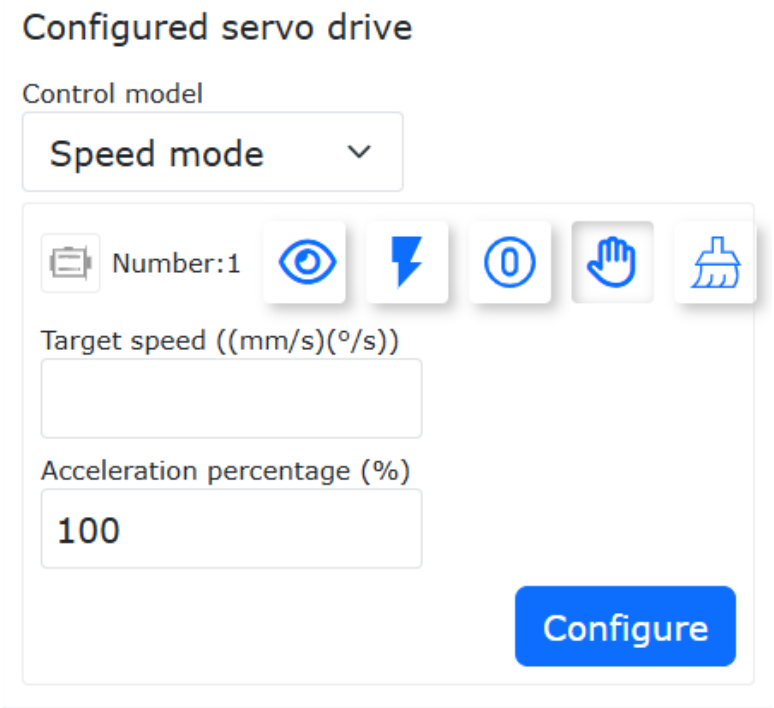
When switching control modes, the “Current Control Mode” display will switch automatically (Please note: After switching the control mode, you need to first disable the servo, then enable the servo again for the servo control mode switch to take effect). If your servo is not currently in “Position Mode”, please switch your servo to position mode. Enter the “Target Position” as 50mm and the run speed as 5mm/s. Under confirmed safe conditions, click the “Set” button. At this point, the servo motor will move according to the parameters you set. You can observe the servo’s position, speed, etc., in real-time in the “Servo” section of the robot’s various status bars.

Figure 8.7-40 Servo Motion Debugging (Position Mode)

Change the servo control mode to “Speed Mode”. Click the “Enable” status button to disable the servo drive, then click the “Disable” status button. Now the servo is switched to speed mode (Please note: After the servo motor starts moving, it can only be stopped by setting the target speed to 0). Enter the target speed as 5mm/s and click the “Set” button. The servo motor will maintain motion at 5mm/s. Similarly, you can observe the servo’s position, speed, etc., in real-time in the “Servo” section of the robot’s various status bars.

Figure 8.7-41 Servo Motion Debugging (Speed Mode)


Configured servo drive
Control model
Location mode
Number:1
Target position ((mm)(°))
Running speed ((mm/s)(°/s))
Acceleration percentage (%)
100
Configure


Configured servo drive
Control model
Speed mode
Number:1
Target speed ((mm/s)(°/s))
Acceleration percentage (%)
100
Configure

1.3.8.7.4.7 Advanced Settings

If the robot collides, the emergency stop is pressed, or other emergency situations occur, the extended axis can trigger an emergency stop and stop motion according to the set emergency stop deceleration. After the collision alarm is reset, commands can continue to be sent to resume extended axis operation. It is necessary to set the servo acceleration/deceleration and servo emergency stop acceleration/deceleration in the Advanced Settings, as shown below:

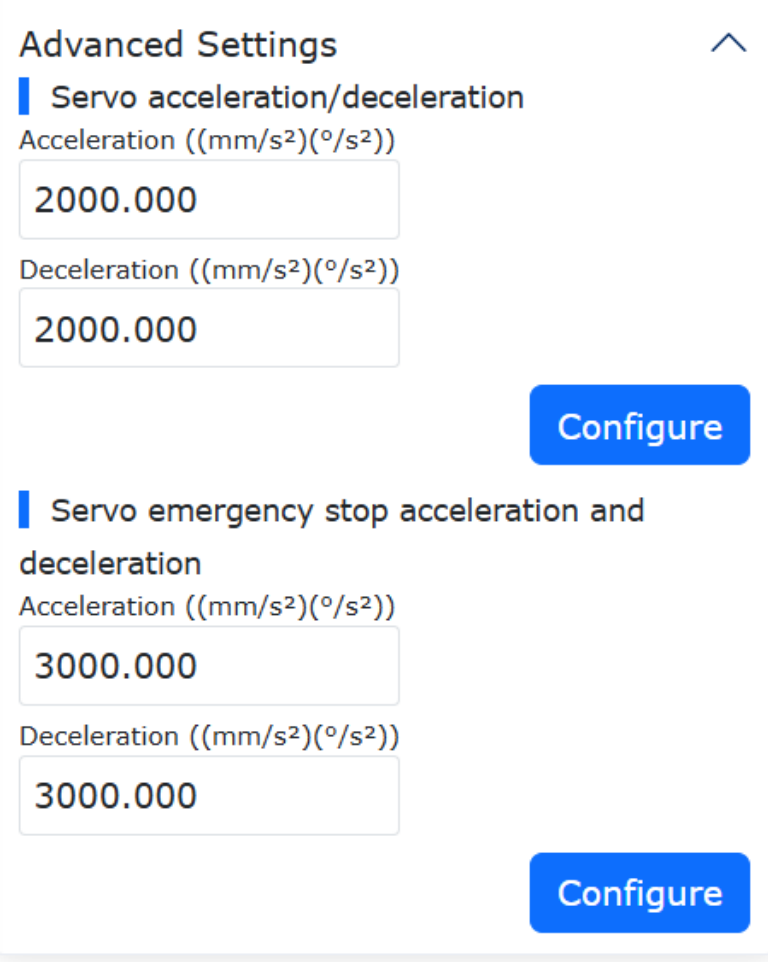


Figure 8.7-42 Advanced Settings

1.3.8.7.4.8 Extended Axis Programming

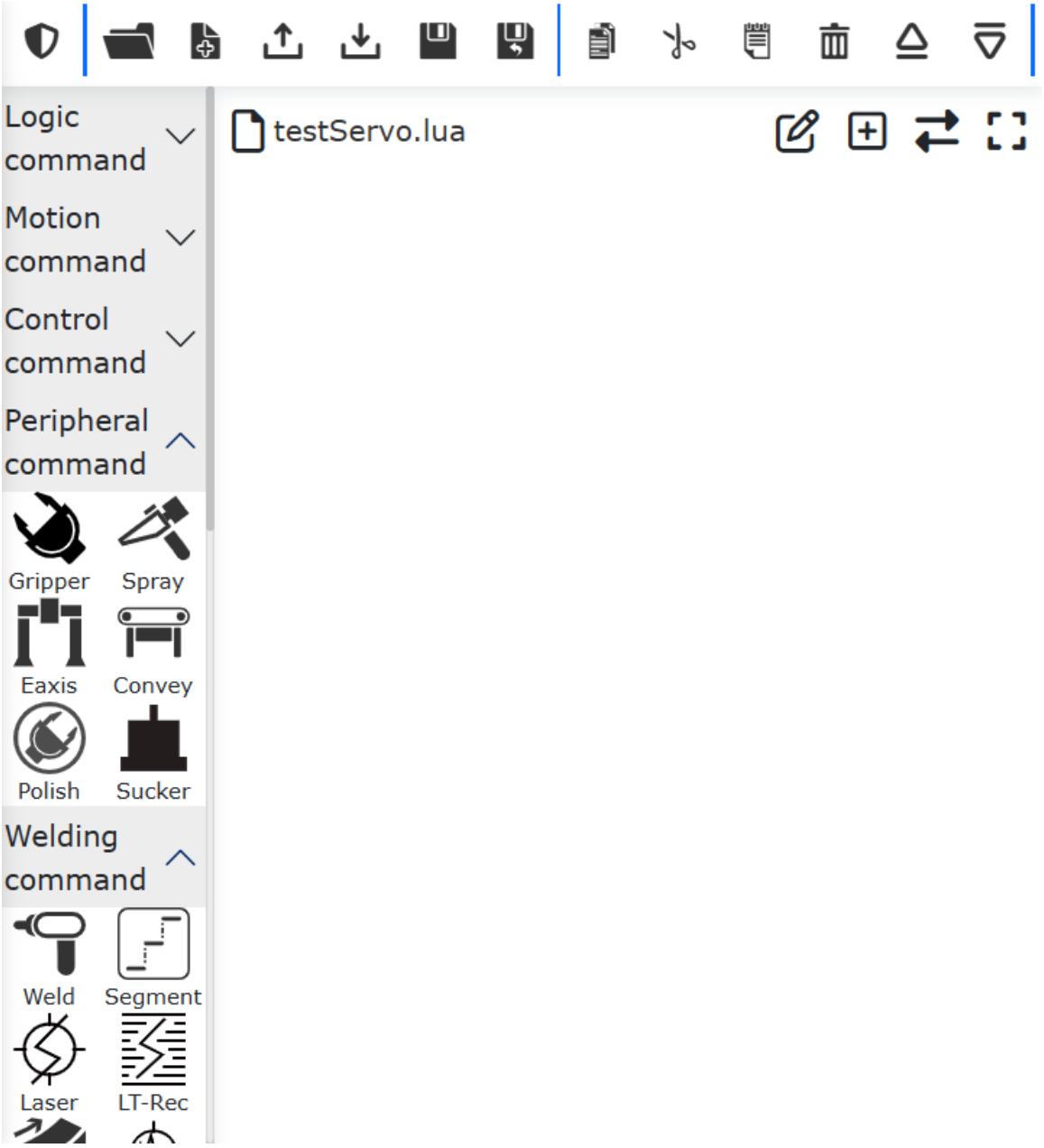
In “Teaching Program” -> “Program Programming”, create a new user program “testServo.lua” and select “Peripheral Instructions”.

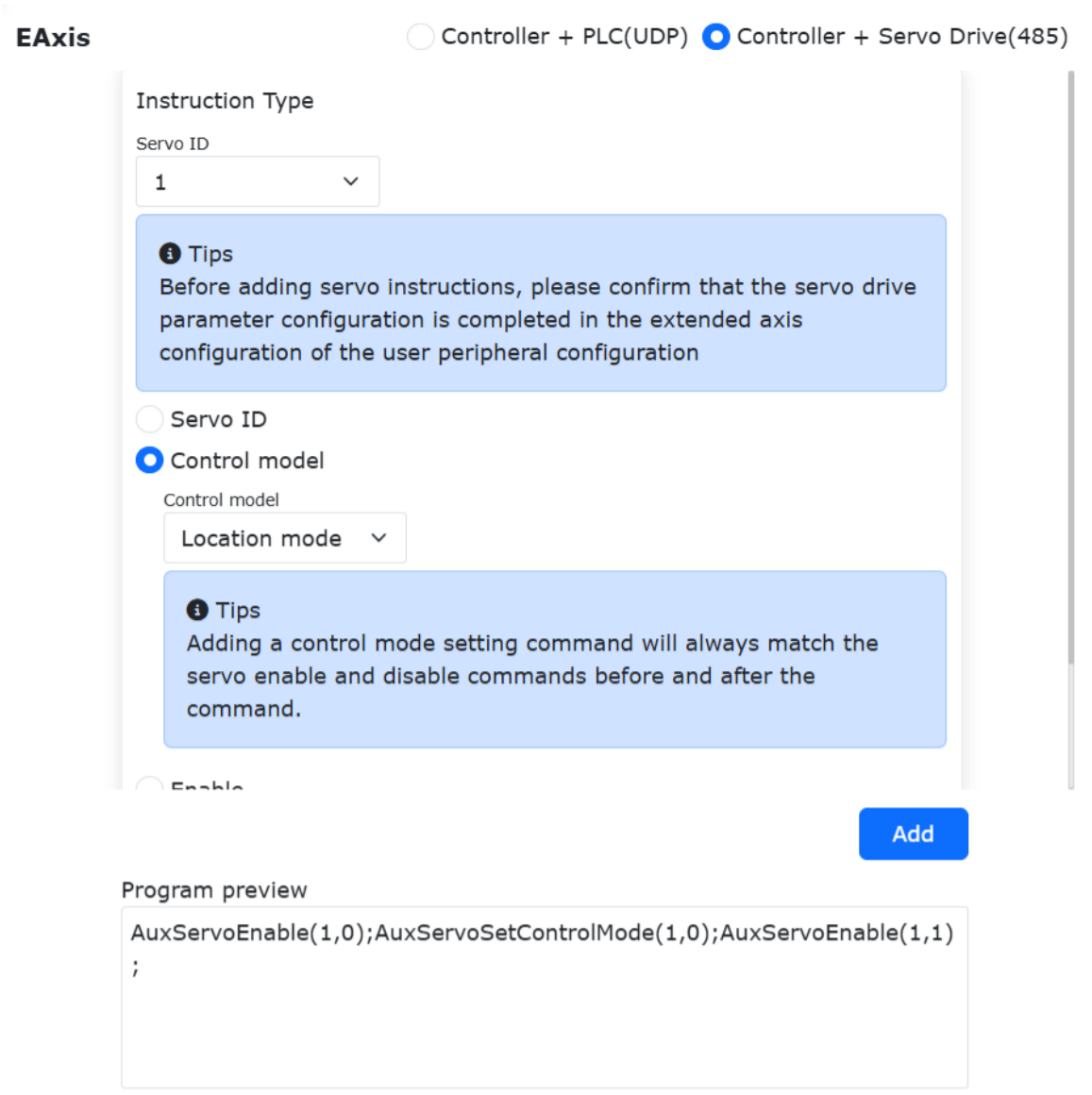
Figure 8.7-43 Open Peripheral Instructions

Click “Extended Axis” to open the Add Extended Axis Instruction interface. Select the combination method as “Controller + Servo Drive (485)”, set the control mode to “Position Mode”, and click the “Add” button on the right. Scroll to the bottom of the Add Extended Axis Instruction interface and click the “Apply” button.

Figure 8.7-44 Set Extended Axis Control Mode

Now, a set of instructions for switching the servo control mode appears in the “testServo.lua” program. You can switch the robot to automatic mode and execute this program.


Logic command
Motion command
Control command
Peripheral command
Gripper
Spray
Eaxis
Convey
Polish
Sucker
Welding command
Weld
Segment
Laser
LT-Rec
testServo.lua


EAxis
Controller + PLC(UDP)
Controller + Servo Drive(485)
Instruction Type
Servo ID
1
Tips
Before adding servo instructions, please confirm that the servo drive parameter configuration is completed in the extended axis configuration of the user peripheral configuration
Servo ID
Control model
Control model
Location mode
Tips
Adding a control mode setting command will always match the servo enable and disable commands before and after the command.
Add
Program preview
AuxServoEnable(1,0);AuxServoSetControlMode(1,0);AuxServoEnable(1,1);

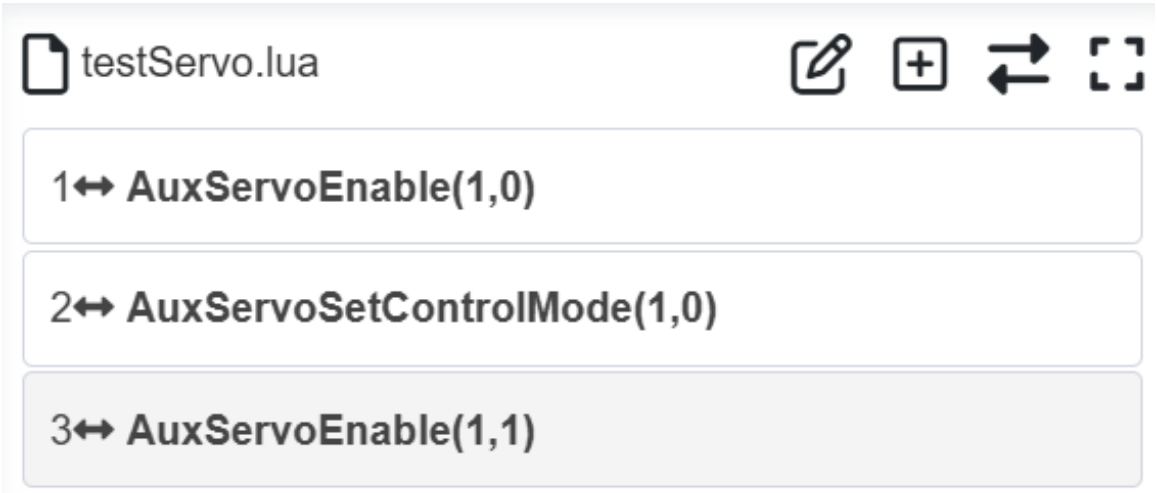

testServo.lua
1 AuxServoEnable(1,0)
2 AuxServoSetControlMode(1,0)
3 AuxServoEnable(1,1)

Figure 8.7-45 Set Servo Control Mode Program

How to control servo motion through a user program? Similarly, open the Add Extended Axis Instruction interface, as shown below, find the parameter configuration bar. Taking position mode as an example, enter the target position and run speed, click the "Add" button; scroll to the bottom of the Add Extended Axis Instruction interface, click the "Apply" button, and close the Add Extended Axis Instruction interface.

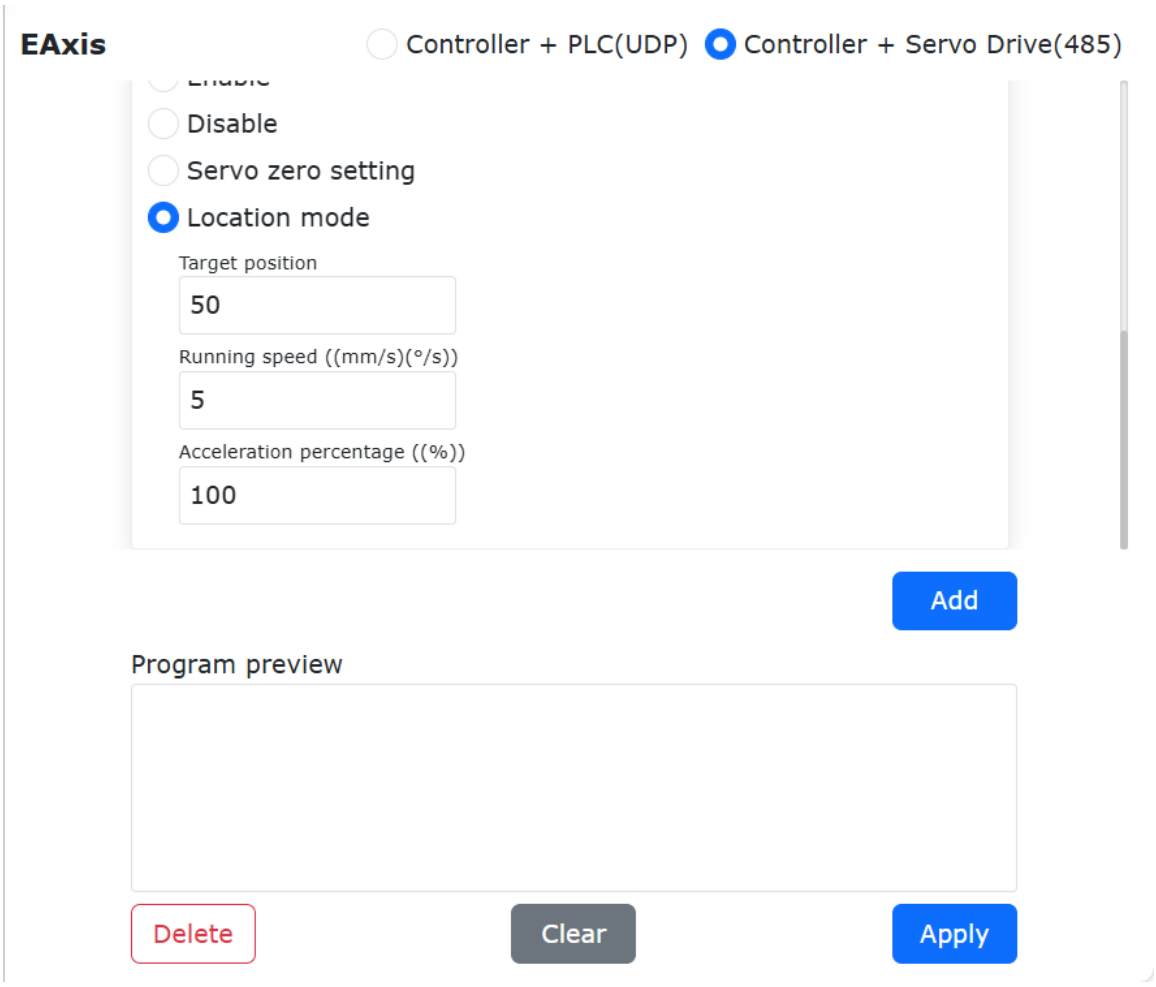


Figure 8.7-46 Add Position Mode Motion Instruction

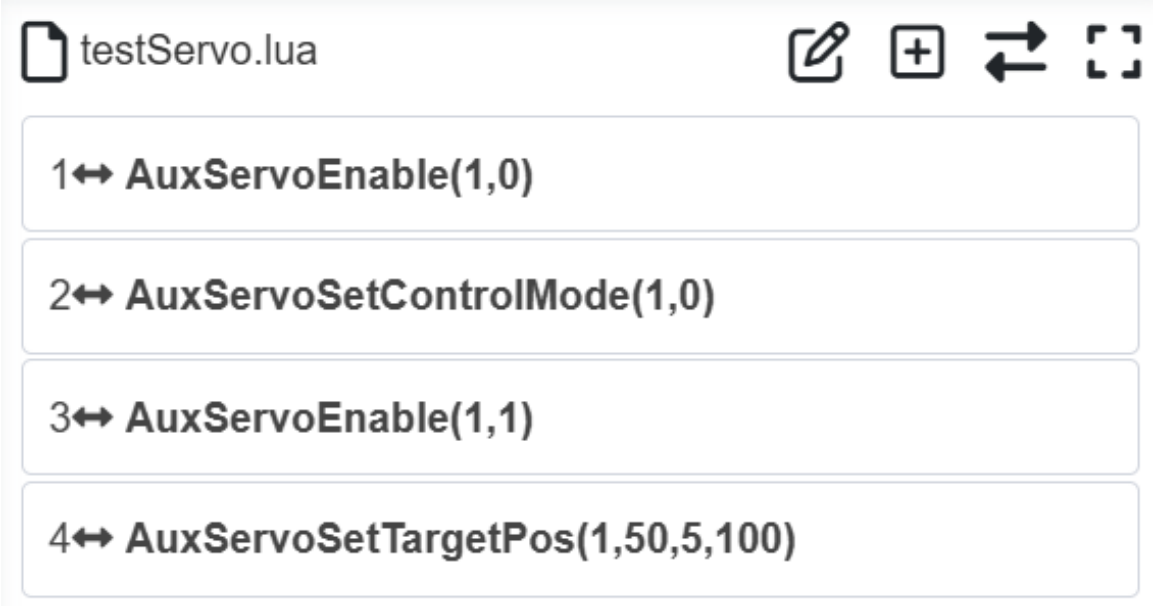
The "testServo.lua" program adds a servo motion instruction: "AuxServoSetTargetPos(1,50,5)". The meanings of the three parameters in the instruction function are:

- 1: Servo number is 1.
- 50: Target position.
- 5: Target speed.

Figure 8.7-47 Position Mode Servo Motion Program

Switch the robot to automatic mode and run the program. Now your servo will move to the 50mm position at a speed of 5mm/s.

So far, we have completed the preliminary configuration and testing of RS485 controlled servo extended axes. You can write programs combining robot motion and servo motion according to the actual situation, such as the example program below.



1.3.8.7.4.9 Example Program for Coordinated Motion between Extended Axis and Robot

No.	Command Format	Comment
1	AuxServoSetTargetPos(1,50,5)	#Extended axis moves to reset point
2	if(GetDI(8,0) == 1) then	#If CI0 input is valid
3	AuxServoSetTargetPos(1,50,5)	#Extended axis moves to 50mm
4	PTP(testptp1,100,-1,0)	#Robot moves to testptp1 point
5	elseif(GetDI(9,0) == 1) then	#If CI1 input is valid
6	AuxServoSetTargetPos(1,150,5)	#Extended axis moves to 150mm
7	PTP(testptp2,100,-1,0)	#Robot moves to testptp2 point
8	else	#If both CI0 and CI1 inputs are invalid
9	AuxServoSetTargetPos(1,300,5)	#Extended axis moves to 300mm
10	PTP(testptp3,100,-1,0)	#Robot moves to testptp3 point
11	end	#End

1.3.8.7.4.10 Summary

In summary, the key points for configuring RS485 communication between the collaborative robot and the servo extended axis are as follows:

1. Correctly connect the RS485 communication cable between the collaborative robot and the servo drive;
2. Correctly select the control mode for the servo extended axis;
3. After switching the control mode, you must first disable the servo, then enable the servo again for the control mode switch to take effect.

1.3.8.8 Line Laser Sensor

The F&R collaborative robot is used in conjunction with a laser sensor to identify feature positions such as welds, achieving the goals of simplified programming and improved production efficiency. The collaborative robot can be adapted to laser sensors from three manufacturers: Ruiniu, Chuangxiang, and Quanshi. When using different sensors, only the corresponding communication protocol needs to be loaded.

1.3.8.8.1 Hardware Connection

Before using the laser sensor, it needs to be installed in a suitable position. Connect the laser sensor's network cable directly or through a switch to any RJ45 interface on the robot control box.

1.3.8.8.2 Sensor Configuration

Please ensure your laser sensor and welding torch are fixedly installed on the robot end flange, the laser sensor is connected to the robot control box via a network cable, and the IP addresses of the laser sensor and the robot control box are on the same subnet. Turn on the power for both the robot and the sensor. The figure below shows the installation of a Ruiniu laser sensor.

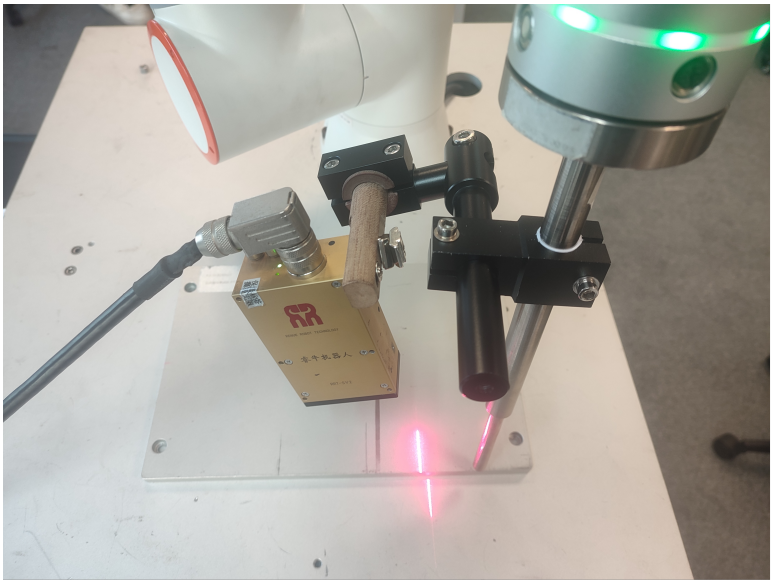


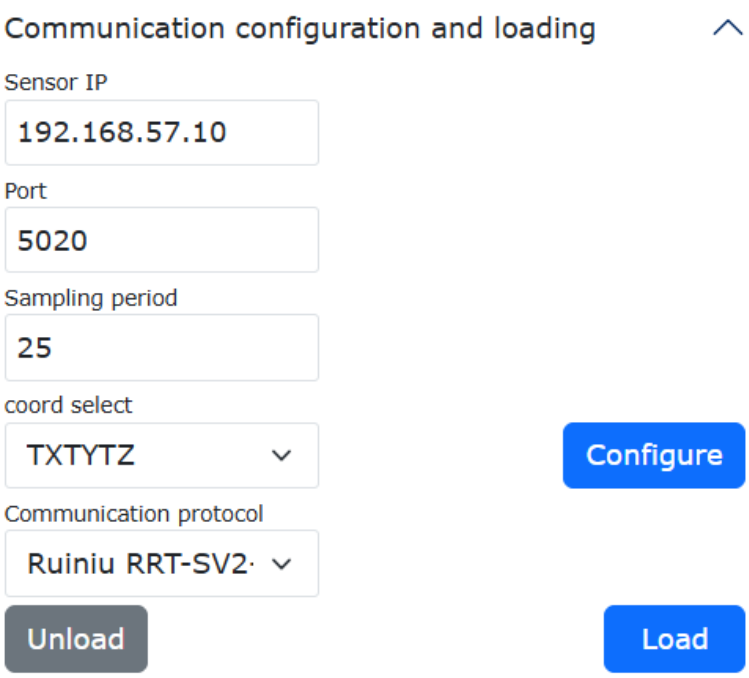
Figure 8.8-1 Laser Sensor Installation

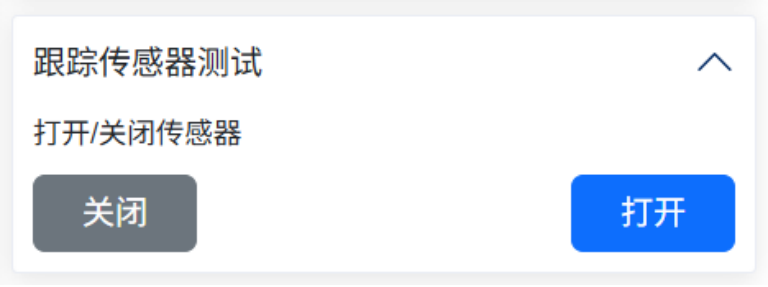
In the communication configuration section, enter the sensor's IP address and port number, click the "Configure" button. The sampling period defaults to 25, select "Laser Plane Coordinate System" for the coordinate system, choose the corresponding communication protocol according to your sensor model, and click the "Load" button.

Figure 8.8-2 Laser Sensor Configuration

In the "Tracking Sensor Test" section, click "Open" and then "Close" the sensor sequentially, observing whether the sensor's laser turns on or off. If the laser turns on and off normally, it indicates that communication has been successfully established between the robot and the sensor. Otherwise, please check if parameters such as the IP address and port number are correct, and if the network connection between the sensor and the robot is proper.

Figure 8.8-3 Laser Sensor Communication Test


Communication configuration and loading
Sensor IP
192.168.57.10
Port
5020
Sampling period
25
coord select
TXTYTZ
Configure
Communication protocol
Ruiniu RRT-SV2·
Unload
Load


跟踪传感器测试
打开/关闭传感器
关闭
打开

1.3.8.8.3 Sensor Calibration

Before using the laser sensor, it needs to be calibrated. The calibration accuracy directly affects the tracking accuracy of the laser sensor. Calibration methods for the laser sensor include the Five-Point Method, Six-Point Method, and Eight-Point Method. Taking the most commonly used Five-Point Method in welding applications as an example, the calibration principle is to first point the tool (welding torch) to a fixed calibration point (as shown in Figure 4), and then have the laser sensor irradiate and recognize this point from four different poses.

Note: This calibration point must be accurately recognizable by the laser sensor; otherwise, precise calibration cannot be achieved.

Then, the sensor coordinate pose is calculated. The calibration process is described in detail below:

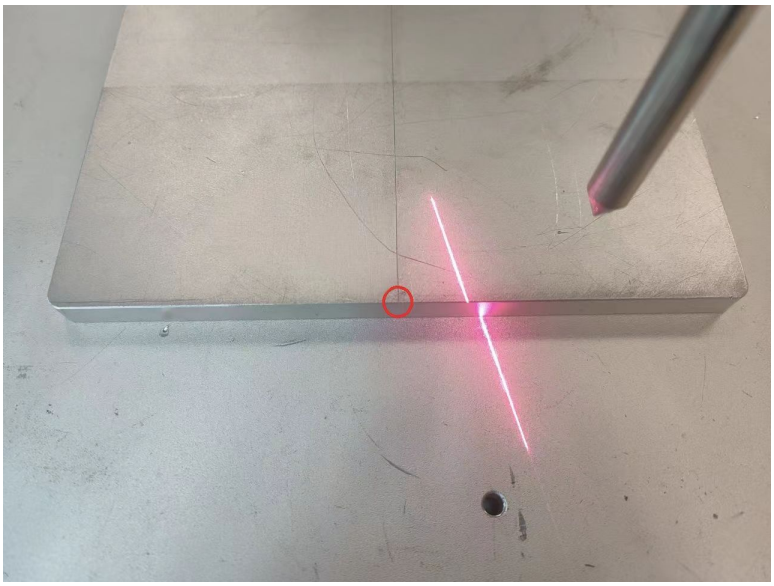


Figure 8.8-4 Laser Sensor Calibration Point

Step1: Open the robot WebApp, navigate to “Initial” -> “Basic” -> “Tool Coordinate” to enter the Tool Coordinate System interface. Select an unused tool coordinate system, click to modify its name to “Welding Torch”, set the tool type to “Tool”, and the installation location to “End Flange”.

Figure 8.8-5 Setting the “Welding Torch” Coordinate System

Select another unused coordinate system, change its name to “Laser Sensor”, set the tool type to “Sensor”, and the installation location to “End Flange”.

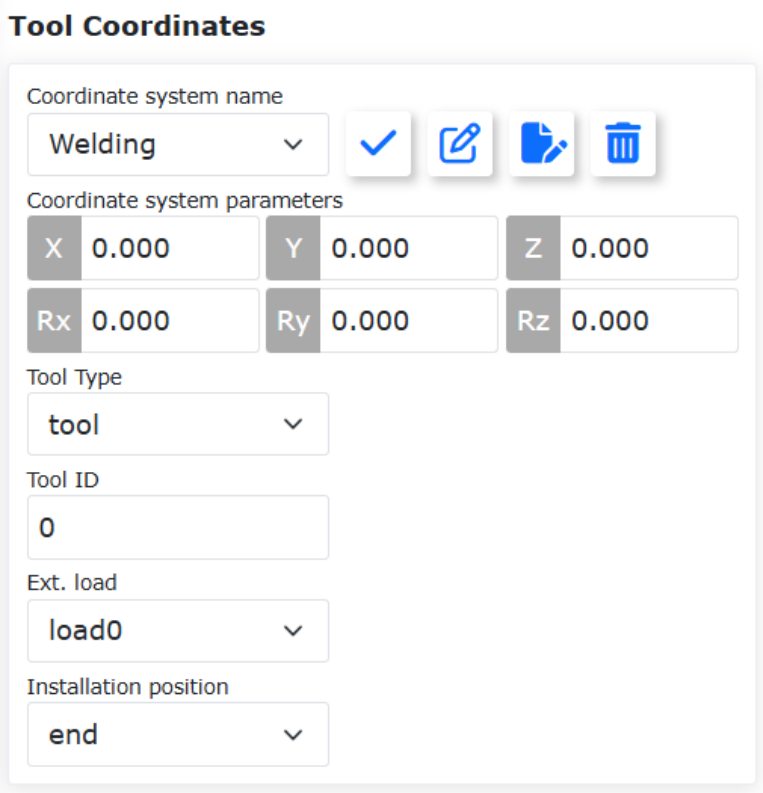
Figure 8.8-6 Setting the “Laser Sensor” Coordinate System

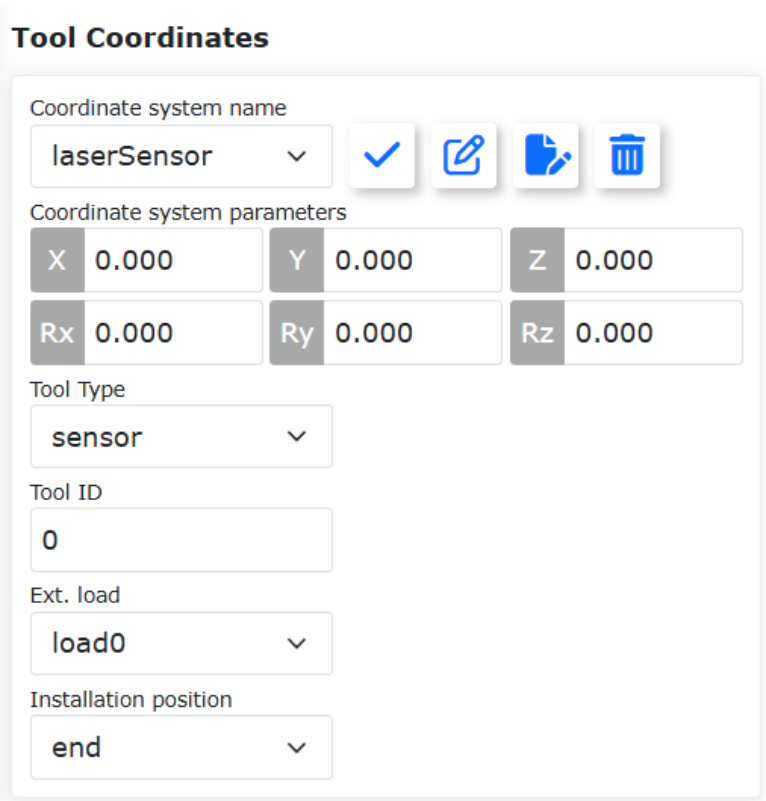
Step2: Calibrate the tool coordinate system of the welding torch using the Six-Point Method: Select the “Welding Torch” coordinate system, click the modify button, and use the Six-Point Method to calibrate the welding torch tool coordinate system (for specific calibration methods, refer to the F&R documentation, which will not be detailed here).

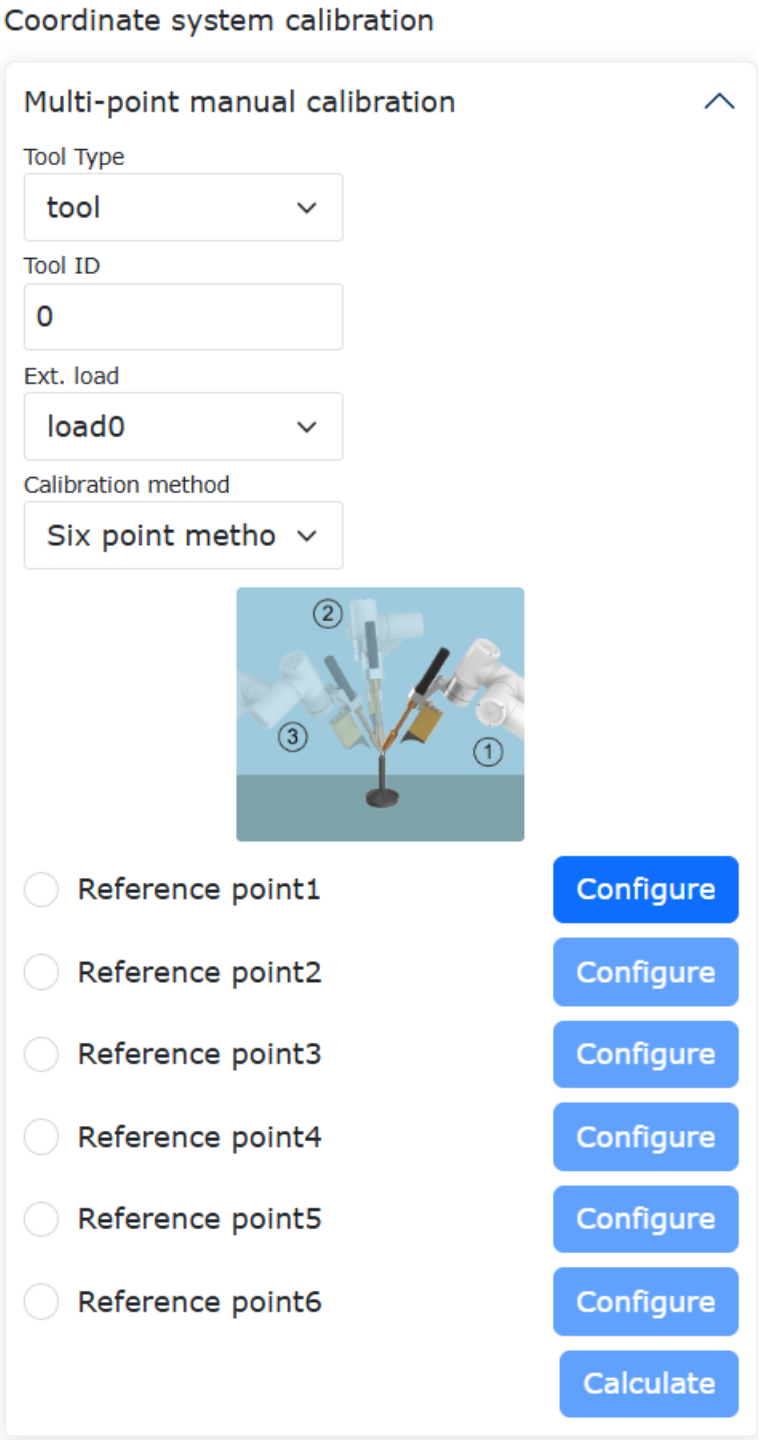
Figure 8.8-7 “Welding Torch” Coordinate System Calibration

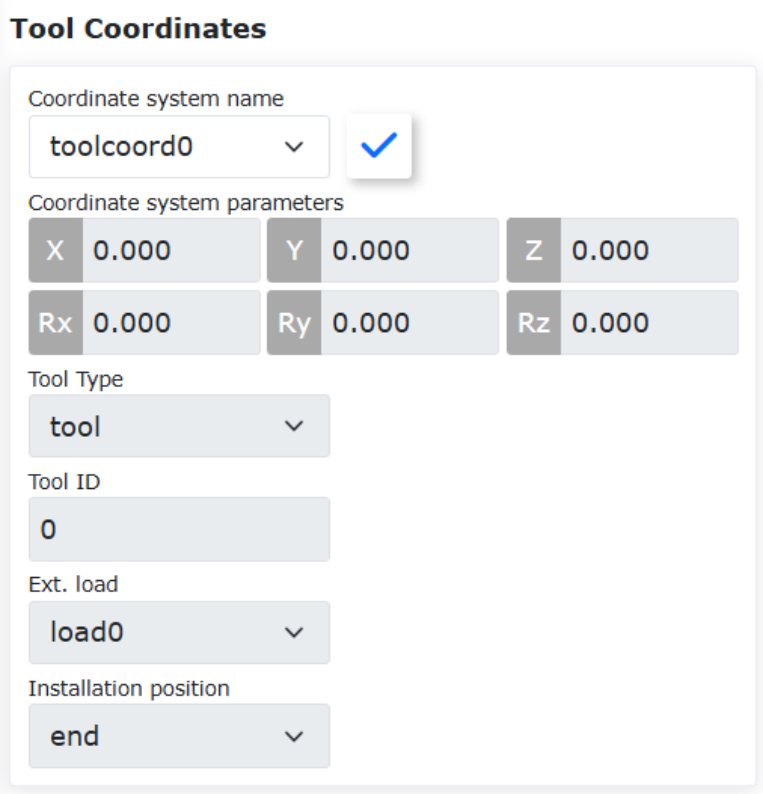
Step3: In the “Tool Coordinate System Settings”, select the 0th coordinate system (Base Coordinate System), default name “toolcoord0”, click “Apply” to switch the current coordinate system to the base coordinate system.

Figure 8.8-8 Sensor Calibration Step 1


Tool Coordinates
Coordinate system name
Welding
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
tool
Tool ID
0
Ext. load
load0
Installation position
end


Tool Coordinates
Coordinate system name
laserSensor
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
sensor
Tool ID
0
Ext. load
load0
Installation position
end


Coordinate system calibration
Multi-point manual calibration
Tool Type
tool
Tool ID
0
Ext. load
load0
Calibration method
Six point metho
2
3
1
Reference point1
Configure
Reference point2
Configure
Reference point3
Configure
Reference point4
Configure
Reference point5
Configure
Reference point6
Configure
Calculate



Step4: Select the previously set “Laser Sensor” coordinate system again (do not click “Apply”), click the “Edit” button, select the tool type as “Sensor”, confirm the sensor is fixed on the “Robot End Flange”, and select the calibration method as “Five-Point Method”.

Figure 8.8-9 Sensor Calibration Step 2

Step5: Jog the robot so that the tip of the welding torch aligns with the calibration point, select the “Welding Torch” coordinate system, click “Apply”, then click “Set Point 1”, as shown in Figure 13.

Figure 8.8-10 Sensor Calibration Step 3

Figure 8.8-11 Sensor Calibration Step 4

Step6: Select the 0th coordinate system (“toolcoord0”) again; then select the “Sensor” coordinate system (do not click “Apply”), and you can continue with the calibration.

Figure 8.8-12 Sensor Calibration Step 5

Figure 8.8-13 Sensor Calibration Step 6

Step7: Move the laser sensor position so that the laser beam just scans the calibration point, click “Set Point 2”; at this time, the sensor output value at the corresponding sequence number on the left will display the current sensor data. If the data is normal, it indicates the current calibration point was successful; otherwise, recalibration is needed.

Figure 8.8-14 Sensor Calibration Step 7

Figure 8.8-15 Sensor Calibration Step 8

Step8: Sequentially make the laser irradiate the calibration point from three more different poses, and click “Set Point 3”, “Set Point 4”, and “Set Point 5” respectively. Finally, ensuring the data for each point is normal, click the “Calculate” button.

Figure 8.8-16 Sensor Calibration Step 9

Coordinate system calibration

Multi-point manual calibration

Tool Type

sensor

Tool ID

0

Ext. load

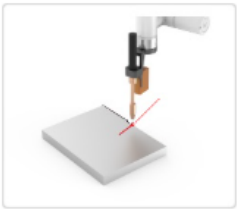
load0

Sensor fixing

Robot end

Calibration method

Five point meth



	Sensor output value		
Serial No	X	Y	Z
1			
2			
3			
4			
5			

Reference point1 Configure

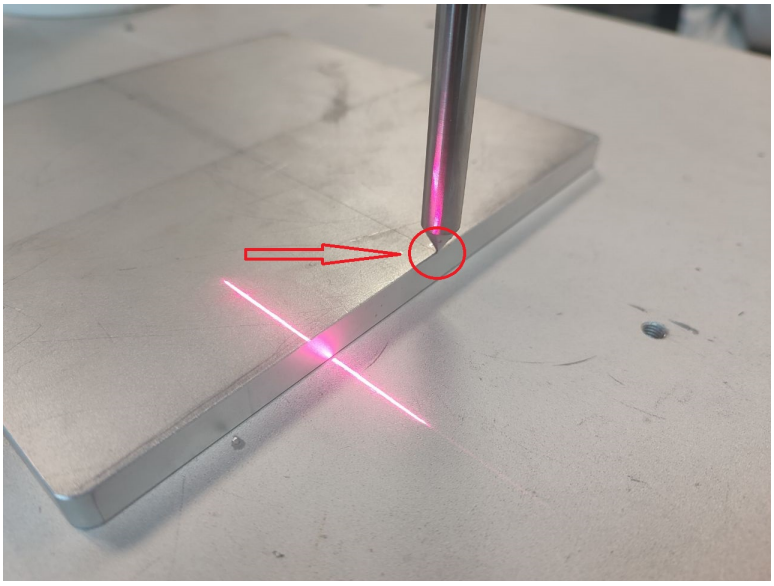
Reference point2 Configure

Reference point3 Configure

Reference point4 Configure

Reference point5 Configure

Calculate



Step9: At this point, the WebApp displays the sensor calibration result and accuracy. Click the "Apply" button to complete the laser sensor calibration. If the calibration accuracy is too poor, you can choose to click the "Cancel" button and recalibrate.

Figure 8.8-17 Sensor Calibration Accuracy

1.3.8.8.4 Laser Sensor Application

Before using the laser sensor, first apply the "Welding Torch" tool coordinate system as the current tool coordinate system.

Figure 8.8-18 Apply Welding Torch Coordinate System

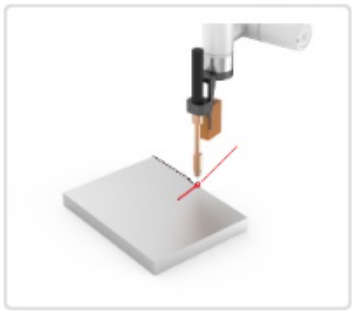
1.3.8.8.4.1 Laser Sensor Teaching Points

Jog the robot so that the laser sensor beam points to the desired weld seam point for teaching. In the WebApp, select the sensor as "Laser Sensor", enter the sensor point name as "laserPt", and click the "Add" button. Create a new user program "testLaser.lua", create a PTP motion command, select "laserPt" as the target point, and execute this instruction step by step. At this point, the welding torch will move to the point previously indicated by the laser sensor.

Figure 8.8-19 Laser Sensor Weld Seam Point

Figure 8.8-20 Teaching Sensor Point

Figure 8.8-21 Welding Torch Pointing to Weld Seam Point



	Sensor output value		
Serial No	**X**	**Y**	**Z**
1	-389.12	-427.23	286.1
2			
3			
4			
5			

Reference point1 Configure

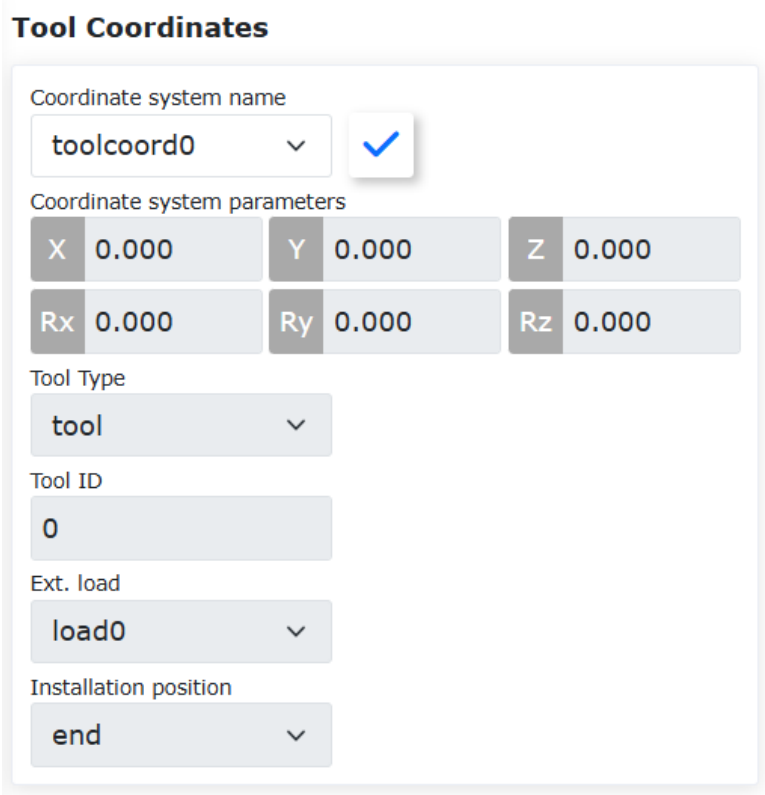
Reference point2 Configure

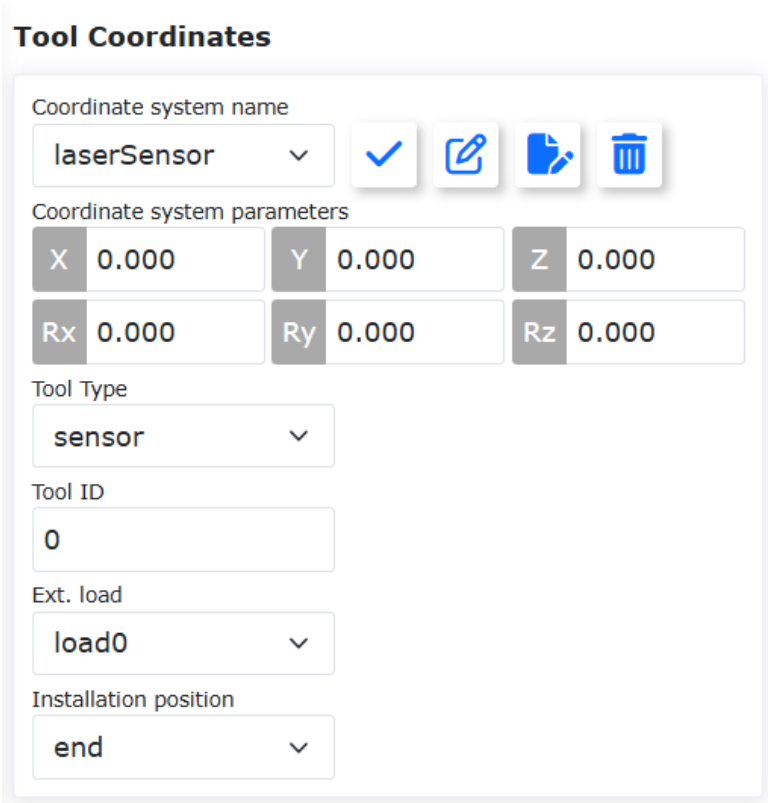
Reference point3 Configure

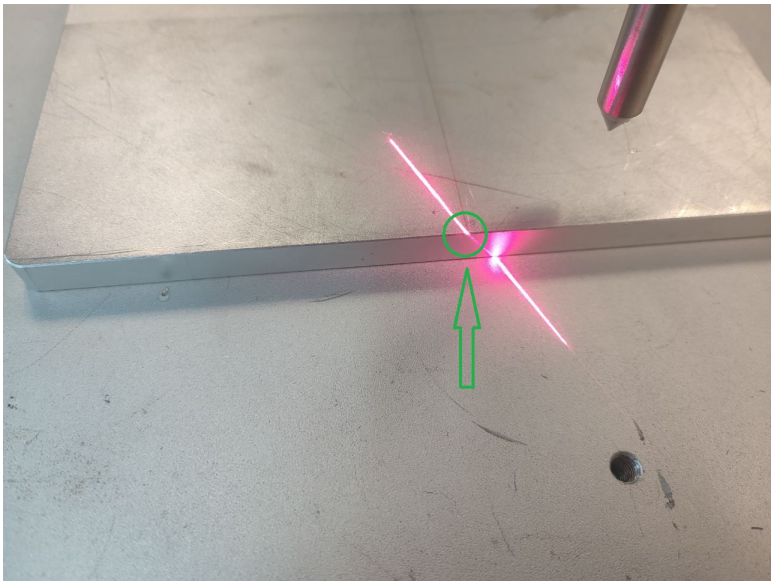
Reference point4 Configure

Reference point5 Configure

Calculate


Tool Coordinates
Coordinate system name
toolcoord0
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
tool
Tool ID
0
Ext. load
load0
Installation position
end


Tool Coordinates
Coordinate system name
laserSensor
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
sensor
Tool ID
0
Ext. load
load0
Installation position
end



1.3.8.8.4.2 Laser Seam Finding + Tracking

The collaborative robot, working with the laser sensor, requires the following steps to complete the laser seam finding + laser tracking function:

(1) The robot moves to a point outside the weld seam;

(2) Start laser seam finding, and the robot moves towards the weld seam position carrying the laser sensor;

(3) The laser sensor identifies the weld seam, and the robot moves the welding torch to the identified weld seam start point;

(4) Laser tracking begins, and simultaneously the robot moves towards the weld seam end point, with the laser sensor recording the position in real time during the movement;

(5) The welding torch moves along the positions recorded by the laser sensor, achieving the tracking effect.

Before debugging seam finding and tracking, please ensure the sensor is correctly installed, the "Welding Torch" tool coordinate system is correctly calibrated, and the laser sensor is also correctly calibrated. Assuming the green line in the figure is the weld seam to be welded, to make the robot automatically find the welding start point A and automatically weld to point B, the following instructions need to be written:

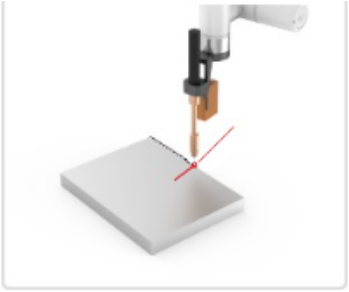
Figure 8.8-22 Sensor Installation

1.3.8.8.4.3 Writing Seam Finding Instructions

Create a new user program "laserTrack.lua", select "Welding Instructions". Click "Laser Tracking", and the laser tracking instruction addition page will pop up.

Figure 8.8-23 Laser Tracking Instruction

Find the "Seam Finding Command", select the coordinate system name as "Laser Sensor", direction select "+x" indicating the robot carrying the laser sensor moves from the current position along the "+x" direction of the "Welding Torch" coordinate system while searching for the weld seam. "Speed" is the moving speed for the laser sensor seam finding, Length is the maximum seam finding length for the laser sensor. If the robot exceeds this length without finding the weld seam, it will report an error. Maximum Seam Finding Time is similar to the length; if this time is exceeded without finding the weld seam, the robot reports an error. Please input the above parameters correctly according to



	Sensor output value		
Serial No	X	Y	Z
1	-389.12	-427.23	286.1
2	0.00	-5.12	-120.75
3			
4			
5			

Reference point1 Configure

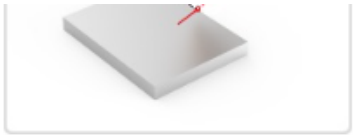
Reference point2 Configure

Reference point3 Configure

Reference point4 Configure

Reference point5 Configure

Calculate



	Sensor output value		
Serial No	X	Y	Z
1	-389.12	-427.23	286.1
2	0.00	-5.12	-120.75
3	0.00	1.37	-116.73
4	0.00	16.07	-117.24
5	0.00	5.90	-117.59

Reference point1 Configure

Reference point2 Configure

Reference point3 Configure

Reference point4 Configure

Reference point5 Configure

Calculate

Tool Type

sensor

Tool ID

0

Ext. load

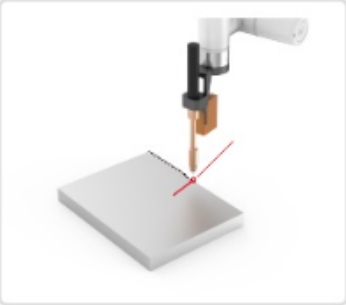
load0

Sensor fixing

Robot end

Calibration method

Five point meth



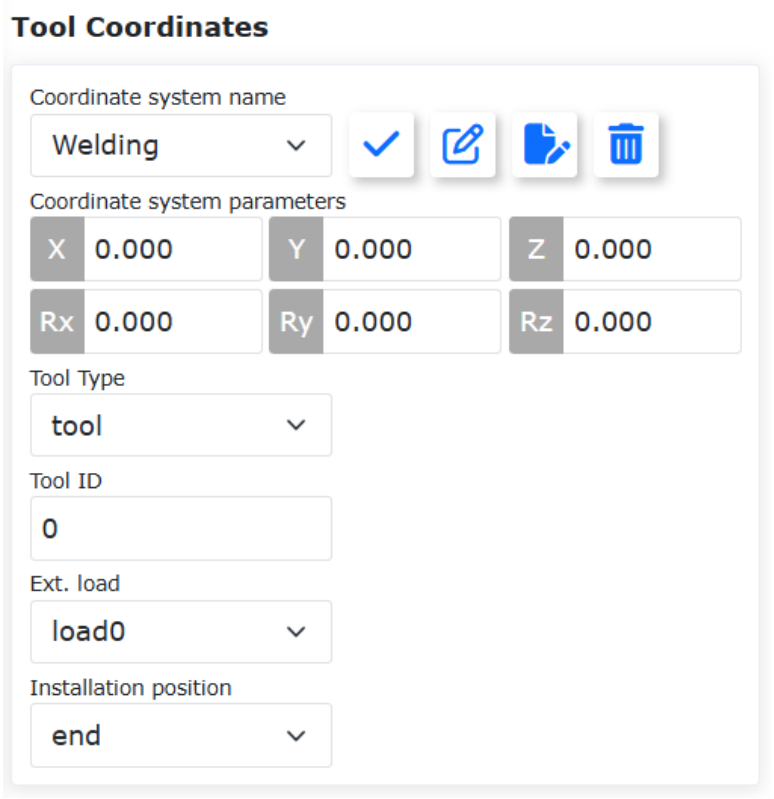
Calculation results

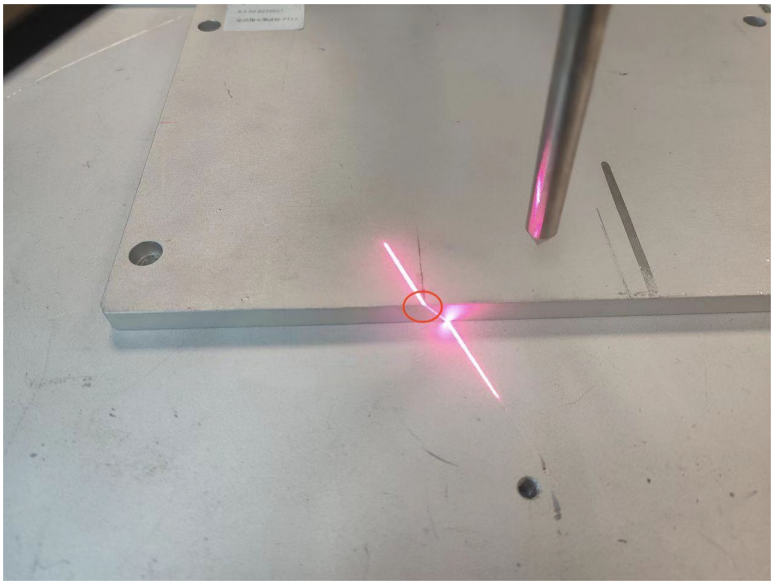
X	-66.27	Y	-730.3
Z	-59.14	Rx	-154.8
Ry	81.712	Rz	23.447

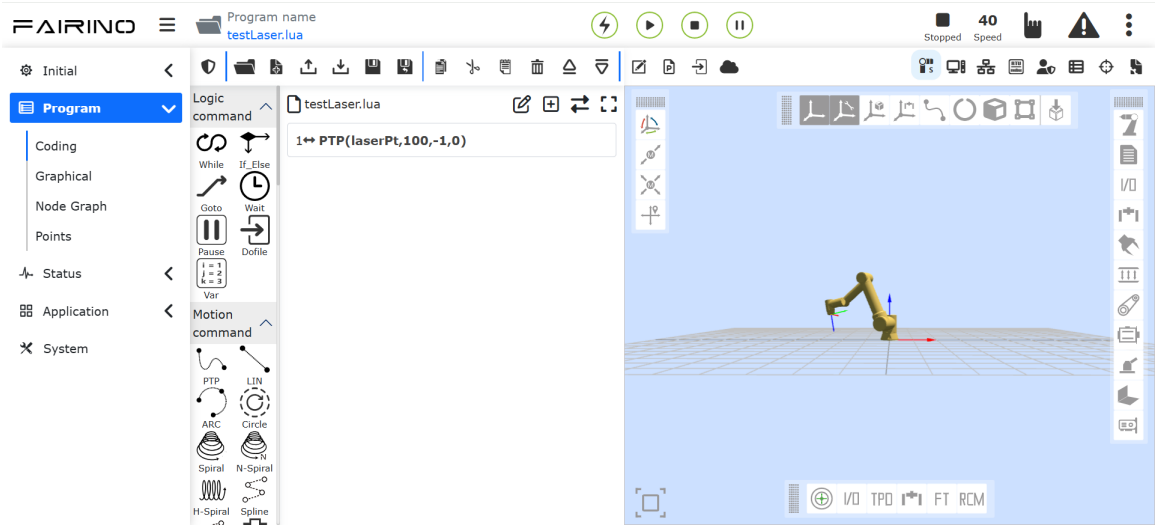
Accuracy (Unit: mm)

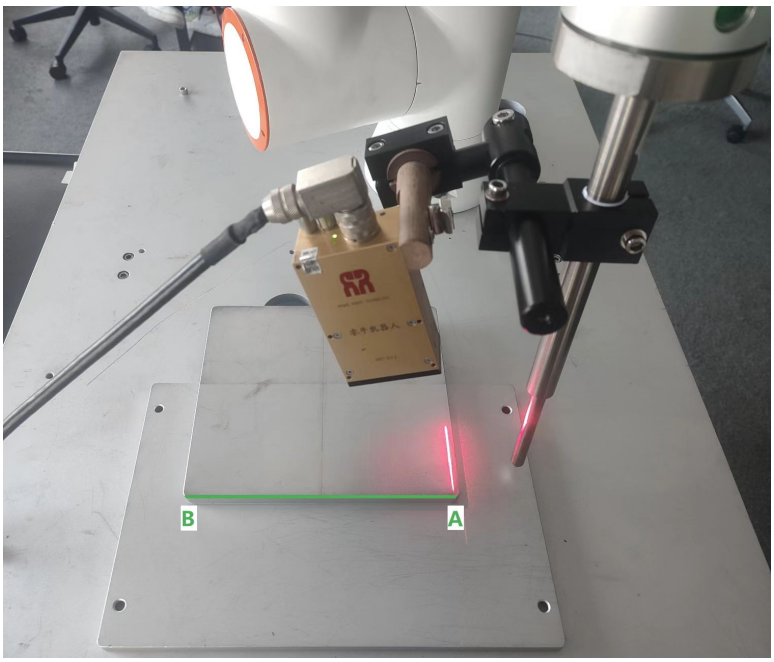
X	237.74	Y	692.83
Z	62.846	Comp.	735.18

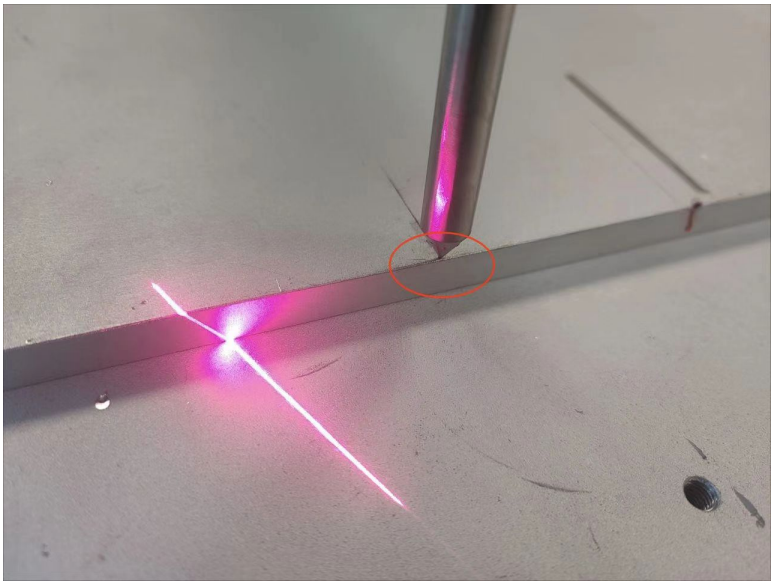
Cancel Apply

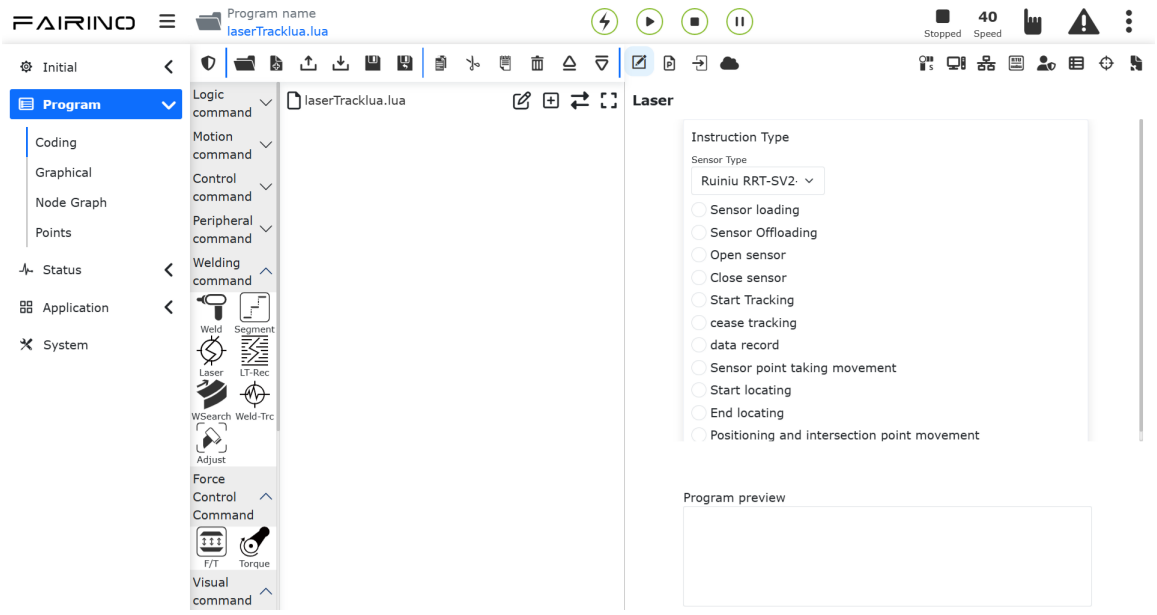

Tool Coordinates
Coordinate system name
Welding
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
tool
Tool ID
0
Ext. load
load0
Installation position
end




FAIRINO
Program name
testLaser.lua
Stopped
40
Speed
Initial
Program
Coding
Graphical
Node Graph
Points
Status
Application
System
Logic command
While
If_Else
Goto
Wait
Pause
Dofile
Var
Motion command
PTP
LIN
ARC
Circle
Spiral
N-Spiral
H-Spiral
Spline
testLaser.lua
1 PTP(laserPt,100,-1,0)
I/O
TPD
FT
RCM


B
A




FAIRINO
Program name
laserTracklua.lua
Stopped
40
Speed
Initial
Program
Coding
Graphical
Node Graph
Points
Status
Application
System
Logic command
Motion command
Control command
Peripheral command
Welding command
Weld
Segment
Laser
LT-Rec
WSearch
Weld-Trc
Adjust
Force Control Command
F/T
Torque
Visual command
laserTracklua.lua
Laser
Instruction Type
Sensor Type
Ruiniu RRT-SV2·
Sensor loading
Sensor Offloading
Open sensor
Close sensor
Start Tracking
cease tracking
data record
Sensor point taking movement
Start locating
End locating
Positioning and intersection point movement
Program preview

the actual scenario. Click the “Seam Finding Start” and “Seam Finding End” instructions sequentially, and click the “Apply” button.

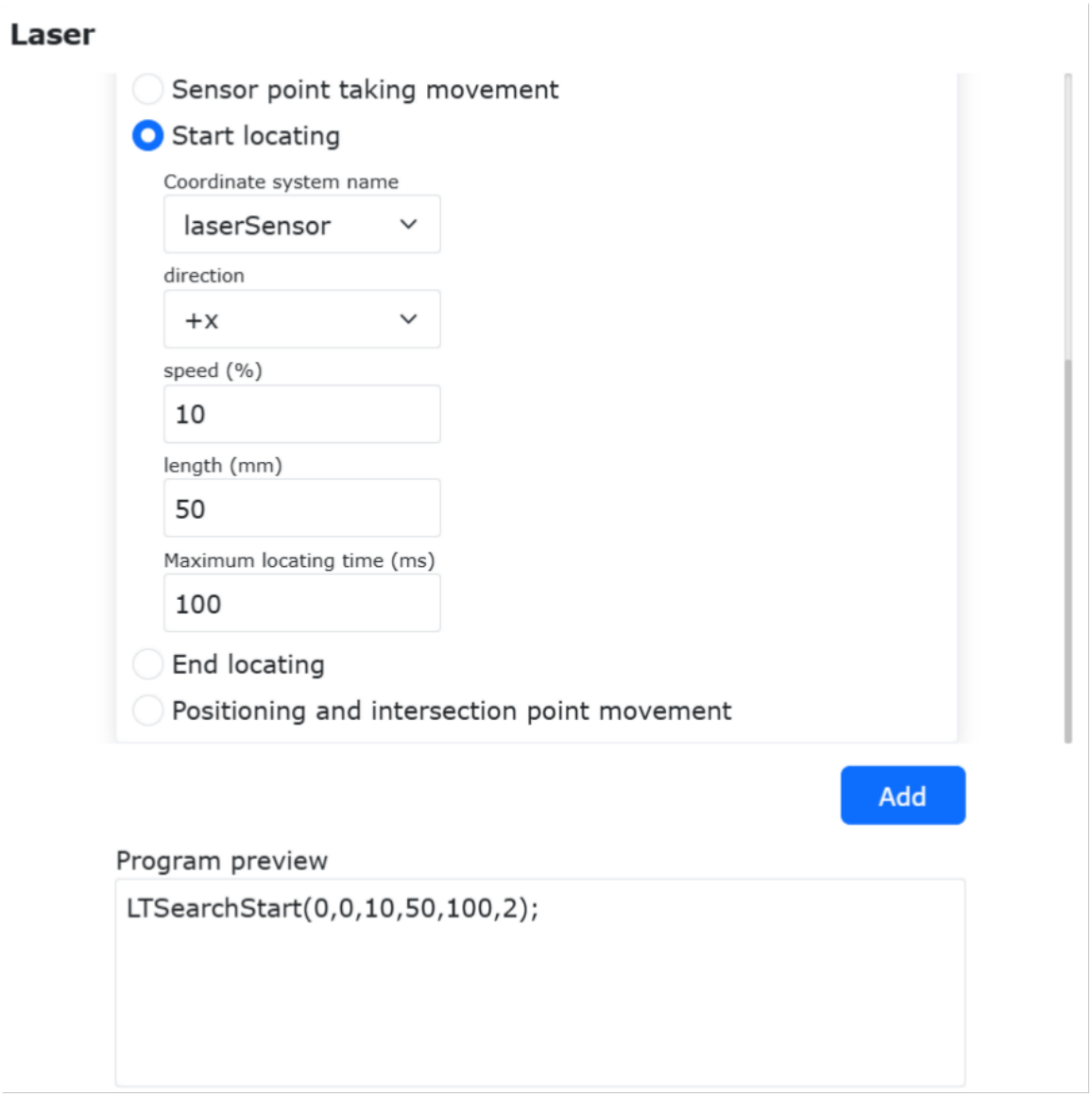
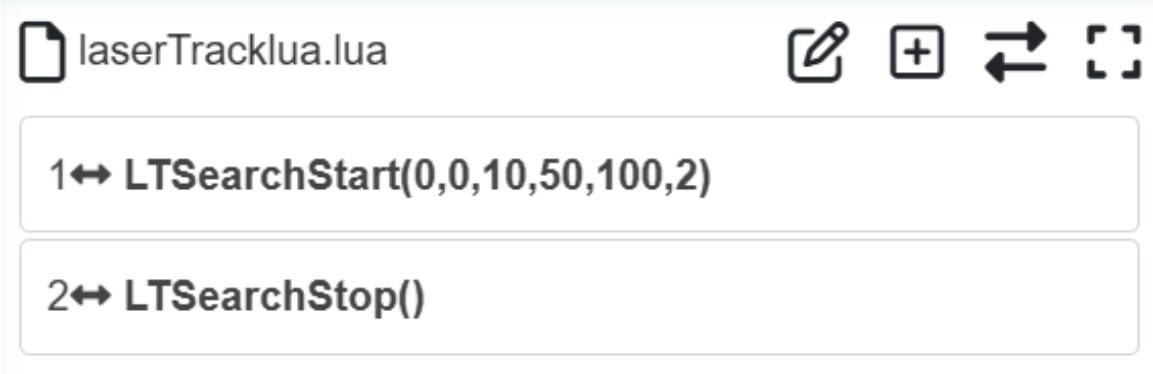


Figure 8.8-24 Adding Seam Finding Instructions

At this point, the corresponding laser seam finding start and end instructions will be added to “laserTrack.lua”.

Figure 8.8-25 Seam Finding Program



1.3.8.8.4.4 Writing Instruction to Move to Seam Finding Point

Add a Linear (LIN) point-to-point motion instruction, with the target point being "seamPos", which is the laser sensor seam finding point.

Note: The "seamPos" point is an internal point in the robot system dedicated for laser sensor seam finding. There is no need to teach this point manually; after laser sensor seam finding, the seam finding point information is automatically stored in the "seamPos" point.

Offset can be set for the seam finding point. The offset type can be "Base Coordinate System Offset", "Tool Coordinate System Offset", or "Laser Raw Data Offset".

Figure 8.8-26 Seam Finding Offset Options

When the seam finding offset function is enabled, offset parameters can be set. "dx" represents the offset distance along the x-direction of the selected coordinate system, "drx" represents the rotation angle around the x-axis of the selected coordinate system. Click the "Add" button, then click the "Apply" button.

Figure 8.8-27 Seam Finding Offset Parameter Settings

At this point, the instruction to move to the seam finding point will be added to "testTrack.lua", as shown in Figure 32.

Figure 8.8-28 Seam Finding Offset Program

1.3.8.8.4.5 Writing Laser Tracking Instructions

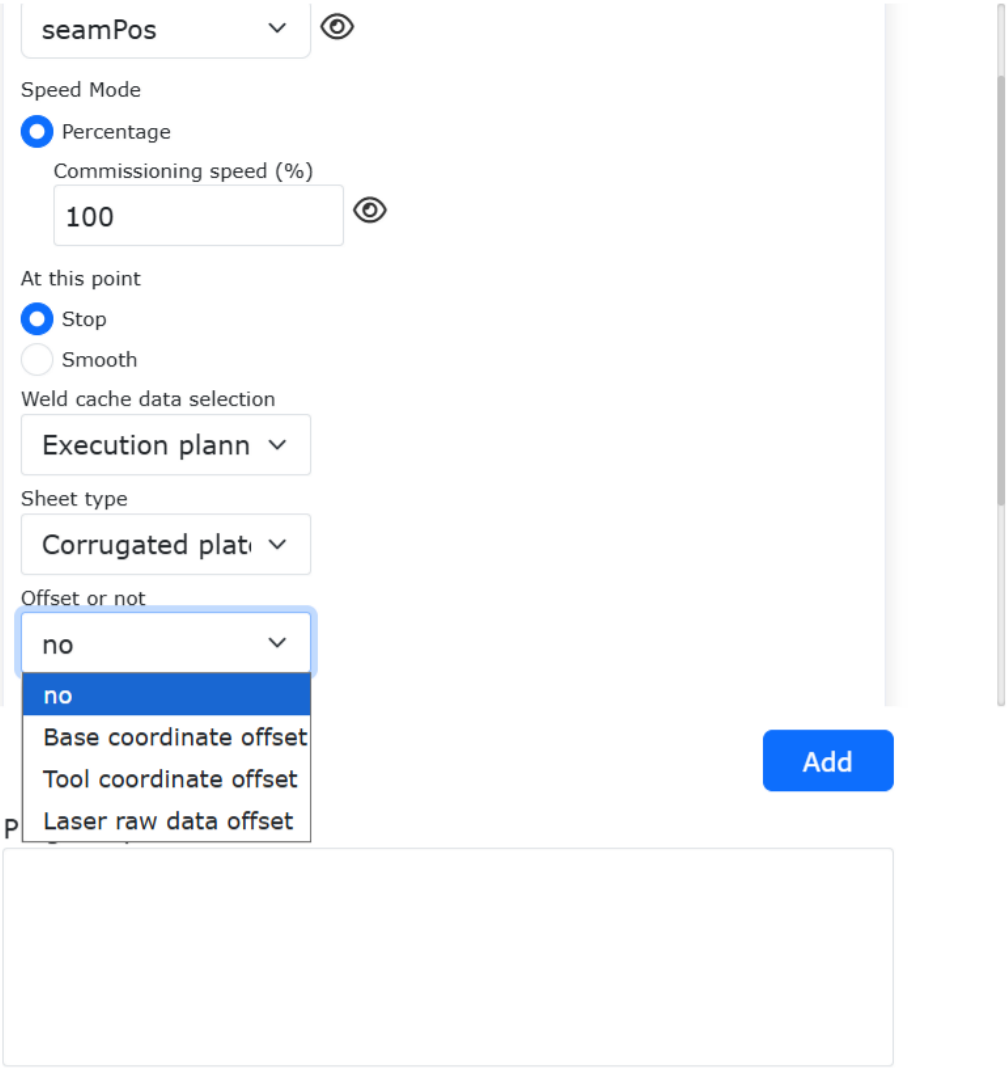
Open the "Laser Tracking" instruction addition page again, click the "Start Tracking" and "Stop Tracking" buttons sequentially, and finally click the "Apply" button at the bottom of the page.

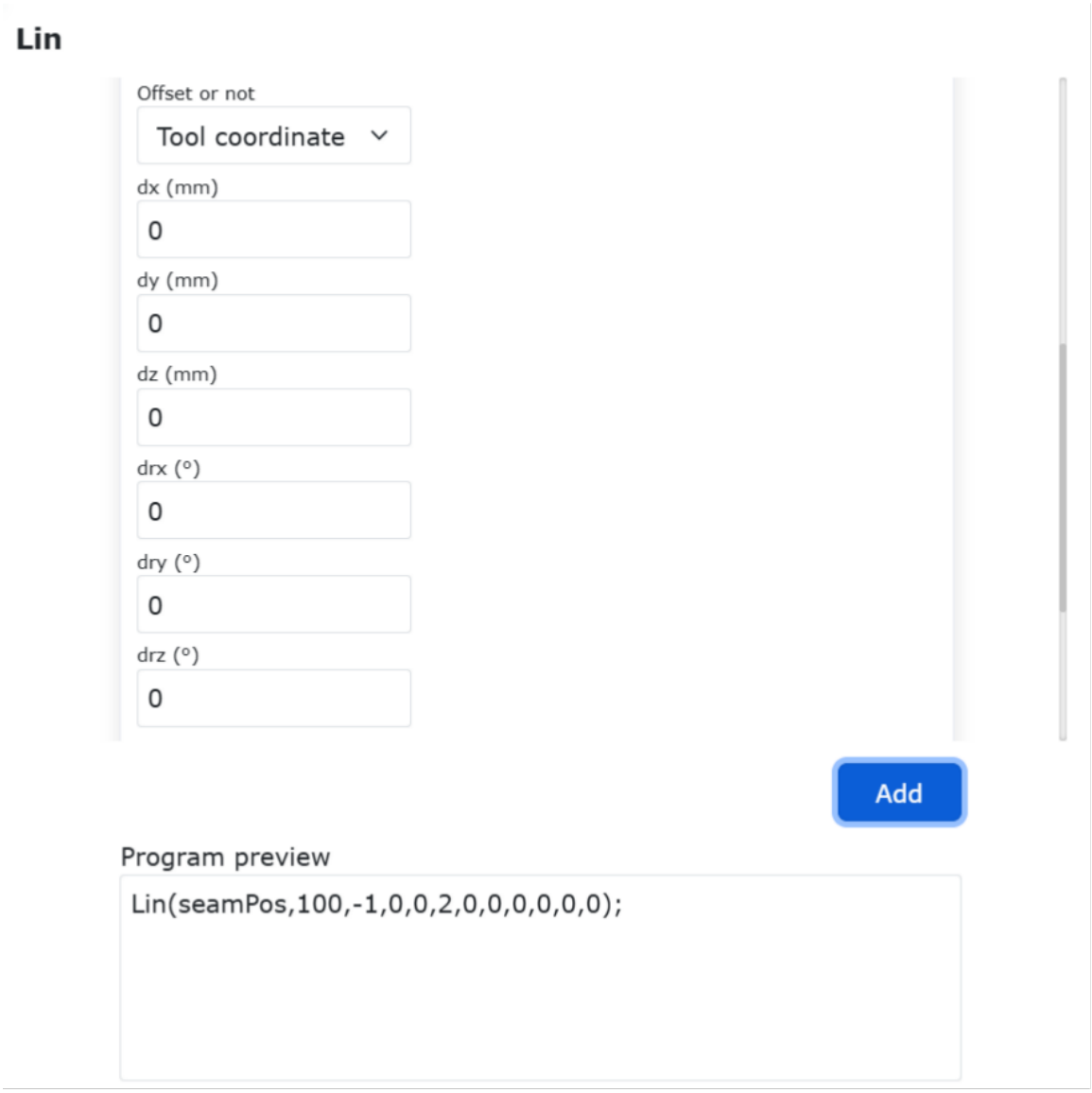
Figure 8.8-29 Laser Tracking Start and Stop

The user program "testTrack.lua" at this stage:

Figure 8.8-30 Laser Tracking Program

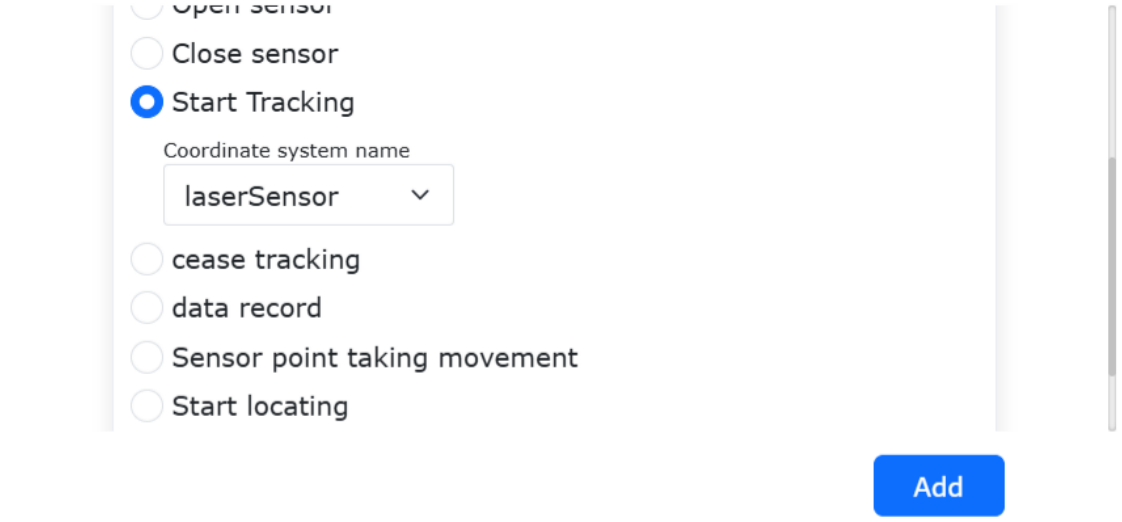
Lin

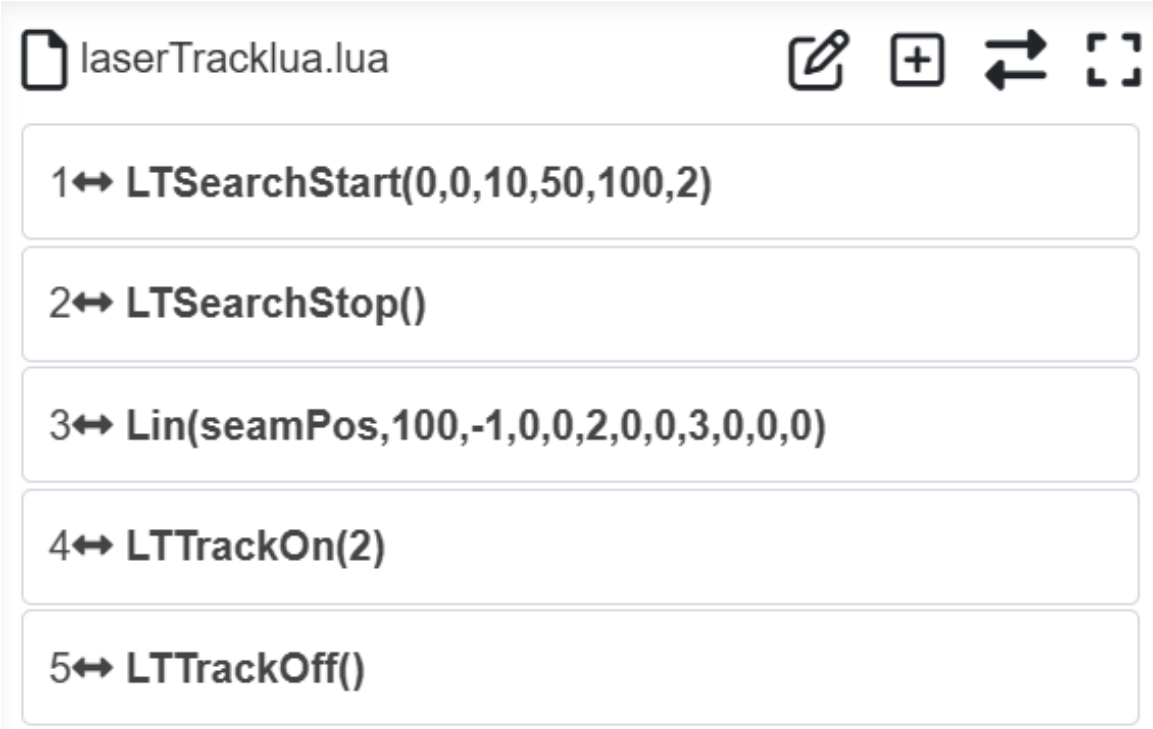

seamPos
Speed Mode
Percentage
Commissioning speed (%)
100
At this point
Stop
Smooth
Weld cache data selection
Execution plann
Sheet type
Corrugated plat
Offset or not
no
no
Base coordinate offset
Tool coordinate offset
Laser raw data offset
Add


Lin
Offset or not
Tool coordinate
dx (mm)
0
dy (mm)
0
dz (mm)
0
drx (°)
0
dry (°)
0
drz (°)
0
Add
Program preview
Lin(seamPos,100,-1,0,0,2,0,0,0,0,0,0);


laserTracklua.lua
1↔ LTSearchStart(0,0,10,50,100,2)
2↔ LTSearchStop()
3↔ Lin(seamPos,100,-1,0,0,2,0,0,3,0,0,0)

Laser


Close sensor
Start Tracking
Coordinate system name
laserSensor
cease tracking
data record
Sensor point taking movement
Start locating
Add
Program preview
LTTrackOn(2);


laserTracklua.lua
1 LTSearchStart(0,0,10,50,100,2)
2 LTSearchStop()
3 Lin(seamPos,100,-1,0,0,2,0,0,3,0,0,0)
4 LTTrackOn(2)
5 LTTrackOff()

1.3.8.8.4.6 Writing Seam Finding Start Point and Tracking End Point Instructions

Before starting laser seam finding, a seam finding start point needs to be specified. The robot first moves to the seam finding start point, then performs seam finding along a certain direction and speed. Teach the seam finding start point "seamStartPt" near where the laser sensor beam is close to the weld seam start point A. Pay attention to matching the seam finding start point with the seam finding direction to ensure the robot can find the weld seam position within the set distance and maximum seam finding time.

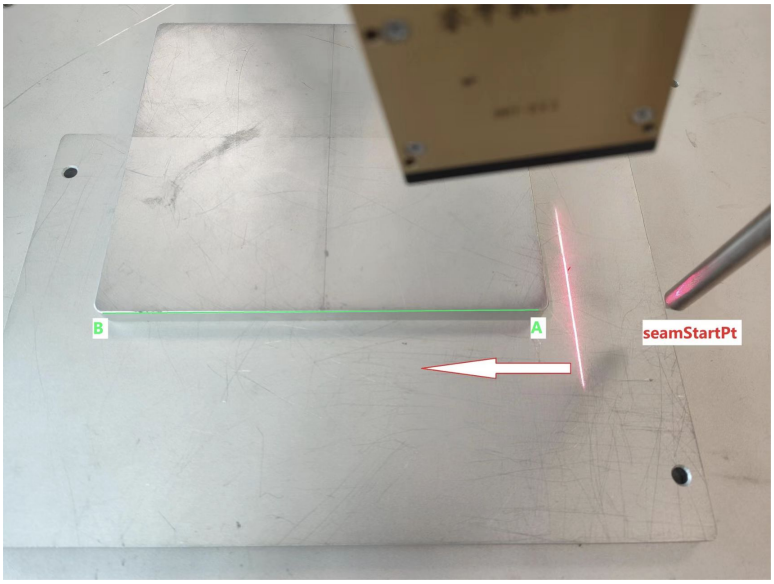


Figure 8.8-31 Seam Finding Start Point

Teach the tracking end point "trackEndPt" at the end of the weld seam.

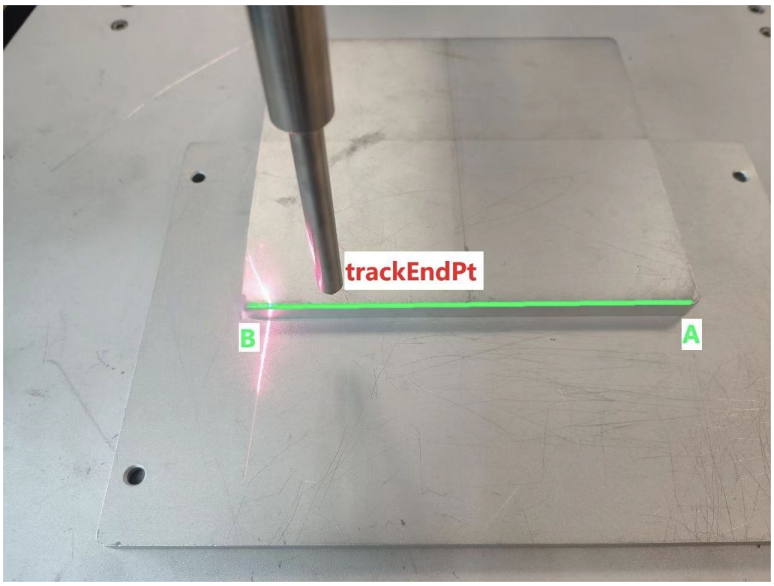
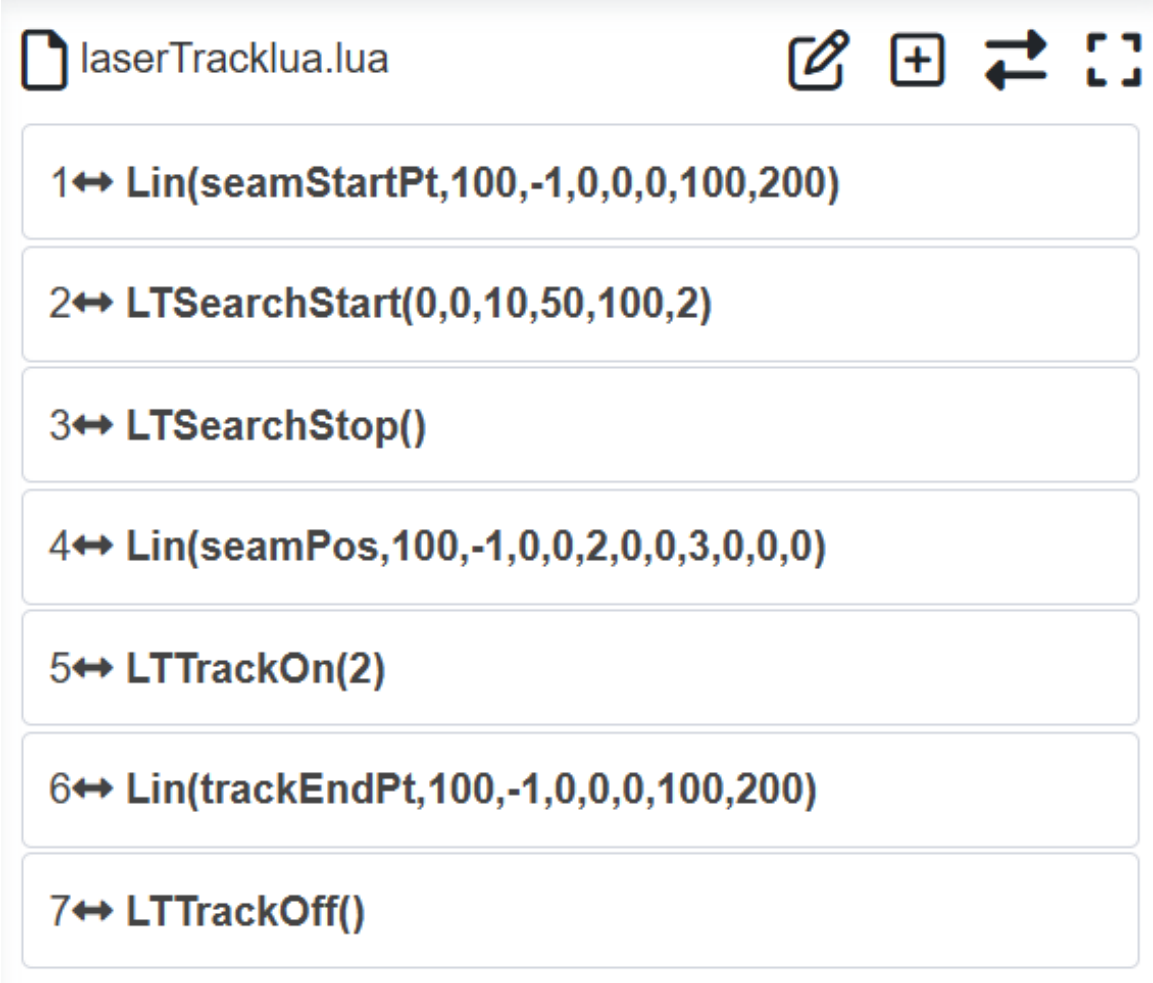


Figure 8.8-32 Seam Finding End Point

Add the above two points to the "testTrack.lua" user program. The final user program is as follows:

Figure 8.8-33 Seam Finding and Tracking Program


laserTracklua.lua
1↔ Lin(seamStartPt,100,-1,0,0,0,100,200)
2↔ LTSearchStart(0,0,10,50,100,2)
3↔ LTSearchStop()
4↔ Lin(seamPos,100,-1,0,0,2,0,0,3,0,0,0)
5↔ LTTrackOn(2)
6↔ Lin(trackEndPt,100,-1,0,0,0,100,200)
7↔ LTTrackOff()

1.3.8.8.4.7 Writing Welding Related Instructions

Finally, add welding instructions between the welding seam finding point "seampos" and "trackEndPt". The final program is as follows:

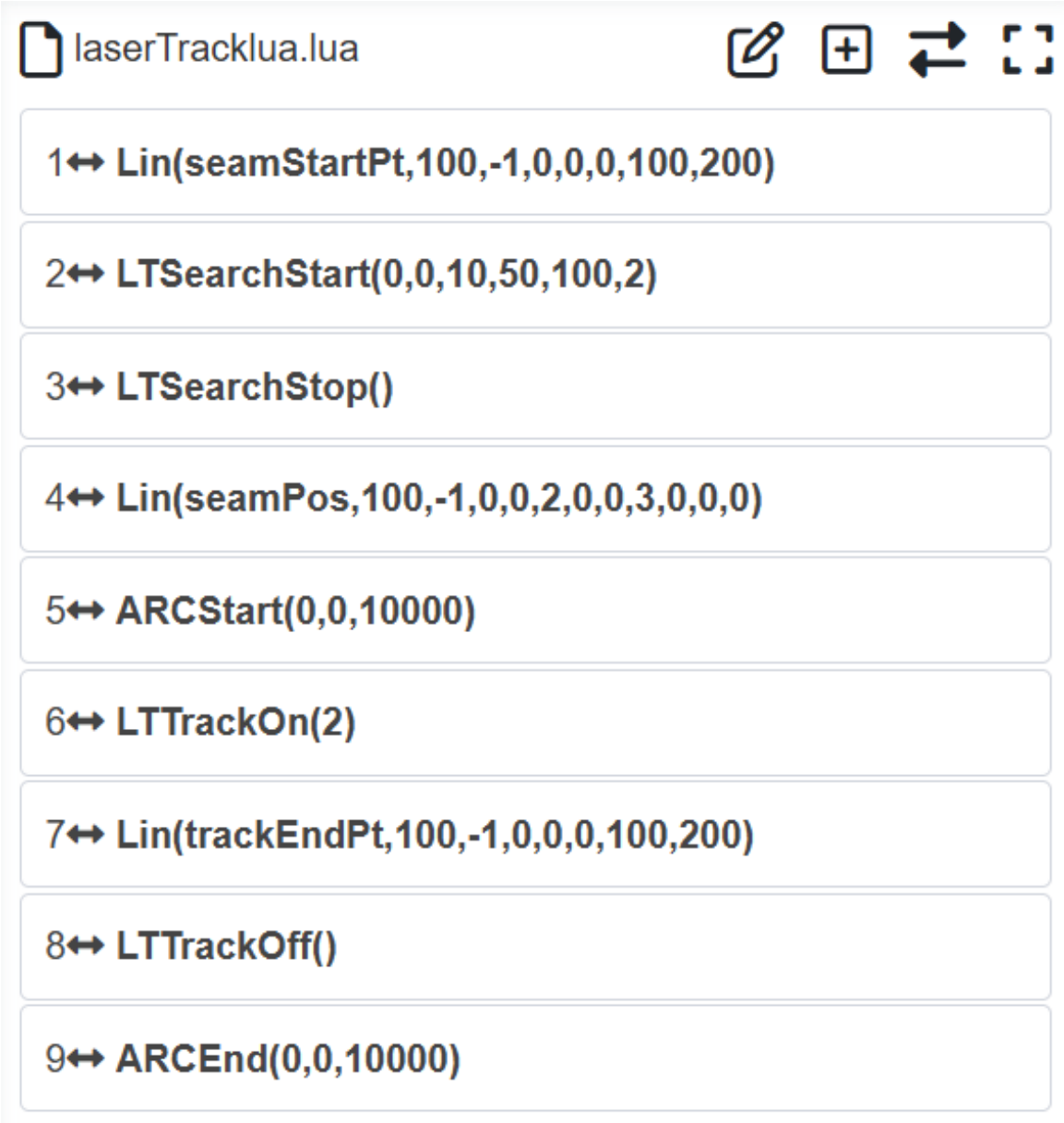


Figure 8.8-34 Seam Finding, Tracking, and Welding Program

Execute the above program. The robot will carry the laser sensor to start the seam finding movement from the seam finding start point. After finding the weld seam, the robot immediately moves to the weld seam start point and executes the arc starting operation. After successful arc starting, the robot moves towards the weld seam end point and tracks the weld seam trajectory during the movement. The robot stops welding upon reaching the weld seam end point.

1.3.8.8.4.8 Laser Trajectory Recording + Trajectory Replay

The workflow for laser trajectory recording + trajectory replay is:

(1) The robot carries the laser sensor to move along a segment of the weld seam, and the laser sensor records the weld seam position trajectory data in real time during the movement;

(2) After the trajectory recording is completed, the robot moves to the starting point of the recorded trajectory;

(3) The robot performs trajectory replay motion along the trajectory recorded by the laser sensor.

1.3.8.8.4.9 Writing Robot Trajectory Recording Instructions

Create a new user program “testRecord.lua”, click “Laser Recording” to open the laser recording instruction addition page, find “Weld Seam Data Recording”, select “Start Recording”, click the “Add” button, select “Stop Recording”, click the “Add” button again; finally click the “Apply” button.

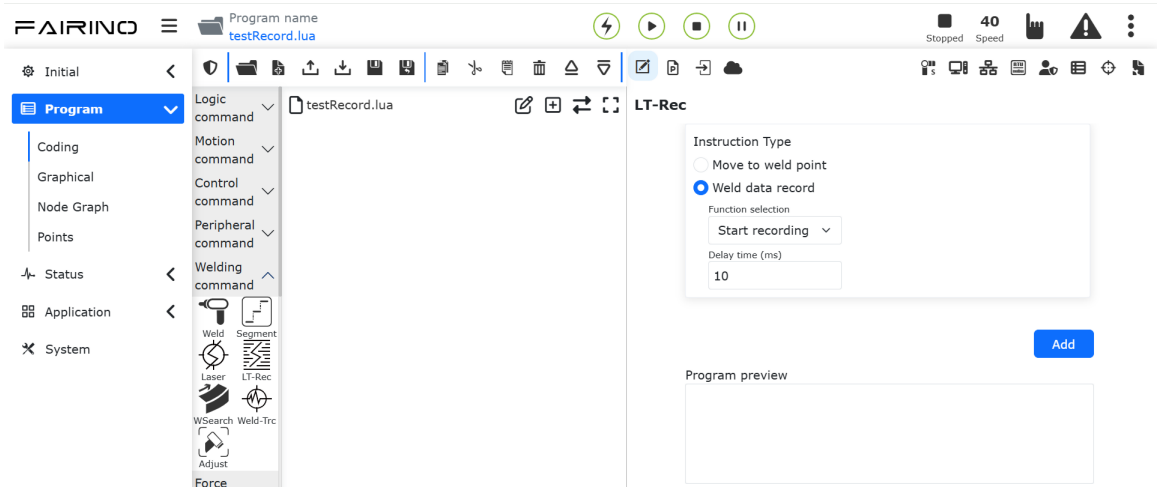


Figure 8.8-35 Laser Recording

Figure 8.8-36 Start Recording and Stop Recording

At this point, the trajectory recording start and stop instructions appear on the page.

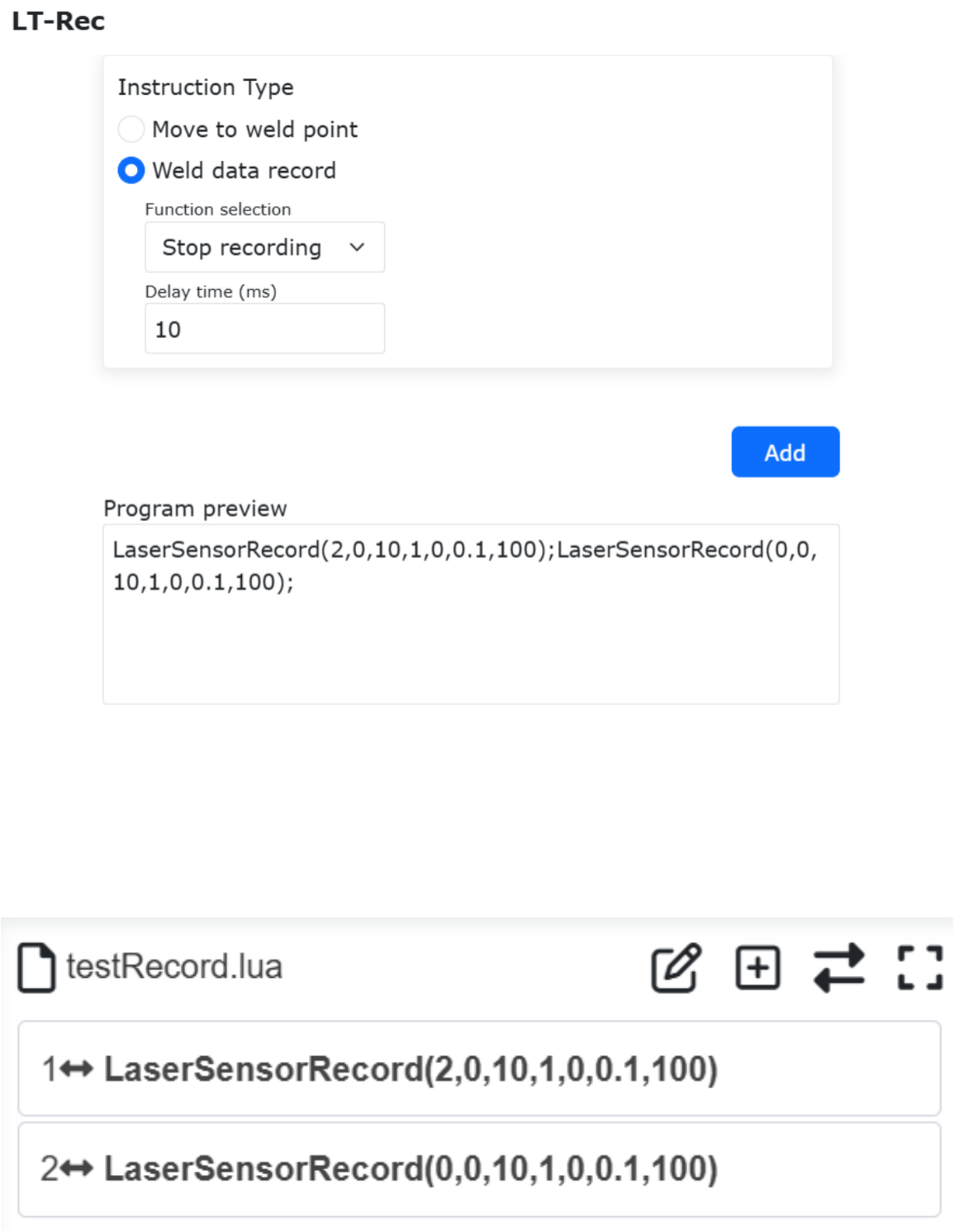
Figure 8.8-37 Trajectory Recording Program

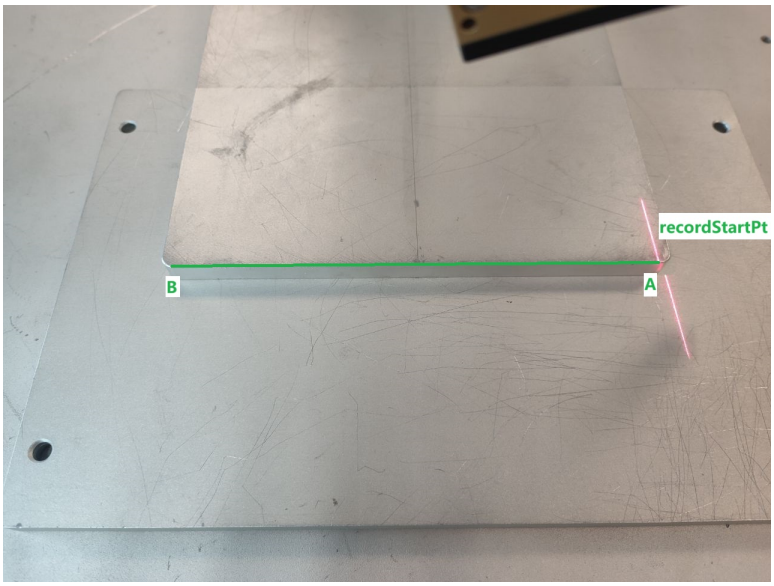
Assuming the green line segment AB in the figure is the weld seam, make the laser irradiate the weld seam start point A and the weld seam end point B respectively, and teach the trajectory recording start point “recordStartPt” and end point “recordEndPt”.

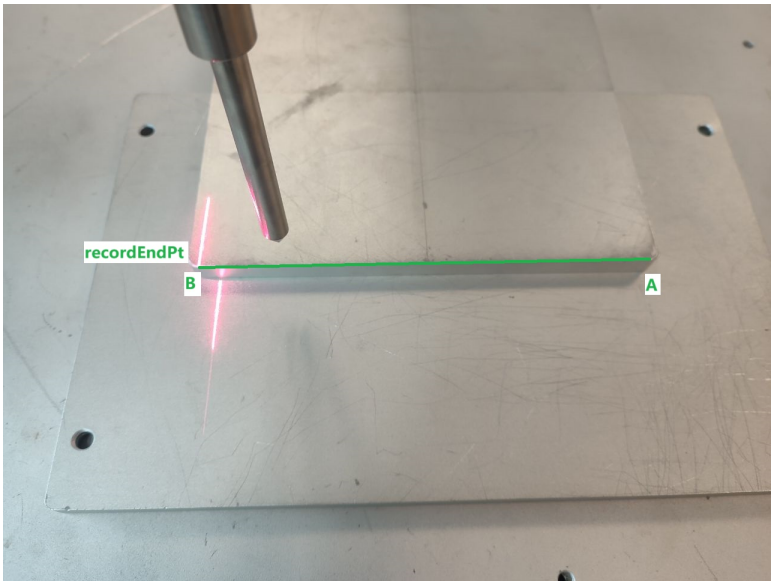
Figure 8.8-38 Trajectory Recording Start and End Points

Add two Linear (LIN) motion instructions to “testRecord.lua”, for moving to the trajectory recording start point “recordStartPt” and end point “recordEndPt” respectively. Adjust the instruction order so that the robot performs the following operations: first move to the “recordStartPt” point, start trajectory recording, robot moves to the “recordEndPt” point, stop trajectory recording.

Figure 8.8-39 Trajectory Recording Program


LT-Rec
Instruction Type
Move to weld point
Weld data record
Function selection
Stop recording
Delay time (ms)
10
Add
Program preview
LaserSensorRecord(2,0,10,1,0,0.1,100);LaserSensorRecord(0,0,10,1,0,0.1,100);
testRecord.lua
1 LaserSensorRecord(2,0,10,1,0,0.1,100)
2 LaserSensorRecord(0,0,10,1,0,0.1,100)


recordStartPt
B
A


recordEndPt
B
A



1.3.8.8.4.10 Writing Instruction for Robot to Move to Trajectory Recording Start Point

Click “Laser Recording” to open the laser recording instruction addition page, find the “Move to Weld Seam Point” section, select the motion mode as PTP, input a certain motion speed, click “Move to Start Point”, then click the “Apply” button.

Figure 8.8-40 Move to Trajectory Start Point

The “testRecord.lua” user program at this stage is as follows:

Figure 8.8-41 Move to Trajectory Start Point Program

1.3.8.8.4.11 Writing Laser Sensor Trajectory Replay Instructions

Click “Laser Recording” to open the laser recording instruction addition page, find “Weld Seam Data Recording”, select “Trajectory Replay”, click the “Add” button, click the “Laser Tracking Replay” button, and finally click the “Apply” button.

Figure 8.8-42 Trajectory Replay

The program after addition is as follows:

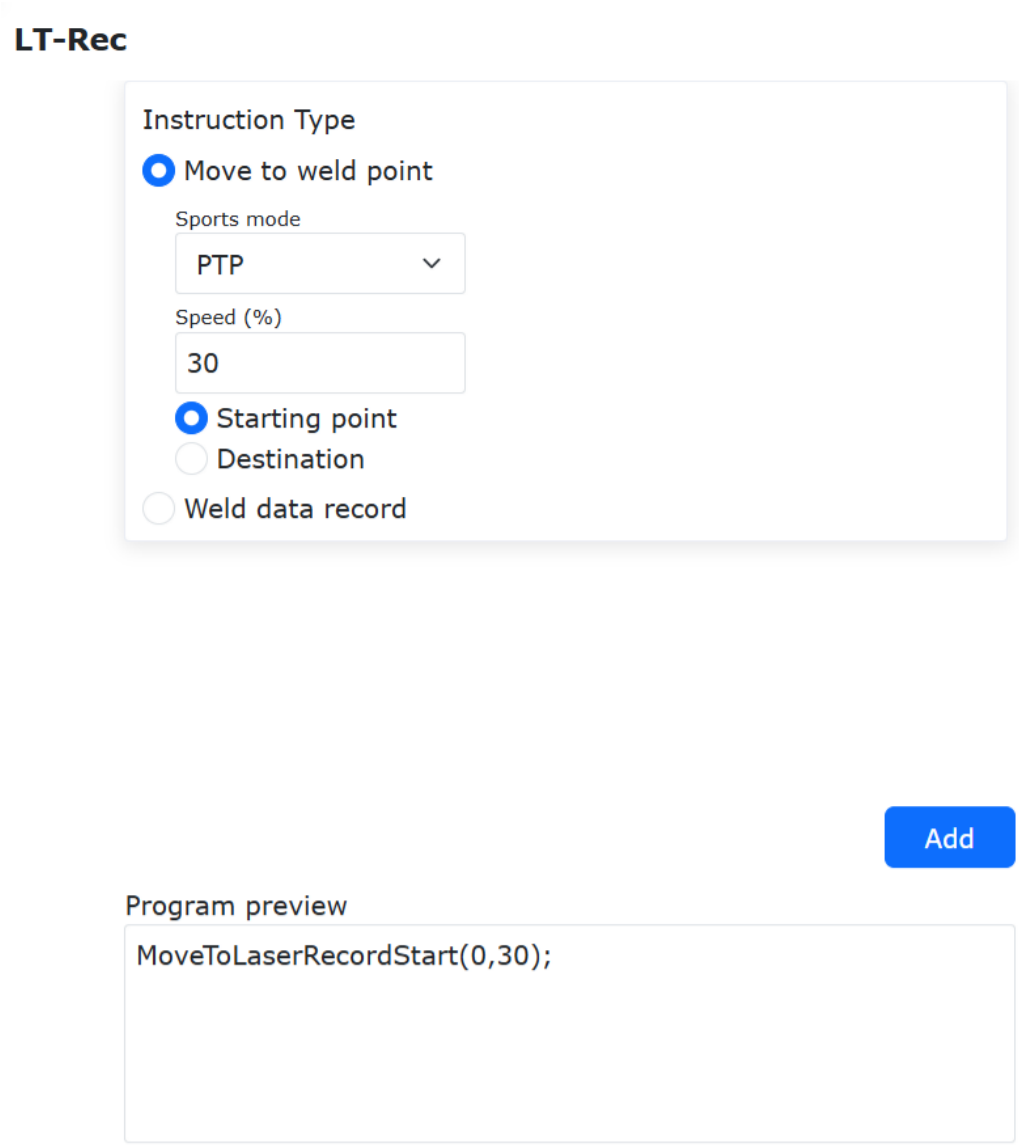
Figure 8.8-43 Trajectory Replay Program

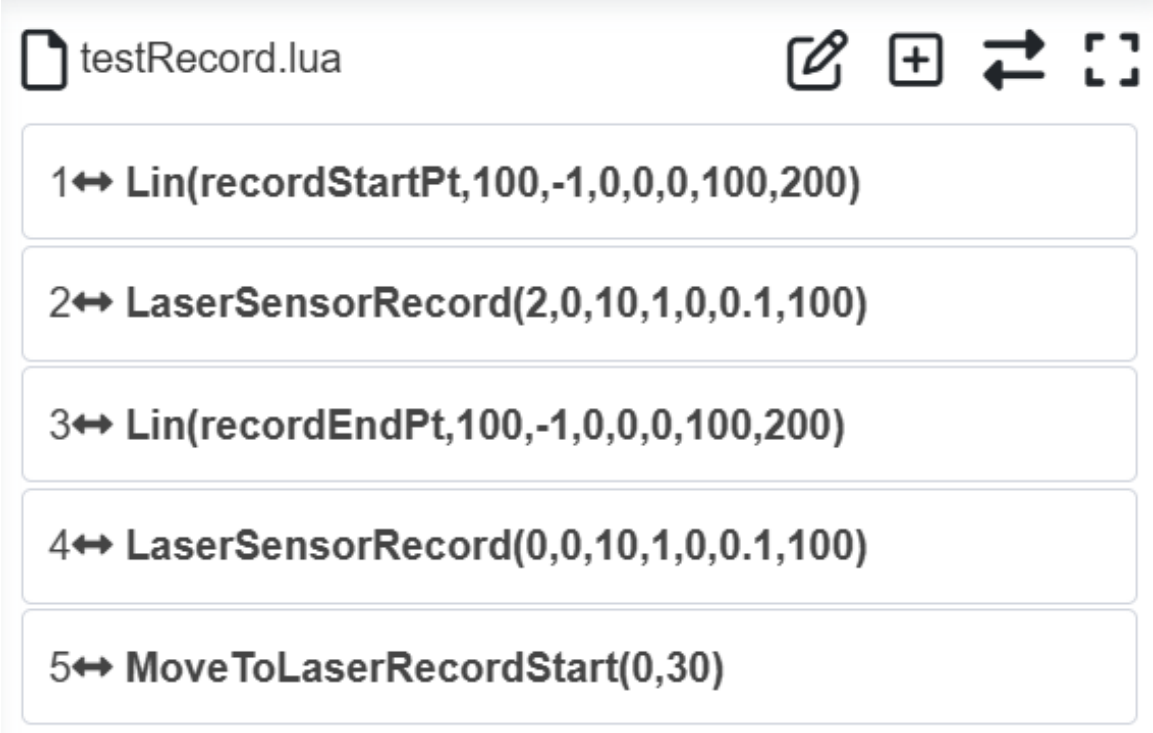
1.3.8.8.4.12 Writing Welding Related Instructions

Finally, add welding start and welding end instructions before the trajectory replay starts and after it ends:

Figure 8.8-44 Trajectory Recording and Replay Welding Program

Execute the above program. The robot will first carry the laser sensor to move along the weld seam trajectory and record the entire trajectory. Then the robot moves to the start point of the recorded trajectory. The robot starts the arc and begins welding along the trajectory recorded by the laser sensor. When the robot completes the trajectory replay, the welding arc is extinguished, completing the welding.


LT-Rec
Instruction Type
Move to weld point
Sports mode
PTP
Speed (%)
30
Starting point
Destination
Weld data record
Add
Program preview
MoveToLaserRecordStart(0,30);



1.3.8.8.5 Laser Sensor Adaptation Controller Peripheral Open Protocol

Step1: If you need to use "Open Protocol Connection" and "Control Laser Sensor", then in the sensor tracking configuration, select "Peripheral Open Protocol" for the "Protocol Type" option. If using the original scheme, select "Adapted Device", and configure and load the laser peripheral in the tracking sensor interface.

Figure 8.8-45 "Open Protocol Connection" and "Control Laser Sensor" Configuration Interface

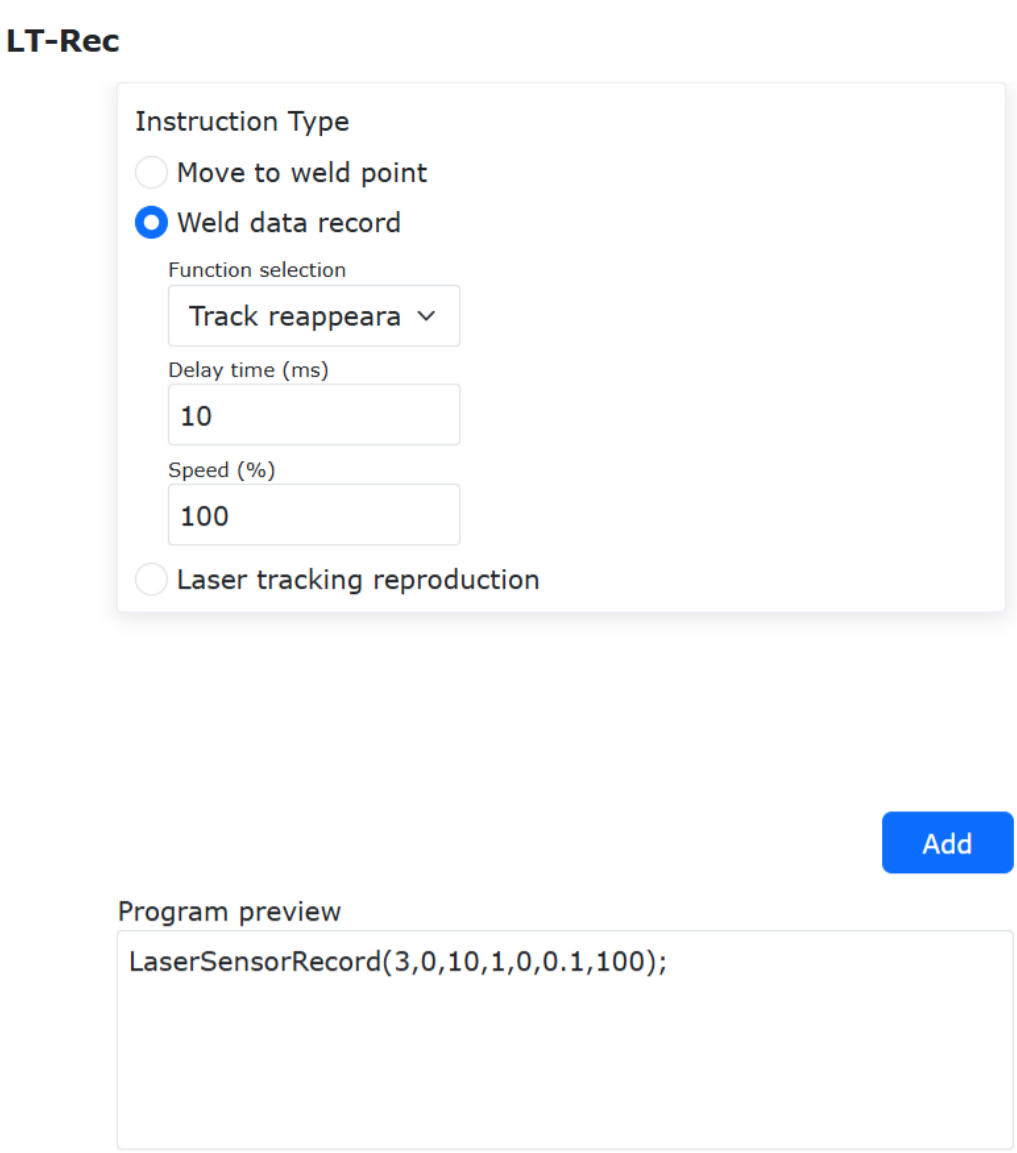
Step2: Click "Peripheral Open Protocol" to enter the interface. In the "Open Protocol Settings", upload the peripheral open protocol corresponding to the laser sensor. After successful upload, select the protocol number and the uploaded file name, click Configure. Then, in the Device Operation and Status section, run the uploaded laser sensor protocol to establish a connection with the corresponding laser sensor.

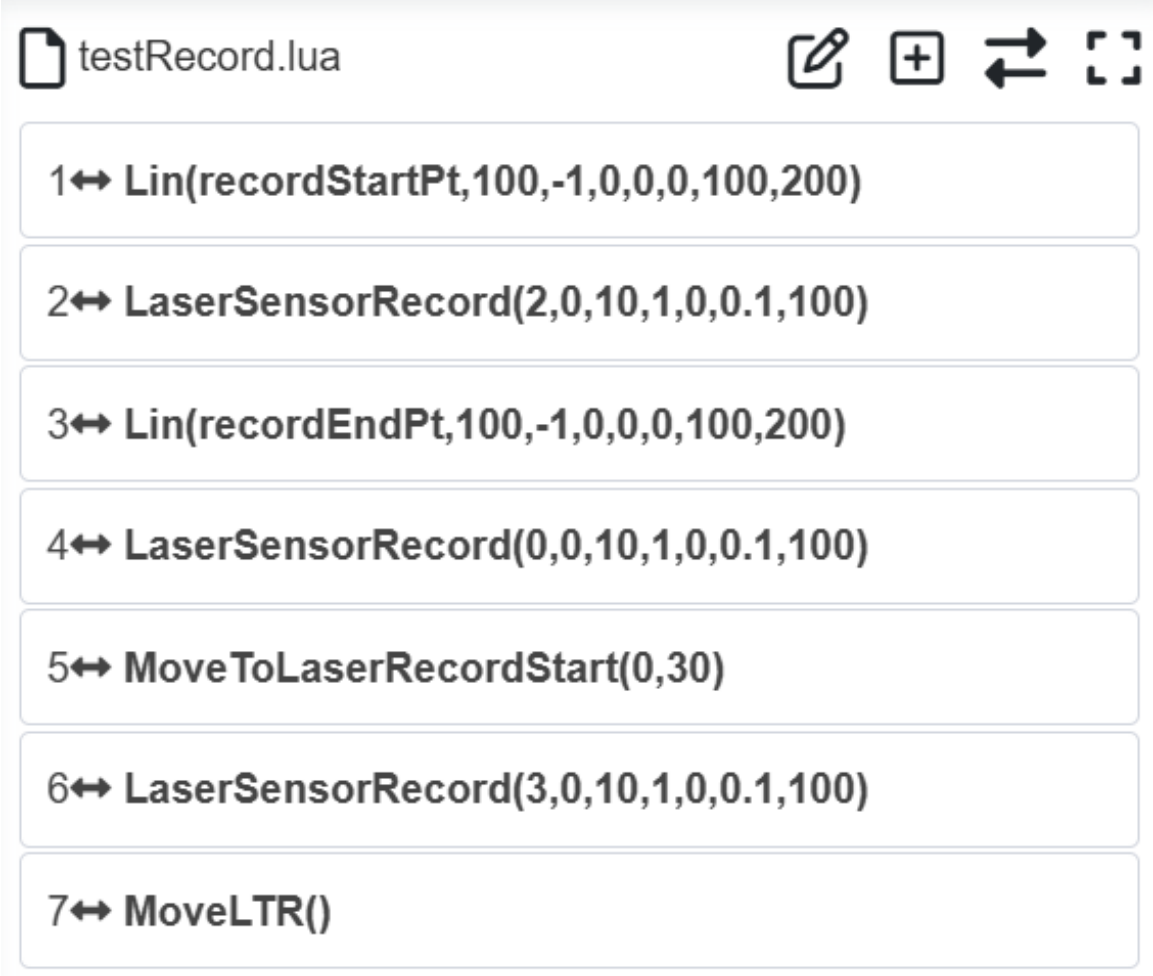
Figure 8.8-46 Laser Sensor Connection Establishment

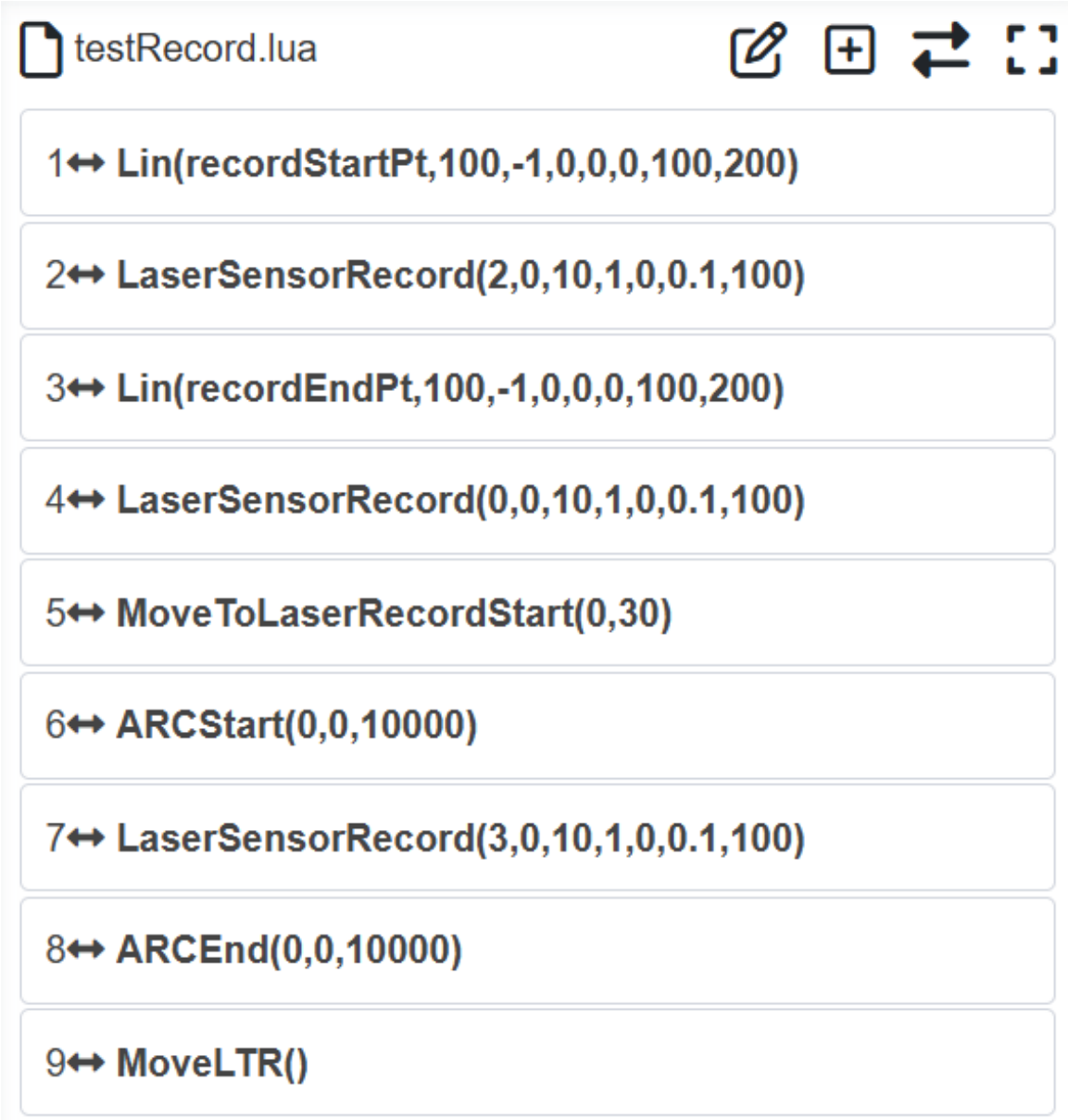
1.3.8.9 Polishing

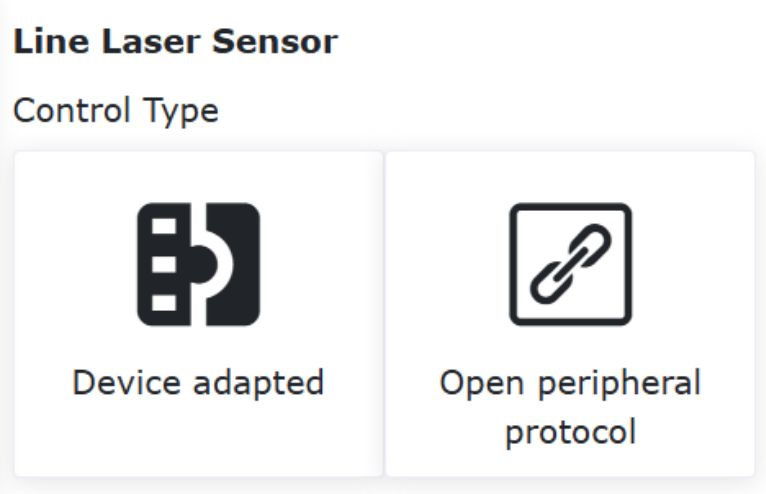
In the "Initial" -> "Peripherals" -> "Polishing" interface, polishing can currently be used via adapted devices or the peripheral open protocol.

Figure 8.9-1 Polishing Status Configuration Page


LT-Rec
Instruction Type
Move to weld point
Weld data record
Function selection
Track reappeara
Delay time (ms)
10
Speed (%)
100
Laser tracking reproduction
Add
Program preview
LaserSensorRecord(3,0,10,1,0,0.1,100);


testRecord.lua
1↔ Lin(recordStartPt,100,-1,0,0,0,100,200)
2↔ LaserSensorRecord(2,0,10,1,0,0.1,100)
3↔ Lin(recordEndPt,100,-1,0,0,0,100,200)
4↔ LaserSensorRecord(0,0,10,1,0,0.1,100)
5↔ MoveToLaserRecordStart(0,30)
6↔ LaserSensorRecord(3,0,10,1,0,0.1,100)
7↔ MoveLTR()


testRecord.lua
1↔ Lin(recordStartPt,100,-1,0,0,0,100,200)
2↔ LaserSensorRecord(2,0,10,1,0,0.1,100)
3↔ Lin(recordEndPt,100,-1,0,0,0,100,200)
4↔ LaserSensorRecord(0,0,10,1,0,0.1,100)
5↔ MoveToLaserRecordStart(0,30)
6↔ ARCStart(0,0,10000)
7↔ LaserSensorRecord(3,0,10,1,0,0.1,100)
8↔ ARCEnd(0,0,10000)
9↔ MoveLTR()



1.3.8.9.1 Adapted Devices

Communication Configuration and Loading: Configure communication information, including IP address, port, sampling period, and communication protocol. Establish communication with the polishing device using the Load/Unload buttons.

Figure 8.9-2 Communication Configuration and Loading

Device Functions: Operations such as device enabling, error clearing, and force sensor zeroing can be performed.

Figure 8.9-3 Device Functions

Parameter Configuration: Parameters such as the polishing device's rotation speed, contact force, extension distance, and control mode can be set. After successful setting, the corresponding data and status are displayed in the right "Polish" status feedback column.

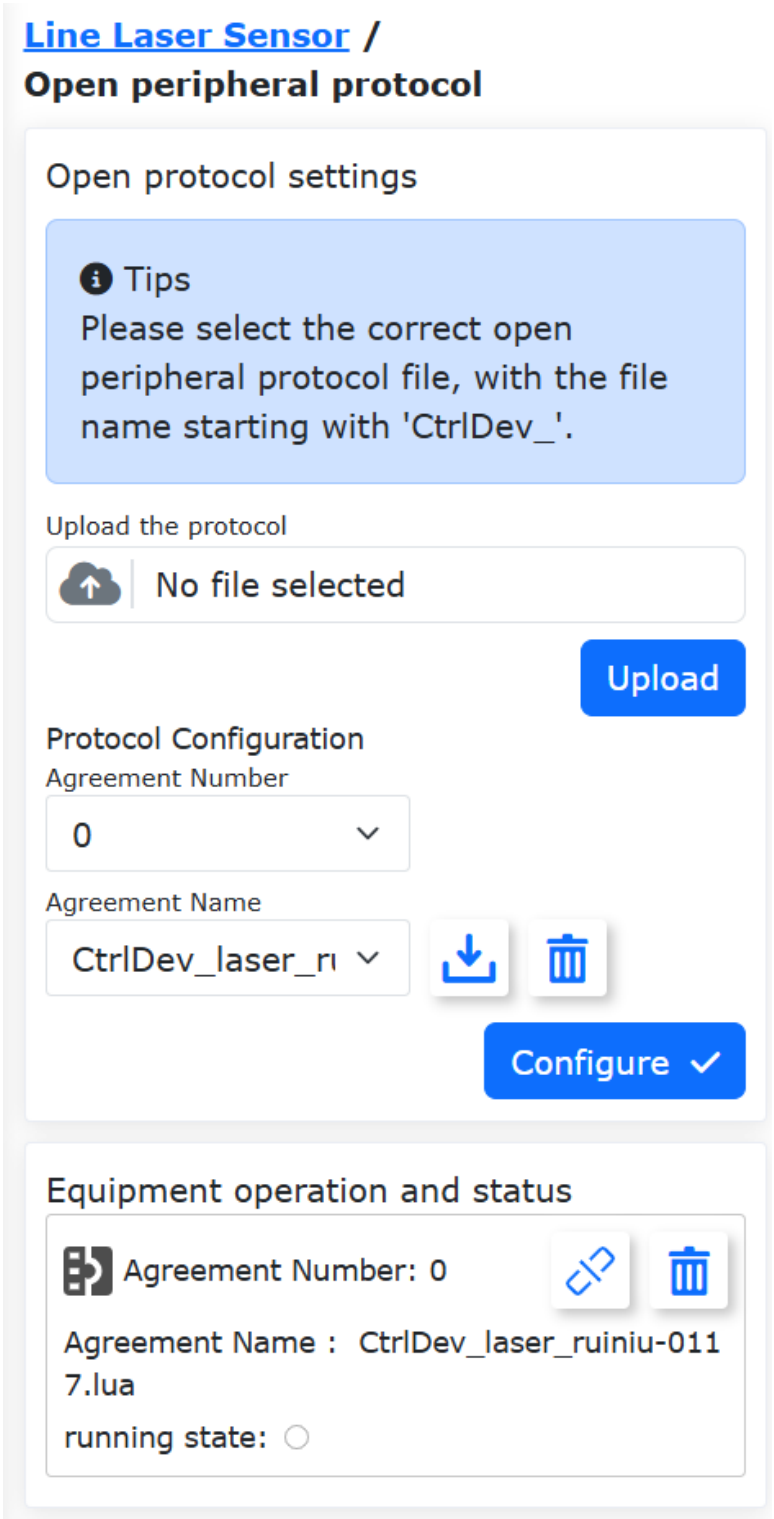
Figure 8.9-4 Parameter Configuration

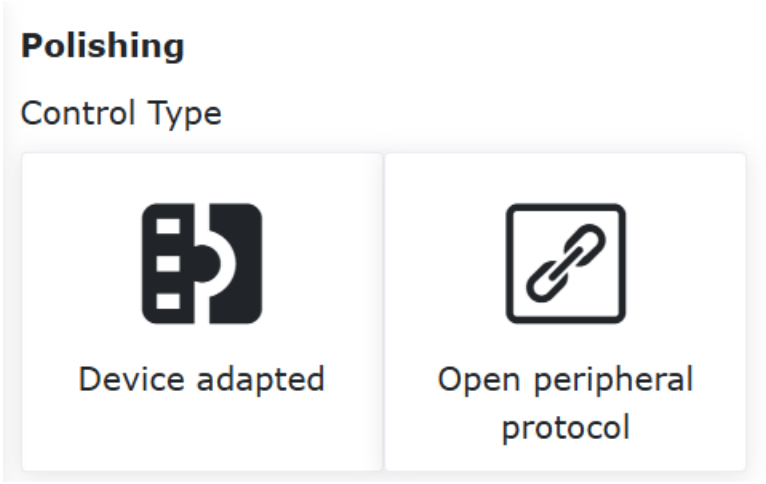
Figure 8.9-5 Parameter Configuration

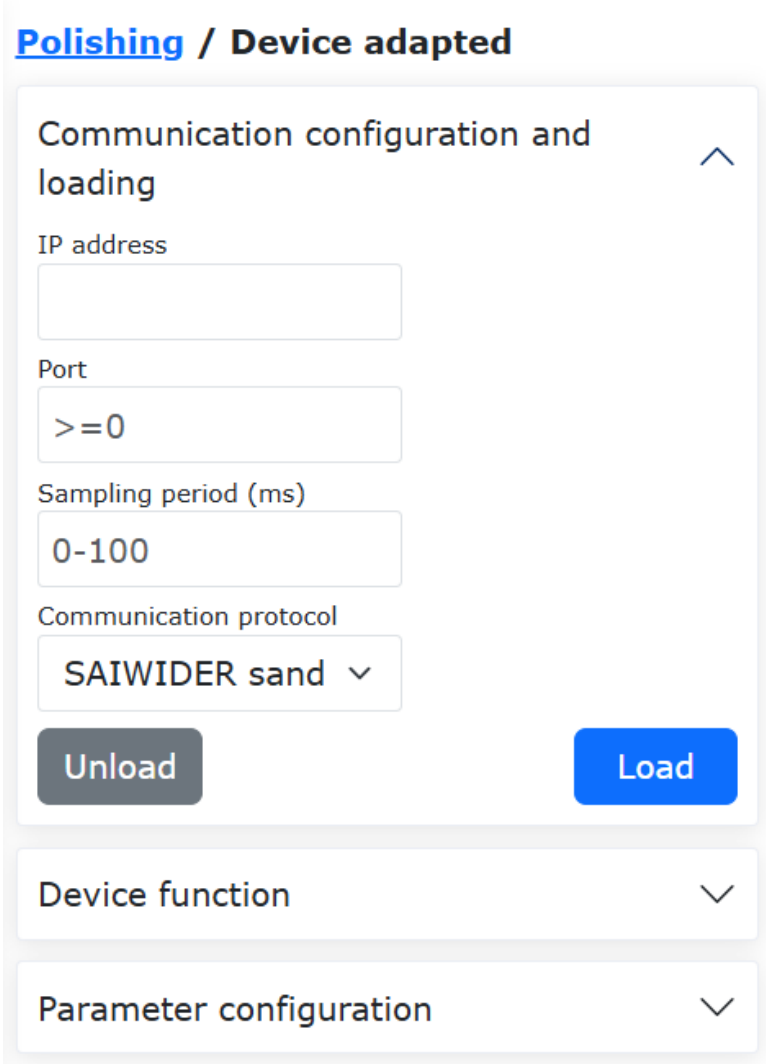
1.3.8.9.2 Peripheral Open Protocol

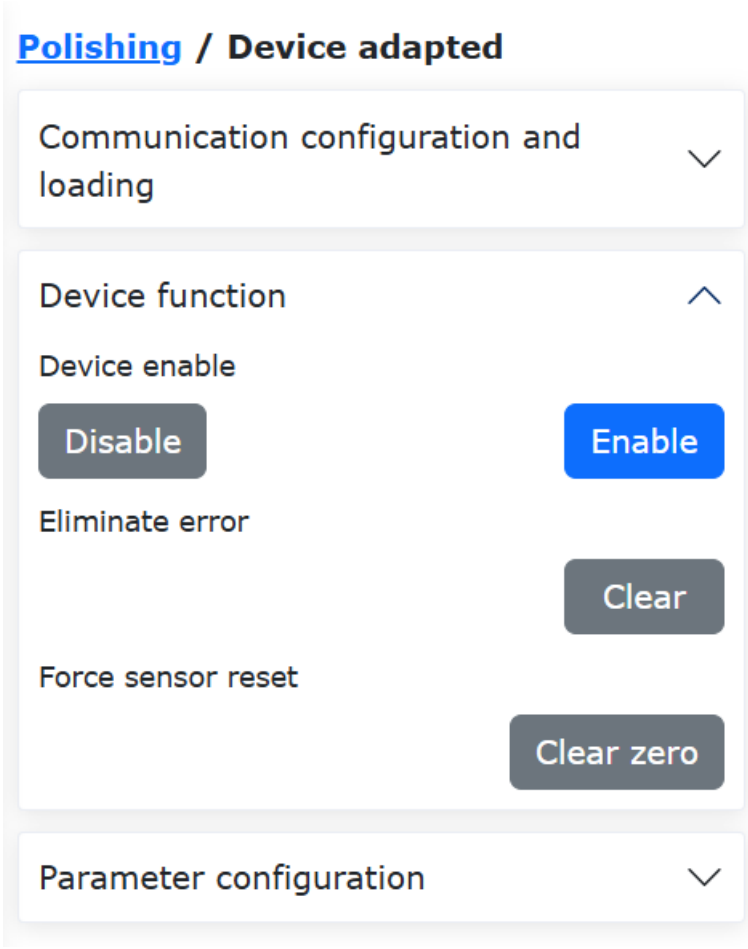
Click "Peripheral Open Protocol" to enter the interface. In the "Open Protocol Settings", upload the peripheral open protocol corresponding to the polishing device. After successful upload, select the protocol number and the uploaded file name, click Configure. Then, in the Device Operation and Status section, run the uploaded polishing peripheral open protocol to establish a connection with the corresponding polishing device.

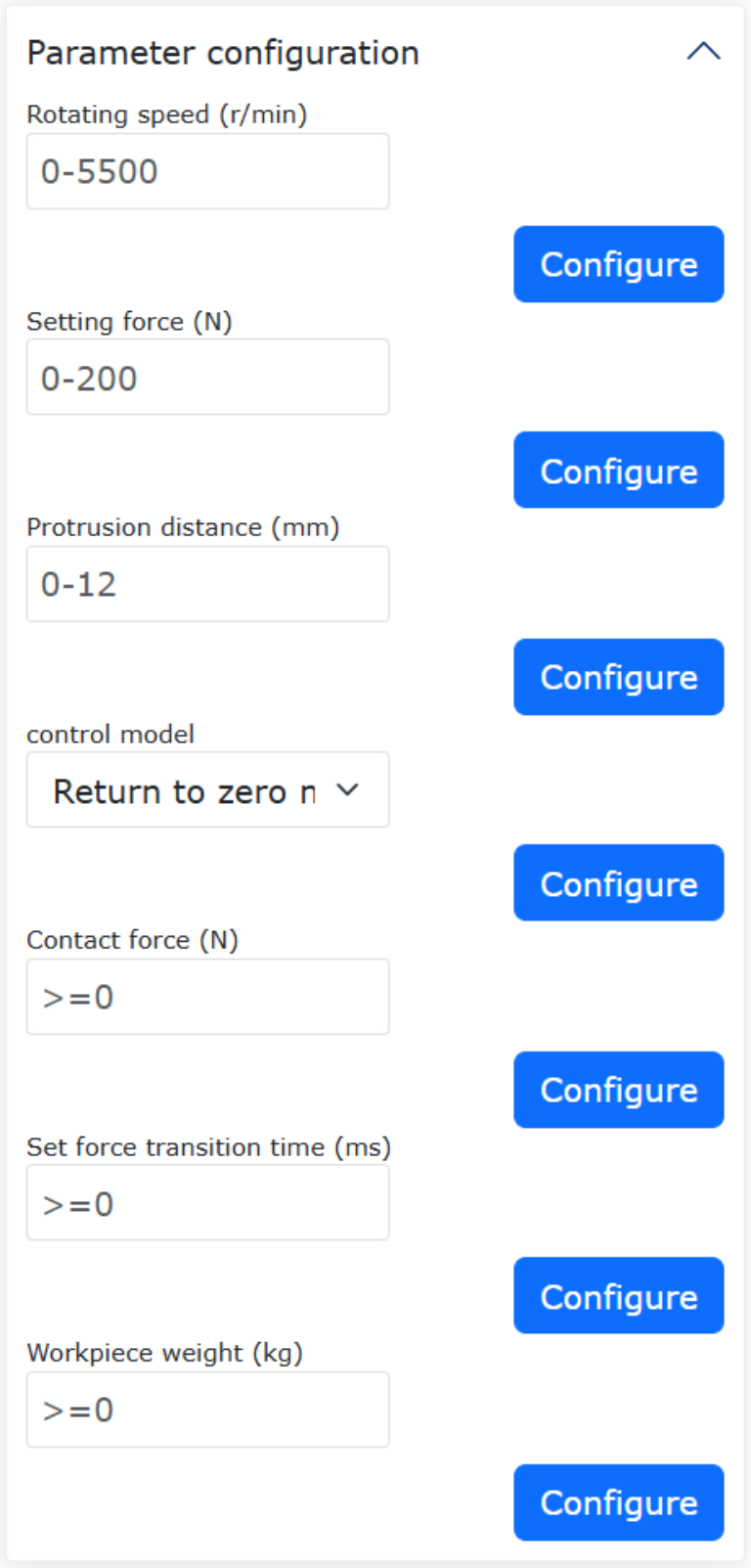
Figure 8.9-6 Polishing Device Connection establishment

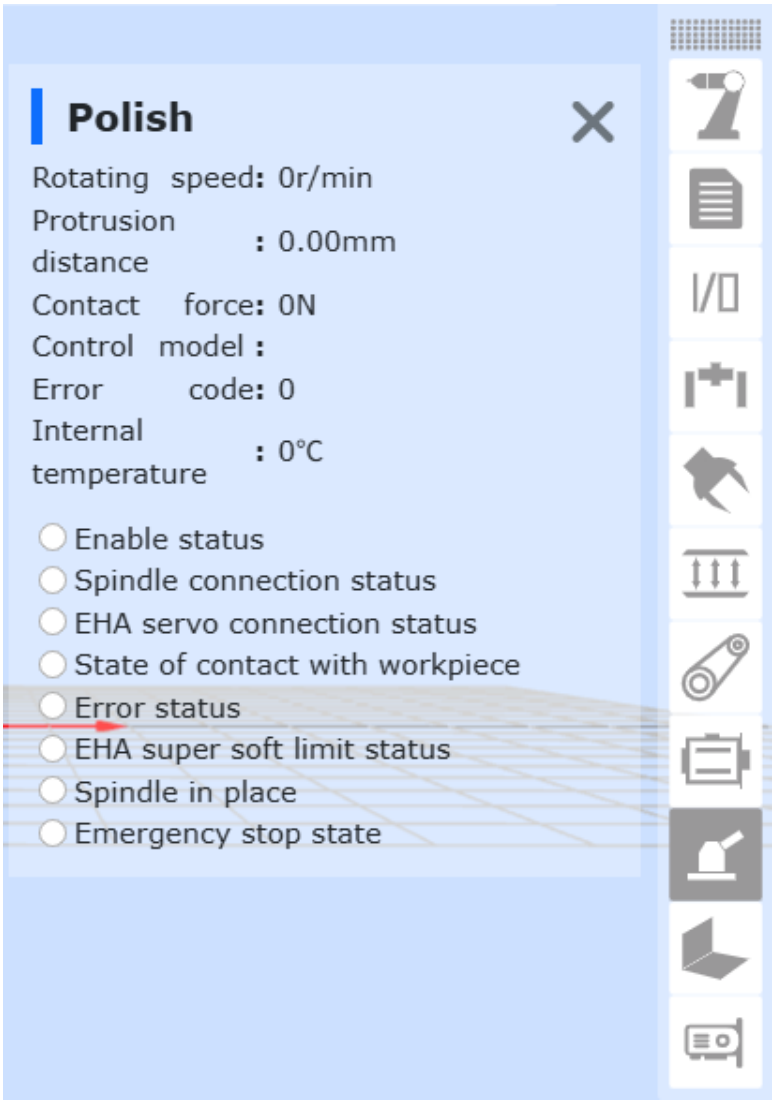

Line Laser Sensor /
Open peripheral protocol
Open protocol settings
Tips
Please select the correct open peripheral protocol file, with the file name starting with 'CtrlDev_'.
Upload the protocol
No file selected
Upload
Protocol Configuration
Agreement Number
0
Agreement Name
CtrlDev_laser_ri
Configure
Equipment operation and status
Agreement Number: 0
Agreement Name : CtrlDev_laser_ruiniu-0117.lua
running state:

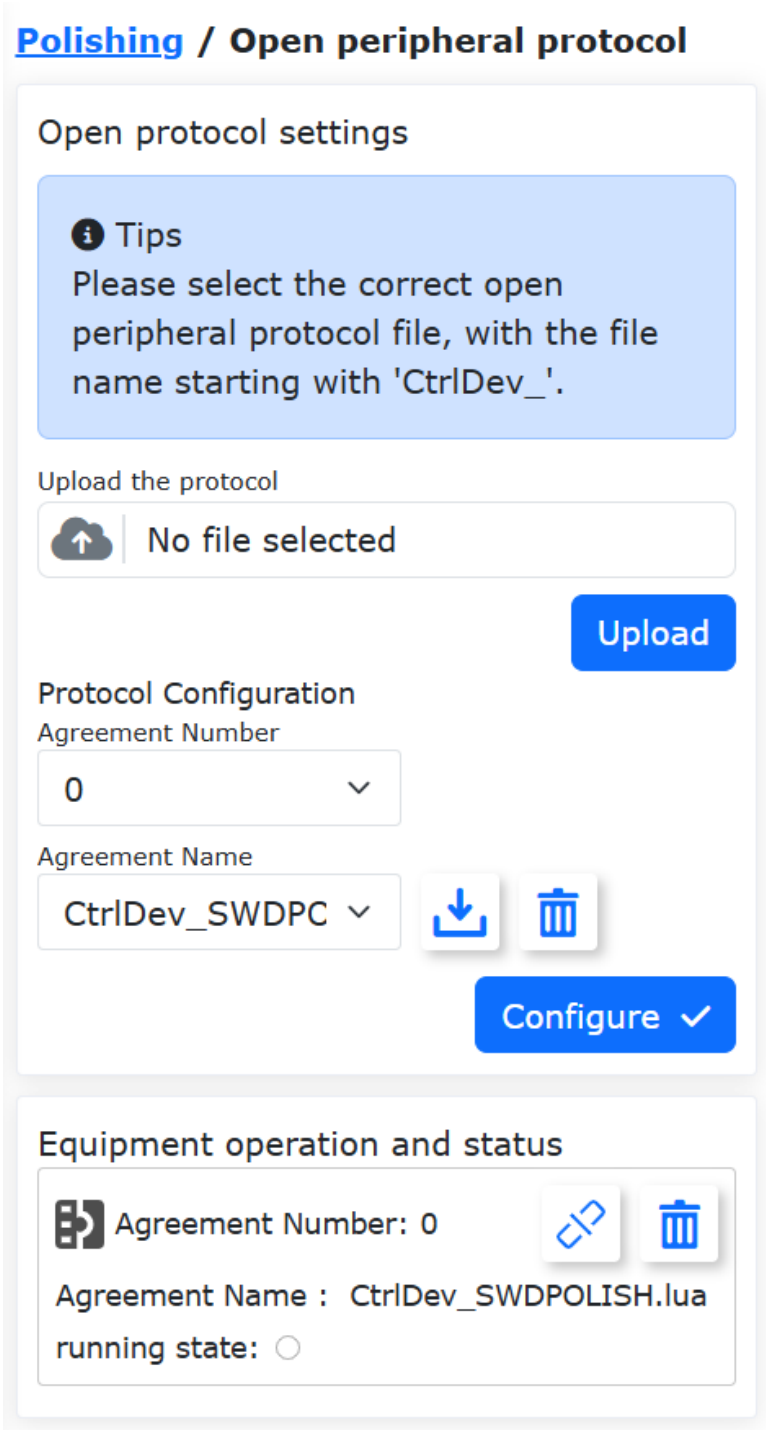

Polishing
Control Type
Device adapted
Open peripheral protocol


Polishing / Device adapted
Communication configuration and loading
IP address
Port
>=0
Sampling period (ms)
0-100
Communication protocol
SAIWIDER sand
Unload
Load
Device function
Parameter configuration


Polishing / Device adapted
Communication configuration and loading
Device function
Device enable
Disable
Enable
Eliminate error
Clear
Force sensor reset
Clear zero
Parameter configuration


Parameter configuration
Rotating speed (r/min)
0-5500
Configure
Setting force (N)
0-200
Configure
Protrusion distance (mm)
0-12
Configure
control model
Return to zero n
Configure
Contact force (N)
>=0
Configure
Set force transition time (ms)
>=0
Configure
Workpiece weight (kg)
>=0
Configure


Polish
Rotating speed: 0r/min
Protrusion distance : 0.00mm
Contact force: 0N
Control model :
Error code: 0
Internal temperature : 0°C
Enable status
Spindle connection status
EHA servo connection status
State of contact with workpiece
Error status
EHA super soft limit status
Spindle in place
Emergency stop state


Polishing / Open peripheral protocol
Open protocol settings
Tips
Please select the correct open peripheral protocol file, with the file name starting with 'CtrlDev_'.
Upload the protocol
No file selected
Upload
Protocol Configuration
Agreement Number
0
Agreement Name
CtrlDev_SWDPC
Configure
Equipment operation and status
Agreement Number: 0
Agreement Name : CtrlDev_SWDPOLISH.lua
running state:

1.3.8.10 Auxiliary Sensors

In the “Initial” -> “Peripherals” -> “Auxiliary Sensors” interface, it is currently available for use with adapted devices. The custom protocol function is not yet open.

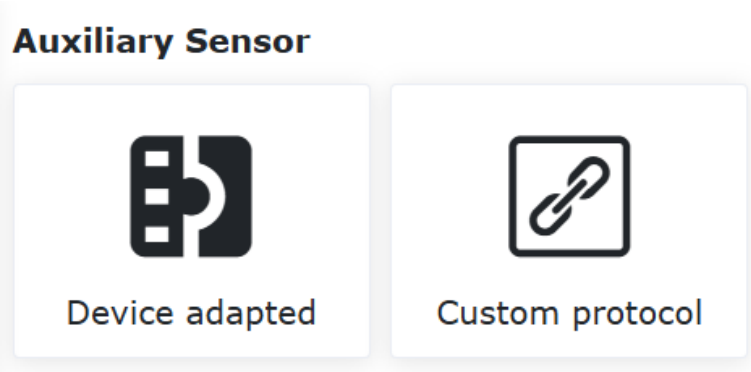


Figure 8.10-1 Auxiliary Sensors–Adapted Devices

1.3.8.10.1 Adapted Devices

Click “Adapted Devices” to enter the auxiliary sensor configuration interface.

The configuration information for auxiliary sensors includes manufacturer, type, software version, and mounting location. Users can configure the corresponding auxiliary sensor information according to specific production requirements.

If users need to change the configuration, they can first select the corresponding auxiliary sensor number, click the “Clear” button to clear the corresponding configuration, and then reconfigure according to requirements.

Figure 8.10-2 Auxiliary Sensors–Adapted Devices

1.3.8.11 Combined Devices (SmartTool + Force Sensor Combination)

In the “Initial” -> “Peripherals” -> “Combined Devices” interface, it is currently available for use with adapted devices. The custom protocol is not yet open.

Figure 8.11-1 Combined Devices

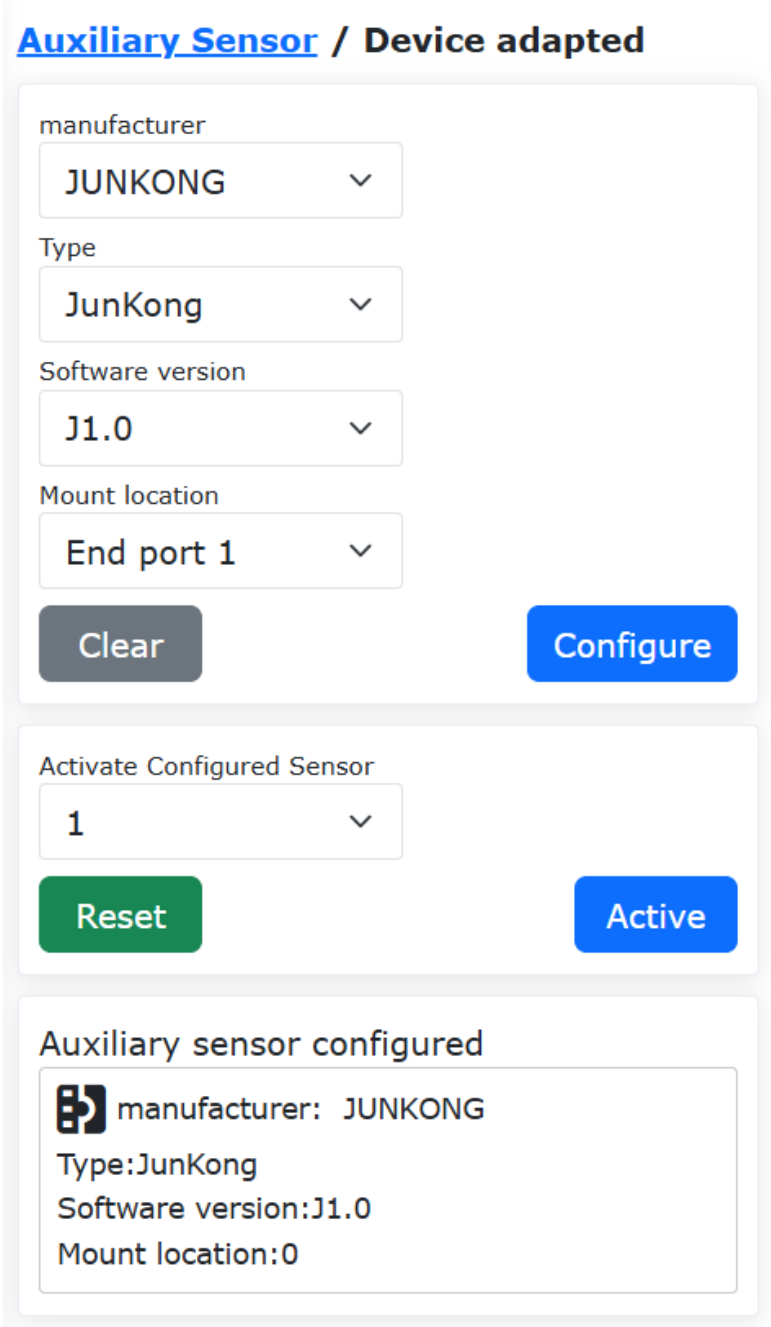
1.3.8.11.1 Adapted Devices

Click “Adapted Devices” to enter the configuration interface.

The configuration information is divided into manufacturer, type, software version, and mounting location. Different manufacturers correspond to different types. The current manufacturer is FR.

Users can configure the corresponding device information according to specific production requirements. After successful configuration, the device information table is displayed. If users need to change the configuration, they can first select the corresponding number, click the “Clear” button to clear the corresponding information, and then reconfigure the device information according to requirements.

Important: Before clicking to clear the configuration, the corresponding device should be in an inactive state.


Auxiliary Sensor / Device adapted
manufacturer
JUNKONG
Type
JunKong
Software version
J1.0
Mount location
End port 1
Clear
Configure
Activate Configured Sensor
1
Reset
Active
Auxiliary sensor configured
manufacturer: JUNKONG
Type:JunKong
Software version:J1.0
Mount location:0

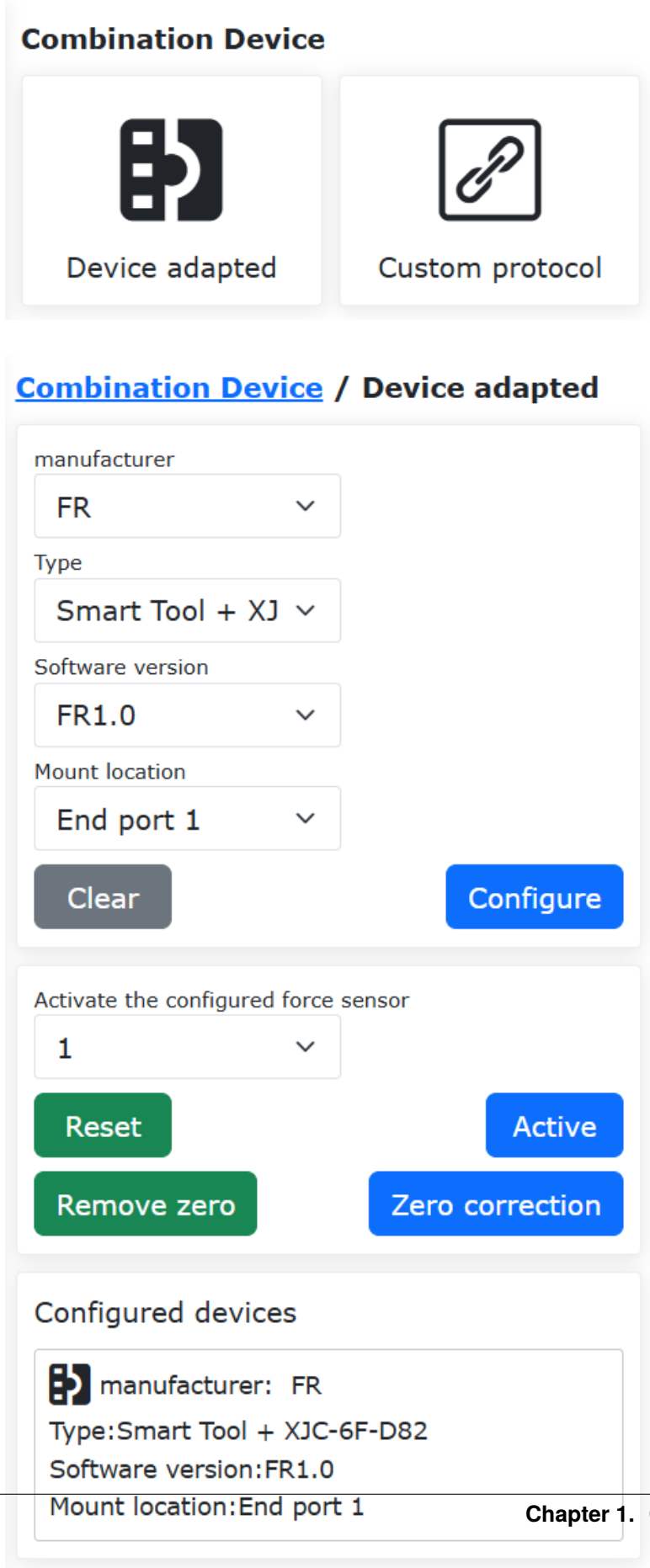

Combination Device
Device adapted
Custom protocol
Combination Device / Device adapted
manufacturer
FR
Type
Smart Tool + XJ
Software version
FR1.0
Mount location
End port 1
Clear
Configure
Activate the configured force sensor
1
Reset
Active
Remove zero
Zero correction
Configured devices
manufacturer: FR
Type:Smart Tool + XJC-6F-D82
Software version:FR1.0
Mount location:End port 1

Figure 8.11-2 Adapted Devices

1.3.8.11.1.1 FR

The type corresponding to FR is “SmartTool” used in combination with a force sensor. The collaborative robot can be adapted to three types of force sensors: XJC, NSR, and GZCX. When using different sensors, only the corresponding communication protocol needs to be loaded, as follows:

- SmartTool + XJC-6F-D82 (XJC).
- SmartTool + NSR-FT Sensor A (NSR).
- SmartTool + GZCX-6F-75A (GZCX).

1. Hardware Installation

1) Disassemble the SmartTool handle, take out the middle fixture, and install it on the robot end. After the fixture is installed, reassemble the SmartTool handle. After successful reassembly, connect the cable to the robot end.

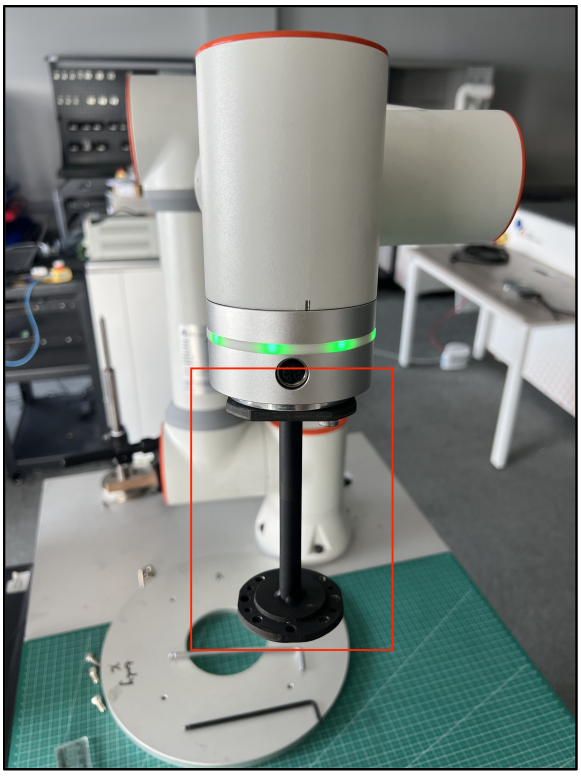


Figure 8.11-3 Installing the Middle Fixture of the SmartTool Handle



Figure 8.11-4 SmartTool Handle Installed Successfully

2) After the SmartTool handle is installed, mount the force sensor (using GZCX as an example) to the end of the SmartTool handle, and connect the cable to the SmartTool handle.



Figure 8.11-5 GZCX Force Sensor Installed at the End of the SmartTool Handle

2. Device Configuration

Important: Please ensure that your SmartTool handle is securely installed on the robot end and correctly connected to the robot end, and that the force sensor is securely installed on the end of the SmartTool handle and correctly connected to the SmartTool handle.

---

1) Configure the SmartTool handle (refer to Welding Handle Key Function Configuration).

2) After configuring the SmartTool handle key functions, set the manufacturer to “FR”, select the “Type”, “Software Version”, and “Mounting Location” information, and click the “Configure” button.

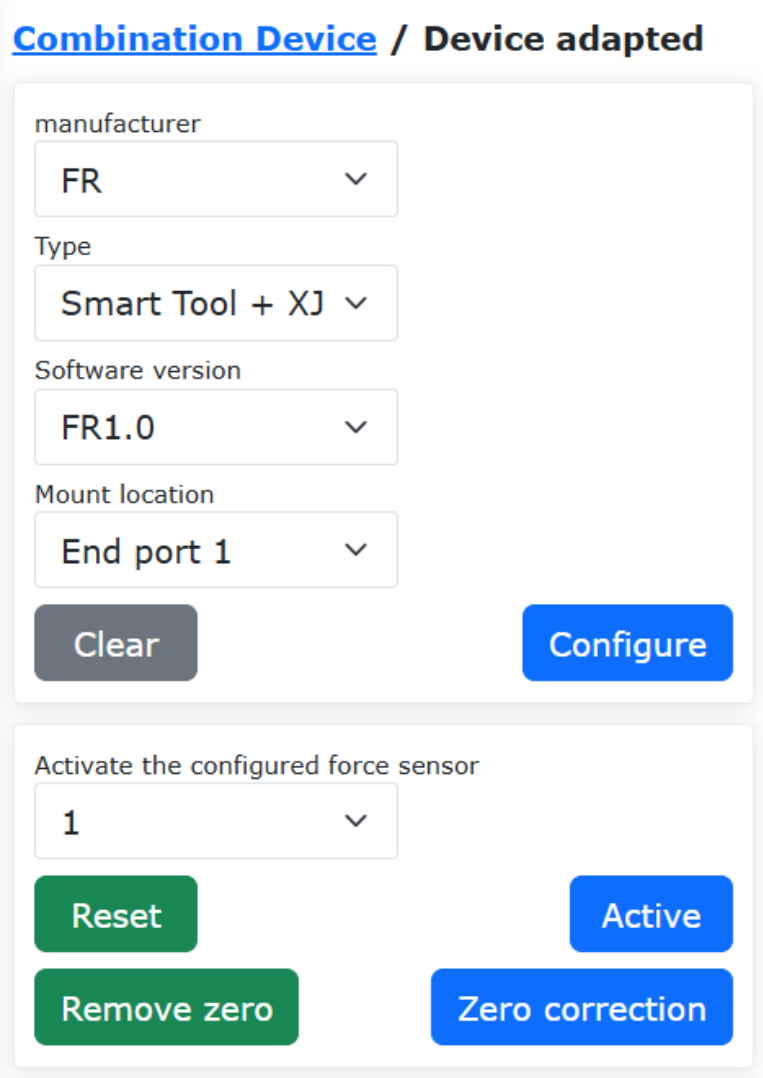


Figure 8.11-6 FR Device Information Configuration Interface

3) After successfully configuring the device information, select the configured force sensor, click the “Activate” button to activate the force sensor. After successful activation, click the “Zero Calibration” button to zero the force sensor, and view the table data.

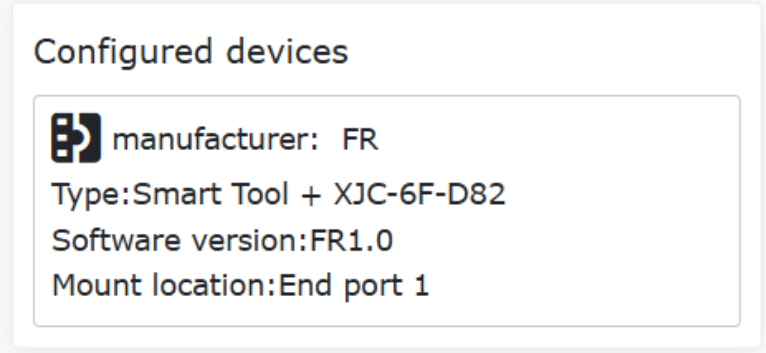


Figure 8.11-7 Force Sensor Zero Calibration

4) According to the current end installation, configure the load data in the “Load” interface, and configure the tool coordinate data, tool type, and installation position in the “Tool Coordinate” interface.

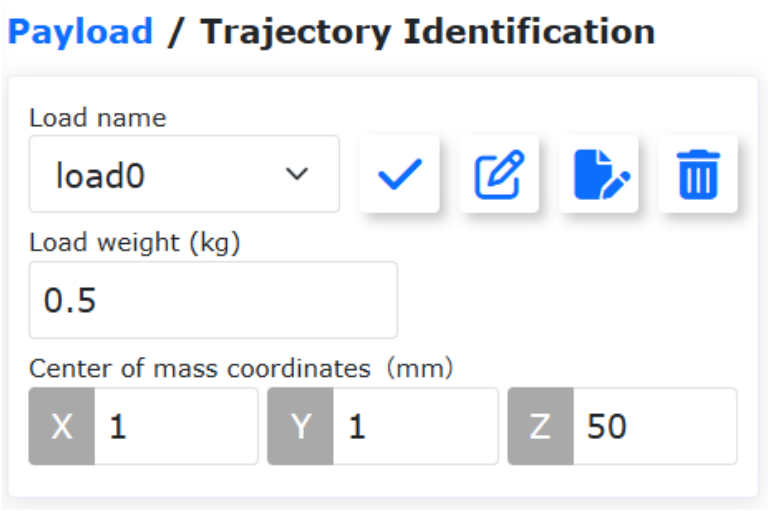


Figure 8.11-8 “End Load” Configuration

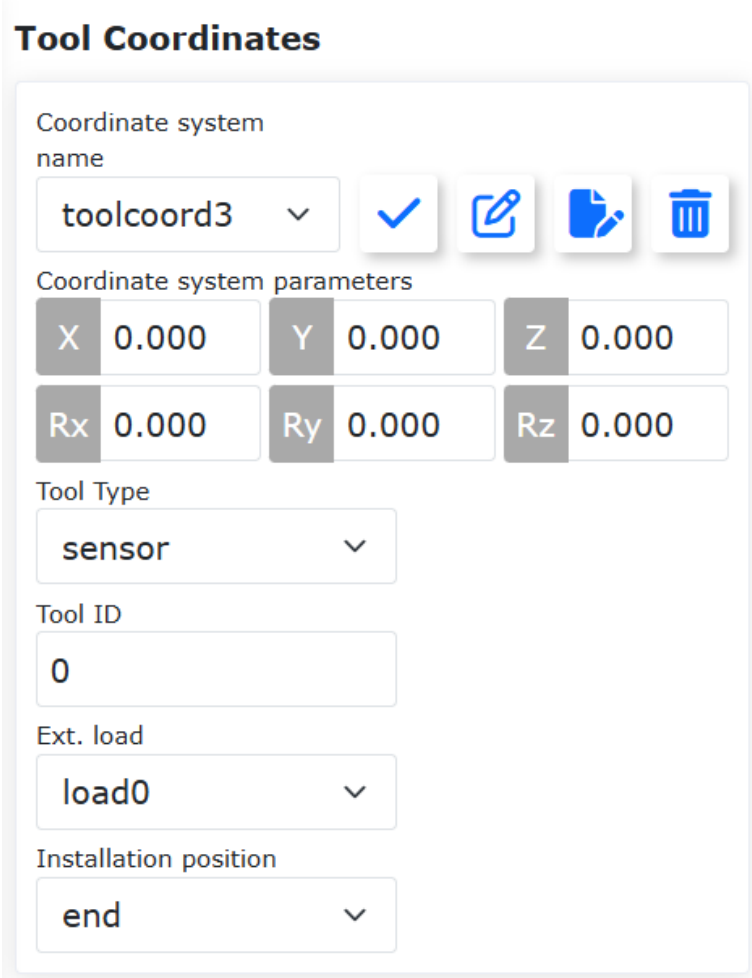


Figure 8.11-9 “Tool Coordinate” Configuration

3. Application

After the device information is successfully configured, it can independently implement the SmartTool button functions and force sensor functions, such as measuring the magnitude and direction of force, and auxiliary drag locking based on the force sensor.

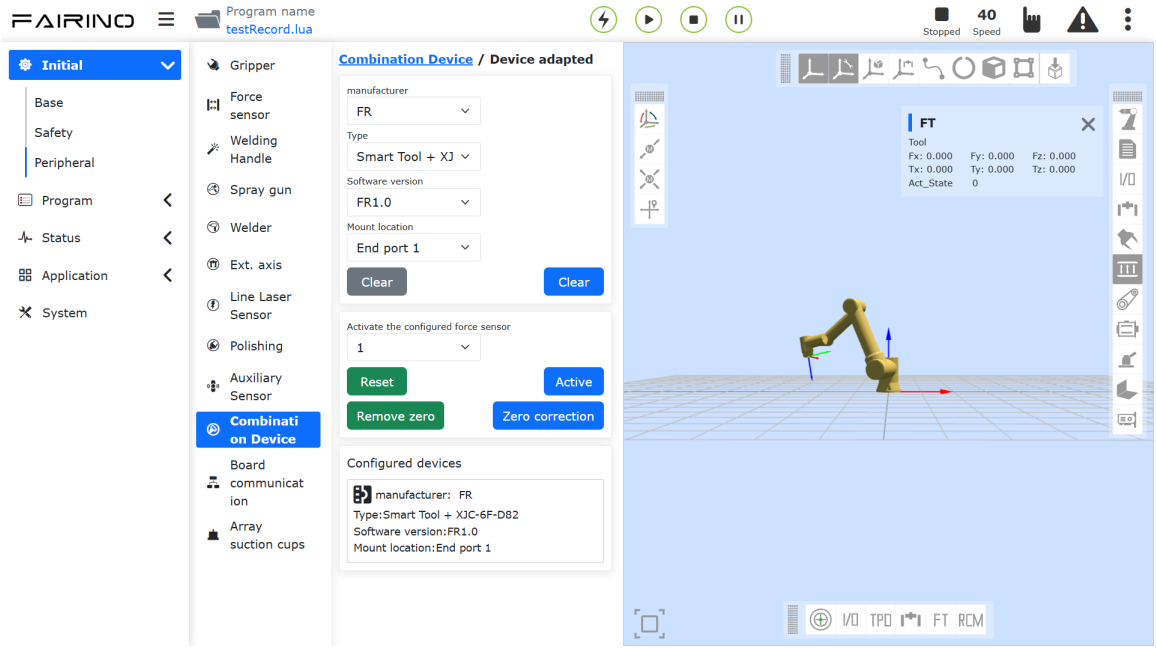


Figure 8.11-10 Measuring Force Magnitude and Direction

1.3.8.11.2 Combined Device End-Effector Lua Protocol

Currently, the end-effector can support the application of combined protocols for two devices. The second device can be connected via a one-to-two communication cable or through the F&S SmartTool 485 interface.

The operation steps are as follows:

Open the WebApp, click “Initial Setup”, “Peripherals” in sequence, and select one of the device types to be combined (e.g., welding handle). Choose “Custom Protocol”. Click “Protocol Management” to configure the end-effector protocol.

Currently, the preset built-in combined device protocols include: Junduo Gripper + Xinjingcheng Force Sensor, SmartTool + Junduo Gripper, SmartTool + Xinjingcheng Force Sensor. The preset protocols for combined devices belong to user-defined protocols, starting with “Custom_End”. They can be downloaded and deleted, as shown in the figure below.

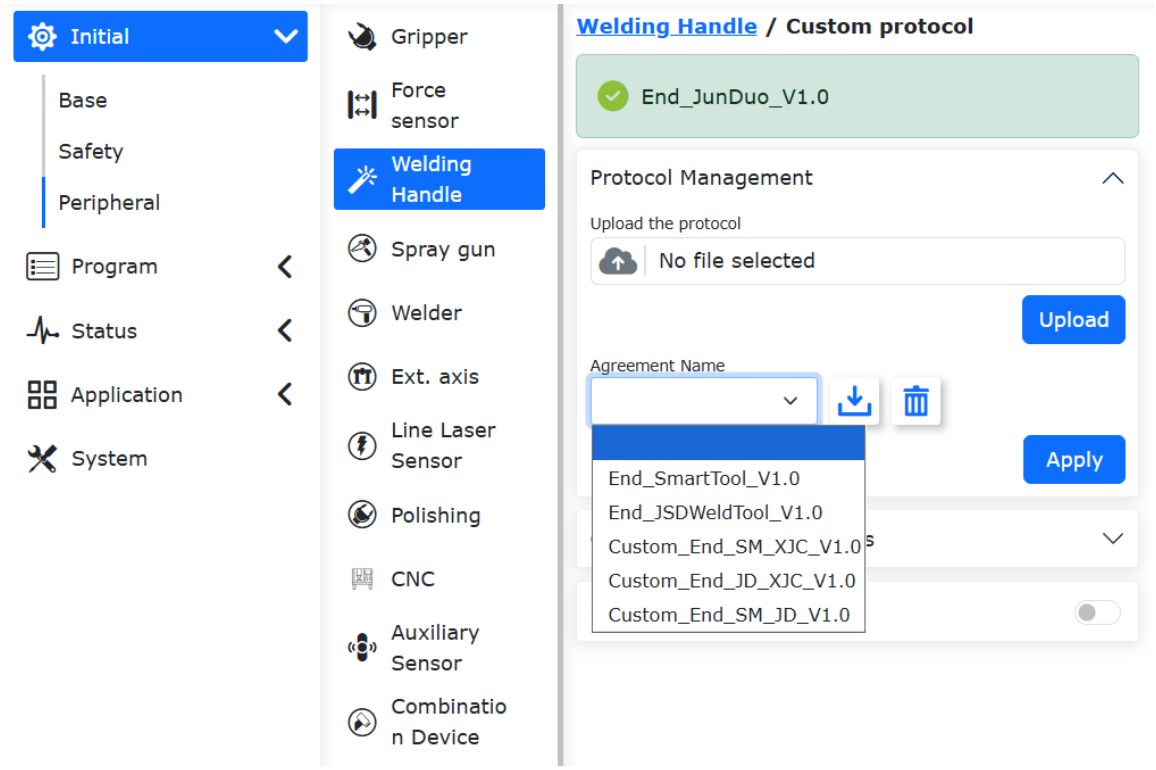


Figure 8.11-11 Welding Handle Preset Built-in Protocol

1.3.8.12 Array Suction Cups

1.3.8.12.1 Overview

Installing array suction cups on the robot end can help quickly deploy material grasping workstations for different scenarios. The number and layout of suction cups can be customized for materials of different sizes and shapes, improving work efficiency and stability.

The collaborative robot supports an array of up to 20 suction cups. Individual suction cups can be controlled for grasping and releasing, or the entire connected array can be controlled to act synchronously. Each suction cup supports configuration of station numbers from 1 to 20, configured based on DynamicLAB software.

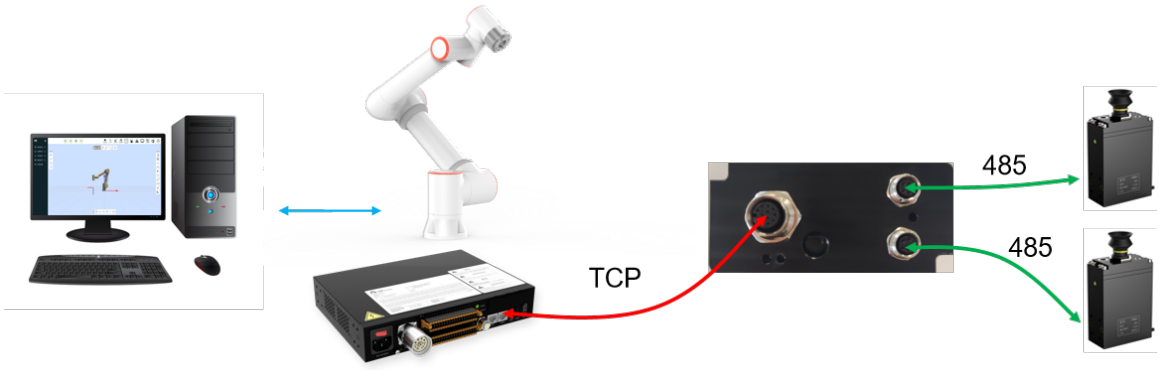
1.3.8.12.1.1 Hardware Description

The collaborative robot communicates with and controls the suction cup array via an Ethernet to 485 module. The WebApp generates the communication protocol for the array suction cups. The protocol sends control data to the Ethernet to 485 module via TCP/IP. The module then sends the received control data to each suction cup via 485, thus achieving control of the array suction cups (the control data format is ModbusRTU protocol format).

The Ethernet to 485 module acts as the server for Ethernet communication and the master for 485 communication. Each suction cup in the array is a 485 communication slave station, and each suction cup should be configured with a different slave station number.

Figure 8.12-1 Collaborative Robot Suction Cup Array Gripper Application

The Ethernet to 485 module usually has two TCP Server ports corresponding to multiple 485 slave ports. Taking CH9121 as an example, its TCP Server port 1 corresponds to 485 slave ports 1-10, and TCP Server port 2 corresponds



to 485 slave ports 11-20. The robot establishes two TCP communications with the Ethernet to 485 module, ultimately controlling 20 suction cups respectively.

The aforementioned Ethernet to 485 module needs to be configured as follows:

- Ethernet end configured as TCP Server, IP address: 192.168.58.10, Port 1 number: 50001, Port 2 number: 50002;
- 485 end configured with baud rate 115200, data bits 8, stop bits 1, no parity. The Ethernet to 485 module usually comes with a debugging software where the above configuration can be performed. The following figure shows the configuration tool page for the CH9121 model Ethernet to 485 module:

1.3.8.12.2 Function Configuration

Open the WebApp, click sequentially “Initial” -> “Peripherals” -> “Array Suction Cups”; The control modes for array suction cups are Unicast Mode and Broadcast Mode:

Unicast Mode: The communication protocol includes control content for each suction cup, allowing independent control of each suction cup in the array.

Broadcast Mode: Generates a communication protocol for all suction cups in the array, allowing synchronous control of grasping and releasing for all suction cups in the array, but cannot control a single suction cup individually.

Depending on the actual scenario, you can configure only Unicast Mode, or configure both modes simultaneously (allowing both individual control of specific suction cups and synchronous control of all suction cups).

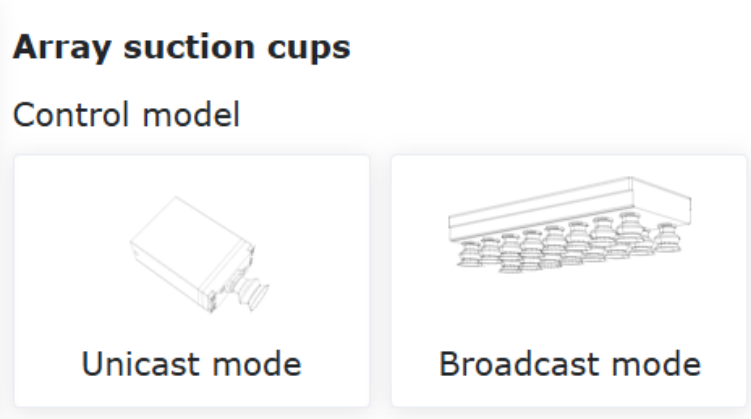


Figure 8.12-3 Array Suction Cup Control Mode

1.3.8.12.2.1 Unicast Mode Configuration

Open the WebApp, click sequentially "Initial" -> "Peripherals" -> "Array Suction Cups" -> "Unicast Mode". There are two methods for configuring the Unicast Mode protocol: "Auto Configuration" and "Manual Configuration":

Figure 8.12-4 Unicast Configuration Mode

Auto Configuration: Upload an existing protocol file directly to the robot controller. The existing protocol file may come from: Downloaded from another robot that has already configured and debugged the array suction cups; Written by technical personnel according to the actual scenario (users writing protocol files can achieve more flexible and efficient suction cup control). If multiple devices use the same array suction cups, using Auto Configuration to directly upload the protocol can improve deployment speed.

Manual Configuration: Configure the communication protocol for each suction cup according to the slave ID and vacuum level of the suction cups in the array. The manual configuration steps are as follows:

Select slave station number 1, input the maximum vacuum, minimum vacuum, grasp timeout time (timeout is not yet open), and click the "Configure" button. At this time, a suction cup protocol with protocol number 1 appears in the "Device Operation and Status" column, and the "Manual Configuration" and "Slave Station Number" labels will display all currently configured slave station numbers.

Figure 8.12-5 Configuring Unicast Suction Cup

Repeat the above steps to configure suction cups for multiple slave station numbers as needed. Each time a suction cup is configured, the robot system automatically updates the suction cup communication protocol content corresponding to "Protocol Number: 1", supporting up to 20 suction cups. After all suction cups are configured, click the "Connect" button in the "Protocol Number 1" box. The communication between the robot and the suction cups starts running, and the "Run Status" indicator lights up (Note: Please complete the configuration of all slave station number suction cups first, then click the "Connect" button. Configuring suction cup slaves after communication is established is invalid).

After the communication between the robot and the suction cups is successfully established, a list of operation boxes for all configured suction cup slave stations appears in the "Device Operation and Status" column; In the operation box page corresponding to each slave station number suction cup, suction cup control and status monitoring can be performed (including "Suction Status", "Current Vacuum", "Suction Cup Pressure", etc.). The suction cup slave station IDs configured in the figure below are 2 and 11 respectively.

Figure 8.12-6 Unicast Suction Cup Connection

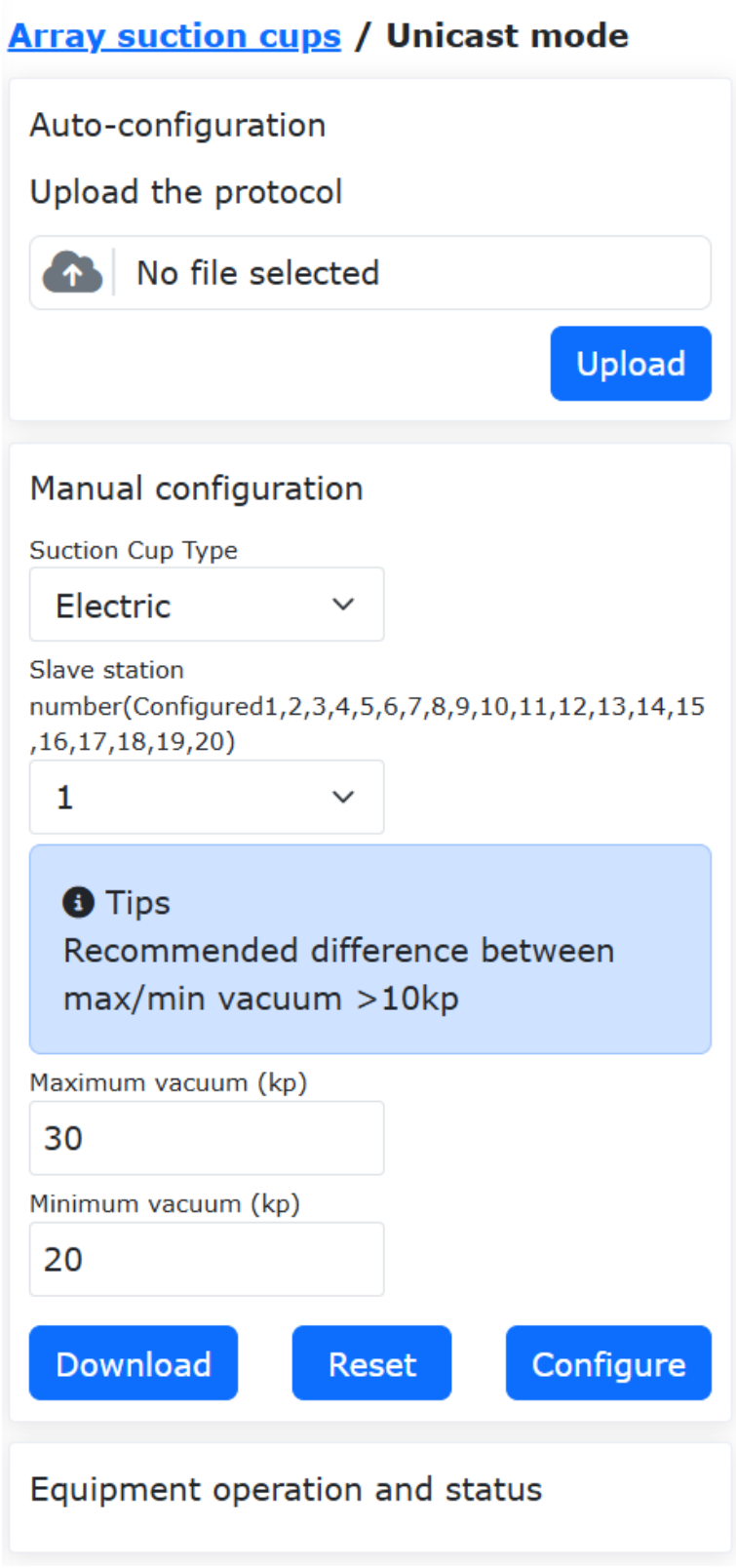
Click the "Suction" button in the upper right corner of the control box for slave station number 1 suction cup, and the suction cup will execute the "Set Vacuum Suction" action. At this time, the "Suction" button becomes the "Release" button. Click this button again, and the suction cup will execute the release action. When the suction cup performs the above actions, the corresponding status items such as "Suction Status" and "Current Vacuum" will display the suction cup's status in real time.

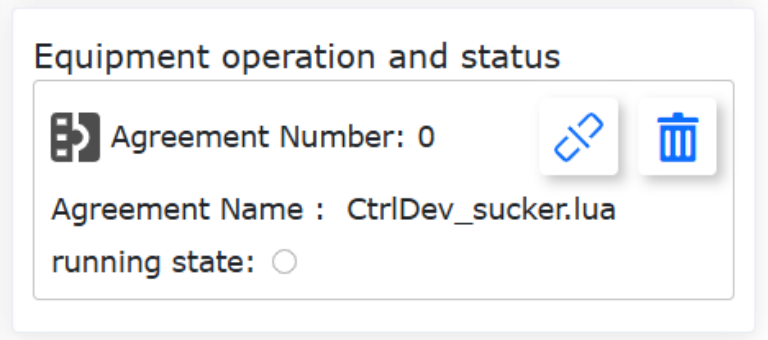
Note: Note: After configuring the suction cup protocol and completing the connection, you need to click the "Suction" button once to activate the suction cup. This can also test whether the communication between the robot and the suction cup is normal.

If the connection between the robot and the suction cup fails, the suction cup control box will not be displayed, and the run status indicator in "Protocol Number: 1" will be off.

Note: Note: If the physical communication connection between the suction cup and the Ethernet to 485 module is disconnected and then reconnected during use, it may cause the protocol to fail to establish a connection. In this case, you can unplug and replug the network cable of the Ethernet to 485 module and try to connect again.

Figure 8.12-7 Robot and Suction Cup Connection Failed


Array suction cups / Unicast mode
Auto-configuration
Upload the protocol
No file selected
Upload
Manual configuration
Suction Cup Type
Electric
Slave station number(Configured1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
1
Tips
Recommended difference between max/min vacuum >10kp
Maximum vacuum (kp)
30
Minimum vacuum (kp)
20
Download
Reset
Configure
Equipment operation and status



1.3.8.12.2.2 Unicast Mode Protocol Download

Click the "Download" button in "Manual Configuration" to download the suction cup protocol to the local computer. The suction cup protocol is a cyclically executed LUA program. The program executes the following steps in each cycle:

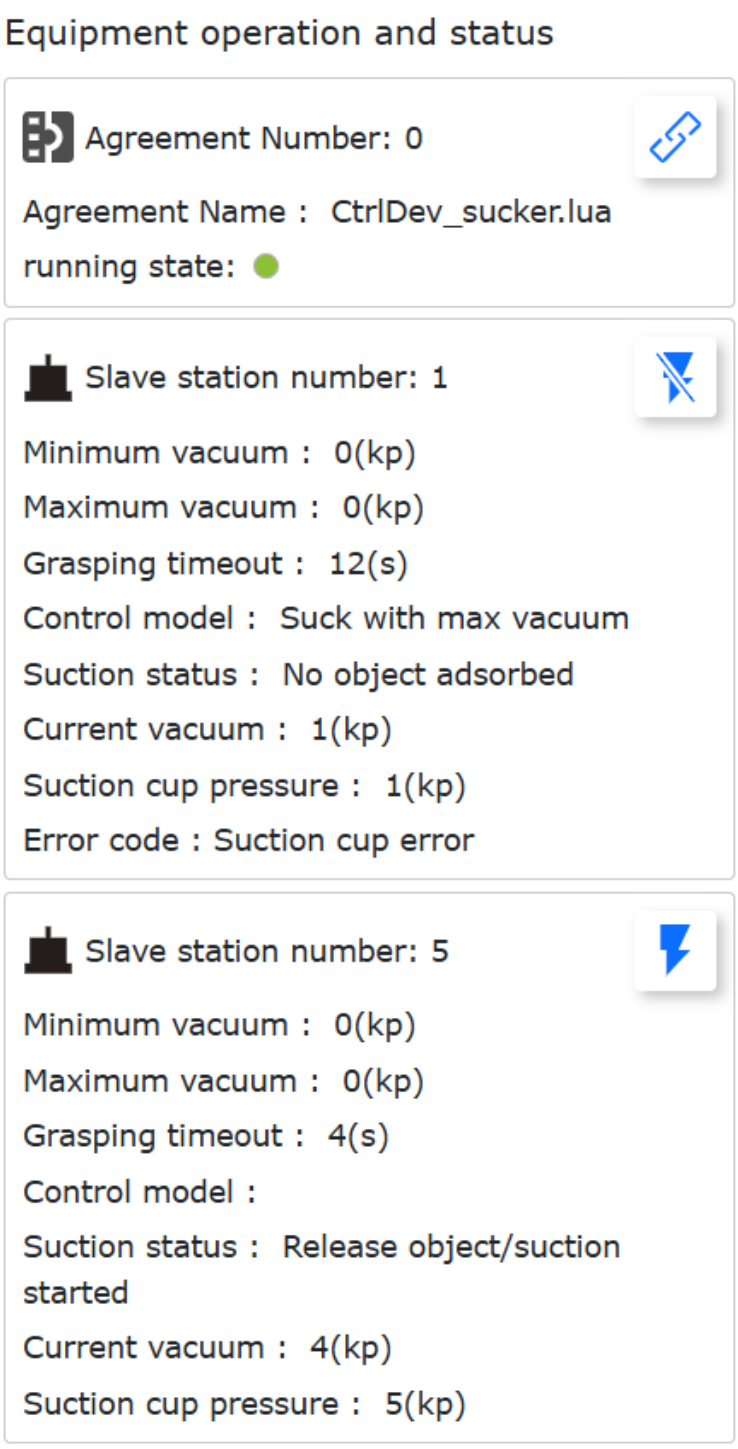
- Read suction cup control data from the robot;
- Write control data to the suction cup via socket;
- Read status data from the suction cup via socket;
- Feedback suction cup status data to the robot;

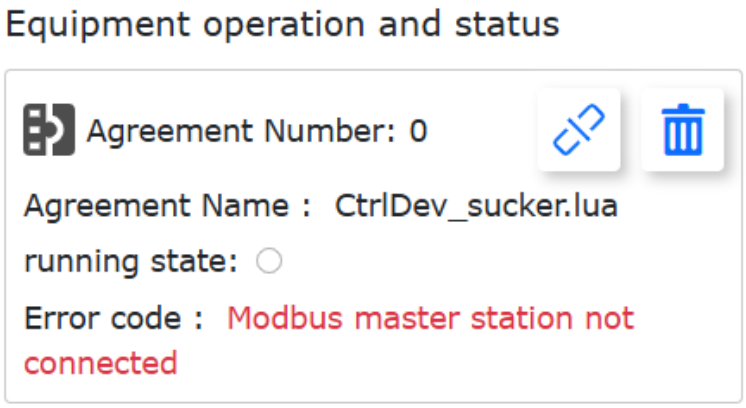
The suction cup communication protocol cyclically executes to achieve communication control between the robot and the suction cups. In the communication protocol, users can customize the cycle period, control data register address, and status data register address, and can modify the protocol content according to the actual situation. The following is a suction cup communication protocol code example:

Suction Cup Protocol Program Example:

```
local id = 1
local ctrlValues = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
local realTimeState = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
local suckerConfig = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
clearSuckerState()
socket1 = TCPClientConnect('192.168.58.10', 50001, 500, 10, 2, 3)
socket2 = TCPClientConnect('192.168.58.10', 50002, 500, 10, 2, 3)
suckerConfig[1] = 30
suckerConfig[2] = 20
suckerConfig[3] = 100
ModbusRTUOverTCPWriteMultiReg(socket1, 0, 0x0501, 3, suckerConfig)
ModbusRTUOverTCPWriteMultiReg(socket2, 0, 0x0501, 3, suckerConfig)
sleep_ms(10)
while(1) do
  setAllCtrl,ctrlValues[1],ctrlValues[2],ctrlValues[3],ctrlValues[4],ctrlValues[5],
↪ctrlValues[6],ctrlValues[7],ctrlValues[8],ctrlValues[9], ctrlValues[10],␣
↪ctrlValues[11], ctrlValues[12],ctrlValues[13],ctrlValues[14],ctrlValues[15],
↪ctrlValues[16],ctrlValues[17],ctrlValues[18],ctrlValues[19], ctrlValues[20] =␣
↪getSuckerCtrlState()
  if(setAllCtrl ~= 0) then
    ModbusRTUOverTCPWriteSingleReg(socket1, 0, 0x0500, setAllCtrl)
    ModbusRTUOverTCPWriteSingleReg(socket2, 0, 0x0500, setAllCtrl)
    ctrlValues = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

(continues on next page)


Equipment operation and status
Agreement Number: 0
Agreement Name : CtrlDev_sucker.lua
running state:
Slave station number: 1
Minimum vacuum : 0(kp)
Maximum vacuum : 0(kp)
Grasping timeout : 12(s)
Control model : Suck with max vacuum
Suction status : No object adsorbed
Current vacuum : 1(kp)
Suction cup pressure : 1(kp)
Error code : Suction cup error
Slave station number: 5
Minimum vacuum : 0(kp)
Maximum vacuum : 0(kp)
Grasping timeout : 4(s)
Control model :
Suction status : Release object/suction started
Current vacuum : 4(kp)
Suction cup pressure : 5(kp)



(continued from previous page)

```
    sleep_ms(1)
  else
    ModbusRTUOverTCPWriteSingleReg(socket1, 2, 0x0500, ctrlValues[2])
    ModbusRTUOverTCPWriteSingleReg(socket2, 11, 0x0500, ctrlValues[11])
  end
  suckerState, pressValue, error, default1, default2 = ModbusRTUOverTCPReadReg(socket1,␣
↪2, 0x0600, 3)
  realTimeState[1] = suckerState
  realTimeState[2] = pressValue
  realTimeState[3] = error
  ctrlState, maxPress, minPress, time, default2 = ModbusRTUOverTCPReadReg(socket1, 2,␣
↪0x0500, 4)
  realTimeState[4] = ctrlState
  realTimeState[5] = maxPress
  realTimeState[6] = minPress
  realTimeState[7] = time
  setSuckerRealtimeState(2, realTimeState)
  suckerState, pressValue, error, default1, default2 = ModbusRTUOverTCPReadReg(socket2,␣
↪11, 0x0600, 3)
  realTimeState[1] = suckerState
  realTimeState[2] = pressValue
  realTimeState[3] = error
  ctrlState, maxPress, minPress, time, default2 = ModbusRTUOverTCPReadReg(socket2, 11,␣
↪0x0500, 4)
  realTimeState[4] = ctrlState
  realTimeState[5] = maxPress
  realTimeState[6] = minPress
  realTimeState[7] = time
  setSuckerRealtimeState(11, realTimeState)
  local stopFlag = GetOpenLUAStopFlag(id)
  if(stopFlag ~= 0) then
    TCPClientDisconnect(socket1)
    TCPClientDisconnect(socket2)
    clearSuckerState()
    break
  end
  sleep_ms(100)
```

(continues on next page)

(continued from previous page)

```
end
```

The above protocol uses the *getSuckerCtrlState()* instruction to obtain suction cup control data, uses the *ModbusRTUOverTCPWriteSingleReg()* instruction to write control data to the suction cups via communication, uses the *ModbusRTUOverTCPReadReg()* instruction to read the status data of the suction cups, and then uses the *setSuckerRealtimeState()* instruction to feedback the suction cup status data to the robot. The detailed definitions of these instructions are as follows:

Table 8.12-1 getSuckerCtrlState() Return Values

No.	Type	Variable Name	Description
1	int	setAllCtrl	Broadcast mode control data: 1-Suction at max vacuum; 2-Suction at set vacuum, i.e., suction cup vacuum maintained between max and min vacuum; 3-Stop suction
2 ~ 21	int	ctrlValues[i]	Suction cup control data corresponding to slave station numbers 1 ~ 20: 1-Suction at max vacuum; 2-Suction at set vacuum, i.e., suction cup vacuum maintained between max and min vacuum; 3-Stop suction

Table 8.12-2 ModbusRTUOverTCPWriteSingleReg() Detailed Parameters

No.	Type	Variable Name	Description
1	int	socket	Socket handle
2	int	slaveID	Slave station number 0-20; 0-Broadcast; 1~20-Slave station number
3	uint16_t	regAddr	Write register address
4	uint16_t	data	Data to write

Table 8.12-3 ModbusRTUOverTCPWriteMultiReg() Detailed Parameters

No.	Type	Variable Name	Description
1	int	socket	Socket handle
2	int	slaveID	Slave station number 0-20; 0-Broadcast; 1~20-Slave station number
3	uint16_t	regStartAddr	Start address for writing multiple registers
4	int	num	Number of registers to write
5	uint16_t[]	data	Array of data content to write

Table 8.12-4 ModbusRTUOverTCPReadReg() Detailed Parameters

No.	Type	Variable Name	Description
1	int	socket	Socket handle
2	int	slaveID	Slave station number 0-20; 0-Broadcast; 1~20-Slave station number
3	uint16_t	regStartAddr	Start address for reading multiple registers
4	int	num	Number of registers to read

Table 8.12-5 ModbusRTUOverTCPReadReg() Return Values

No.	Type	Variable Name	Description
1	int	suckState	Suction cup current status: 0-Object released or suction cup started successfully; 1-Workpiece detected, object adsorbed; 2-No object adsorbed; 3-Object detached
2	float	pressValue	Current vacuum/pressure
3	int	err	Error code: 0-Normal; Others: Abnormal

Table 8.12-6 setSuckerRealtimeState() Detailed Parameters

No.	Type	Variable Name	Description
1	int	slaveID	Slave ID
2	int[]	states	**states[1]: Current status 0-Object released or suction cup started successfully; 1-Workpiece detected, object adsorbed; 2-No object adsorbed; 3-Object detached.** states[2]: Current vacuum/pressure; states[3]: Wait register value; states[4]: Control status; states[5]: Maximum vacuum; states[6]: Minimum vacuum; state[7]: Timeout time; states[8~10]: Reserved.

1.3.8.12.2.3 Broadcast Mode

The collaborative robot can control all connected suction cups to act simultaneously through Broadcast Mode.

Note: Note: Unicast Mode must be configured first before configuring Broadcast Mode.

Open the WebApp, click sequentially “Initial” -> “Peripherals” -> “Array Suction Cups”, first complete the configuration of all required suction cup slave stations in Unicast Mode (configure only, do not establish communication protocol connection).

Click “Broadcast Mode”, input the “Maximum Vacuum”, “Minimum Vacuum”, “Grasp Timeout Time” (timeout is not yet open) for the suction cups in the “Parameter Configuration”, and click the “Configure” button. At this time, the broadcast mode communication protocol appears in the “Device Operation and Status” box. In broadcast mode, setting the vacuum parameters takes effect for every connected suction cup.

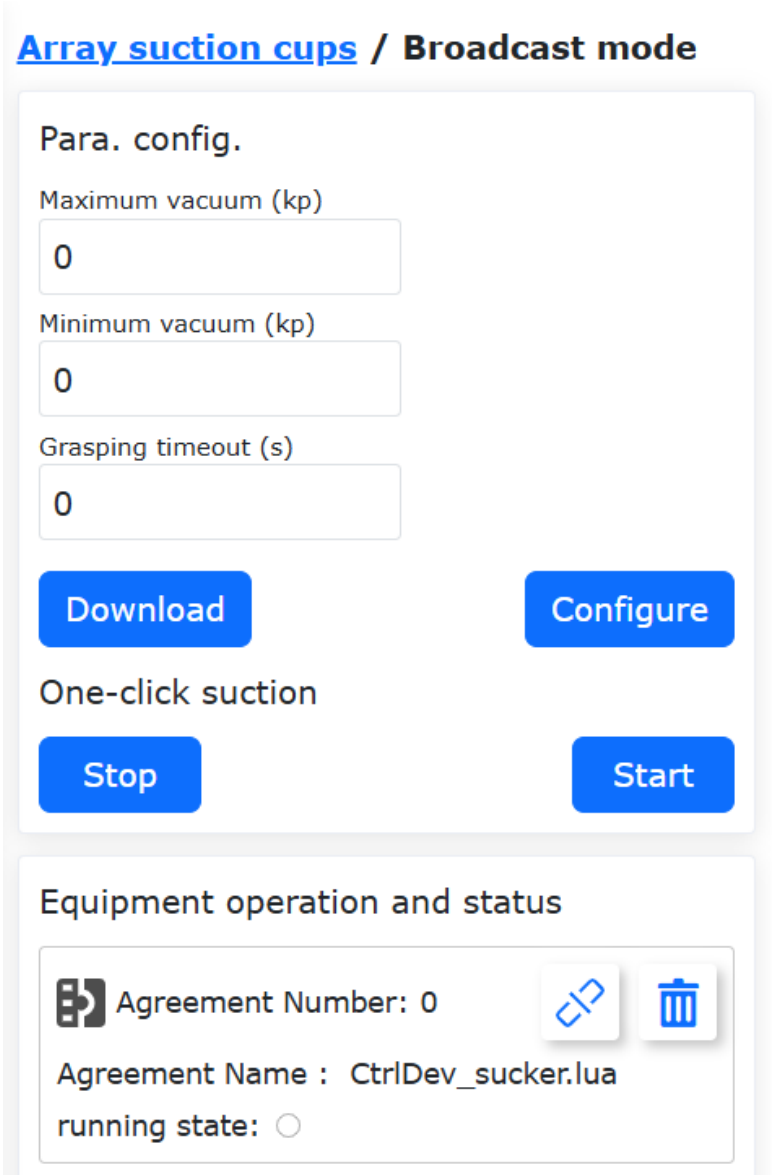
Figure 8.12-8 Broadcast Mode Parameter Configuration

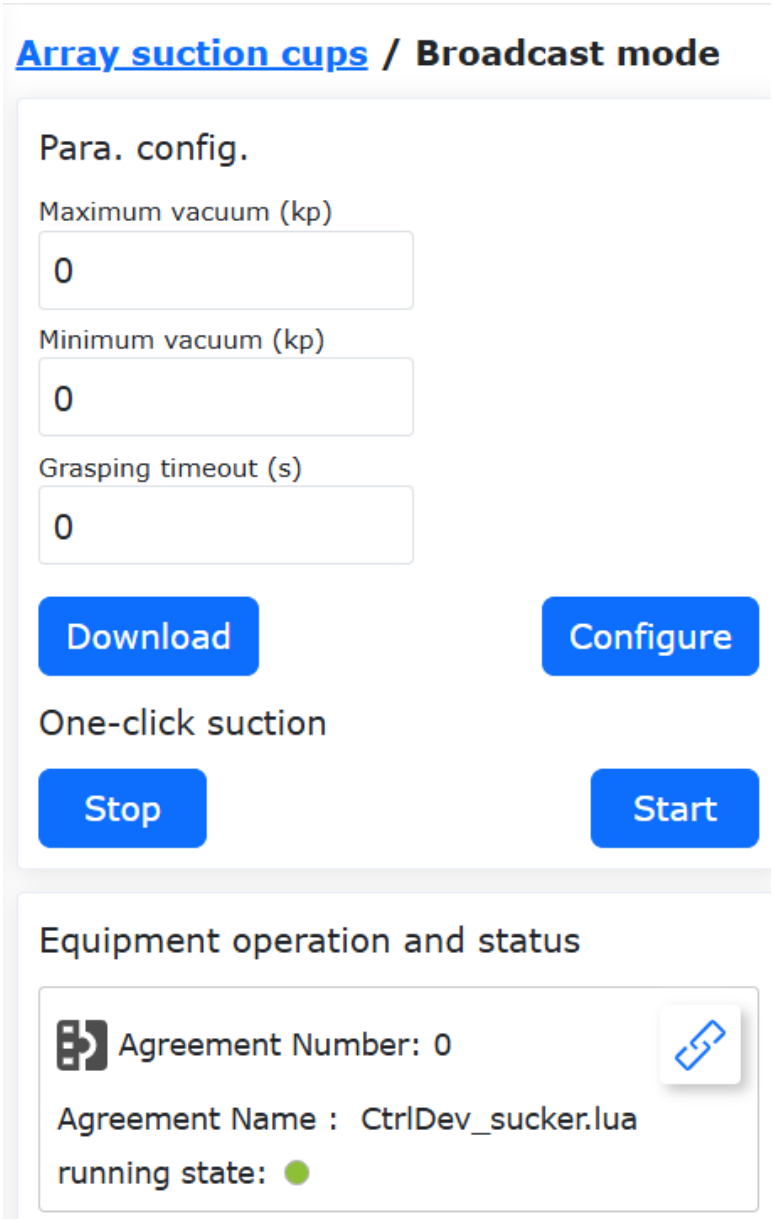
Click the “Connect” button in the “Protocol Number 1” operation box. The “Run Status” indicator lights up, indicating that communication has been established between the robot and the array suction cups. After successful connection, the operation box list for all connected suction cups is displayed in the “Device Operation and Status” column.

Click “Start” in “Parameter Configuration” -> “One-key Suction”, and each suction cup in the array suction cups will perform the “Set Vacuum Suction” action. Click “Stop”, and each suction cup in the array suction cups will stop the suction action.

Figure 8.12-9 Broadcast Mode Communication Established

Downloading the protocol file in Broadcast Mode is the same as in Unicast Mode. The protocol files downloaded from both locations can be uploaded to the robot via the “Auto Configuration” in the Unicast Mode page.


Array suction cups / Broadcast mode
Para. config.
Maximum vacuum (kp)
0
Minimum vacuum (kp)
0
Grasping timeout (s)
0
Download
Configure
One-click suction
Stop
Start
Equipment operation and status
Agreement Number: 0
Agreement Name : CtrlDev_sucker.lua
running state:


Array suction cups / Broadcast mode
Para. config.
Maximum vacuum (kp)
0
Minimum vacuum (kp)
0
Grasping timeout (s)
0
Download
Configure
One-click suction
Stop
Start
Equipment operation and status
Agreement Number: 0
Agreement Name : CtrlDev_sucker.lua
running state:

1.3.8.12.3 Array Suction Cup LUA Program Application

Adding array suction cup control, status acquisition, and other instructions to the robot LUA program, combined with robot motion instructions, can flexibly and conveniently implement material grasping and handling applications.

Open the WebApp, click sequentially “Teach Program” -> “Program Programming”, and create a new LUA program “testSucker.lua”.

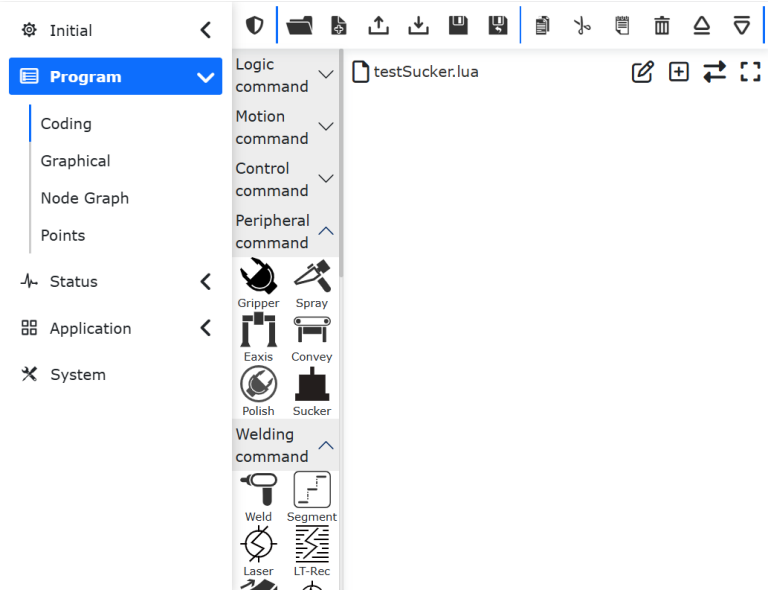


Figure 8.12-10 Create New “testSucker.lua” Program

Select the instruction type as “Peripheral Instruction”, and click the “Suction Cup” button in the peripheral instructions. At this time, the “Sucker” array suction cup instruction addition page appears on the right side of the WebApp.

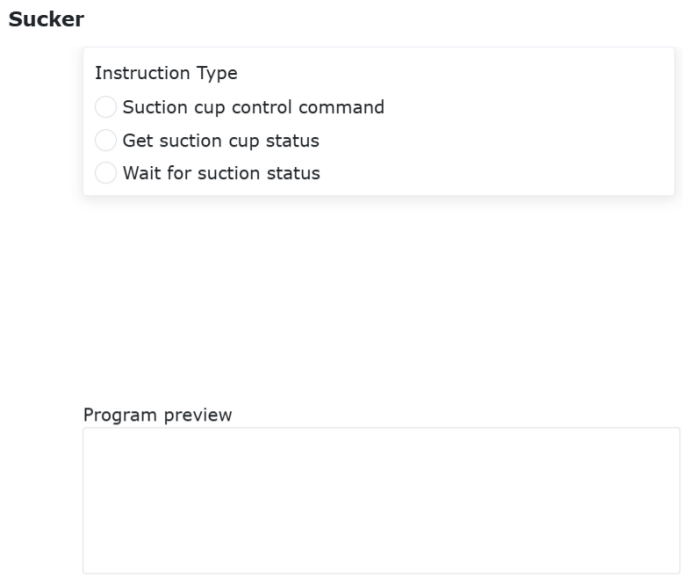


Figure 8.12-11 Array Suction Cup Instruction Addition

1.3.8.12.3.1 Suction Cup Control Instruction Addition

Writing suction cup control instructions in the LUA program allows for suction control and release control of the suction cups. Unicast Mode and Broadcast Mode controls have different logical effects.

1.3.8.12.3.2 Unicast Mode Control Instruction Addition

Unicast Mode control can control single or multiple suction cups based on the starting slave station address and quantity, and can set different control states for each suction cup.

In the suction cup instruction addition page, click “Suction Cup Control Instruction”, select the control mode as “Unicast Mode”, input the slave station number as 1, the write quantity as 2, and the suction states as “1,2”. Click the “Add” button to add a Unicast Mode suction cup control instruction in the “Program Preview”.

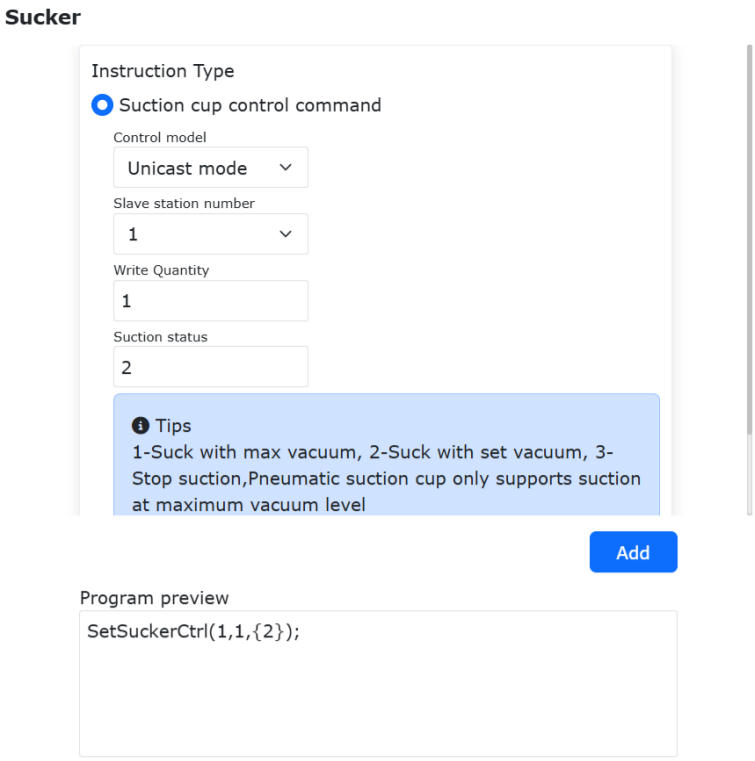


Figure 8.12-12 Add Suction Cup Control Instruction

The meanings of the parameters in the suction cup control instruction are as follows:

- **Slave Station Number**: The starting slave station number for Unicast Mode control of suction cups.
- **Write Quantity**: The number of suction cups to control starting from the starting slave station number in Unicast Mode.
- **Suction States**: In Unicast Mode, starting from the starting slave station number, the control state flag for each suction cup (1-Suction at max vacuum; 2-Suction at set vacuum, i.e., suction cup vacuum maintained between max and min vacuum; 3-Stop suction); The control state flags for each suction cup are separated by “,”, and the number of control flags must match the number of suction cups to be controlled; If you want to control two suction cups, with control operations being “Suction at max vacuum” and “Suction at set vacuum” respectively, then the input for this item is “1,2”.

Click the “Apply” button. At this time, a suction cup control instruction is added to the “testSucker.lua” program. Switch the robot to automatic mode and execute this LUA program. The robot will control the two suction cups with slave station numbers 1 and 2 to perform suction actions at maximum vacuum and set vacuum respectively.



Figure 8.12-13 Add Suction Cup Instruction in LUA Program

1.3.8.12.3.3 Adding Broadcast Mode Control Commands

The suction state set by the broadcast mode control command takes effect on all currently connected suction cups.

Click “Suction Cup Control Command”, select the control mode as “Broadcast Mode”, and enter the suction state as 1 (suction with maximum vacuum). Click the “Add” button.

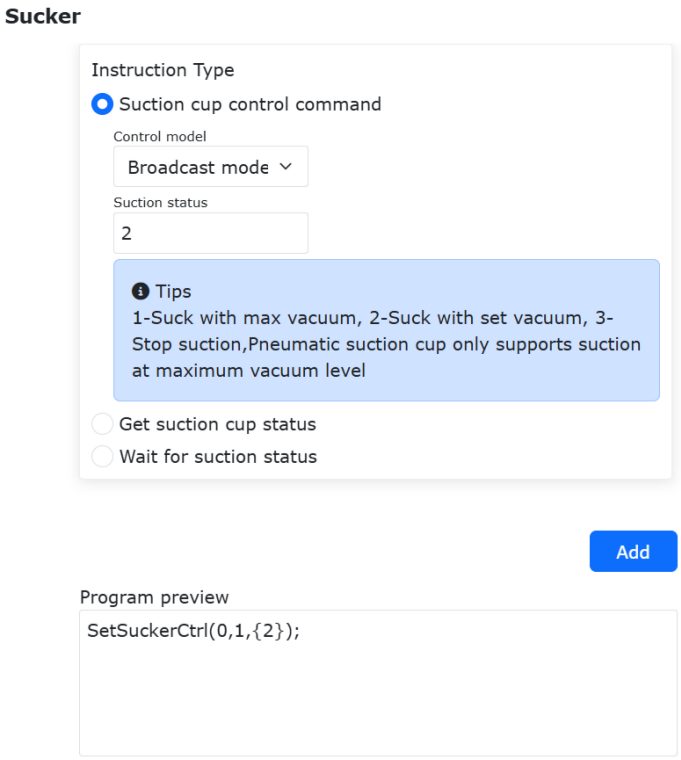
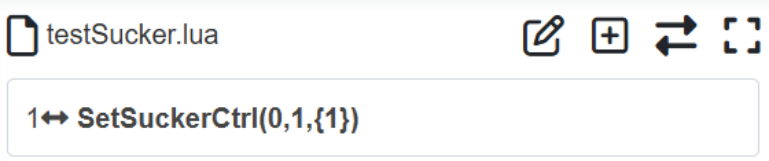


Figure 8.12-14 Adding a Broadcast Control Command

Click the “Apply” button. At this point, a broadcast mode suction cup control command is added to the “testSucker.lua” file. Switch the robot to automatic mode and execute this program, then all connected suction cups will start the suction action with maximum vacuum.

Figure 8.12-15 Adding a Broadcast Control Command in the LUA Program



1.3.8.12.3.4 Adding Suction Cup Status Acquisition Command

Click “Get Suction Cup Status”, select the slave station number of the suction cup whose status you want to obtain, then click the “Add” and “Apply” buttons sequentially. This adds a command “GetSuckerState(1)” to get the suction cup status in “testSucker.lua”.

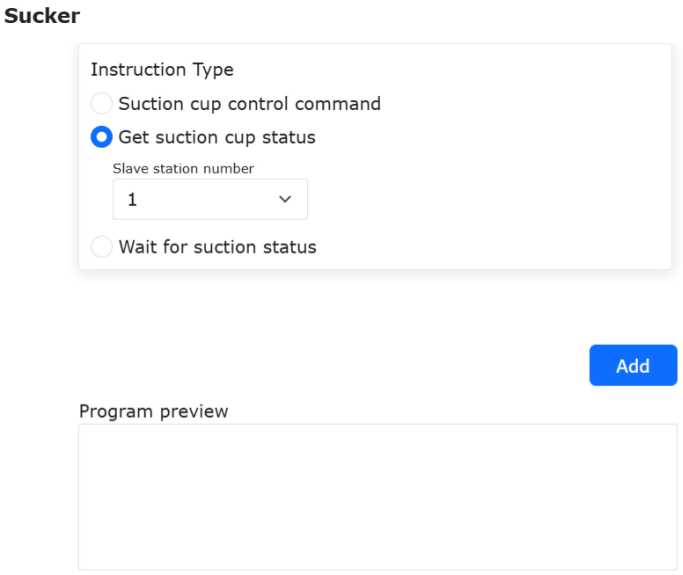


Figure 8.12-16 Adding the Get Suction Cup Status Command

The GetSuckerState() command returns 3 values, as follows:

- **state**: The current state of the suction cup: 0-Object released or suction cup started successfully; 1-Workpiece detected, object adsorbed; 2-No object adsorbed; 3-Object detached.
- **pressValue**: Current vacuum/pressure value;
- **err**: Error code: 0-Normal; Others: Abnormal.

In “testSucker.lua”, use three variables to receive the return values of the GetSuckerState() function. Use Lua variable queries to display the above information in the WebApp Variable Query Display Area.

Figure 8.12-17 Program for Getting Suction Cup Status



1.3.8.12.3.5 Adding Wait for Suction Cup Adsorption Status Command

In practical applications of array suction cups, it is often necessary to wait for the suction cup to complete suction (release) before executing the next action. The collaborative robot provides a command to wait for the suction cup action to complete. This command ends execution when the suction cup reaches the set state; otherwise, it blocks and waits for the suction cup action to complete within the set timeout period.

On the array suction cup command addition page, click "Wait for Suction Cup Adsorption Status", select the corresponding suction cup slave station number 1, select the control mode as "Workpiece detected, object adsorbed", and enter the timeout time as 10000ms. Click the "Add" button.

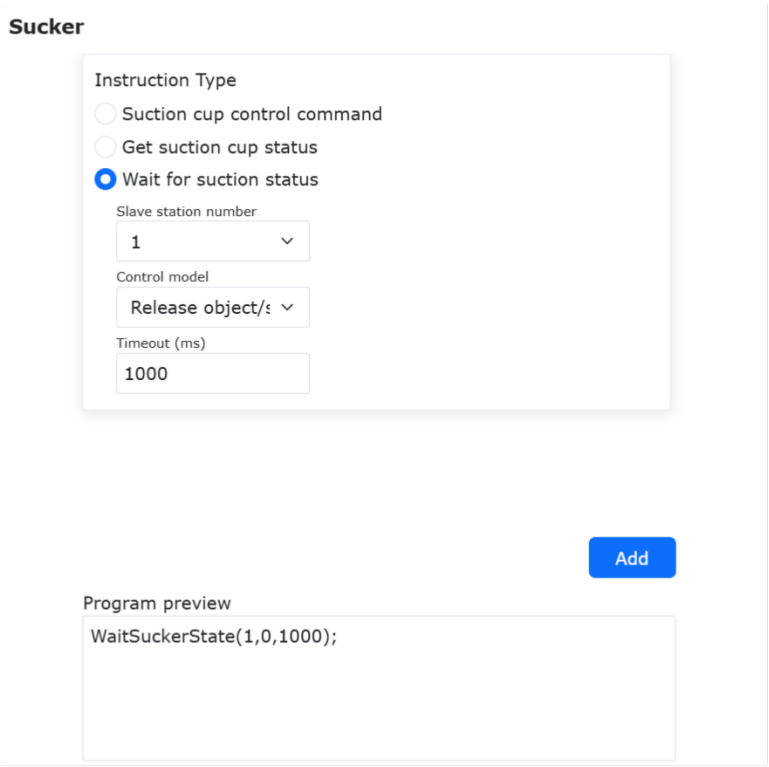


Figure 8.12-18 Adding the Wait for Suction Cup Status Command

Click the “Apply” button. A command to wait for the suction cup to adsorb an object is added to “testSucker.lua”.

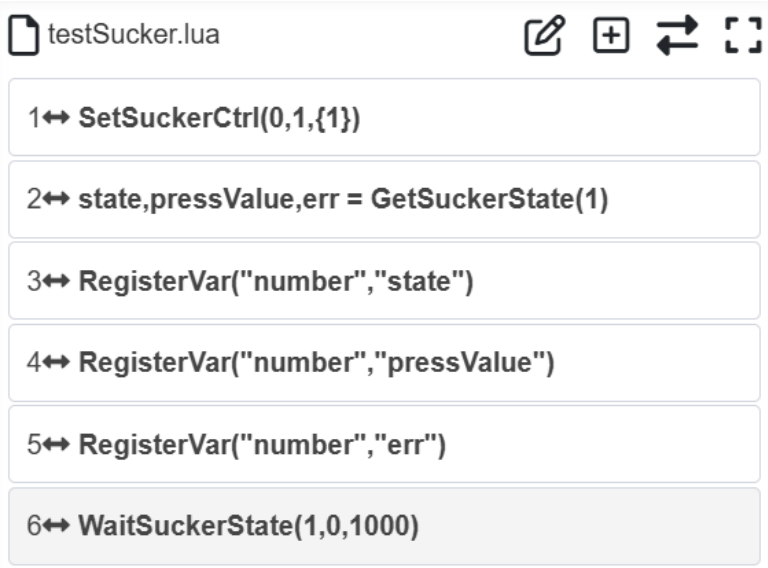


Figure 8.12-19 Adding Wait for Suction Cup to Adsorb Object in LUA Program

1.3.8.12.3.6 Application Example

Example LUA program for suction cup handling control:

```
while (1) do
::satety_suck::
PTP(sucker_safey,100,-1,0)
PTP(sucker_suck,100,-1,0)
SetSuckerCtrl(2, 1, {2})
SetSuckerCtrl(11, 1, {2})
loop1 = 0
while (loop1 < 10) do
    state, press, errorcode = GetSuckerState(2)
    RegisterVar("number","state")
    RegisterVar("number","press")
    RegisterVar("number","errorcode")
    state11, press11, errorcode11 = GetSuckerState(11)
    RegisterVar("number","state11")
    RegisterVar("number","press11")
    RegisterVar("number","errorcode11")
    loop1 = loop1 + 1
    WaitMs(50)
end

if(state11 == 1) then
    PTP(sucker_safey,100,-1,0)
    PTP(sucker_release,100,-1,0)
    WaitMs(1000)
    SetSuckerCtrl(2, 1, {3})
    SetSuckerCtrl(11, 1, {3})
```

(continues on next page)

(continued from previous page)

```
    WaitMs(500)
else
    PTP(sucker_safey,100,-1,0)
    SetSuckerCtrl(2, 1, {3})
    SetSuckerCtrl(11, 1, {3})
    WaitMs(2000)
    goto satety_suck
end
::satety_release::
PTP(sucker_safey,100,-1,0)
PTP(sucker_release,100,-1,0)
SetSuckerCtrl(2, 1, {2})
SetSuckerCtrl(11, 1, {2})
loop1 = 0
while (loop1 < 10) do
    state, press, errorcode = GetSuckerState(2)
    RegisterVar("number","state")
    RegisterVar("number","press")
    RegisterVar("number","errorcode")
    state11, press11, errorcode11 = GetSuckerState(11)
    RegisterVar("number","state11")
    RegisterVar("number","press11")
    RegisterVar("number","errorcode11")
    loop1 = loop1 + 1
    WaitMs(50)
end

if(state11 == 1) then
    PTP(sucker_safey,100,-1,0)
    PTP(sucker_suck,100,-1,0)
    WaitMs(1000)
    SetSuckerCtrl(2, 1, {3})
    SetSuckerCtrl(11, 1, {3})
    WaitMs(500)
else
    PTP(sucker_safey,100,-1,0)
    SetSuckerCtrl(2, 1, {3})
    SetSuckerCtrl(11, 1, {3})
    WaitMs(2000)
    goto satety_release
end
end
```

1.3.8.13 CNC Function Package Based on FOCAS (For Linux Systems Only)

1.3.8.13.1 Overview

To automate the loading and unloading process in machine tool processing, a CNC function package based on FOCAS communication has been developed, enabling communication interaction and coordinated motion between the collaborative robot and the CNC machine tool.

As shown in the figure, FOCAS communication is based on Ethernet. By connecting the robot control box network port to the embedded network port of the machine tool with an Ethernet cable, FOCAS communication between the robot and the machine tool can be established, enabling CNC control and machine tool status monitoring from the robot side.

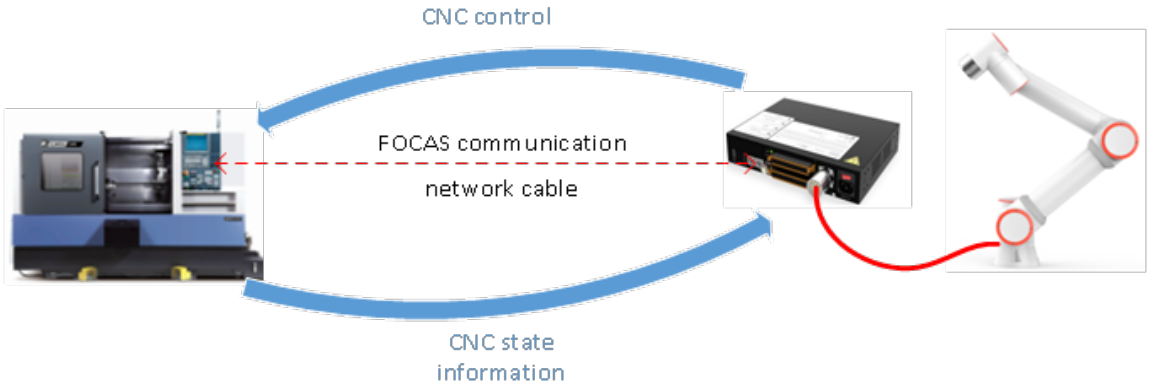


Figure 8.13-1 FOCAS Communication Topology Diagram Between Robot and CNC

The functions currently supported by the control box's CNC function package based on FOCAS communication for machine tool control and status feedback are shown in the table.

Table 8.13-1 Supported Functions of the FOCAS-based CNC Function Package

No.	Function Name	Description
1	Machine Tool Type	Status Feedback
2	FOCAS Communication Status	Status Feedback
3	Auto Mode Operation	Control, Status Feedback
4	Alarm Status	Status Feedback
5	Safety Door	Status Feedback
6	Chuck	Control, Status Feedback
7	Emergency Stop	Control, Status Feedback

1.3.8.13.2 Related Operation Instructions

1.3.8.13.2.1 Establishing FOCAS Communication

FOCAS communication is based on Ethernet. It requires forming a local area network (LAN) connecting the robot, CNC machine tool, and PC to achieve physical link connection, and finally establishing FOCAS communication through the robot's open protocol.

1.3.8.13.2.2 Network Configuration

Step1: First, change the IP address of the PC to the same subnet as the robot control box. The robot control box's IP address is "192.168.58.2".

If no switch is used for networking, you can use the two built-in network ports on the robot control box for networking. The operation is as follows: Log in to the robot's WebAPP, go to System Settings -> General Settings -> Network Settings, set the IP address of Port 0 to: 192.168.58.2; set the IP address of Port 1 to 192.168.57.2. Simultaneously, set WebAPP to use Port 0 and WebRecovery to use Port 1, as shown in the figure. After completing all settings, click Set Network.

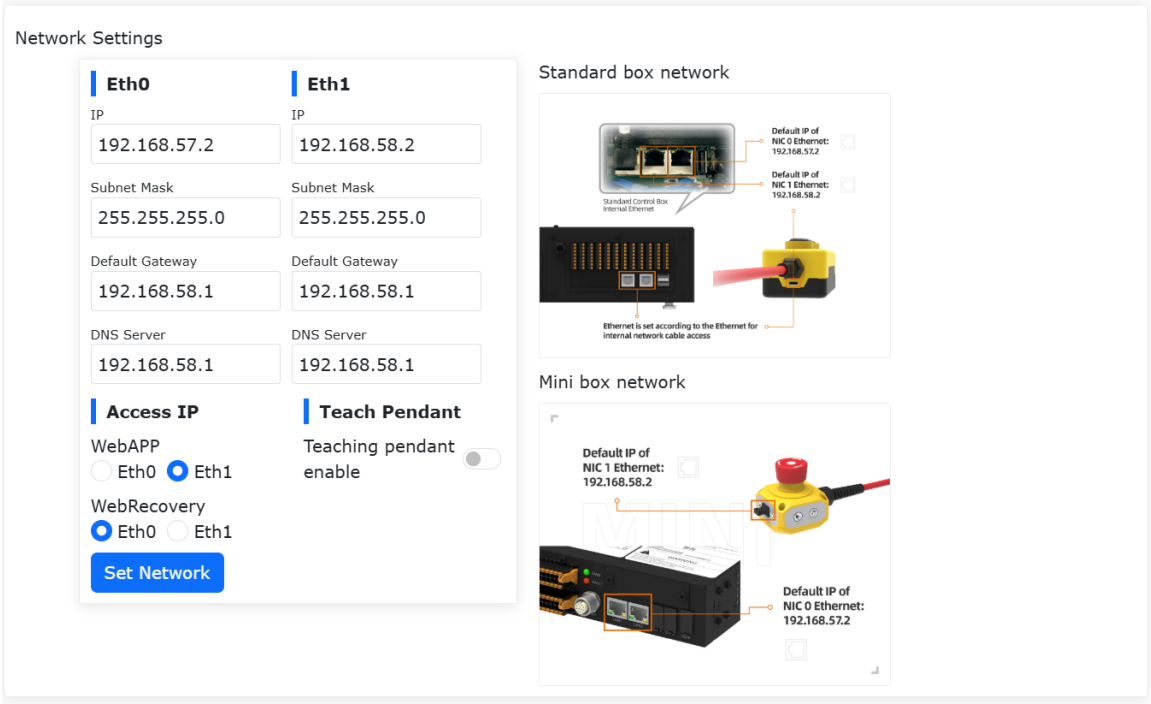


Figure 8.13-2 Robot Network Configuration Diagram

Step2: Then restart the control box and connect it to the PC via the Port 0 network port, then log in to the robot WebApp. Also, configure the IP address of the CNC machine tool that needs communication to be in the same subnet as the PC and the robot control box, i.e., 192.168.58.xx, and change the machine tool's port to 8193. This completes all network configuration.

1.3.8.13.2.3 Open Protocol File Configuration

Step1: Then proceed with the peripheral open protocol configuration. First, you need to create a new lua file named starting with *CtrlDev_CNC* as the open protocol file for establishing FOCAS communication, such as *CtrlDev_CNC_demo.lua*.

This file needs to set the Open Protocol ID and use the *CNCComSet* function to establish or disconnect the connection with the CNC. The parameter descriptions for the *CNCComSet* function are shown in the table below. Example code is as follows.

Table 8.13-2 CNCComSet Function Parameter Description Table

No.	Parameter Name	Description
1	Machine Tool Manufacturer	0-Invalid 1-Machine Tool (FOCAS)
2	Communication Command	1-Establish Connection 1001-Disconnect
3	Machine Tool IP Address	–
4	Machine Tool Port Number	–

Open protocol example code for establishing FOCAS connection:

```
local id = 1      --Open LUA Protocol ID
--FOCAS Disconnect
CNCComSet(1, 1001, '192.168.57.100', 8193)
sleep_ms(1000)
--FOCAS Establish Connection
CNCComSet(1, 1, '192.168.57.100', 8193)
sleep_ms(1000)
while(1) do
sleep_ms(5000)
end
```

Step2: After writing the open protocol lua file, select the just-created *CtrlDev_CNC_fanuc.lua* file and upload it. Select the ID set in the file, choose the uploaded open protocol file from the dropdown, and click Configure.

Figure 8.13-3 Open Protocol File Upload and Configuration

Step3: Then check that all communication links are normal and confirm the CNC machine tool is powered on. Click the Connect button in the open protocol. You can confirm whether the connection with the machine tool is established through the CNC -> FOCAS Communication Status in the status feedback panel on the right (Red light: Connection established; Gray: Connection disconnected), as shown in the figure.

Figure 8.13-4 FOCAS Communication Connection Established

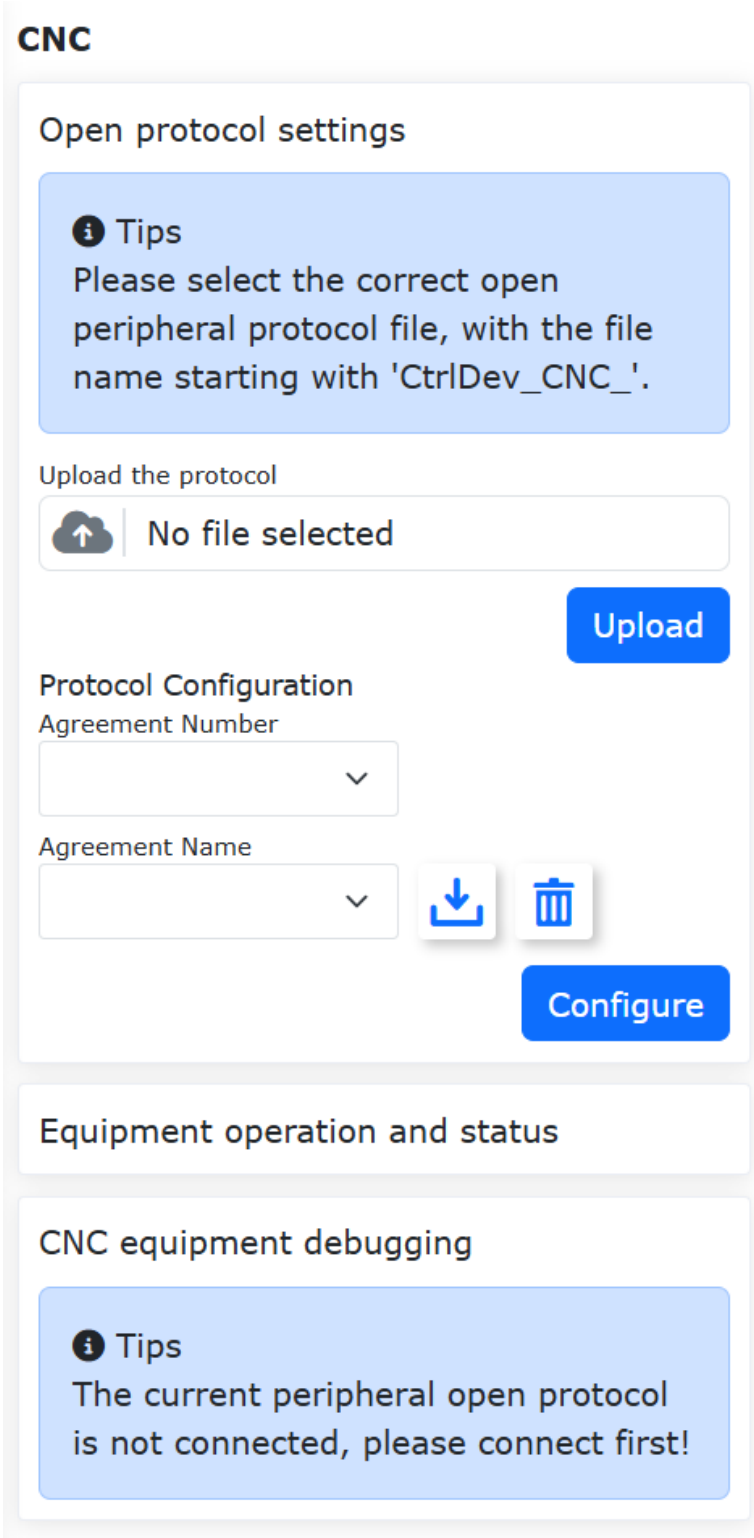
1.3.8.13.2.4 CNC Status Feedback Description

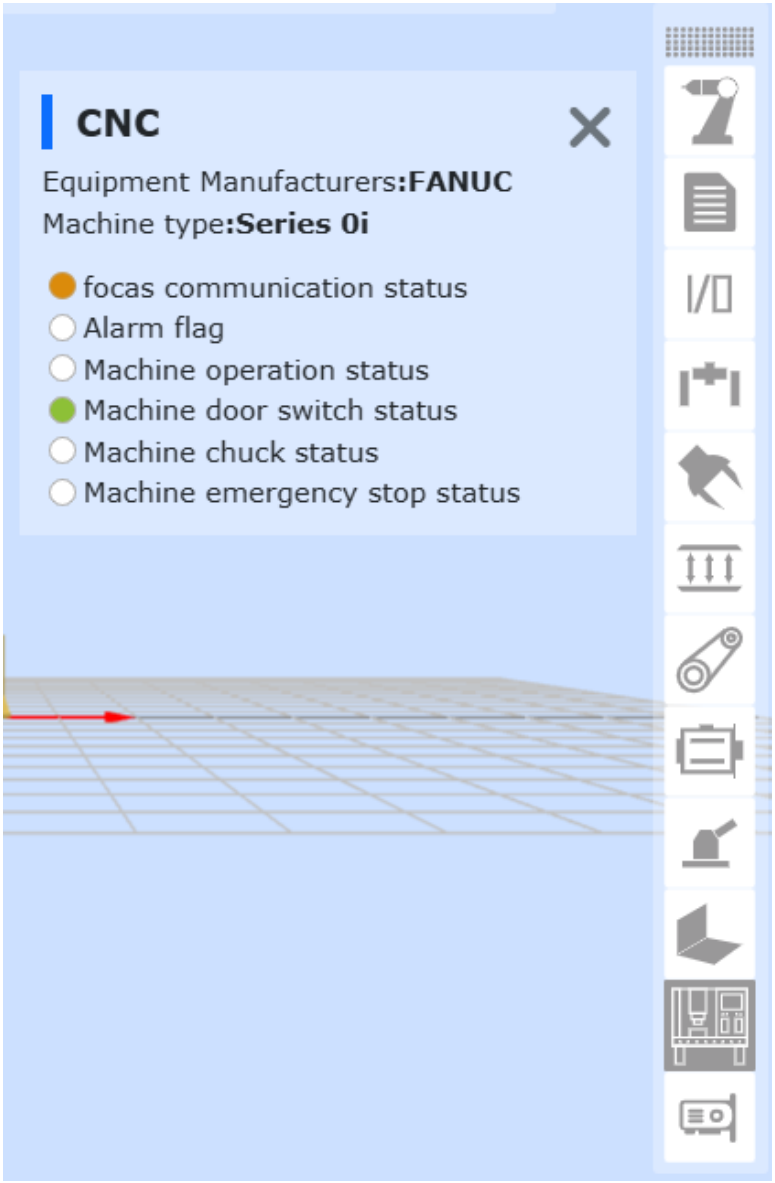
The status feedback of the CNC machine tool is displayed on the CNC icon in the peripheral status feedback area on the far right of the WebAPP, as shown in the figure. Clicking it will display all current statuses of the machine tool, including equipment manufacturer, machine tool type, FOCAS communication status, alarm flag, machine tool operation status, machine door switch status, machine chuck status, and machine emergency stop status.

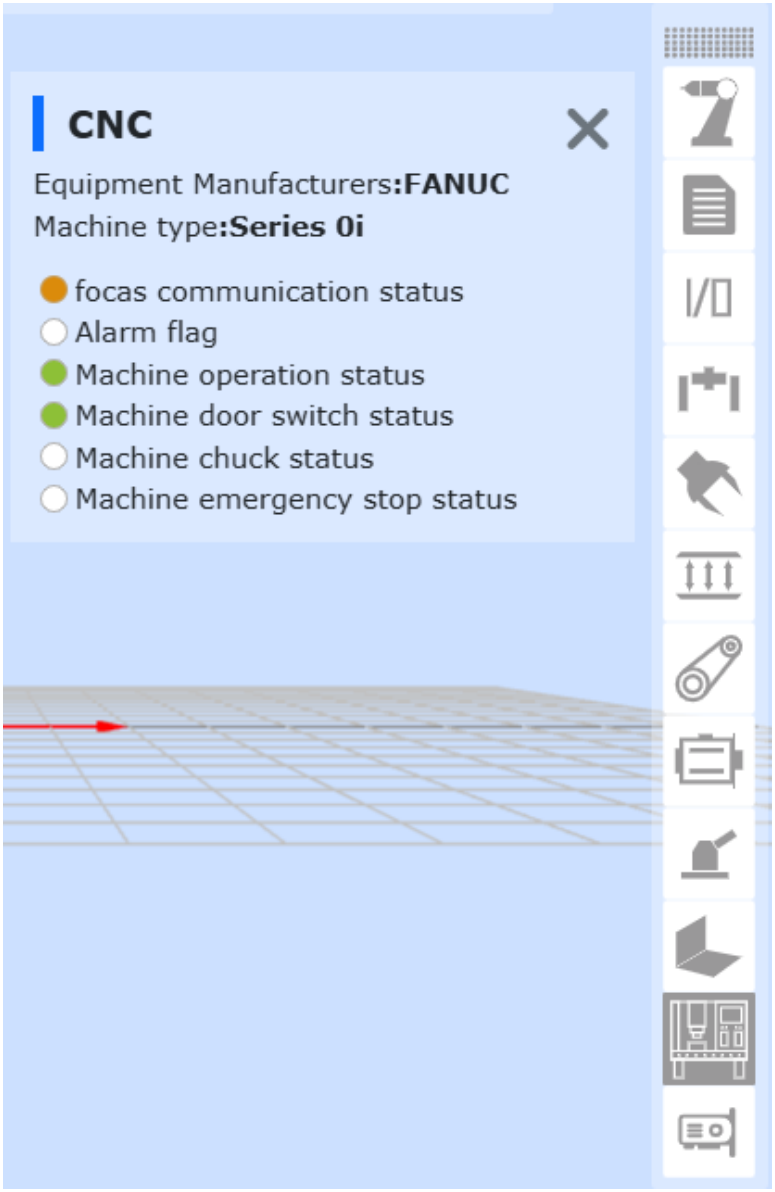
Figure 8.13-5 CNC Status Feedback Panel

The meanings of the status feedback indicator lights for CNC are shown in the table below.

Table 8.13-3 Meanings of CNC Status Feedback Indicator Lights


CNC
Open protocol settings
Tips
Please select the correct open peripheral protocol file, with the file name starting with 'CtrlDev_CNC_'.
Upload the protocol
No file selected
Upload
Protocol Configuration
Agreement Number
Agreement Name
Configure
Equipment operation and status
CNC equipment debugging
Tips
The current peripheral open protocol is not connected, please connect first!


CNC
Equipment Manufacturers:FANUC
Machine type:Series 0i
focas communication status
Alarm flag
Machine operation status
Machine door switch status
Machine chuck status
Machine emergency stop status


CNC
Equipment Manufacturers:FANUC
Machine type:Series 0i
focas communication status
Alarm flag
Machine operation status
Machine door switch status
Machine chuck status
Machine emergency stop status

No.	Function Name	Description
1	FOCAS Communication Status	Gray-Communication disconnected Red-Communication normal
2	Alarm Flag	Gray-No alarm Red-Alarm exists
3	Machine Tool Operation Status	Gray-Stopped Green-Running
4	Machine Door Status	Gray-Door closed Green-Door open
5	Machine Chuck Status	Gray-Loosened Green-Clamped
6	Machine Emergency Stop Status	Gray-E-stop inactive Green-E-stop active

1.3.8.13.2.5 CNC Control Description

The control for the CNC machine tool is located in the Peripheral Open Protocol. After the FOCAS communication connection is completed, click the upper right corner of the configured peripheral open protocol to open the CNC control page, as shown in the figure.

Note: The control buttons include Door Control (Open, Close), Chuck Control (Clamp, Loosen), Start/Stop Control (Run, Stop), and Emergency Stop Control (E-stop, Inactive). All control signals are edge-triggered.

Figure 8.13-6 CNC Control Page

1.3.8.13.2.6 CNC Teach Program Description

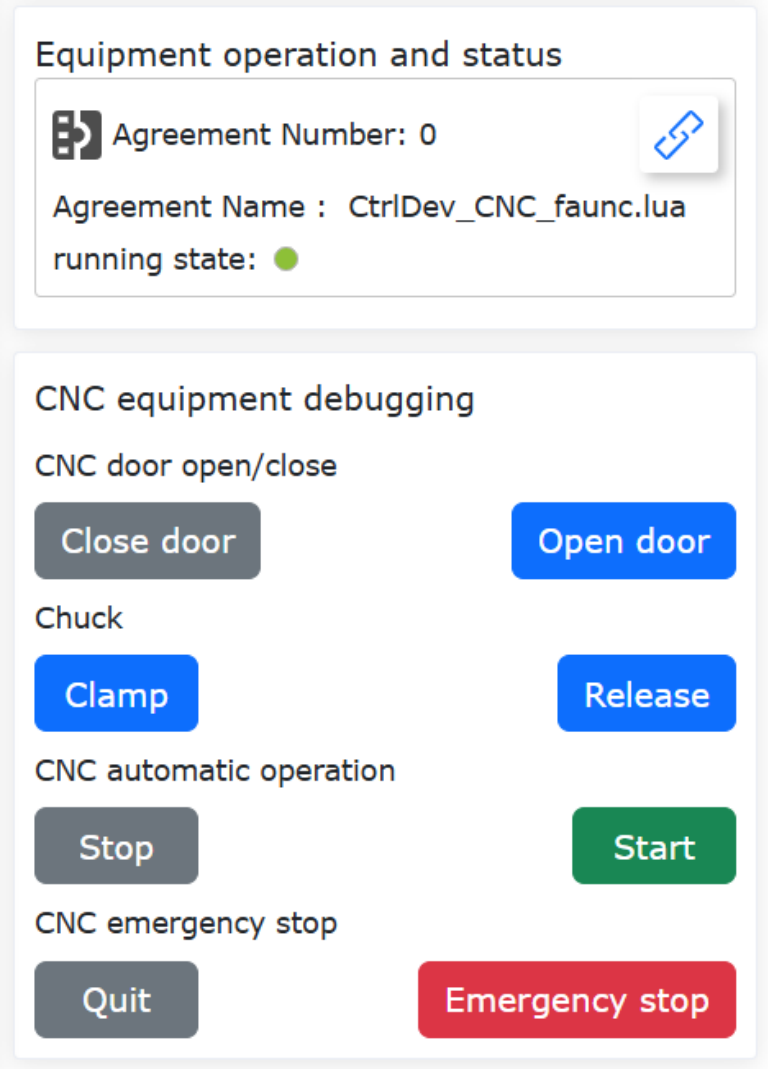
The CNC function package supports calling control commands in teach programs and obtaining machine tool status in real time. Navigate sequentially to “Teach Program” -> “Program Programming” -> “Peripheral Commands” -> “CNC” to see all supported CNC teach commands, as shown in the figure.

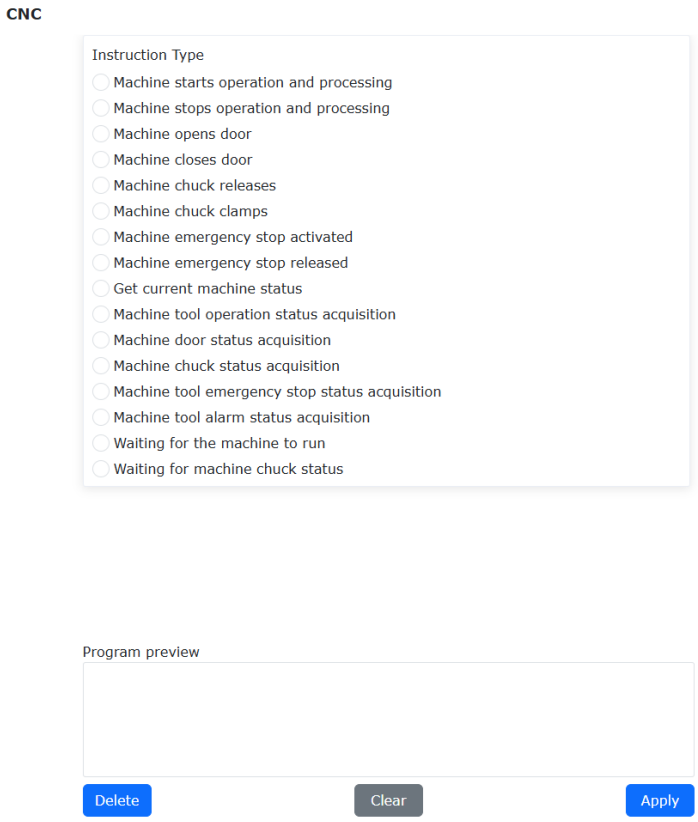
Figure 8.13-7 CNC Teach Commands

Note: The control commands correspond one-to-one with the CNC controls and are all edge signal effective, meaning that after a start command is executed, a stop command must be executed before the next start command will take effect.

“Get Current Machine Tool Status” is a Lua function. This function returns 9 parameters, whose meanings are shown in the table below.

Table 8.13-4 Return Value Description for “Get Current Machine Tool Status”


Equipment operation and status
Agreement Number: 0
Agreement Name : CtrlDev_CNC_faunc.lua
running state:
CNC equipment debugging
CNC door open/close
Close door
Open door
Chuck
Clamp
Release
CNC automatic operation
Stop
Start
CNC emergency stop
Quit
Emergency stop



No.	Name	Meaning
1	Equipment Manufacturer	0-Invalid 1-Other-Reserved
2	FOCAS Communication Status	0-Communication normal Other-Communication disconnected
3	Machine Tool Model (string)	'15' : Series 150/150i '16' : Series 160/160i '18' : Series 180/180i '21' : Series 210/210i '30' : Series 300i '31' : Series 310i '32' : Series 320i '0' : Series 0i
4	Machine Tool Type (string)	'15' : Series 150/150i '16' : Series 160/160i '18' : Series 180/180i '21' : Series 210/210i '30' : Series 300i '31' : Series 310i '32' : Series 320i '0' : Series 0i
5	Machine Tool Operation Status	0-Stopped 1-Running
6	Machine Tool Emergency Stop Status	0-E-stop active Other-E-stop inactive
7	Machine Tool Alarm Status	0-No alarm Other-Alarm exists
8	Machine Tool Door Status	0-Door open 1-Door closed
9	Machine Tool Chuck Status	0-Loosened 1-Clamped

A Lua teach program example is written for the robot loading/unloading process. This example program includes controlling the CNC door (close, open), operation (run, stop), chuck (loosen, clamp), and uses the obtained CNC current status as judgment conditions to set the robot's movement between three points: safety point, pick point, and place point, as shown in the code.

Example Lua Teach Program for Coordinated Motion between Robot and CNC:

```
while (1) do
   CNCDoorClose()
   CNCWorkStart()
   WaitMs(1000)
   t1,t2,t3,t4,t5,t6,t7,t8,t9=CNCGetStatus()
   if t5 == 1 then
       PTP(CNCsafe,100,-1,0)
   else
       CNCWorkStop()
       CNCDoorOpen()
       WaitMs(1000)
       PTP(CNCg1,100,-1,0)
       WaitMs(1000)
       CNCChuckOpen()
       PTP(CNCg2,100,-1,0)
       PTP(CNCsafe,100,-1,0)
   end
   t1,t2,t3,t4,t5,t6,t7,t8,t9=CNCGetStatus()
   if t8 == 0 then
       if t5 == 0 then
           PTP(CNCg2,100,-1,0)
            PTP(CNCg1,100,-1,0)
            CNCChuckFastening()
            WaitMs(1000)
            PTP(CNCsafe,100,-1,0)
        end
    end
end
```

1.3.8.14 Virtual Wall Configuration Based on Force Sensor

The virtual wall function based on the force sensor allows setting virtual walls to limit the robot's workspace and avoid direct collision contact.

1.3.8.14.1 Force Sensor Installation and Configuration

Step1: Taking the "Kunwei" sensor as an example, during installation, the coordinate system direction of the force sensor must be consistent with the end flange coordinate system, as shown in Figure 1 (In Figure 1, red indicates the X+ direction of the end flange coordinate system, green indicates the Y+ direction, and blue indicates the Z+ direction);

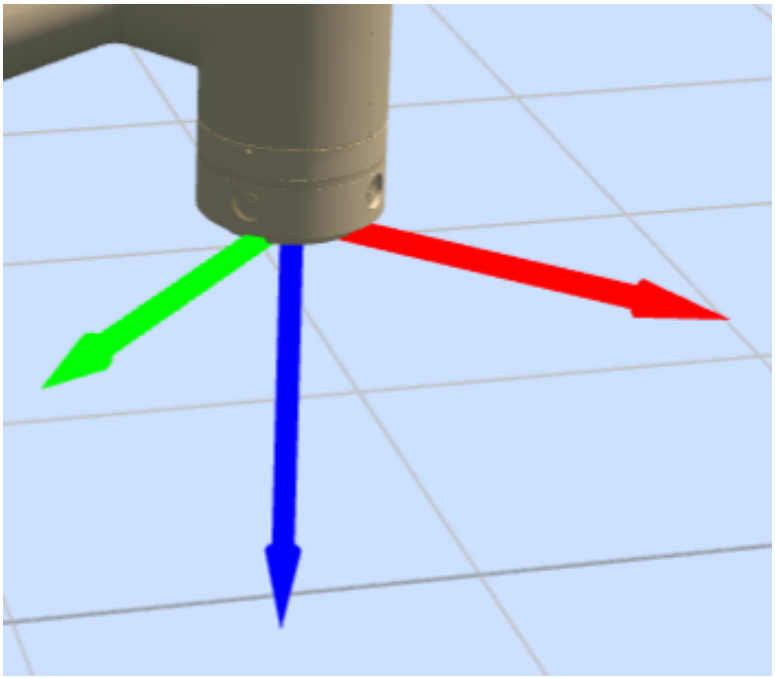
Figure 8.14-1 Force Sensor Installation

Step2: Under the "Initial" -> "Peripherals" -> "Force Sensor" menu bar, click "Adapted Devices" to enter the force sensor device configuration interface.

The force sensor configuration information includes manufacturer, type, software version, and mounting location. Users can configure the corresponding force sensor information according to specific production needs. If users need to change the configuration, they can first select the corresponding number, click the "Clear" button to clear the corresponding information, and reconfigure according to requirements; the specific operation is shown in the figure.

Step3: Select the configured force sensor number, click the "Reset" button. After the page pops up indicating the command was sent successfully, click the "Activate" button. Check the activation status in the force sensor information table to determine if activation was successful. Additionally, the force sensor will have initial values. Users can choose


Y+
X+



"Zero Calibration" and "Remove Zero" according to usage needs. Force sensor zero calibration requires ensuring the force sensor is level and vertically downward, and the robot has no configured load.

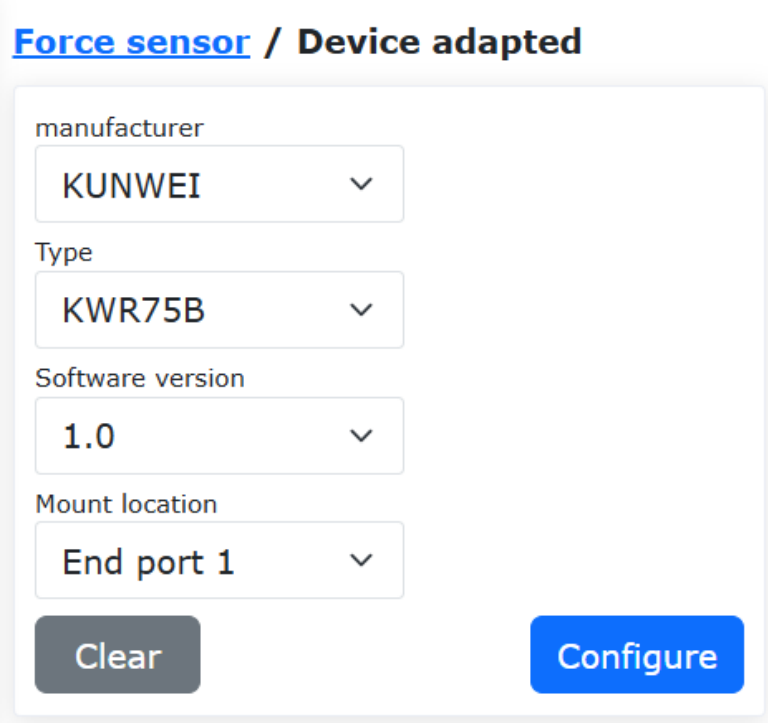


Figure 8.14-2 Force Sensor Configuration and Activation

Figure 8.14-3 Force Sensor Activation

1.3.8.14.2 Virtual Wall Configuration

To use the force sensor for assisted dragging, a dragging handle needs to be installed under the force sensor, and a tool coordinate system needs to be configured. The specific operation is shown in Figure 4. At this point, the method for detecting the interference zone uses the set tool coordinate system position as a reference; if not set, the end flange is used as the reference.

Step1: Under the "Initial" -> "Safety" -> "Interference Zone" menu bar, click "Single" to enter the interference zone configuration function interface;

Step2: It is necessary to configure the interference method and the operation upon entering the interference zone; click "Cube Interference" to enter the configuration interface, set the operation upon entering the interference zone during dragging to "Do Not Restrict Dragging", and the operation upon entering during motion can be set as needed;

Step3: According to requirements, the parameter configuration can be modified. The detection method is divided into two types: "Command Position" and "Feedback Position". The interference zone mode is divided into two types: "Interference Within Range" and "Interference Outside Range". The reference coordinate system is selected as "Base Coordinate". Set according to actual use. Detailed operations are shown in the figure;

Figure 8.14-4 Installing Dragging Handle and Setting Tool Coordinate System

Figure 8.14-5 Virtual Wall Parameter Configuration

Step4: The interference zone mode under parameter configuration is divided into two types: "Interference Within Range" and "Interference Outside Range";

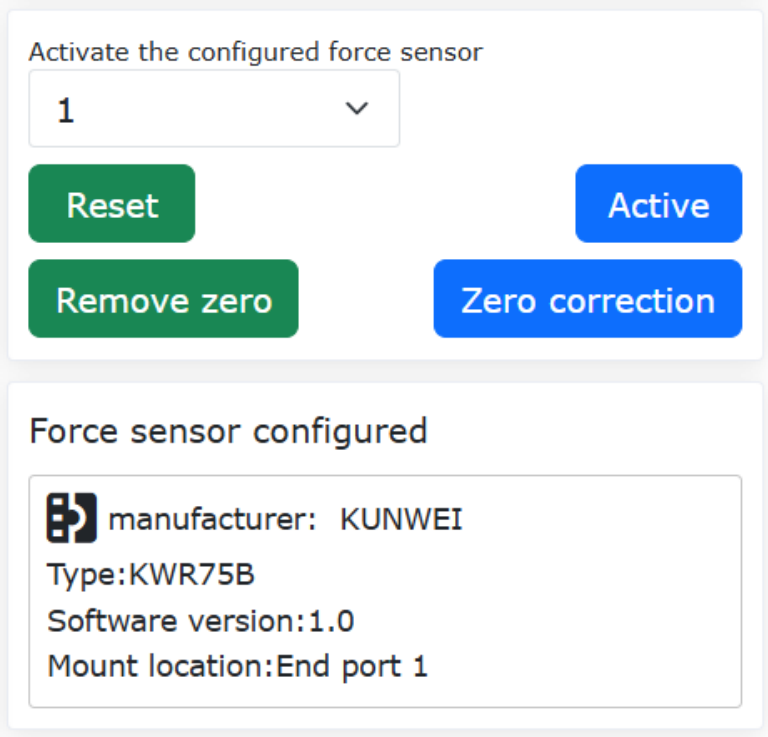


Figure 8.14-6 Interference Within Range

Figure 8.14-7 Interference Outside Range

Step5: Establish the interference zone. The specific operation is shown in Figure 7 and Figure 8; it is recommended to set the interference area as large as possible when selecting "Interference Outside Range".

Figure 8.14-8 Establishing Interference Zone Using Two-Point Method

Figure 8.14-9 Establishing Interference Zone Using Center Point + Side Length Method

1.3.8.14.3 Force Sensor Assisted Dragging

Step1: Under the "Application" -> "Tool App" menu, click "Drag Lock" to enter the force sensor assisted locking function interface;

Step2: Set the parameters as shown in the figure below to enable the virtual wall function based on the force sensor. The specific effect is: approaching the virtual wall, resistance increases; moving away from the virtual wall, the force sensor assisted dragging function works normally.

Figure 8.14-10 Parameter Settings for Force Sensor Assisted Dragging

Specific functions of the parameters:

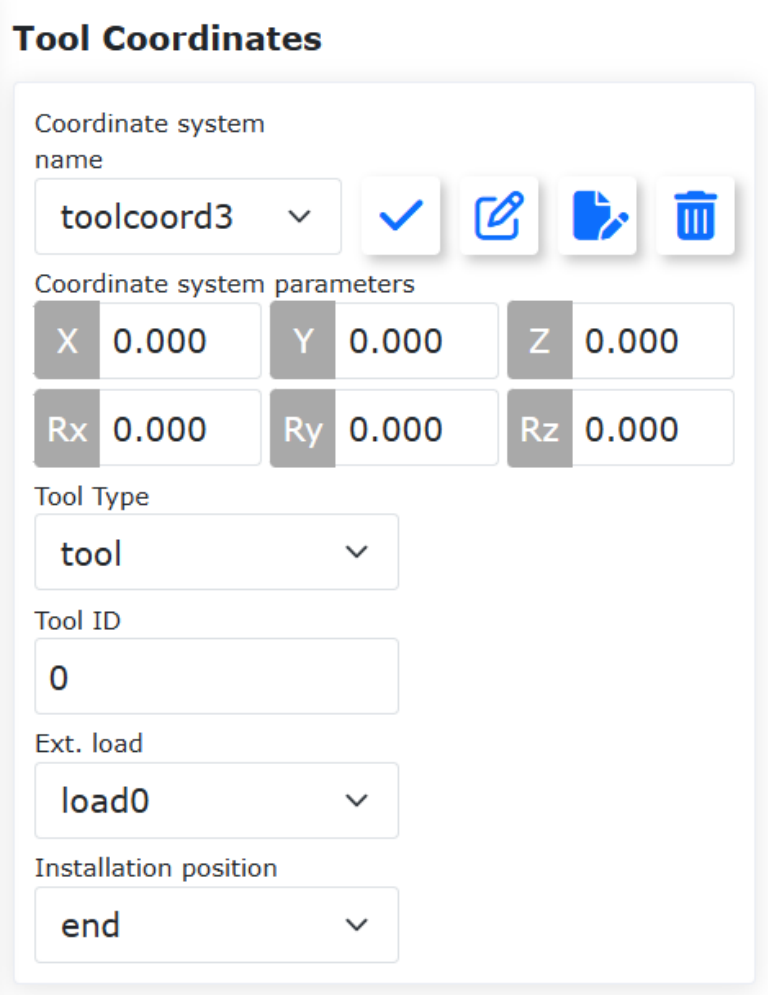
Adaptive Selection: Enable when assembly is needed; dragging becomes heavier after enabling;

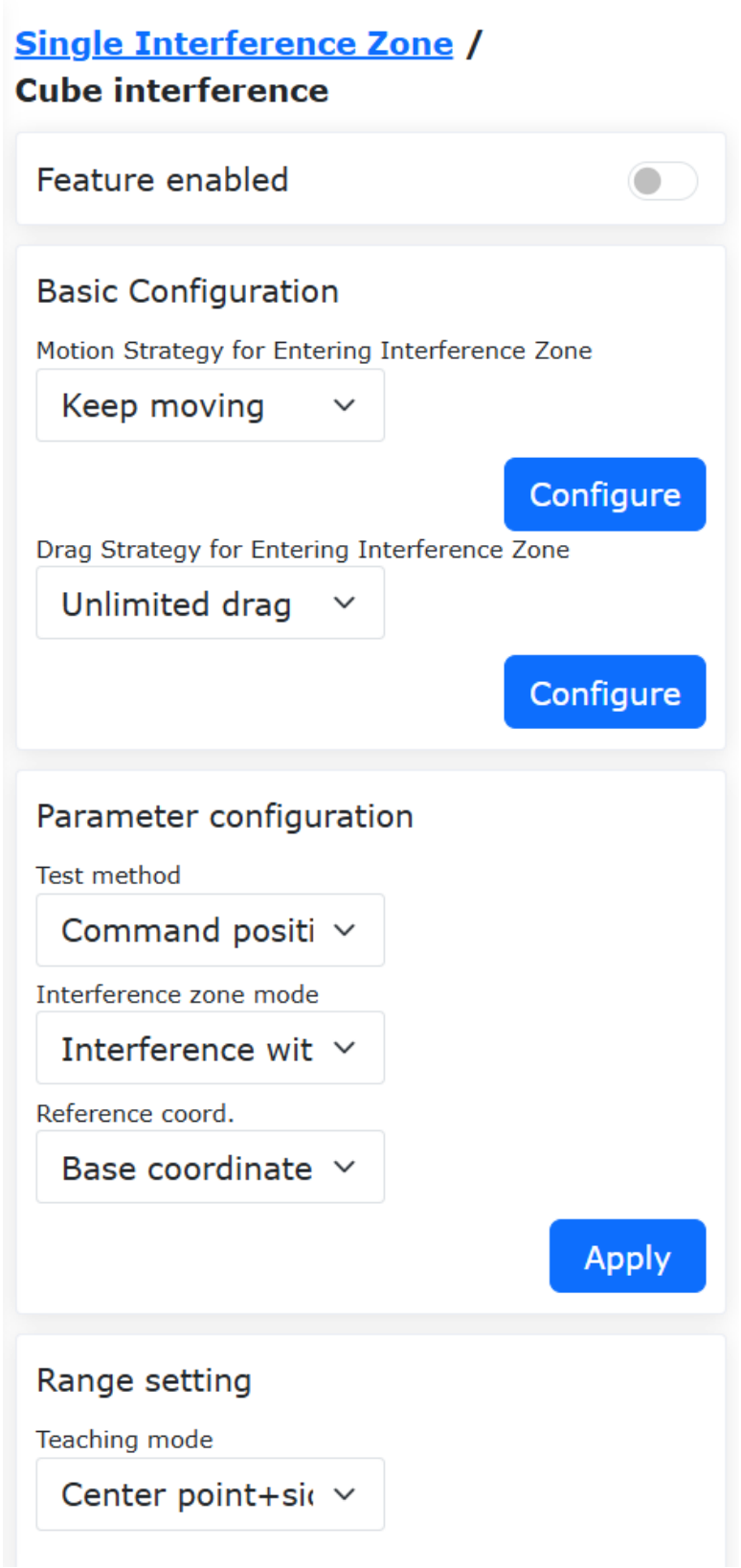
Inertia Parameter: Adjusts the feel during dragging; operate with caution under the guidance of technical personnel.

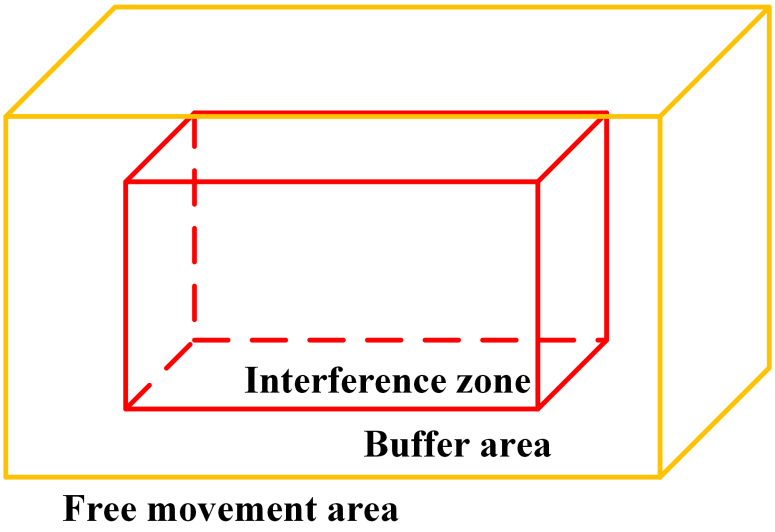
Damping Parameter:

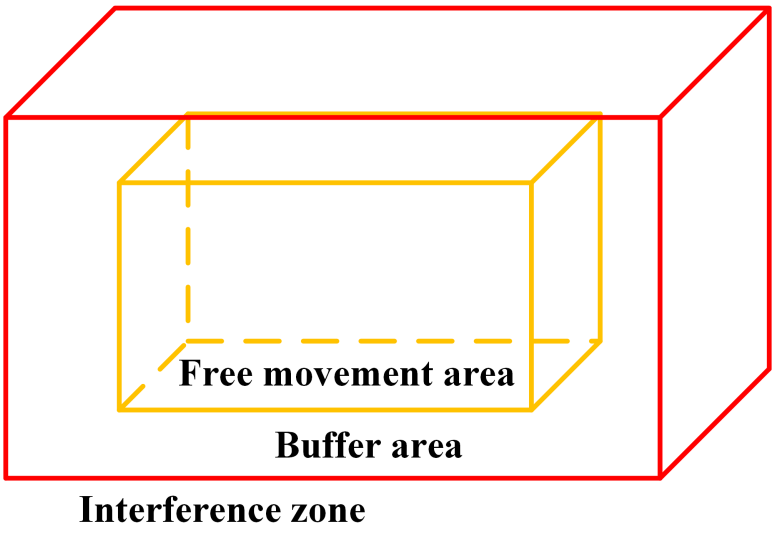
- Translational Direction: Recommended parameter range [100-200];
- Rotational Direction: Recommended parameter range [3-10], with the RZ direction range being [0.1-5];


Y+
X+


Tool Coordinates
Coordinate system name
toolcoord3
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
tool
Tool ID
0
Ext. load
load0
Installation position
end


Single Interference Zone /
Cube interference
Feature enabled
Basic Configuration
Motion Strategy for Entering Interference Zone
Keep moving
Configure
Drag Strategy for Entering Interference Zone
Unlimited drag
Configure
Parameter configuration
Test method
Command positi
Interference zone mode
Interference wit
Reference coord.
Base coordinate
Apply
Range setting
Teaching mode
Center point+si

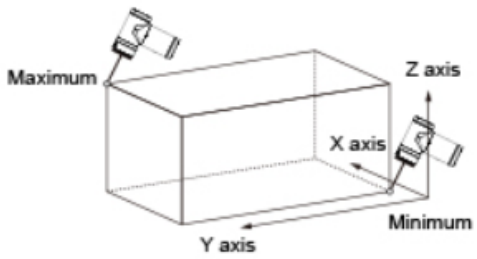

Interference zone
Buffer area
Free movement area


Free movement area
Buffer area
Interference zone

Range setting

Teaching mode

Two-point meth



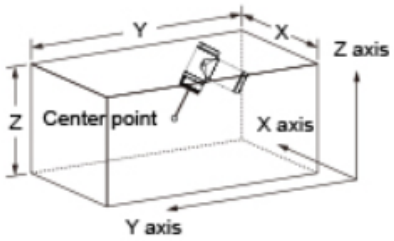
Joint	Minimum	Maximum
X	0.000	0.000
Y	0.000	0.000
Z	0.000	0.000

Apply

Range setting

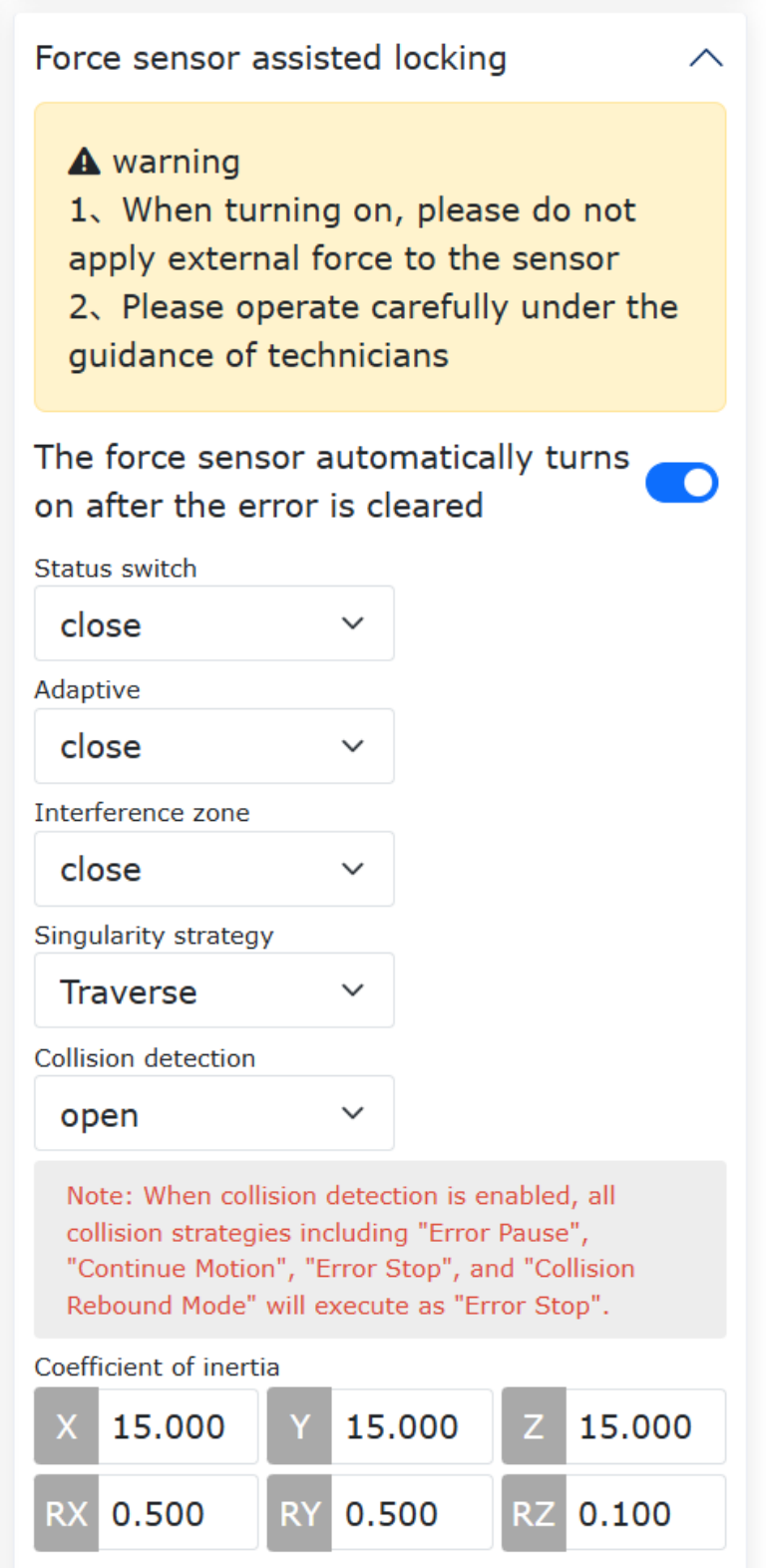
Teaching mode

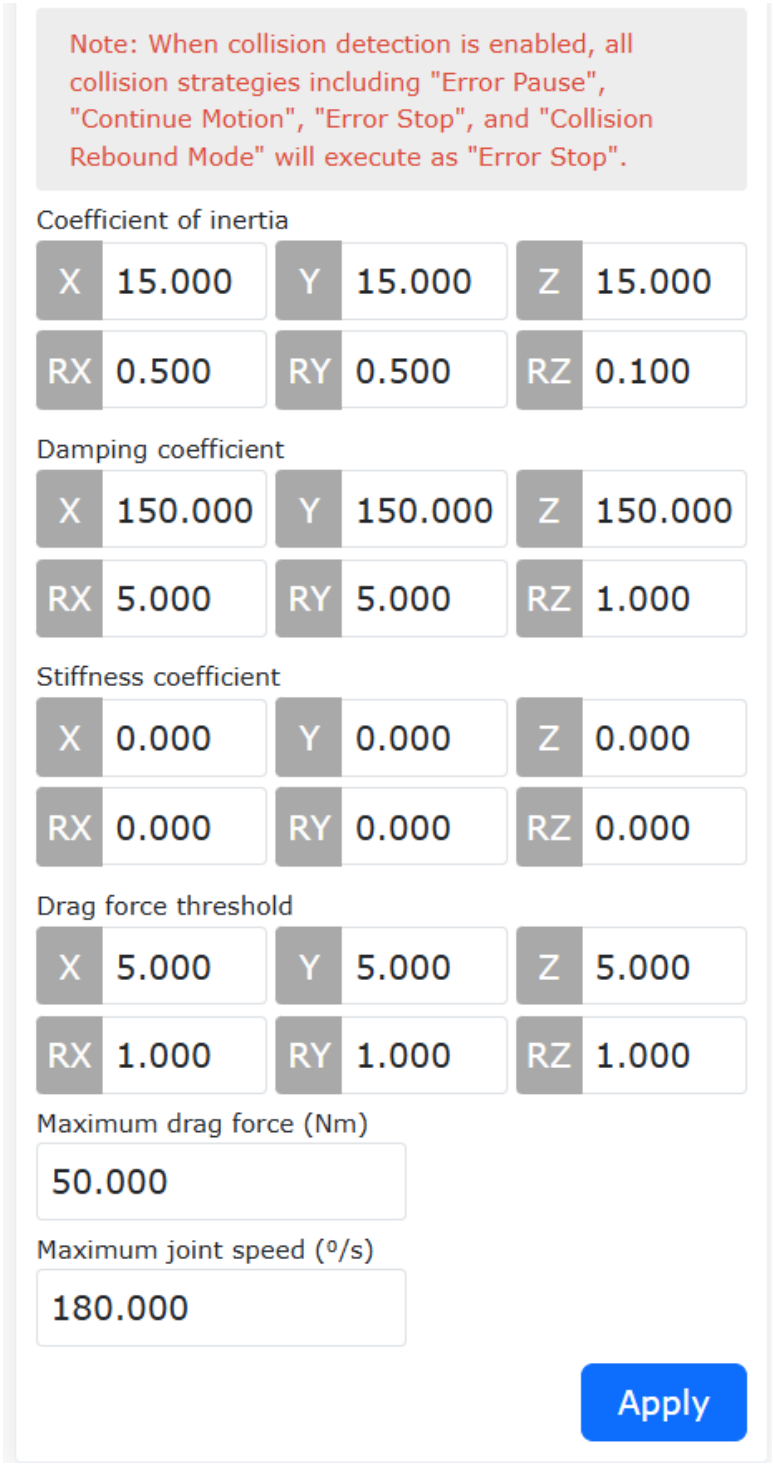
Center point+si



Joint	Center point	Side length of each axis
X	-633.428	50.000
Y	485.353	50.000
Z	161.638	50.000

Apply


Force sensor assisted locking
warning
1、When turning on, please do not apply external force to the sensor
2、Please operate carefully under the guidance of technicians
The force sensor automatically turns on after the error is cleared
Status switch
close
Adaptive
close
Interference zone
close
Singularity strategy
Traverse
Collision detection
open
Note: When collision detection is enabled, all collision strategies including "Error Pause", "Continue Motion", "Error Stop", and "Collision Rebound Mode" will execute as "Error Stop".
Coefficient of inertia
X 15.000
Y 15.000
Z 15.000
RX 0.500
RY 0.500
RZ 0.100


Note: When collision detection is enabled, all collision strategies including "Error Pause", "Continue Motion", "Error Stop", and "Collision Rebound Mode" will execute as "Error Stop".
Coefficient of inertia
X 15.000
Y 15.000
Z 15.000
RX 0.500
RY 0.500
RZ 0.100
Damping coefficient
X 150.000
Y 150.000
Z 150.000
RX 5.000
RY 5.000
RZ 1.000
Stiffness coefficient
X 0.000
Y 0.000
Z 0.000
RX 0.000
RY 0.000
RZ 0.000
Drag force threshold
X 5.000
Y 5.000
Z 5.000
RX 1.000
RY 1.000
RZ 1.000
Maximum drag force (Nm)
50.000
Maximum joint speed (°/s)
180.000
Apply

- Effect: When dragging with the sensor, increasing damping makes dragging difficult, decreasing damping makes dragging the robot too easy (recommended not to set it too small);
- Overall Damping Parameter Range: Translational XYZ: [100-1000]; Rotational RX, RY: [3-50], RZ: [2-10];
- Maximum Drag Force is 50, Maximum Drag Speed is 180.

Stiffness Parameter: Set all to 0;

Drag Force Threshold: Translational XYZ is [5-10]; Rotational RX, RY, RZ is [0.5-5];

Maximum Drag Force: 50;

Maximum Drag Speed: 180;

1.3.8.14.4 Six-Axis Force and Joint Impedance Hybrid Dragging Function

1.3.8.14.4.1 Overview

The six-axis force and joint impedance hybrid dragging function utilizes the force sensor to perceive external forces, allowing the robot to perform assisted dragging in drag mode. Different dragging experiences can be achieved by adjusting the gain coefficients. Joint impedance uses impedance control to limit the drag force.

1.3.8.14.4.2 Force Sensor Installation Configuration and Zeroing Operation

1. Force Sensor Installation Configuration

For detailed operations on force sensor installation configuration, refer to the section above: Virtual Wall Configuration Based on Force Sensor.

2. Force Sensor Zeroing

To facilitate dragging the robot, a drag handle needs to be installed below the sensor, as shown in Figure 1.

Figure 8.14-11 Drag Handle

Step1: Set the tool coordinate system according to the actual handle length, as shown in Figure 2.

Step2: Under the “Initial” -> “Basic” -> “Load” menu, click “Sensor” to enter the Force/Torque Sensor Load interface.

Using the drag button, adjust the robot end effector to face horizontally downwards. Then, click “Load” -> “Sensor Identification” to enter the interface, find the “Record Initial Position” button in the “Sensor Auto Zeroing” section. Then, switch the robot mode to automatic mode and click the “Auto Zero” button. After the program finishes running, the sensor zeroing is complete. Detailed operations are shown in the figures.

Figure 8.14-12 Tool Coordinate System Setup

Figure 8.14-13 Force/Torque Sensor Auto Zeroing



1.3.8.14.4.3 Six-Axis Force and Joint Impedance Hybrid Dragging

1. Assisted Dragging

Step1: Under the "Application" -> "Tool App" menu, click "Drag Lock" to enter the drag lock function interface.

Step2: In the "Six-Axis Force and Joint Impedance Hybrid Dragging" section, set the Control Status to "On", the Impedance Enable Status to "Off", set the Drag Gain, the End Linear Speed to 1000 mm/s, the Angular Speed Limit to 100 °/s, then click the "Apply" button to enable the function. The specific configuration is shown in Figure 4.

Step3: Switch the robot mode to drag mode to drag the robot. The specific effect is: dragging the robot end effector feels easy and provides a good experience; dragging the robot joints feels heavy.

Figure 8.14-14 Configuration Parameters for Six-Axis Force Assisted Dragging

2. Joint Impedance Control

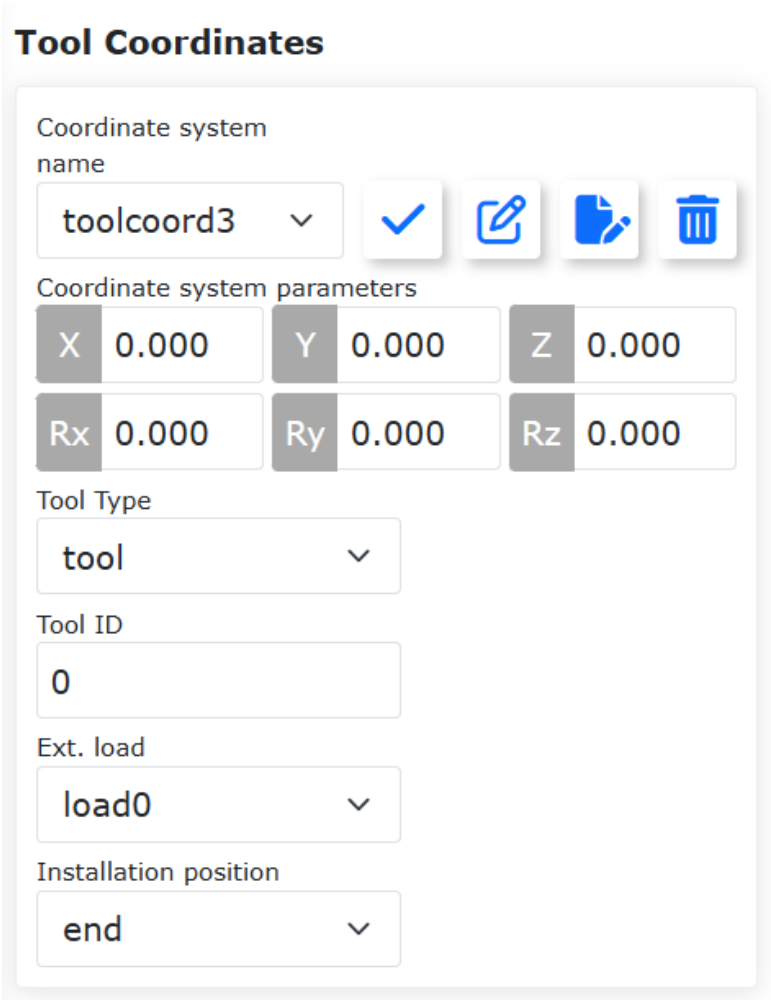
The role of impedance control is to limit the drag force and drag position. Its default status is "Off".

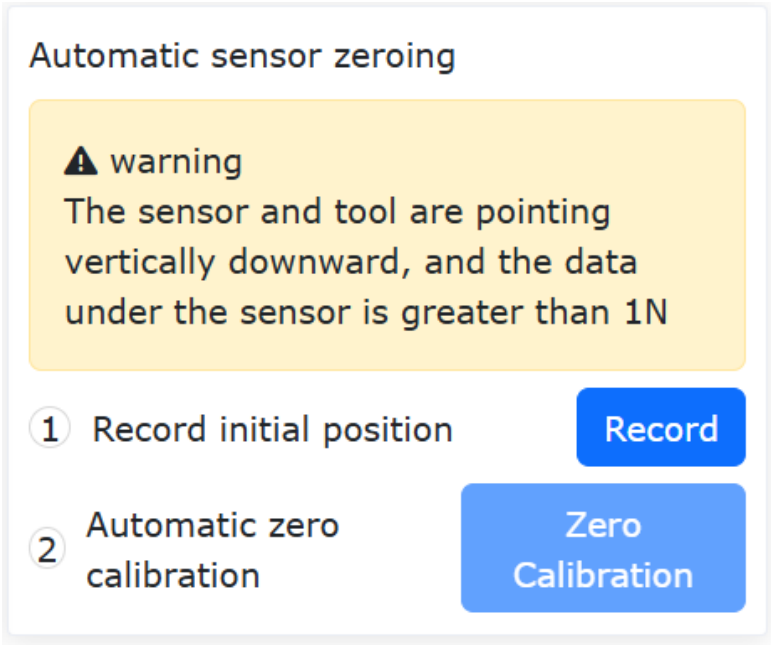
The specific operation is shown in Figure 5. Set the Impedance Enable Status to "On", then set the Damping Coefficient and Stiffness Coefficient as shown in Figure 5. The function of the Stiffness Coefficient is currently not available.

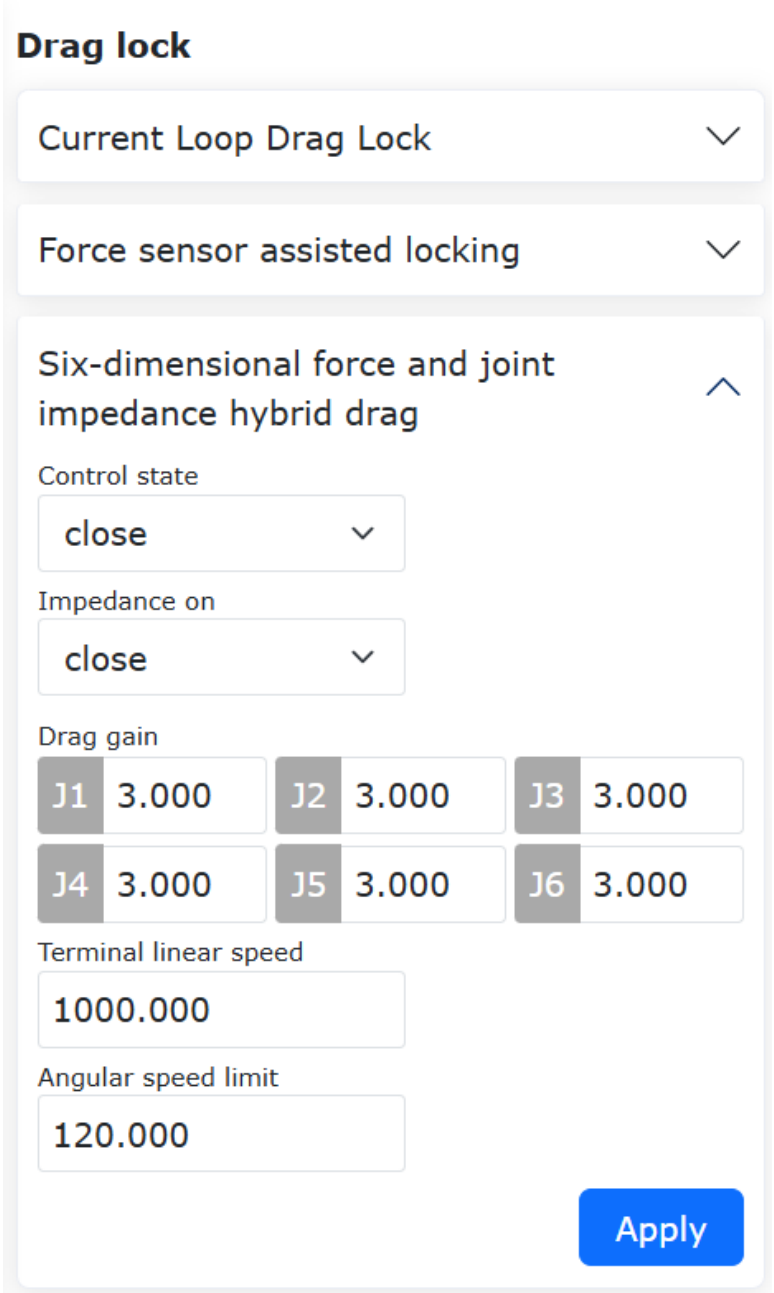
Figure 8.14-15 Configuration Parameters for Joint Impedance

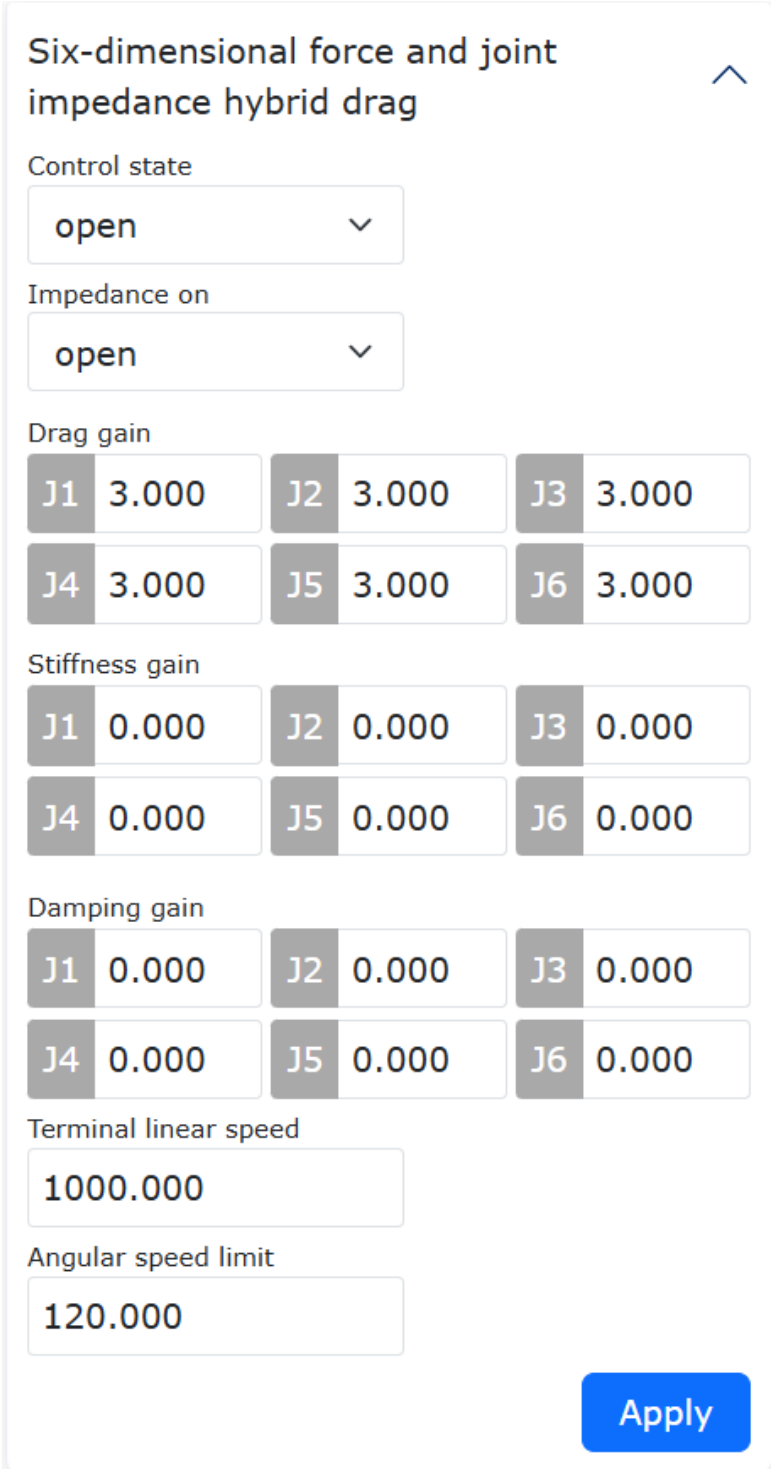
Specific functions of the parameters:

- **Control Status**: After enabling, this function can be used in drag mode.
- **Impedance Enable**: After enabling, stiffness and damping parameters need to be configured. Its role is to limit the drag force and drag position.


Tool Coordinates
Coordinate system name
toolcoord3
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
tool
Tool ID
0
Ext. load
load0
Installation position
end


Automatic sensor zeroing
warning
The sensor and tool are pointing vertically downward, and the data under the sensor is greater than 1N
1 Record initial position
Record
2 Automatic zero calibration
Zero Calibration


Drag lock
Current Loop Drag Lock
Force sensor assisted locking
Six-dimensional force and joint impedance hybrid drag
Control state
close
Impedance on
close
Drag gain
J1 3.000
J2 3.000
J3 3.000
J4 3.000
J5 3.000
J6 3.000
Terminal linear speed
1000.000
Angular speed limit
120.000
Apply


Six-dimensional force and joint impedance hybrid drag
Control state
open
Impedance on
open
Drag gain
J1 3.000
J2 3.000
J3 3.000
J4 3.000
J5 3.000
J6 3.000
Stiffness gain
J1 0.000
J2 0.000
J3 0.000
J4 0.000
J5 0.000
J6 0.000
Damping gain
J1 0.000
J2 0.000
J3 0.000
J4 0.000
J5 0.000
J6 0.000
Terminal linear speed
1000.000
Angular speed limit
120.000
Apply

- **Drag Gain**: Recommended parameter range [0-5]. Setting the parameter to 0 makes the robot undraggable. Setting to 1 shows no improvement in dragging effect. Setting greater than 1 makes dragging light and provides a good experience. The larger the parameter, the easier the dragging.
- **Stiffness Gain**: Set to 0. Its role is to return to the initial position before dragging after dragging.
- **Damping Gain**: Its role is to limit the drag force. Parameter range for axes 1-3 is [0-0.5], for axes 4-5 is [0-0.1]; for axis 6 it is [0-0.05].
- **End Linear Speed**: 1000 mm/s. When the end linear speed limit is exceeded, the robot switches mode to manual mode and prompts TCP overspeed.
- **Angular Speed Limit**: 100 °/s. When the angular speed limit is exceeded, the robot switches mode to manual mode and prompts TCP overspeed.

1.3.8.15 Extended Axis with Laser Point Tracking Function

1.3.8.15.1 Robot Extended Axis with Laser Point Tracking System Composition

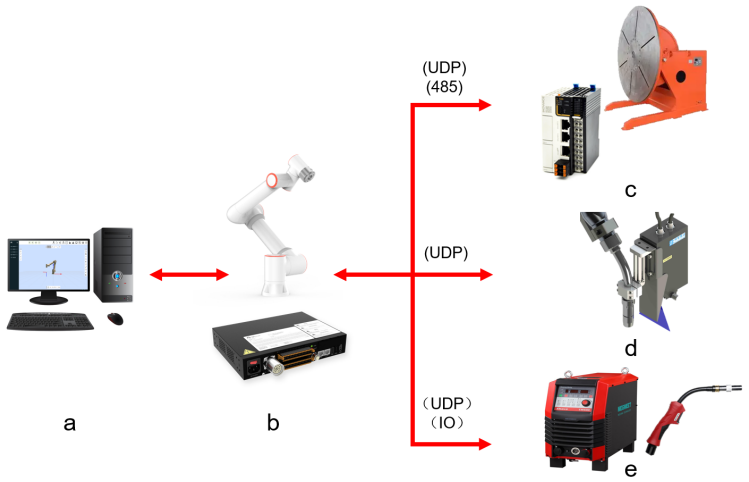


Figure 8.15-1 Robot Extended Axis with Laser Point Tracking System Composition

In the system, (a) is the computer, (b) is the robot and its control box, (c) is the positioner and drive equipment, (d) is the weld seam tracking laser sensor, (e) is the welder and supporting equipment.

Figure 8.15-2 Peripheral Installation Diagram

The weld seam tracking laser sensor and welding torch (b) are installed on the end flange of the robot (a), and the positioner (c) is fixedly installed outside the robot.

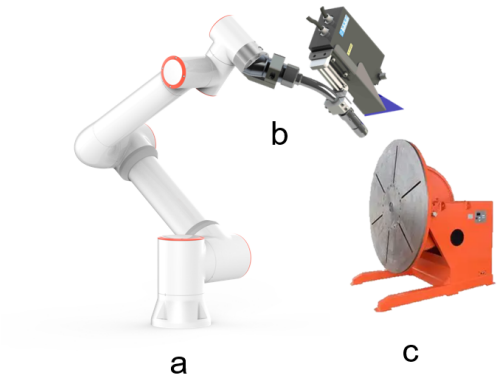
1.3.8.15.2 Extended Axis Communication Configuration

The communication methods between the robot and the extended axis include using UDP or RS485.

Figure 8.15-3 Extended Axis Configuration Page

Click “Initial” -> “Peripherals” -> “Extended Axis” on the robot operation interface to enter the extended axis configuration page. Taking connecting the robot to a PLC via UDP communication as an example, click the “UDP Communication” icon to enter the UDP communication extended axis configuration page.

Figure 8.15-4 UDP Communication Configuration Interface



On the UDP communication extended axis configuration page, you can select the corresponding extended axis number, connect and configure UDP communication parameters (address, port, cycle, packet loss detection, etc.), and the extended axis positioning completion time.

The extended axis configuration content is not the focus of this function introduction. For detailed configuration, refer to the corresponding part of the user manual.

1.3.8.15.3 Weld Seam Tracking Laser Sensor Connection Configuration

Connect the weld seam tracking laser sensor through the following configuration page:

Figure 8.15-5 Laser Sensor Connection and Configuration Page

Click “Initial” -> “Peripherals” -> “Line Laser Sensor” -> “Adapted Devices” to enter the configuration page. The configuration page includes “Sensor Configuration”, “Communication Configuration and Loading”, “Base Calculation”. Click “Sensor Configuration” to set the sensor input filter parameters. Click “Communication Configuration and Loading” to input the corresponding communication parameters to connect the laser sensor.

The laser sensor configuration content is not the focus of this function introduction. For detailed configuration, refer to the corresponding part of the user manual.

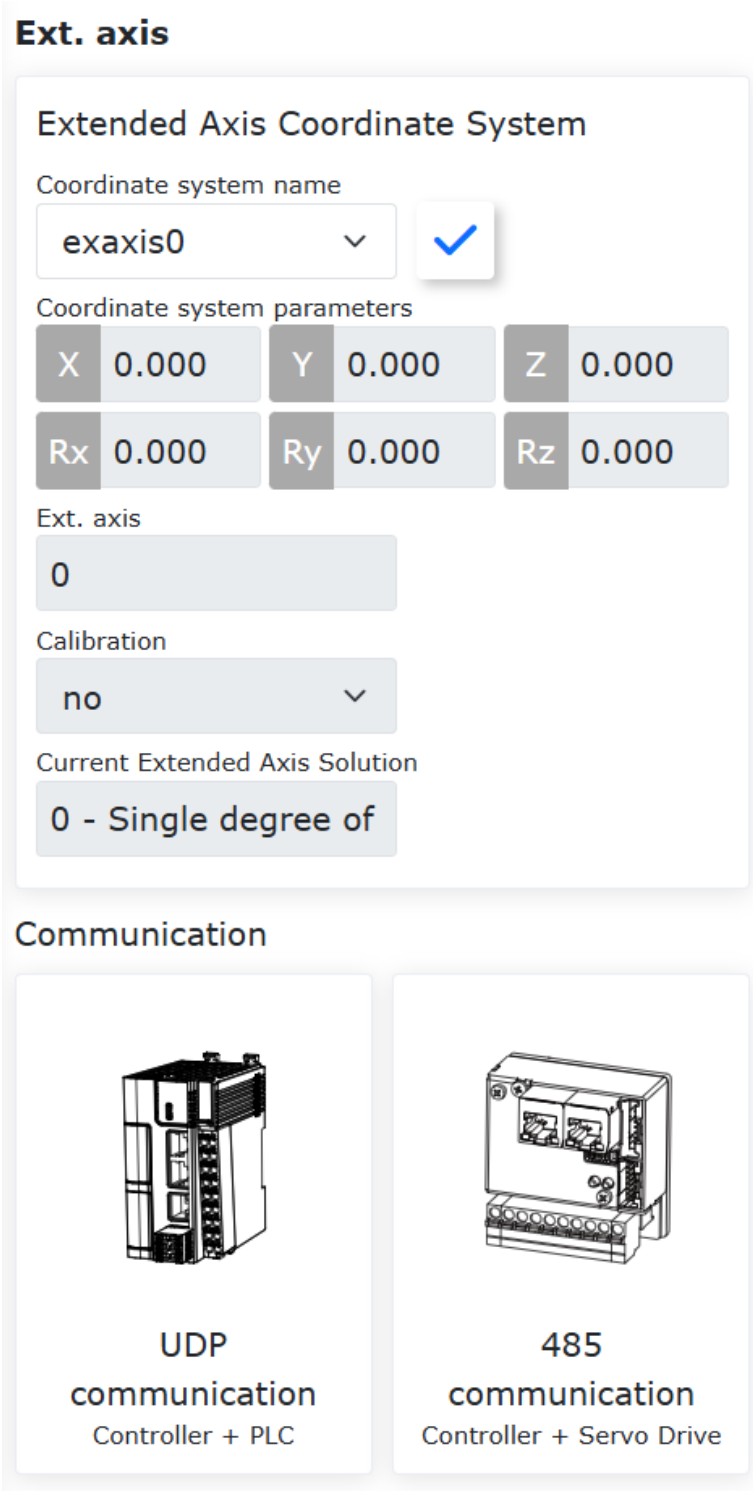
1.3.8.15.4 Welder Connection Configuration

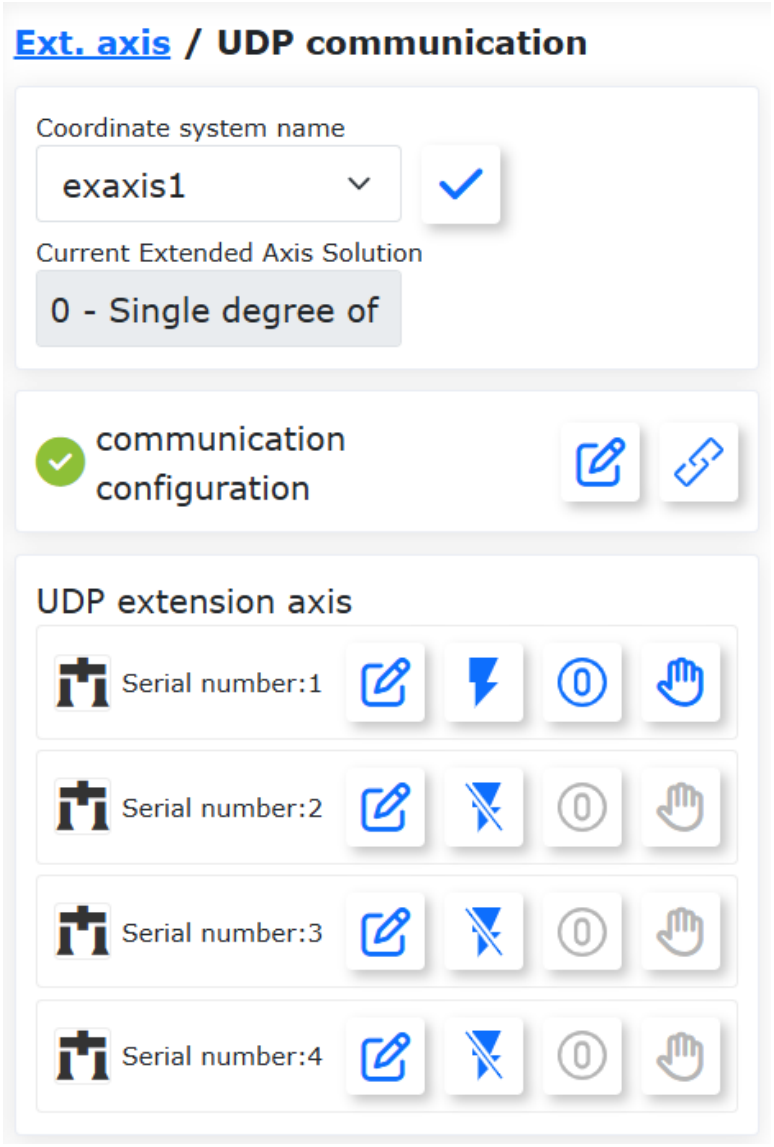
Configure the welder through the following configuration page:

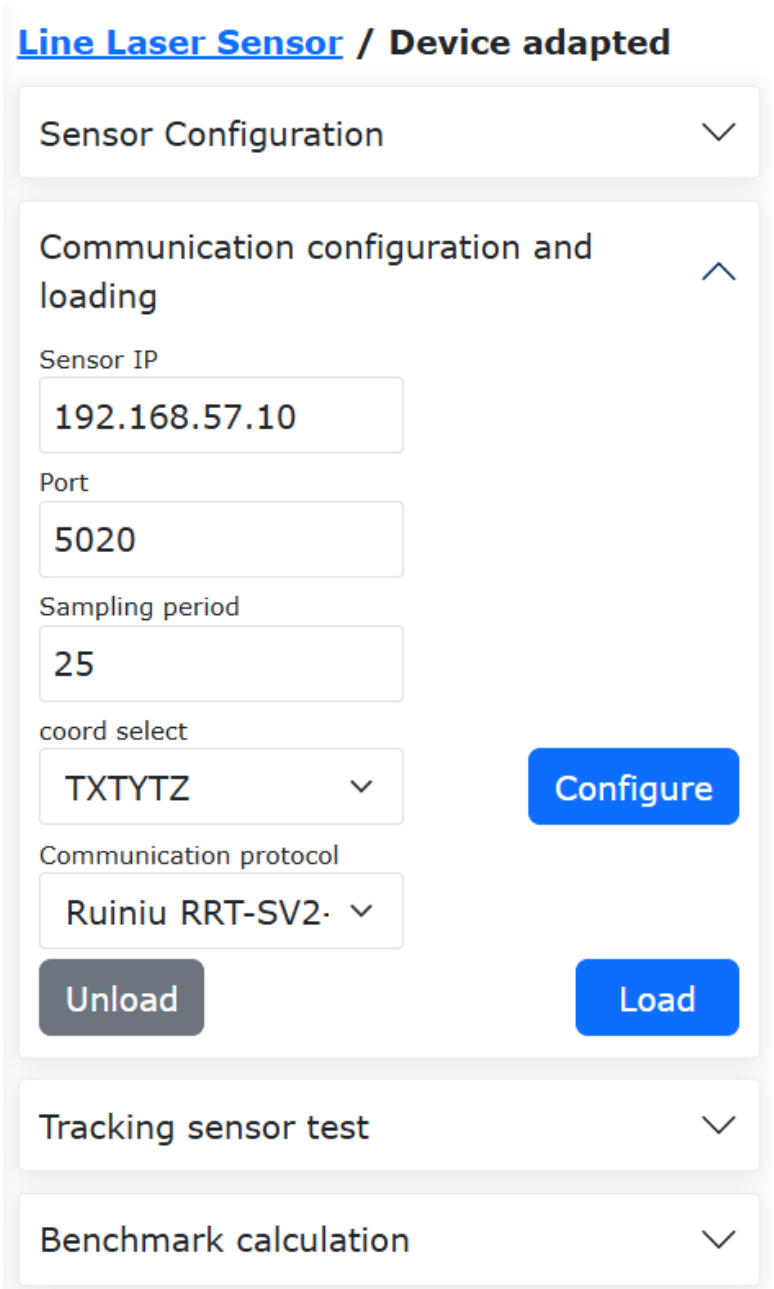
Figure 8.15-6 Welder Configuration Page

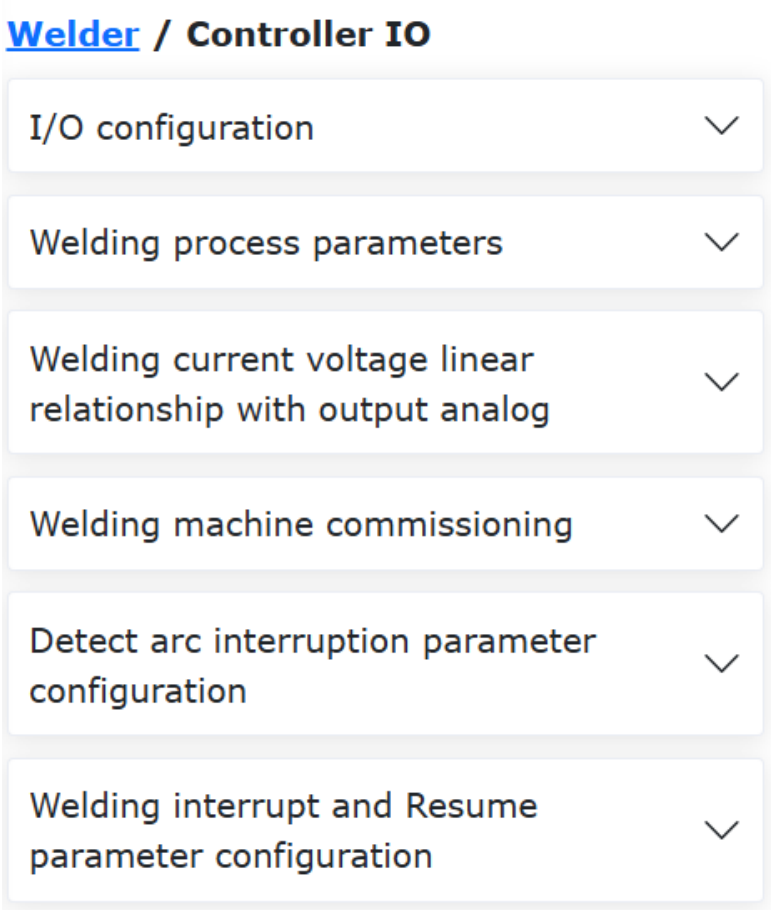
Welder communication can use IO communication or RS485 communication. Click “Initial”, “Peripherals”, “Welder” to enter the configuration and connection interface, where you can configure modules such as “Control Type”, “Signal Corresponding IO”, “Welding Process Parameters”, “Welder Debugging”, etc.

The welder configuration content is not the focus of this function introduction. For detailed configuration, refer to the corresponding part of the user manual.


Ext. axis
Extended Axis Coordinate System
Coordinate system name
exaxis0
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Ext. axis
0
Calibration
no
Current Extended Axis Solution
0 - Single degree of
Communication
UDP communication
Controller + PLC
485 communication
Controller + Servo Drive


Ext. axis / UDP communication
Coordinate system name
exaxis1
Current Extended Axis Solution
0 - Single degree of
communication configuration
UDP extension axis
Serial number:1
Serial number:2
Serial number:3
Serial number:4


Line Laser Sensor / Device adapted
Sensor Configuration
Communication configuration and loading
Sensor IP
192.168.57.10
Port
5020
Sampling period
25
coord select
TXTYTZ
Configure
Communication protocol
Ruiniu RRT-SV2·
Unload
Load
Tracking sensor test
Benchmark calculation


Welder / Controller IO
I/O configuration
Welding process parameters
Welding current voltage linear relationship with output analog
Welding machine commissioning
Detect arc interruption parameter configuration
Welding interrupt and Resume parameter configuration

1.3.8.15.5 Tool Coordinate System and Laser Sensor Coordinate System Calibration

After installing the welding torch on the robot end, calibrate the extrinsic parameters between the welding torch and the laser sensor:

Figure 8.15-7 Tool Coordinate System Configuration Page

Click “Initial”, “Basic”, “Coordinate System”, “Tool” to enter the tool coordinate system setup page.

Figure 8.15-8 Selecting the 6-Point Method to Calibrate the Welding Torch

Select an empty coordinate system, select the tool type as “Tool”, and choose the 6-point method to calibrate the welding torch tool.

Figure 8.15-9 Selecting the 5-Point Method to Calibrate the Laser Sensor

Select an empty coordinate system, select the tool type as “Sensor”, and choose the 5-point method to calibrate the laser sensor.

The tool coordinate system and laser sensor coordinate system calibration content is not the focus of this function introduction. For detailed calibration methods, refer to the corresponding part of the user manual.

1.3.8.15.6 Extended Axis and Laser Point Tracking Function

Extended axis and laser point tracking are divided into two methods: laser data with transformation executes a “record first, then replay” tracking strategy, and laser data without transformation executes a “record while replaying” tracking strategy.

1.3.8.15.6.1 Extended Axis Coordinate System Calibration

Calibrating the extended axis coordinate system is required when using the extended axis coordinate system to achieve synchronized laser tracking between the extended axis and the robot.

Figure 8.15-10 Extended Axis Coordinate System Setup Page

Click “Initial” -> “Peripherals” -> “Extended Axis” to enter the extended axis coordinate system setup interface. Select the extended axis number that needs to be set, click the edit button, select “4-Single DOF Positioner” and save.

Figure 8.15-11 Extended Axis Calibration Page

When calibrating the extended axis, pay attention to select “Robot Position Relative to Extended Axis” as “Outside Extended Axis”. For the case of a positioner, select the 4-point method for calibration.

The extended axis calibration content is not the focus of this function introduction. For detailed calibration methods, refer to the corresponding part of the user manual.

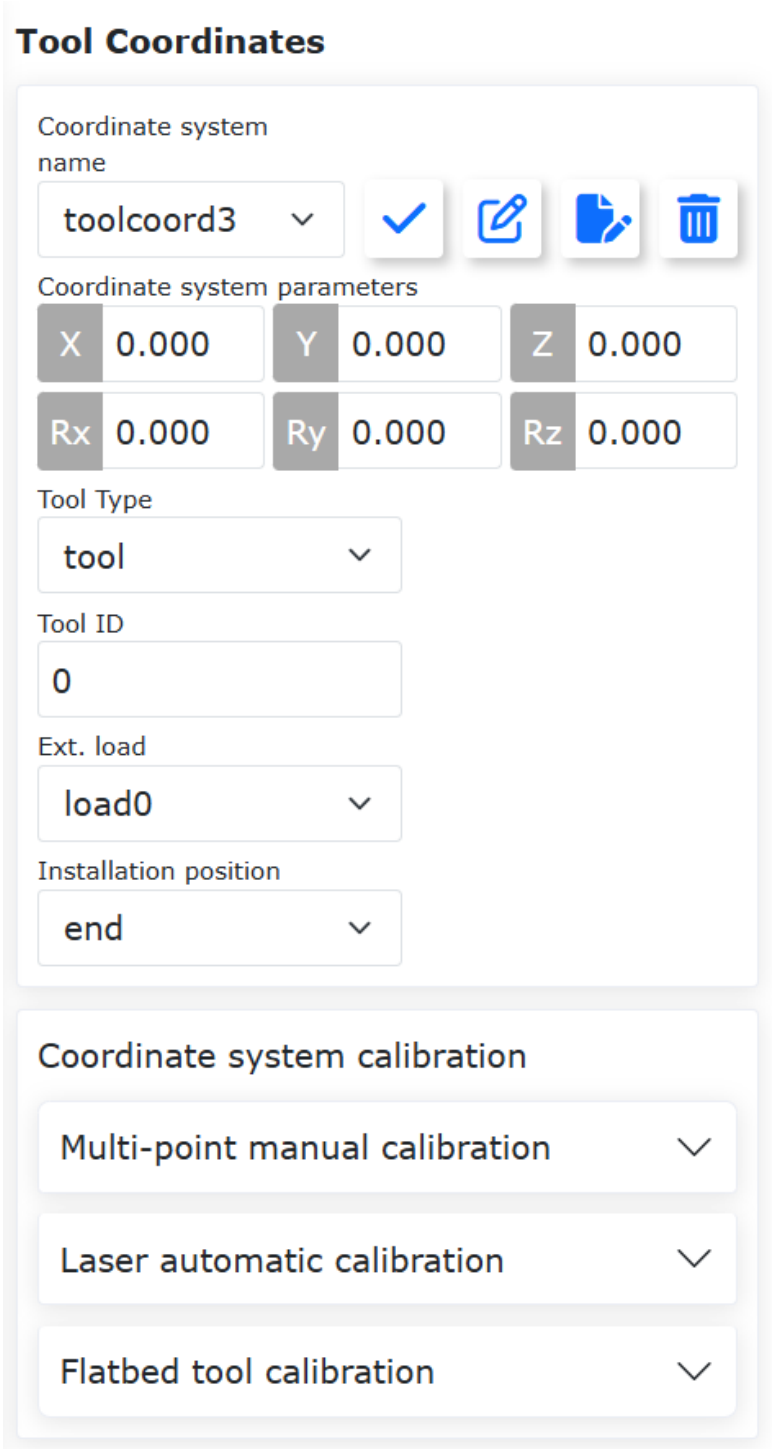
1.3.8.15.6.2 Synchronized Laser Tracking of Extended Axis and Robot

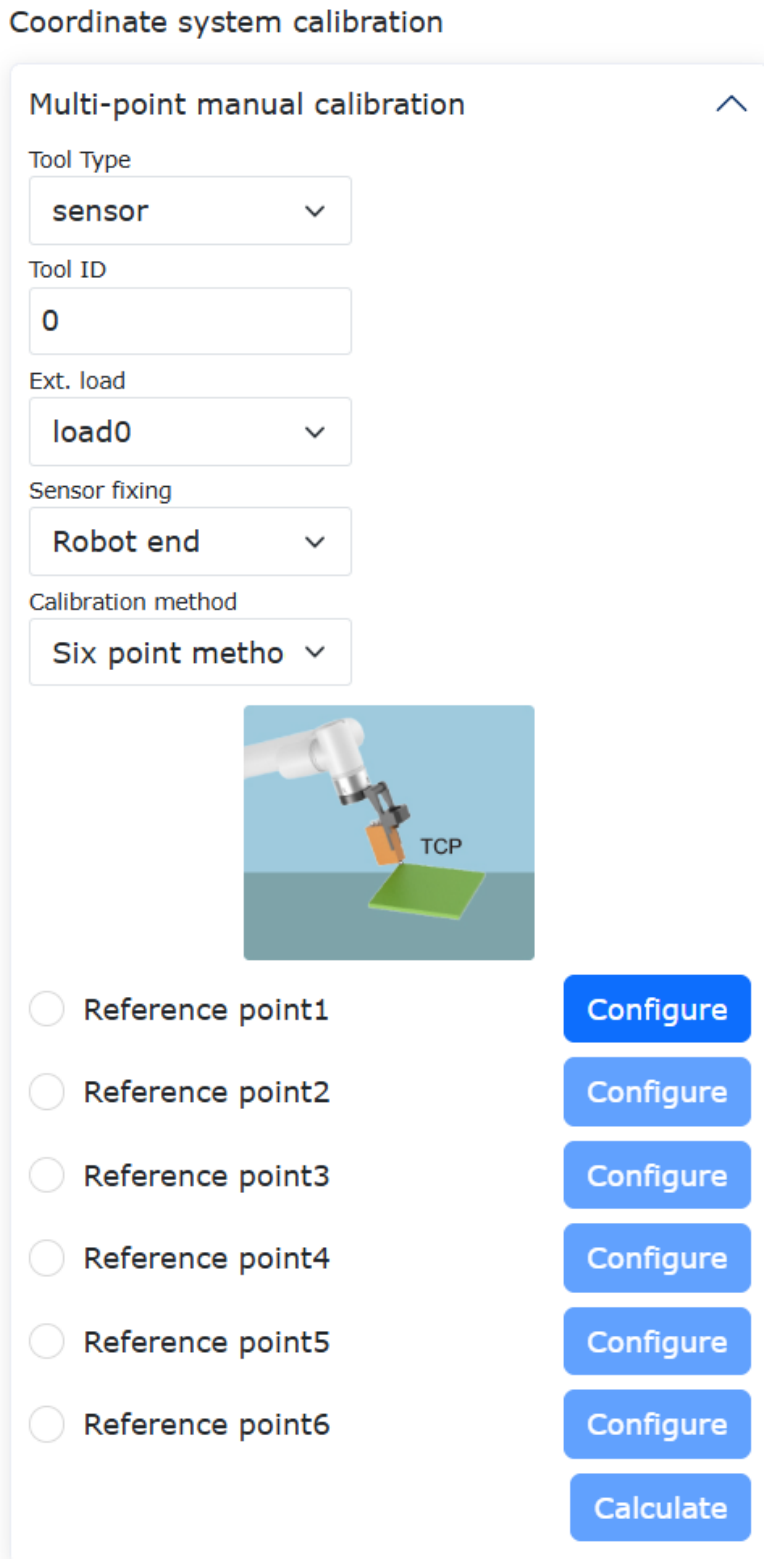
1.3.8.15.6.3 Laser Data with Transformation Method

Synchronized laser tracking of the extended axis and robot in the base coordinate system does not require external axis calibration. Other function settings and composition are consistent with synchronized tracking in the extended axis coordinate system.

First, configure the laser tracking data, setting the laser tracker data to the type with transformation.

Figure 8.15-12 Setting Laser Data to Type with Transformation


Tool Coordinates
Coordinate system name
toolcoord3
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
tool
Tool ID
0
Ext. load
load0
Installation position
end
Coordinate system calibration
Multi-point manual calibration
Laser automatic calibration
Flatbed tool calibration


Coordinate system calibration
Multi-point manual calibration
Tool Type
sensor
Tool ID
0
Ext. load
load0
Sensor fixing
Robot end
Calibration method
Six point metho
TCP
Reference point1
Configure
Reference point2
Configure
Reference point3
Configure
Reference point4
Configure
Reference point5
Configure
Reference point6
Configure
Calculate

Coordinate system calibration

Multi-point manual calibration

Tool Type

sensor

Tool ID

0

Ext. load

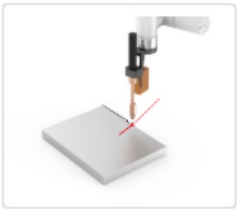
load0

Sensor fixing

Robot end

Calibration method

Five point meth



	Sensor output value		
Serial No	X	Y	Z
1			
2			
3			
4			
5			

Reference point1 Configure

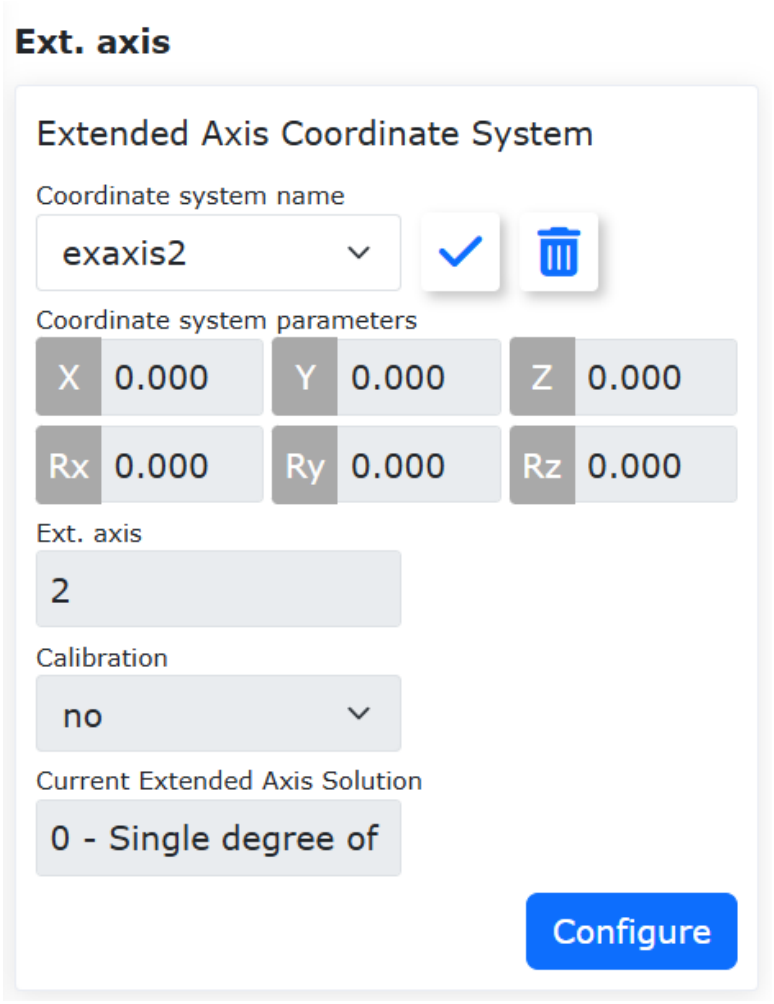
Reference point2 Configure

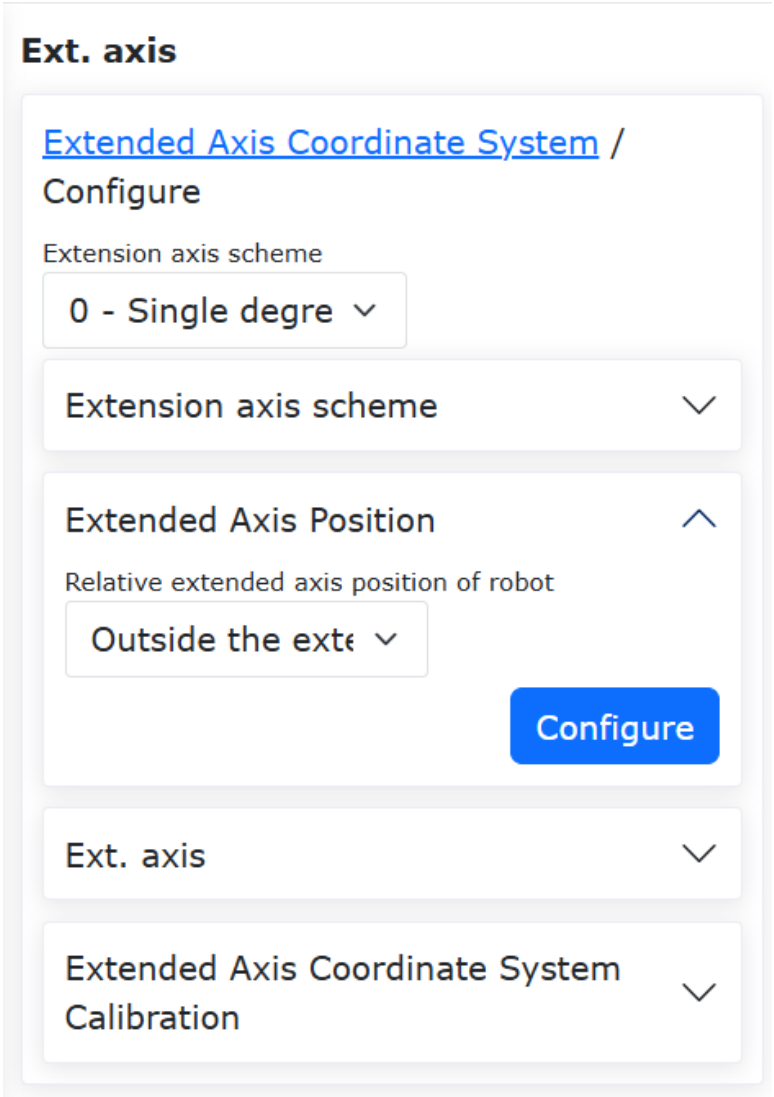
Reference point3 Configure

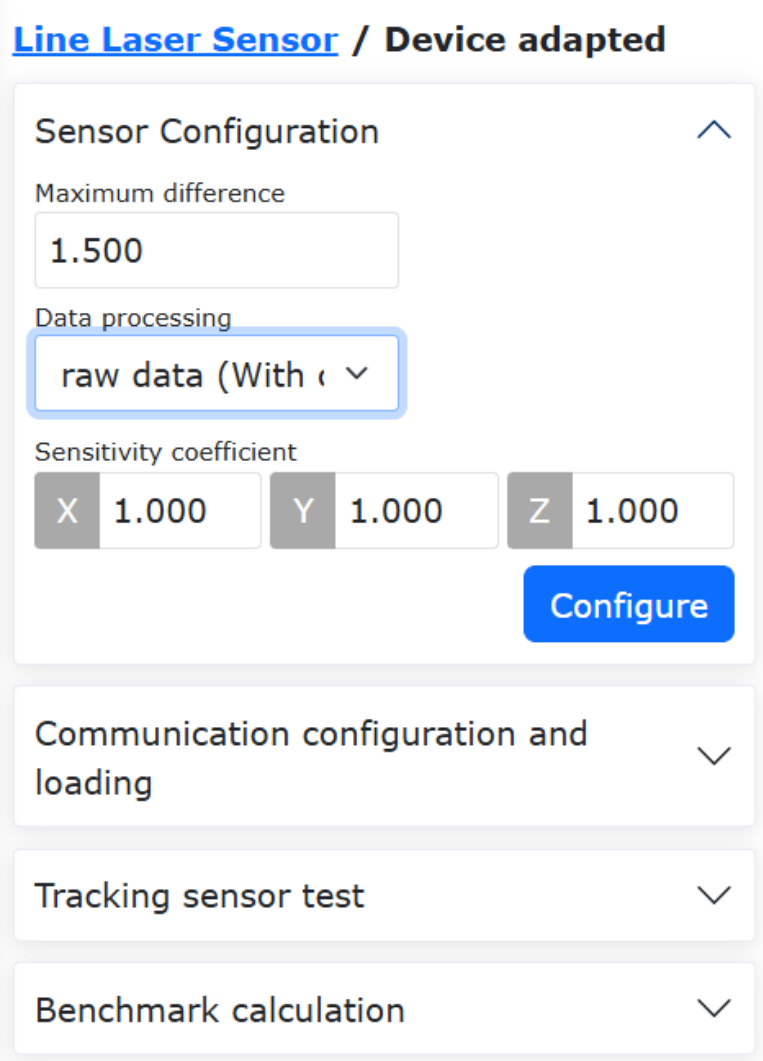
Reference point4 Configure

Reference point5 Configure

Calculate


Ext. axis
Extended Axis Coordinate System
Coordinate system name
exaxis2
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Ext. axis
2
Calibration
no
Current Extended Axis Solution
0 - Single degree of
Configure


Ext. axis
Extended Axis Coordinate System /
Configure
Extension axis scheme
0 - Single degre
Extension axis scheme
Extended Axis Position
Relative extended axis position of robot
Outside the exte
Configure
Ext. axis
Extended Axis Coordinate System Calibration


Line Laser Sensor / Device adapted
Sensor Configuration
Maximum difference
1.500
Data processing
raw data (With
Sensitivity coefficient
X 1.000
Y 1.000
Z 1.000
Configure
Communication configuration and loading
Tracking sensor test
Benchmark calculation

Click “Initial”, “Peripherals”, “Tracking”, “Sensor”, click “Sensor Configuration” in the page dropdown, and adjust “Data Processing” to the data type with transformation.

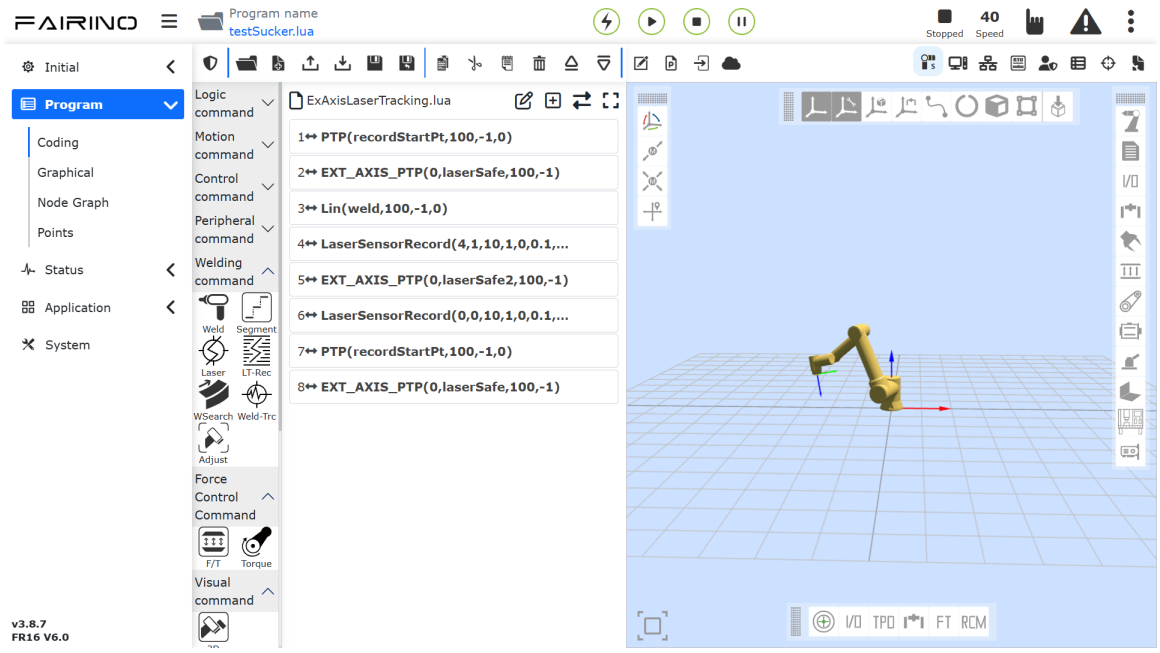


Figure 8.15-13 Laser Tracking Function Page

This function is implemented through a combination of multi-functional modules. The main functional modules are included within the “Laser Tracking” function. Click “Teach Program” -> “Program Programming” -> “Laser Tracking” to enter the laser tracking page, or click “Laser Record” to directly enter the recording page.

Figure 8.15-14 Adding the Start Recording Laser Data Instruction

Add the start recording laser data instruction after the extended axis moves to the welding start point.

Figure 8.15-15 Adding the Stop Recording Laser Data Instruction

Add the stop recording laser data instruction after the extended axis moves to the welding end point.

After the robot finishes recording the movement trajectory of the weld seam during the extended axis motion in place, the extended axis can be returned to the welding start point to prepare for synchronized tracking welding.

At the beginning of welding, the welding torch needs to be moved to the starting point position of the laser sensor recorded data. Add the Move to Welding Point instruction:

Figure 8.15-16 Adding the Move to Welding Point Instruction

Click “Teach Program” -> “Program Programming” -> “Laser Record” button, select “Move to Welding Point”, set the motion mode and speed, click the “Start Point” button and apply.

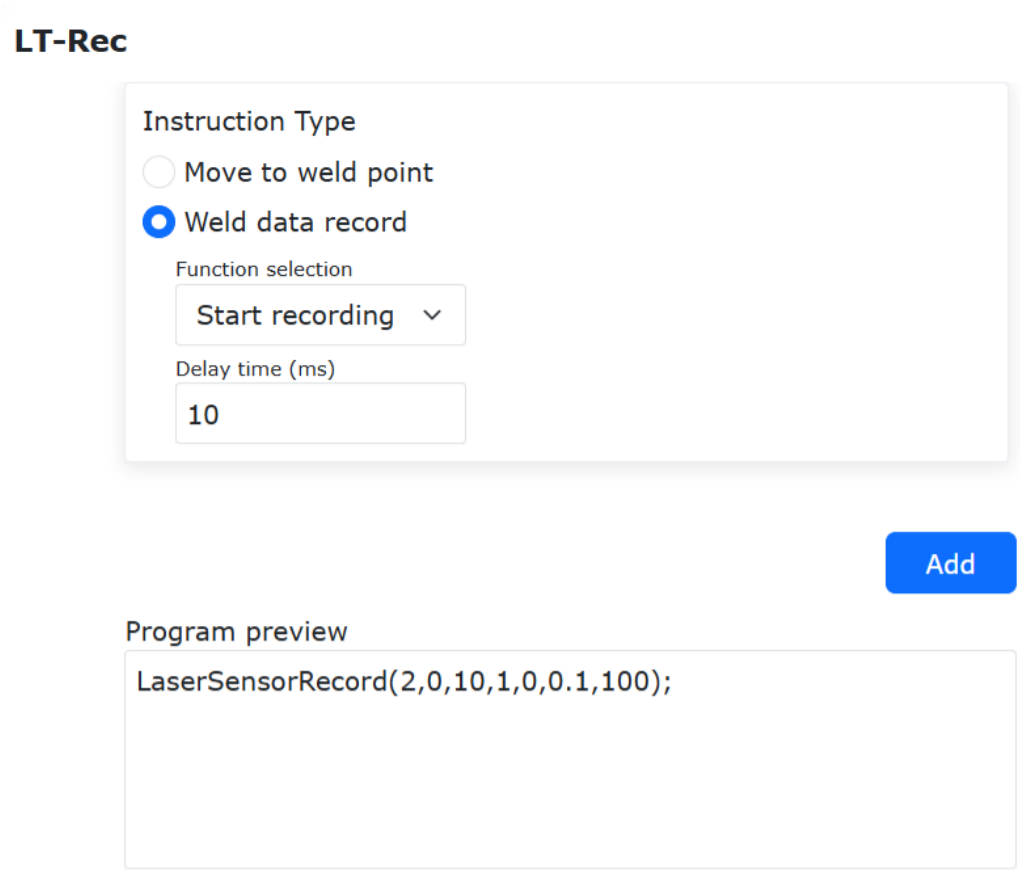
Figure 8.15-17 Adding the Trajectory Replay Instruction for Recorded Laser Data

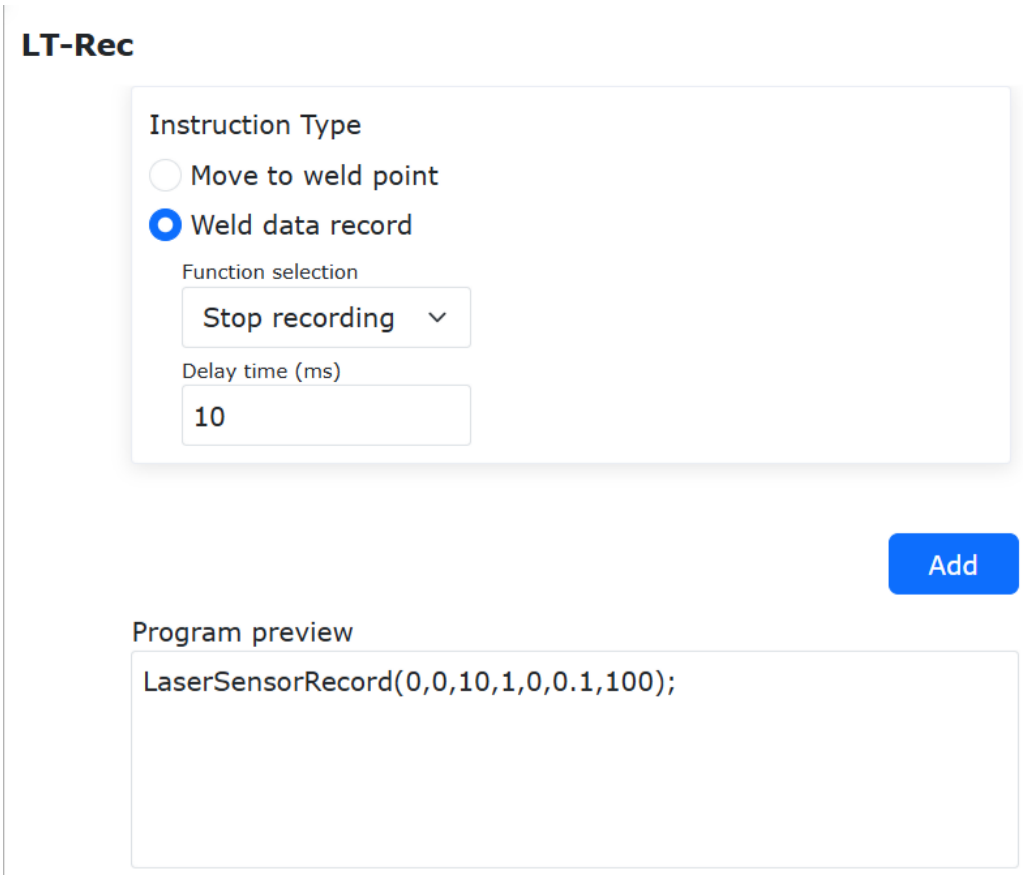
In the “Laser Tracking” page, select the “Data Record” -> “Trajectory Replay” instruction, click “Add” and apply. In the instruction, the waiting time defaults to 0ms, and the speed is the ratio of the replay speed to the recording speed, recommended to be greater than 50%.

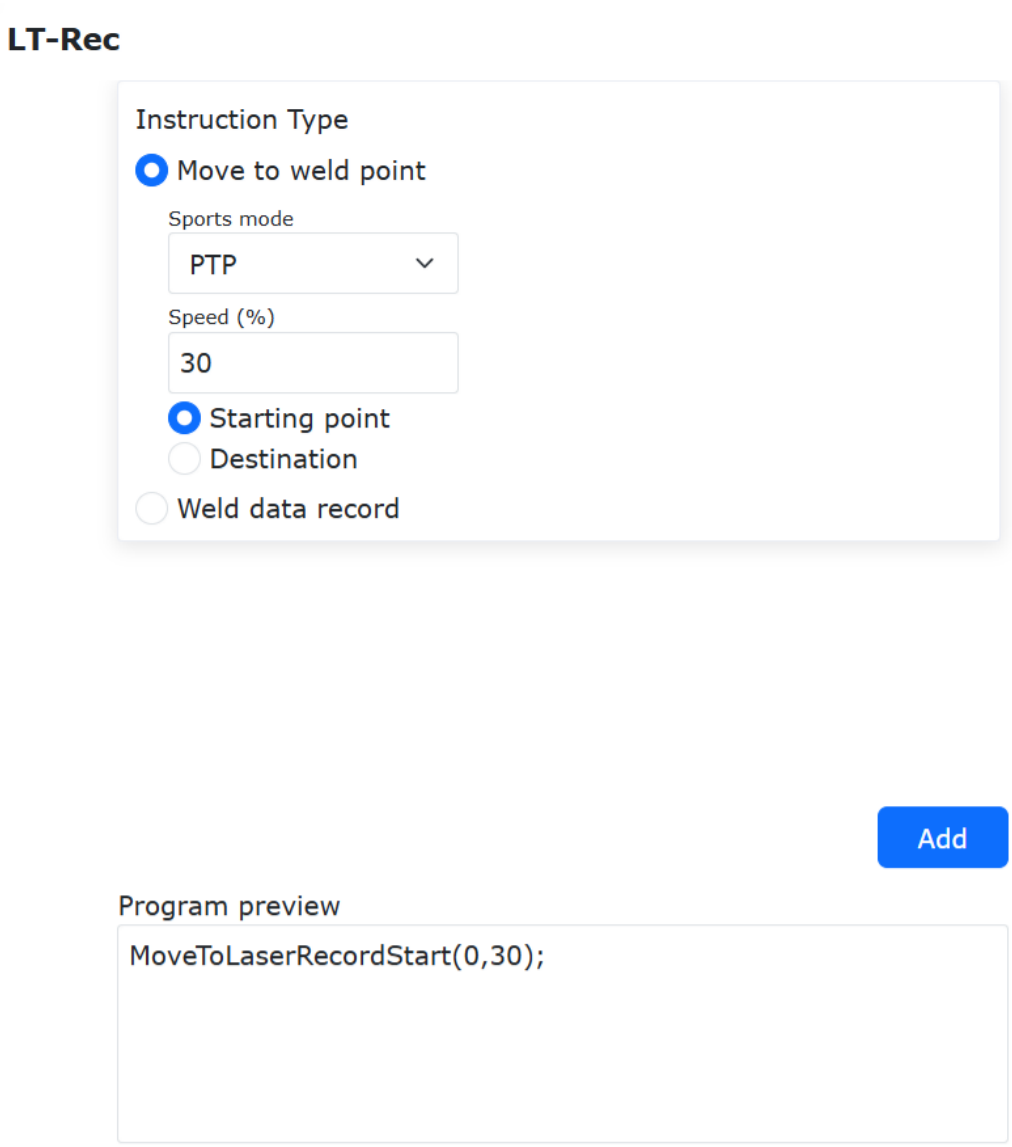
Adding an extended axis motion instruction after the “Trajectory Replay” instruction enables synchronized motion of the extended axis and robot laser tracking.

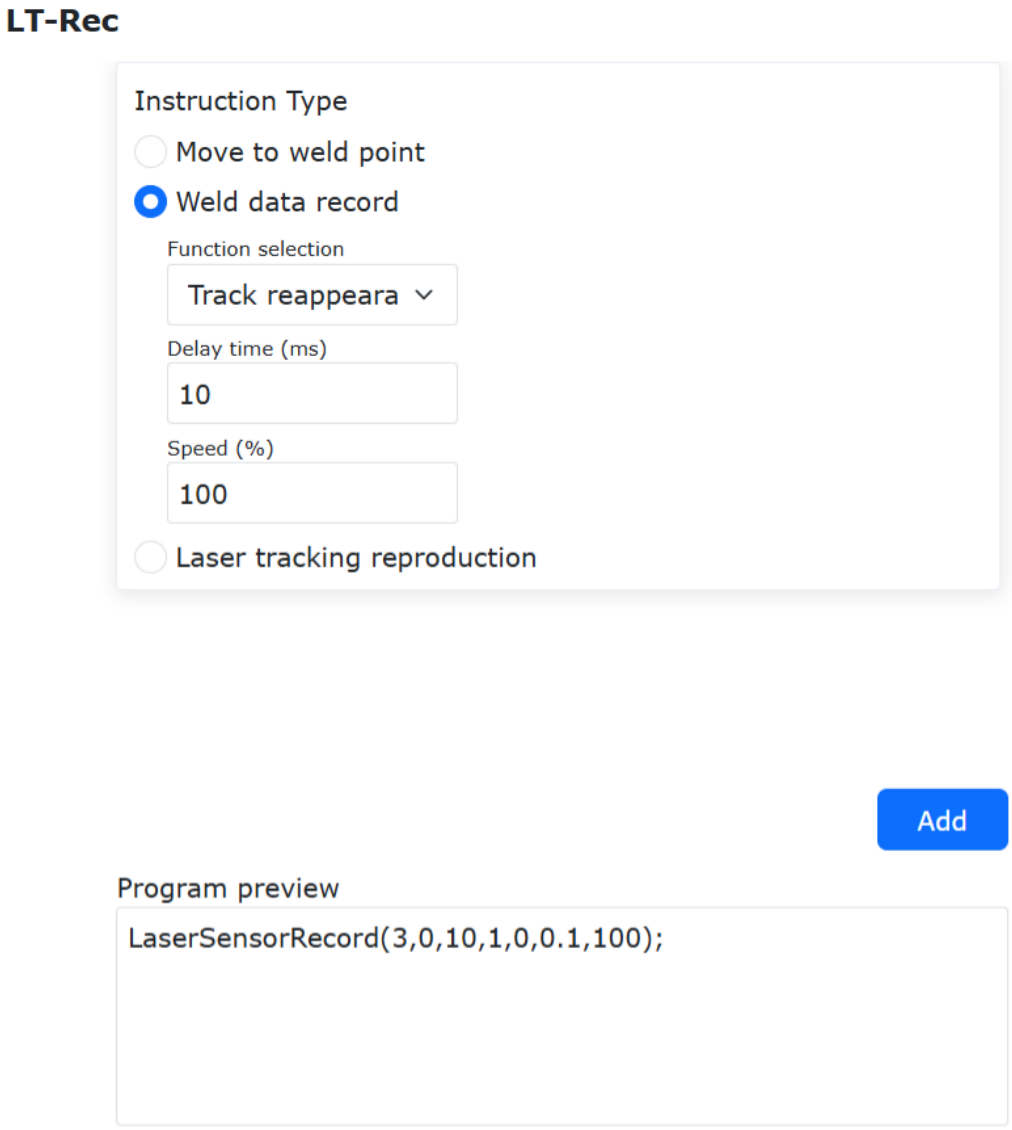
The following is a typical LUA program for extended axis with laser data transformation point tracking:

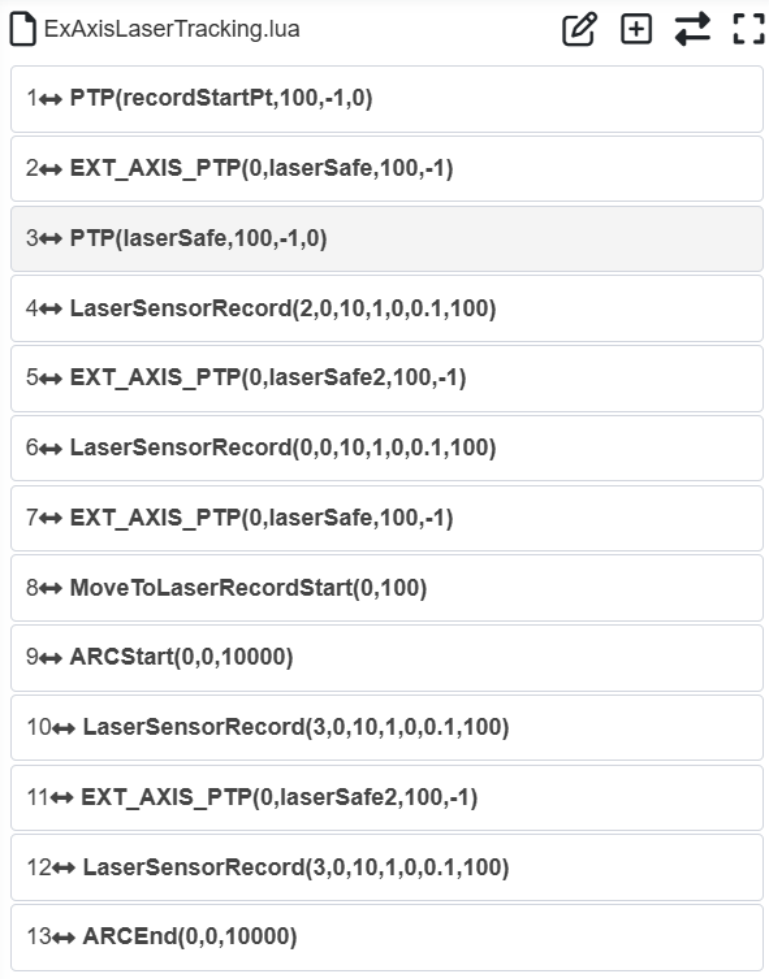
Figure 8.15-18 Example Program for Extended Axis with Laser Data Transformation Point Tracking


LT-Rec
Instruction Type
Move to weld point
Weld data record
Function selection
Start recording
Delay time (ms)
10
Add
Program preview
LaserSensorRecord(2,0,10,1,0,0.1,100);


LT-Rec
Instruction Type
Move to weld point
Weld data record
Function selection
Stop recording
Delay time (ms)
10
Add
Program preview
LaserSensorRecord(0,0,10,1,0,0.1,100);


LT-Rec
Instruction Type
Move to weld point
Sports mode
PTP
Speed (%)
30
Starting point
Destination
Weld data record
Add
Program preview
MoveToLaserRecordStart(0,30);


LT-Rec
Instruction Type
Move to weld point
Weld data record
Function selection
Track reappeara
Delay time (ms)
10
Speed (%)
100
Laser tracking reproduction
Add
Program preview
LaserSensorRecord(3,0,10,1,0,0.1,100);


ExAxisLaserTracking.lua
1↔ PTP(recordStartPt,100,-1,0)
2↔ EXT_AXIS_PTP(0,laserSafe,100,-1)
3↔ PTP(laserSafe,100,-1,0)
4↔ LaserSensorRecord(2,0,10,1,0,0.1,100)
5↔ EXT_AXIS_PTP(0,laserSafe2,100,-1)
6↔ LaserSensorRecord(0,0,10,1,0,0.1,100)
7↔ EXT_AXIS_PTP(0,laserSafe,100,-1)
8↔ MoveToLaserRecordStart(0,100)
9↔ ARCStart(0,0,10000)
10↔ LaserSensorRecord(3,0,10,1,0,0.1,100)
11↔ EXT_AXIS_PTP(0,laserSafe2,100,-1)
12↔ LaserSensorRecord(3,0,10,1,0,0.1,100)
13↔ ARCEnd(0,0,10000)

The robot executes the "record first, then replay" process. It first records the changing trajectory of the workpiece weld seam during extended axis motion, and then during welding, the extended axis and trajectory replay are executed synchronously.

1.3.8.15.6.4 Laser Data without Transformation Method

Using the laser data without transformation method for point tracking does not require calibration of the extended axis coordinate system.

Set the laser tracking sensor data to the type without transformation.

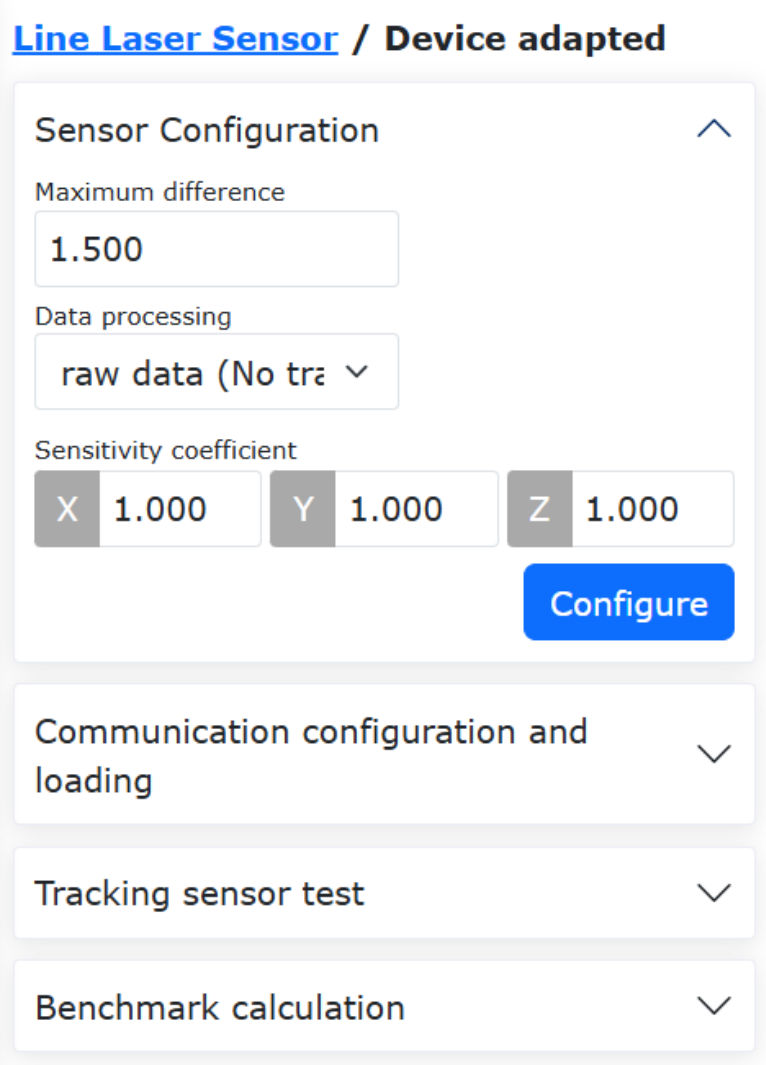


Figure 8.15-19 Setting Laser Data to Type without Transformation

Click "Initial" -> "Peripherals" -> "Line Laser Sensor", click "Sensor Configuration" in the page dropdown, and adjust "Data Processing" to the data type without transformation.

Figure 8.15-20 Laser Tracking Function Page

Click "Teach Program" -> "Program Programming" -> "Laser Tracking" to enter the laser tracking page, or click "Laser Record" to directly enter the recording page.

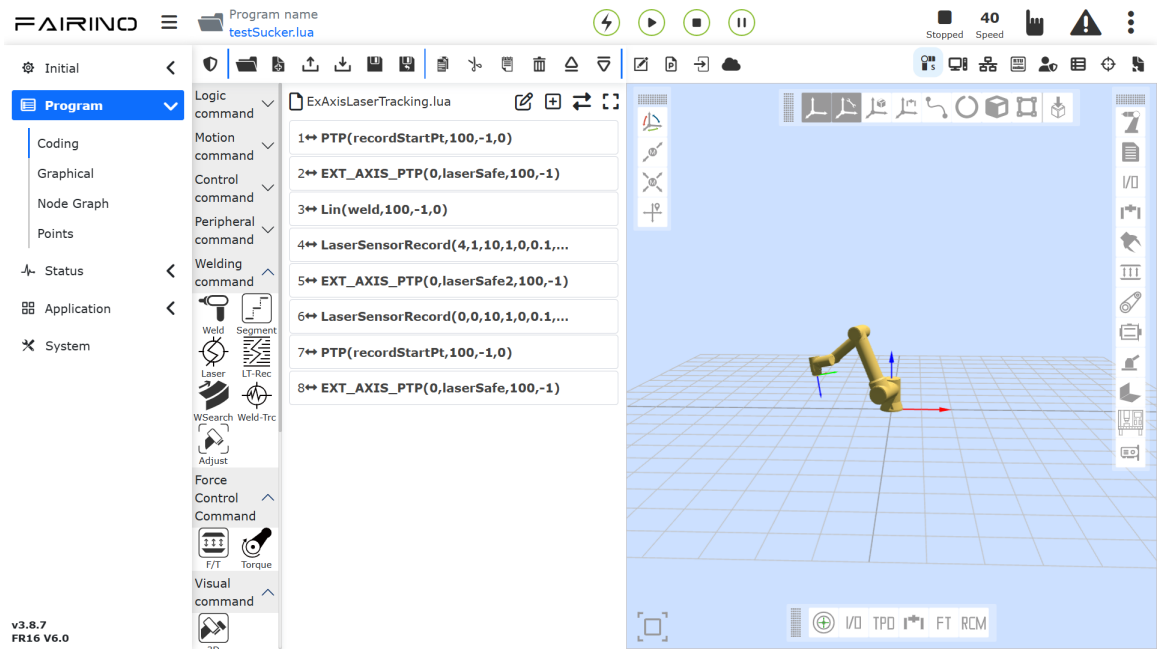


Figure 8.15-21 Adding the Record While Replaying Instruction

In the “Laser Record” page, select the “Record While Replaying” instruction, click “Add” and apply. In the instruction, you can choose “Delay Time” or “Delay Distance” (distance is recommended). The compensation sensitivity coefficient is adjusted according to the actual sensor laser data; a lower value means lower adjustment sensitivity and better anti-interference. The replay speed defaults to 100%.

Adding an extended axis motion instruction after the “Record While Replaying” instruction enables synchronized motion of the extended axis and robot laser tracking.

The following is a typical LUA program for extended axis with laser data without transformation point tracking:

Figure 8.15-22 Example Program for Extended Axis with Laser Data without Transformation Point Tracking

After aligning the welding torch with the offset from the front-mounted laser, the robot’s extended axis moves and executes the “record while replaying” process. The front-mounted laser tracker first records the changing trajectory of the workpiece weld seam during extended axis motion, and then adjusts at the welding torch after the set delay distance or time.

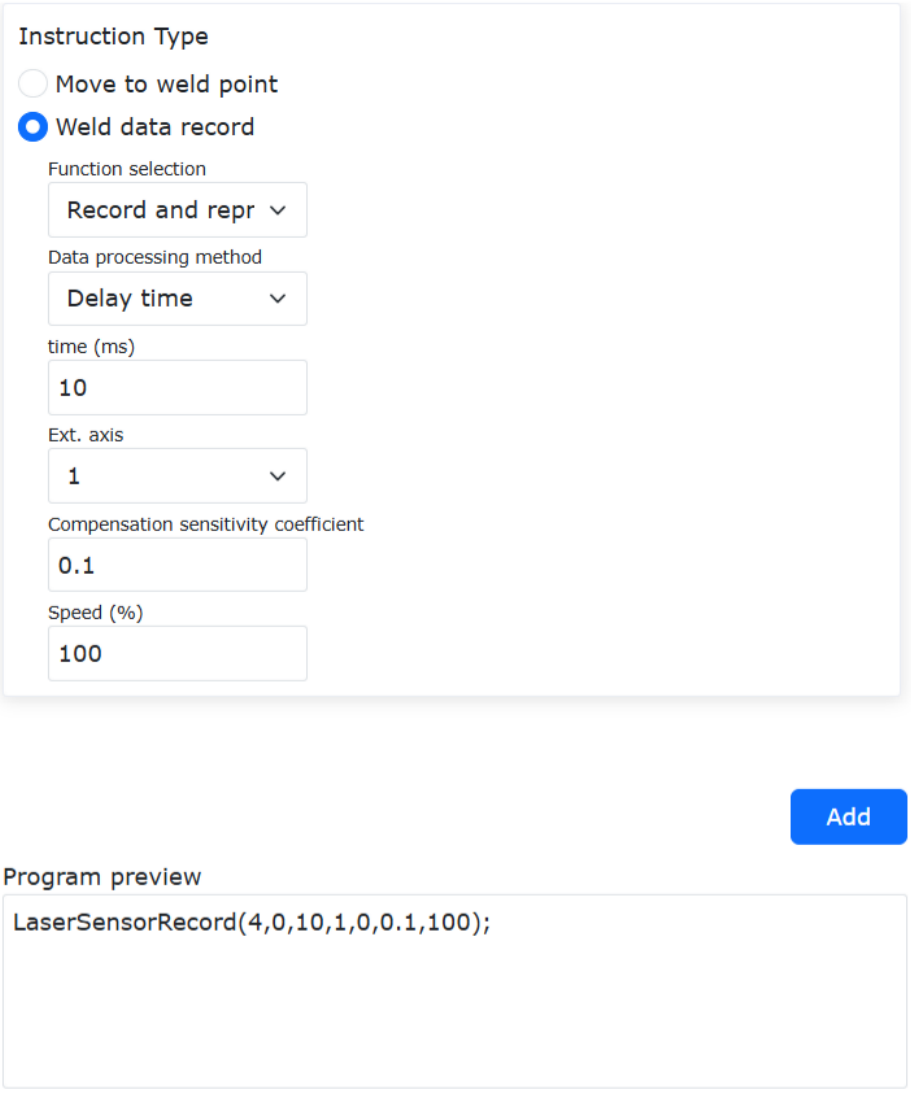
1.3.8.16 Laser Seam Finding Point Position Acquisition Function

1.3.8.16.1 Robot Laser Seam Finding Point Position Acquisition System Composition

Figure 8.16-1 Robot Laser Seam Finding Point Position Acquisition System Topology Diagram

In the system, (a) is the computer, (b) is the robot and its control box, (c) is the laser sensor.

LT-Rec


Instruction Type
Move to weld point
Weld data record
Function selection
Record and repr
Data processing method
Delay time
time (ms)
10
Ext. axis
1
Compensation sensitivity coefficient
0.1
Speed (%)
100
Add
Program preview
LaserSensorRecord(4,0,10,1,0,0.1,100);

ExAxisLaserTracking.lua

```
1↔ PTP(recordStartPt,100,-1,0)
2↔ EXT_AXIS_PTP(0,laserSafe,100,-1)
3↔ Lin(weld,100,-1,0)
4↔ LaserSensorRecord(4,1,10,1,0,0.1,100)
5↔ EXT_AXIS_PTP(0,laserSafe2,100,-1)
6↔ LaserSensorRecord(0,0,10,1,0,0.1,100)
7↔ PTP(recordStartPt,100,-1,0)
8↔ EXT_AXIS_PTP(0,laserSafe,100,-1)
```



a b c

1.3.8.16.2 Laser Sensor Communication Configuration

Open WebApp, click sequentially “Initial” -> “Peripherals” -> “Line Laser Sensor” to configure the sensor communication.

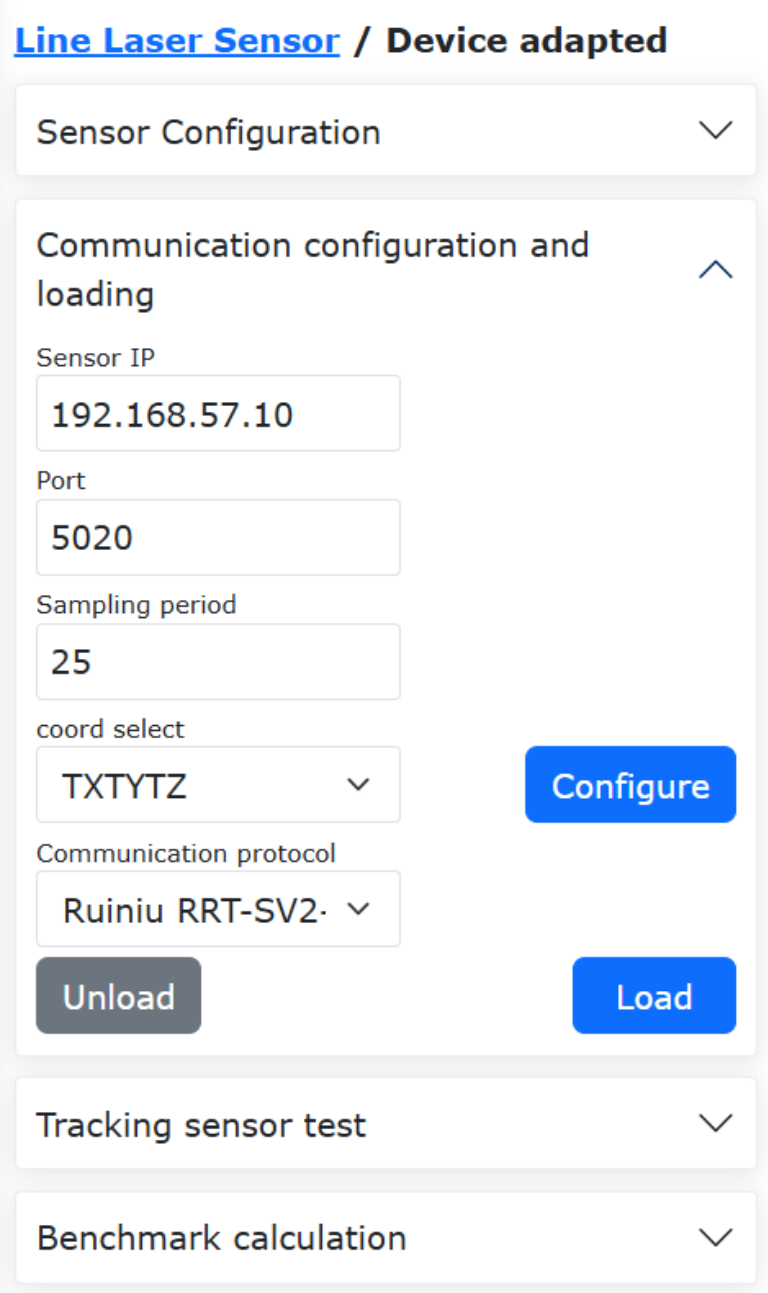


Figure 8.16-2 Sensor Communication Configuration

1.3.8.16.3 Laser Seam Finding Point Position Acquisition Function

The operation process for acquiring the laser seam finding point position is as follows:

Step 1: Before laser seam finding, first designate the seam finding start points “seamStartPt1”, “seamStartPt2”. Then click “Teach Program”, “Program Programming”, select “Point to Point”, and move the laser sensor’s beam close to the seam finding start point 1 “seamStartPt1” near the start of weld seam 1.

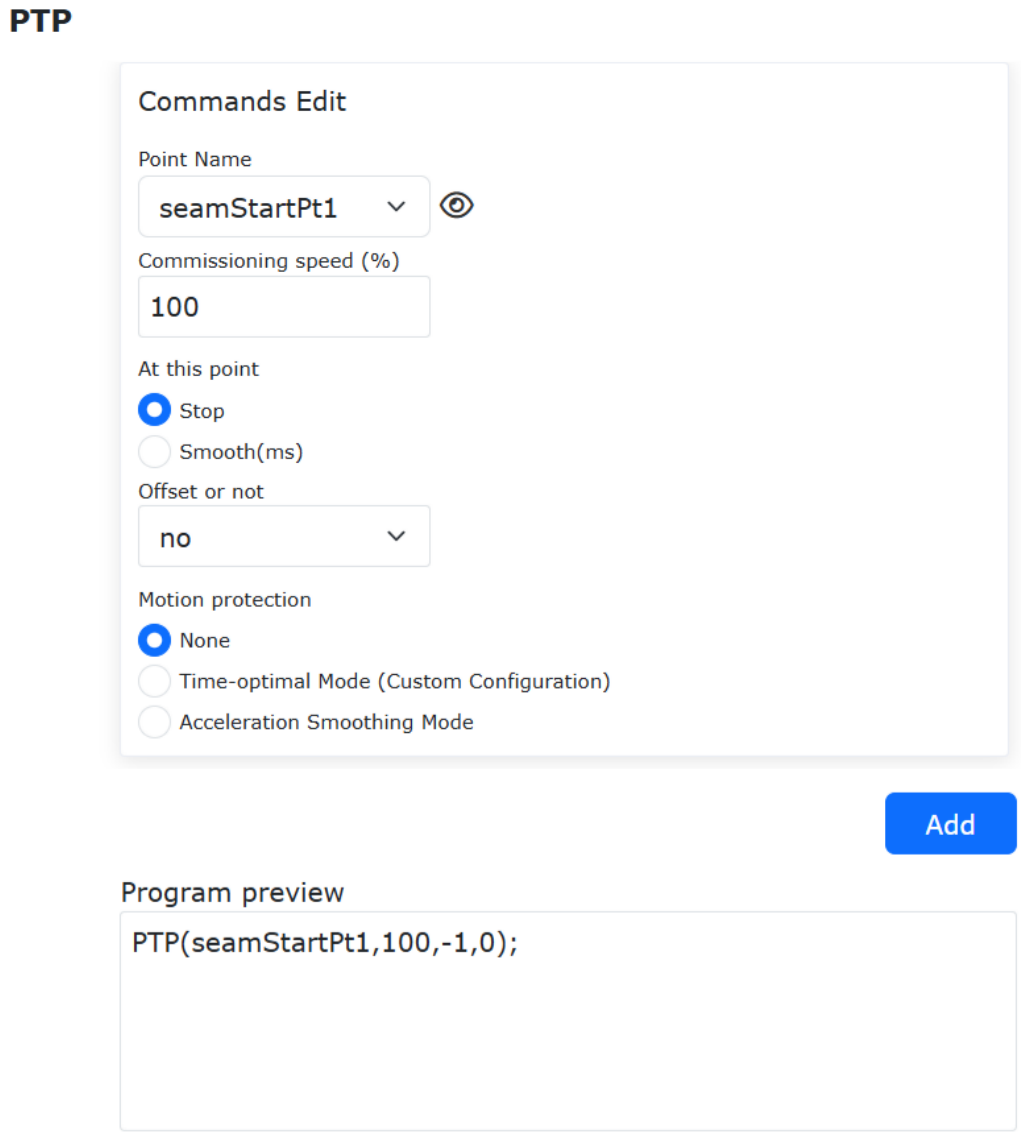
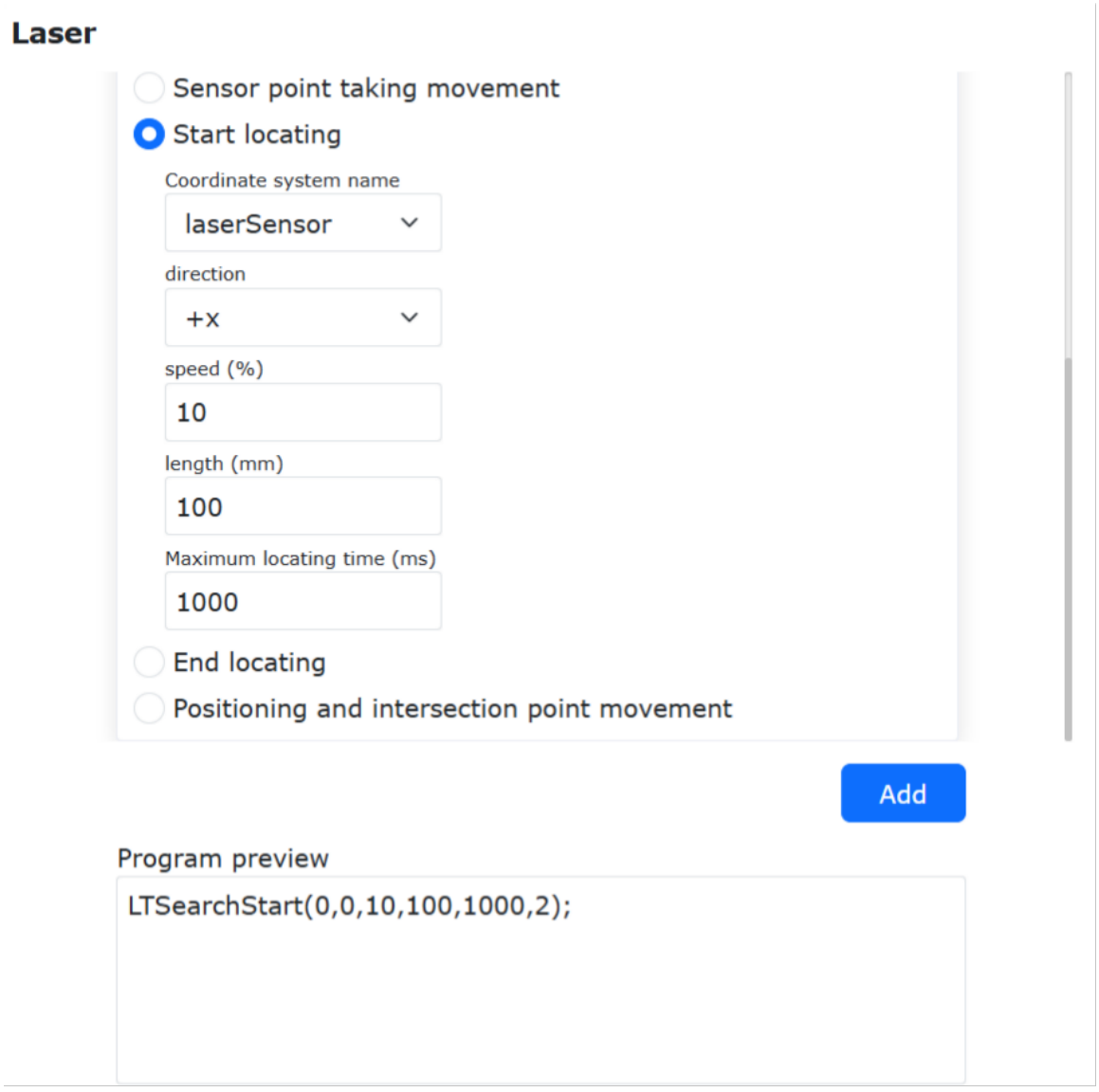


Figure 8.16-3 Adding the Move to Seam Finding Start Point 1 Instruction

Step 2: In the instruction type, click “Seam Finding Start”, then select the calibrated sensor coordinate system, set the seam finding direction, speed, length, and maximum seam finding time, and click the “Add” button. Then click “Seam Finding End”, and click the “Add” button.

Figure 8.16-4 Adding the Seam Finding Start Instruction

Step 3: Select “Sensor Point Acquisition Motion”, select the calibrated “Laser Sensor” for the coordinate system name, select “PTP” or “LIN” for the motion mode, set the debugging speed and choose whether to “Configure Pose”, click the “Add” button, then click the “Apply” button to add it to the LUA program.


Laser
Sensor point taking movement
Start locating
Coordinate system name
laserSensor
direction
+x
speed (%)
10
length (mm)
100
Maximum locating time (ms)
1000
End locating
Positioning and intersection point movement
Add
Program preview
LTSearchStart(0,0,10,100,1000,2);

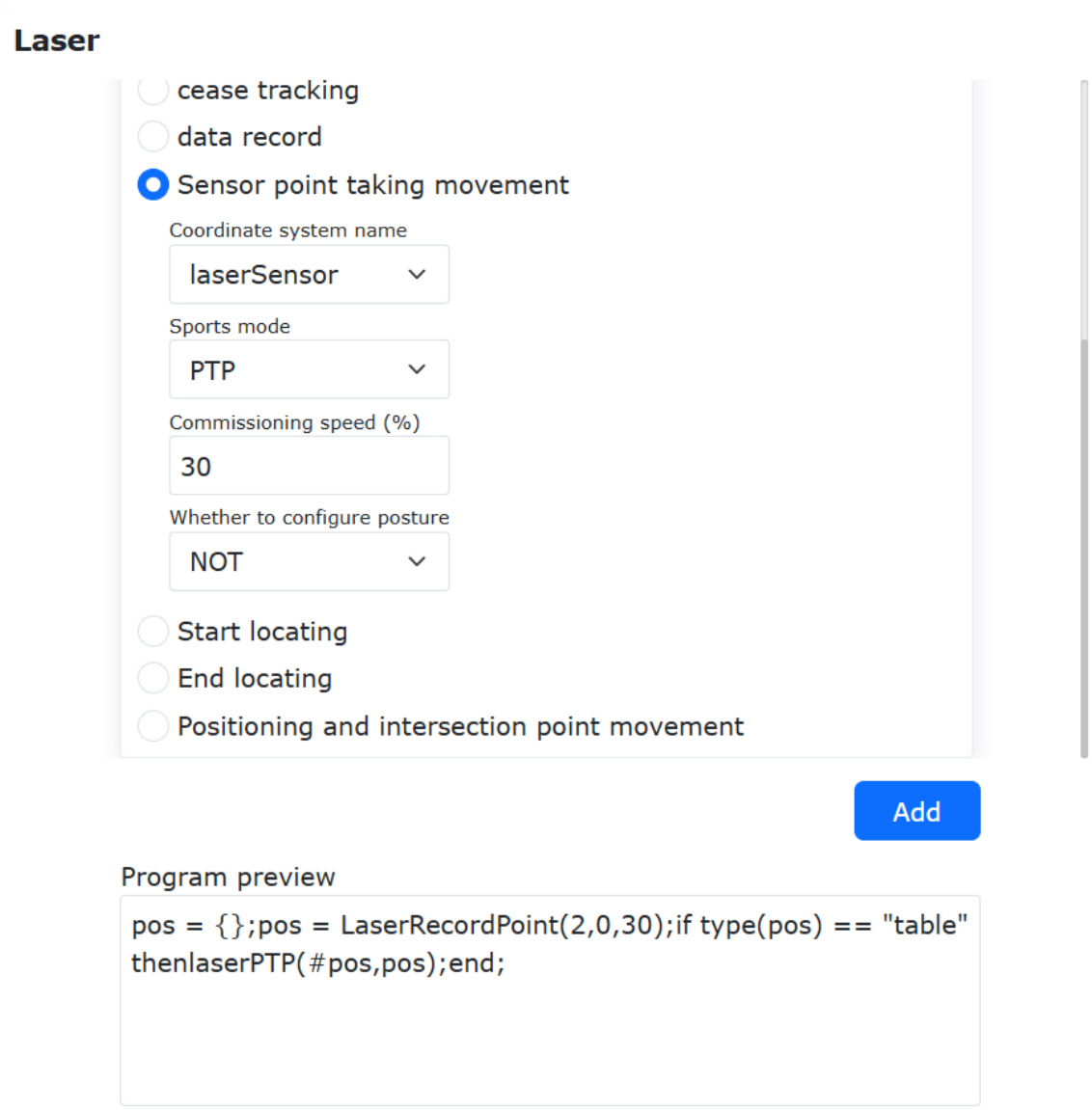

Laser
cease tracking
data record
Sensor point taking movement
Coordinate system name
laserSensor
Sports mode
PTP
Commissioning speed (%)
30
Whether to configure posture
NOT
Start locating
End locating
Positioning and intersection point movement
Add
Program preview
pos = {};pos = LaserRecordPoint(2,0,30);if type(pos) == "table" thenlaserPTP(#pos,pos);end;

Figure 8.16-5 Adding the Sensor Point Acquisition Motion Instruction

Step 4: In the "Program Programming" interface, click the "Switch Mode" button, change the variable "pos" to "pos1", and delete the move to seam finding point instruction.

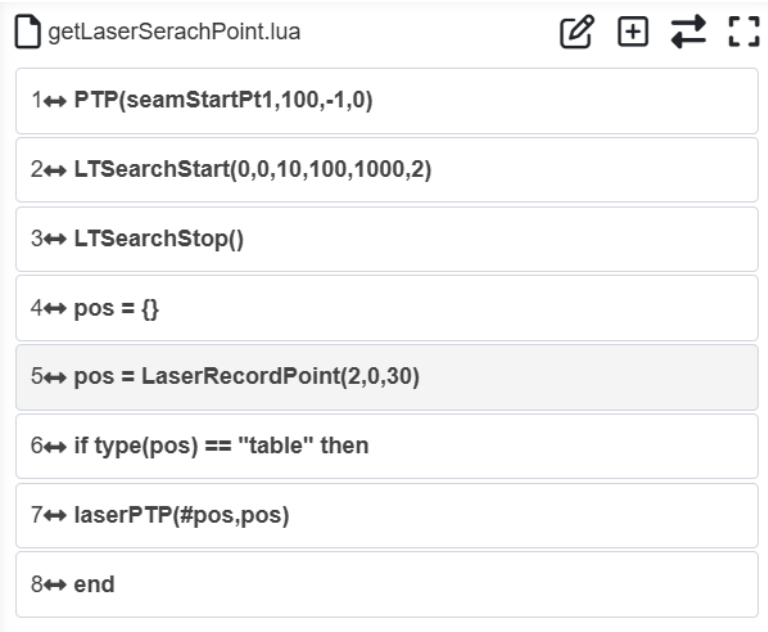


Figure 8.16-6 Program Programming Switch Mode

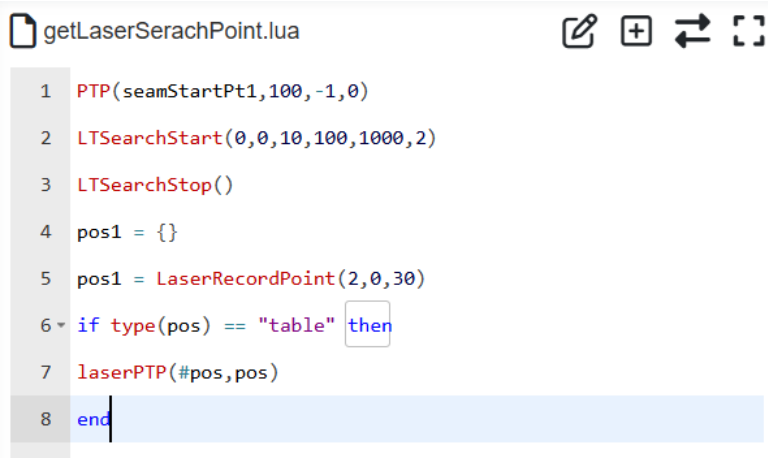


Figure 8.16-7 Modifying the Acquire Laser Seam Finding Point Program

Step 5: Follow Steps 1-4 to perform seam finding for the second weld seam and acquire the laser seam finding point position.

Figure 8.16-8 Second Weld Seam Seam Finding Point Acquisition

getLaserSerachPoint.lua

```lua
PTP(seamStartPt1,100,-1,0)
LTSearchStart(0,0,10,100,1000,2)
LTSearchStop()
pos1 = {}
pos1 = LaserRecordPoint(2,0,30)
PTP(seamStartPt2,100,-1,0)
LTSearchStart(0,0,10,100,1000,2)
LTSearchStop()
pos2 = {}
pos2 = LaserRecordPoint(2,0,30)
if type(pos) == "table" then
laserPTP(#pos,pos)
end
```

1.3.8.17 DARU DFC Force Control Polishing Head Application

1.3.8.17.1 Overview

Installing the DFC polishing head on the robot end-effector helps the robot quickly deploy grinding, polishing, and deburring tasks in different scenarios. It allows customizing the force control magnitude for workpieces of different sizes and shapes, improving the precision and effectiveness of grinding work.

1.3.8.17.1.1 Hardware Description

The collaborative robot communicates with and controls the DARU DFC polishing head via Ethernet. The WebApp generates the DARU DFC polishing head communication protocol, which sends control data to the DARU force control controller module via TCP/IP. The module then sends the received control data to the DFC force control actuator, thereby achieving control of the polishing head. The force control controller module acts as the server for Ethernet communication and can connect to polishing head actuators on two channels.

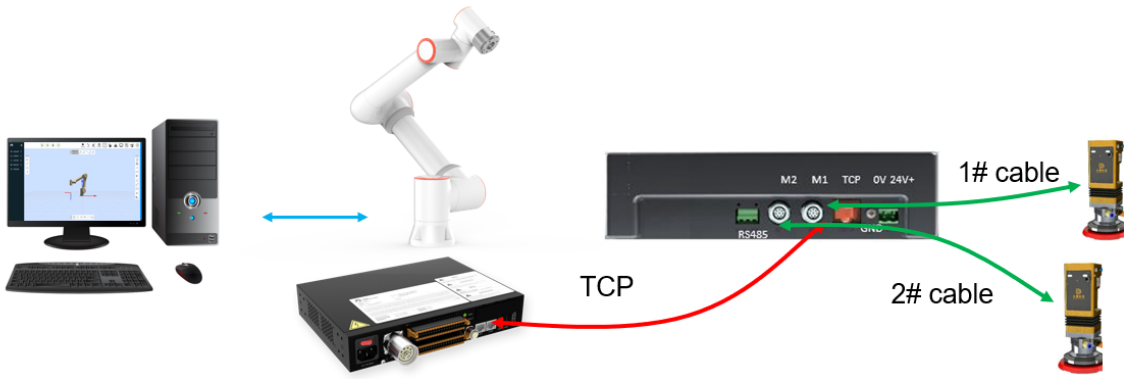


Figure 8.17-1 Collaborative Robot DARU DFC Polishing Head Application

The force control controller module requires the following configuration: Ethernet port configured with IP address: 192.168.58.88, port number: 2000.

1.3.8.17.2 Function Configuration

Open the WebApp, click sequentially on “Initial Setup”, “Peripherals”, “Polishing”. The control types for the polishing head are Pre-adapted Devices and Peripheral Open Protocol: Pre-adapted Devices: Automatically generates and loads the open protocol for adapted polishing head device types, no user programming required. Peripheral Open Protocol: Users write the open protocol for the polishing head to be adapted using Lua to achieve communication control.

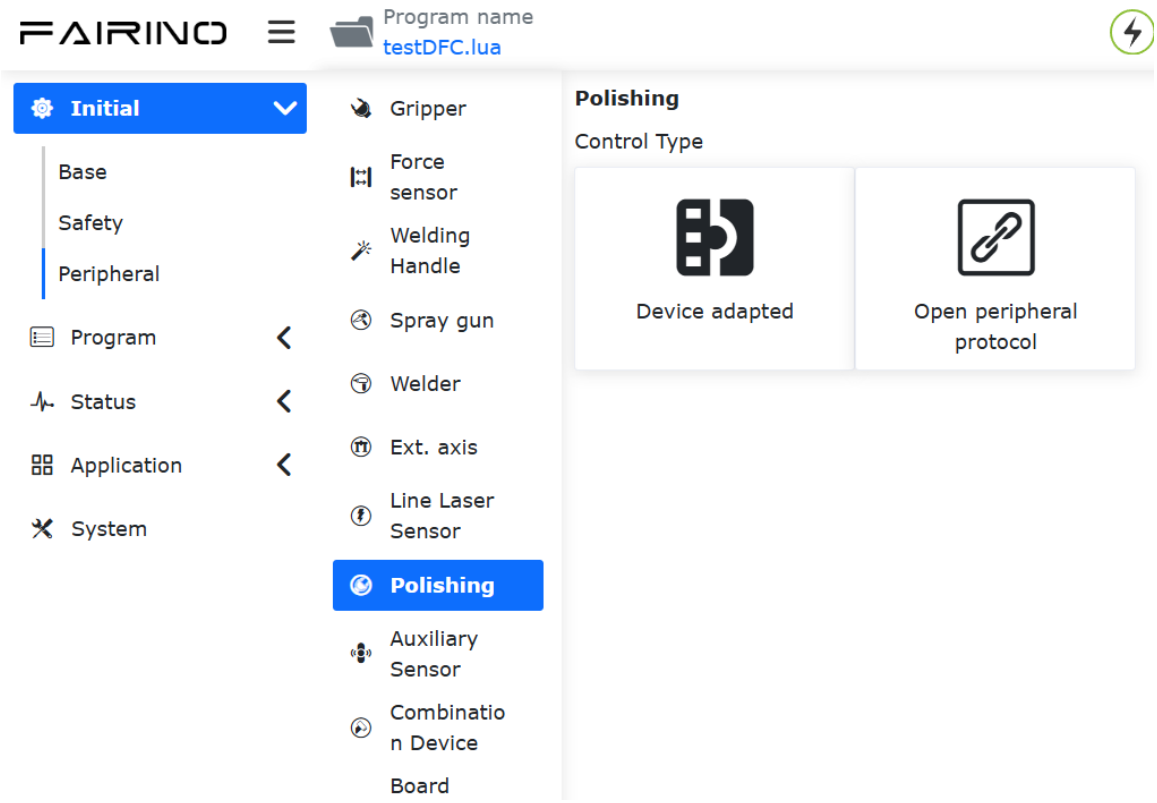


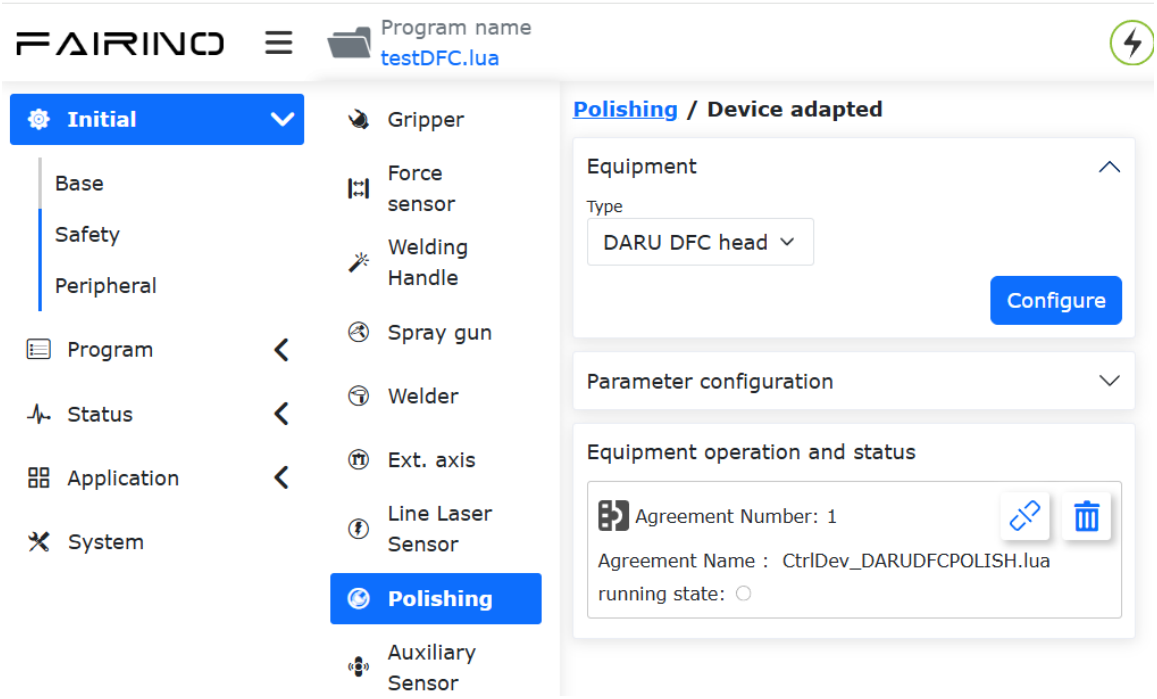
Figure 8.17-2 Polishing Control Types

1.3.8.17.2.1 Pre-adapted Device Configuration

Open the WebApp, click sequentially on “Initial Setup”, “Peripherals”, “Polishing Head”, “Pre-adapted Devices”. In the device status, select “DARU DFC Polishing Head” for the type, then click “Configure”. This will automatically load the embedded peripheral open protocol “CtrlDev_DARUDFCPOLISH.lua”.

Figure 8.17-3 DARU DFC Device Peripheral Open Protocol Auto-loading

With the hardware connection confirmed correct, the open protocol can be started. When the run status is green and the communication status feedback in the Polish status on the right shows “Connection Established”, it indicates successful communication between the robot and the polishing head controller. Parameters can then be configured to set the polishing head channel for force control and the set force value. The open protocol will cyclically send the set value, channel, and the robot’s current rx, ry, rz to the polishing head, as shown in Figure 2-3. Additionally, the Polish status feedback will also display the current force feedback value and force limit warning from the polishing head in



real-time. When a warning occurs, an alarm reminder will also appear in the upper right corner of the page, as shown in Figure 2-4.

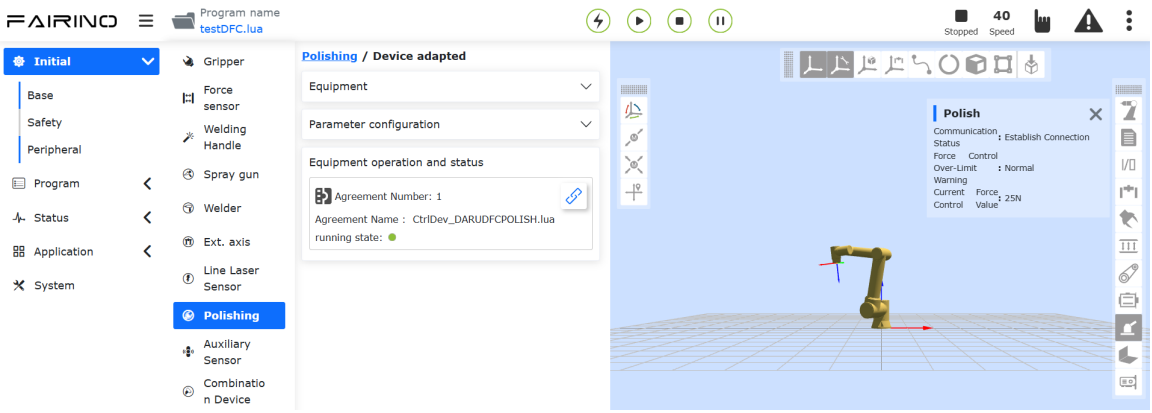
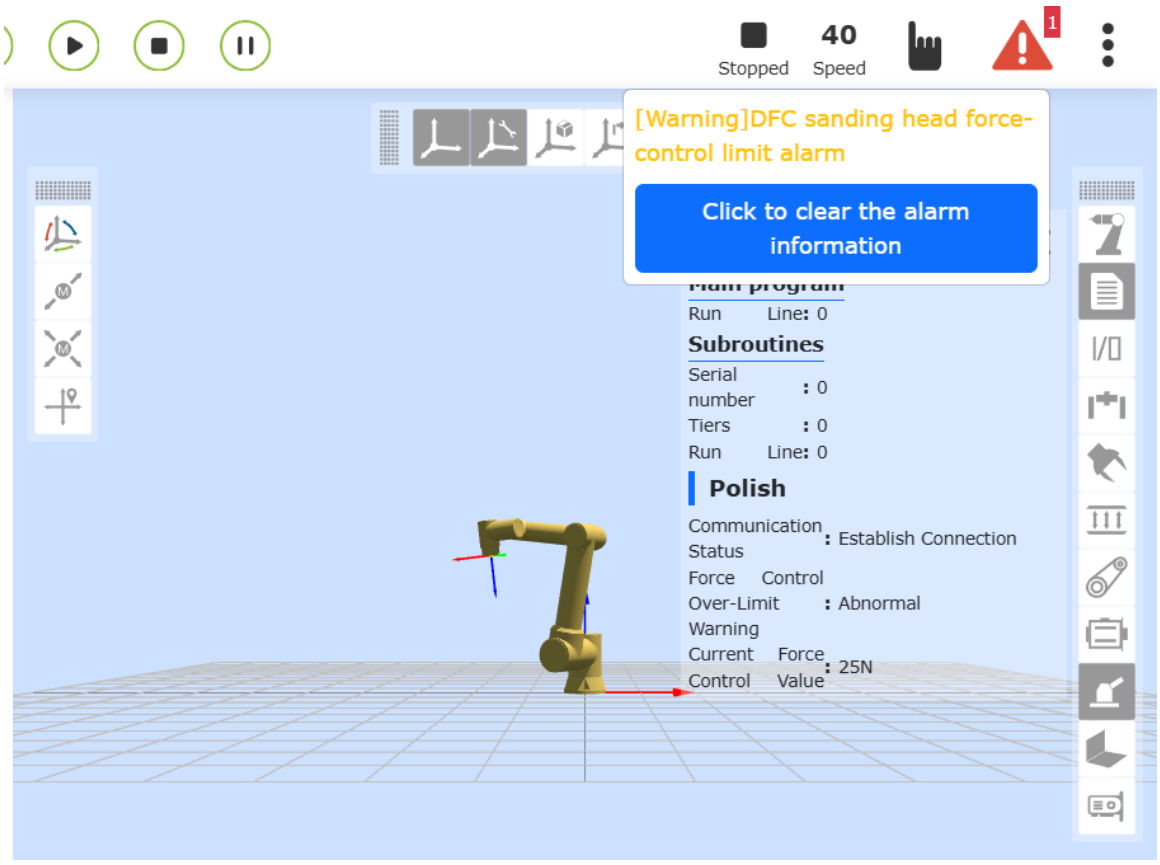


Figure 8.17-4 DFC Polishing Head Page Settings and Status Feedback

Figure 8.17-5 DFC Polishing Head Force Control Limit Exceeded Alarm



1.3.8.17.2.2 Peripheral Open Protocol Download

Click the "Download" button in "Peripheral Open Protocol" to download the protocol to the local computer. The peripheral open protocol is a cyclically executed LUA program. In each cycle, the program performs the following steps:

Read control data for the DFC polishing head from the robot;

Write control data to the DFC polishing head via socket;

Read status data from the DFC polishing head via socket;

Feedback DFC polishing head status data to the robot;

The communication protocol executes cyclically to achieve communication control between the robot and the polishing head. Within the communication protocol, users can customize the cycle period, the server port to connect to, and the IP address.

Below is a sample communication protocol code for the DARU DFC polishing head:

```
local id = 1
local ctrlValues = {0,0, 0,0, 0,0, 0,0}
local realTimeState = {0,0, 0,0, 0,0, 0,0}
socket1 = TCPClientConnect('192.168.58.88', 2000, 500, 10, 2, 3)
sleepCnt = 100
while(sleepCnt > 0) do
    local stopFlag = GetOpenLUAStopFlag(id)
    if(stopFlag ~= 0) then
```

(continues on next page)

(continued from previous page)

```
      TCPClientDisconnect(socket1)
      setDFCPolishRealtimeState(0, 0, 0)
      break
    end
  sleepCnt = sleepCnt -1
  sleep_ms(50)
end
local cnt = 5
while(1) do
    channel, force = getDFCPolishSet()
    comState, sendBuff = DFCPolishInput(socket1, channel, force)
    sleep_ms(50)

    byte, error, forceFeedback = DFCPolishOutput(socket1)
    setDFCPolishRealtimeState(comState, error, forceFeedback)
    sleep_ms(50)

  if(comState == 0) then
      TCPClientDisconnect(socket1)
      while(cnt > 0) do
        socket1 = TCPClientConnect('192.168.58.88', 2000, 500, 10, 2, 3)
        cnt = cnt - 1
        if(socket1 > 0)then
          break
        end
      end
  end

    local stopFlag = GetOpenLUAStopFlag(id)
    if(stopFlag ~= 0 or cnt == 0) then
      TCPClientDisconnect(socket1)
      setDFCPolishRealtimeState(0, 0, 0)
      break
    end
end
```

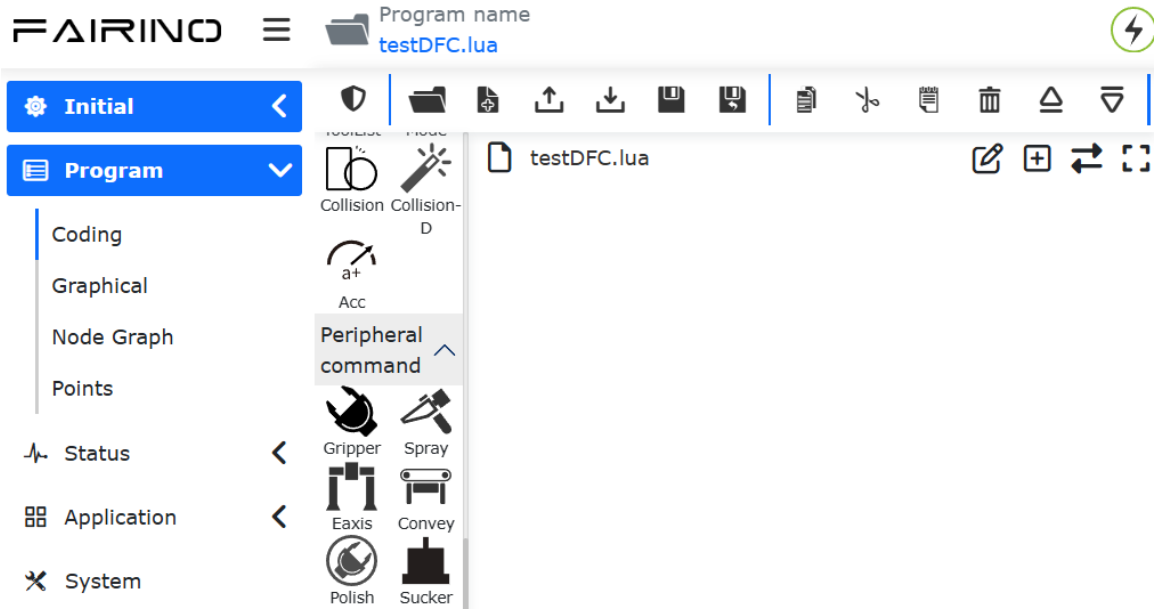
1.3.8.17.3 DFC Polishing Head LUA Program Application

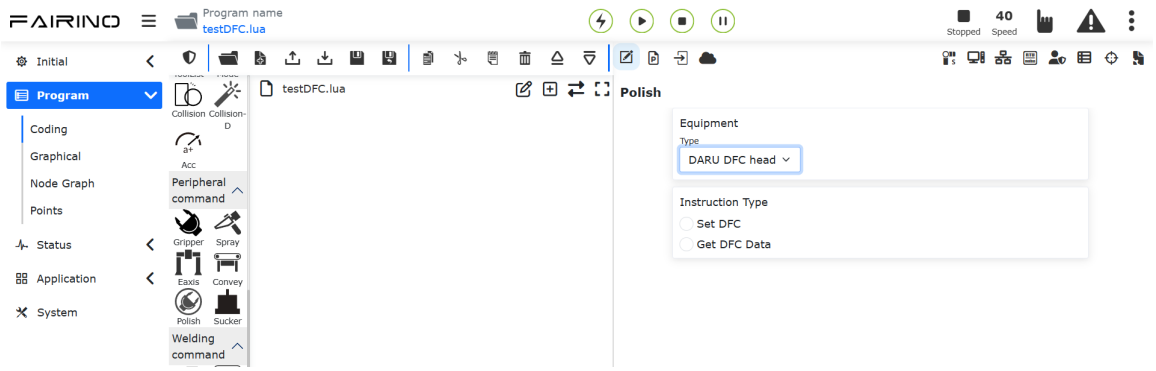
Adding DFC force control configuration, channel switching, status acquisition instructions, etc., to the robot LUA program, combined with robot motion instructions, enables flexible and convenient implementation of polishing applications. Open the WebApp, click sequentially on “Teach Program”, “Program Programming”, and create a new LUA program “testDFC.lua”.

Figure 8.17-6 Create New “testDFC.lua” Program

Select the instruction type as “Peripheral Instruction”, and click the “Polishing Device” button in the peripheral instructions. The “Polish” polishing instruction addition page will then appear on the right side of the WebApp. Select “DARU DFC Polishing Head” for the device type.

Figure 8.17-7 Polishing Head Instruction Addition


Program name
testDFC.lua
Initial
Program
Coding
Graphical
Node Graph
Points
Status
Application
System
testDFC.lua
Collision
Collision-D
Acc
Peripheral command
Gripper
Spray
Eaxis
Convey
Polish
Sucker


Program name
testDFC.lua
Stopped
40
Speed
Initial
Program
Coding
Graphical
Node Graph
Points
Status
Application
System
testDFC.lua
Collision
Collision-D
Acc
Peripheral command
Gripper
Spray
Eaxis
Convey
Polish
Sucker
Welding command
Polish
Equipment
Type
DARU DFC head
Instruction Type
Set DFC
Get DFC Data

1.3.8.17.3.1 Polishing Head Control Instruction Addition

Writing polishing head control instructions in the LUA program allows force control setting and channel selection for the DFC.

In the polishing device instruction addition page, click “Set DFC”, select the polishing head channel mode as “2”, and the set force as “10”. Click the “Add” button to add the polishing head setting instruction in the “Program Preview”.

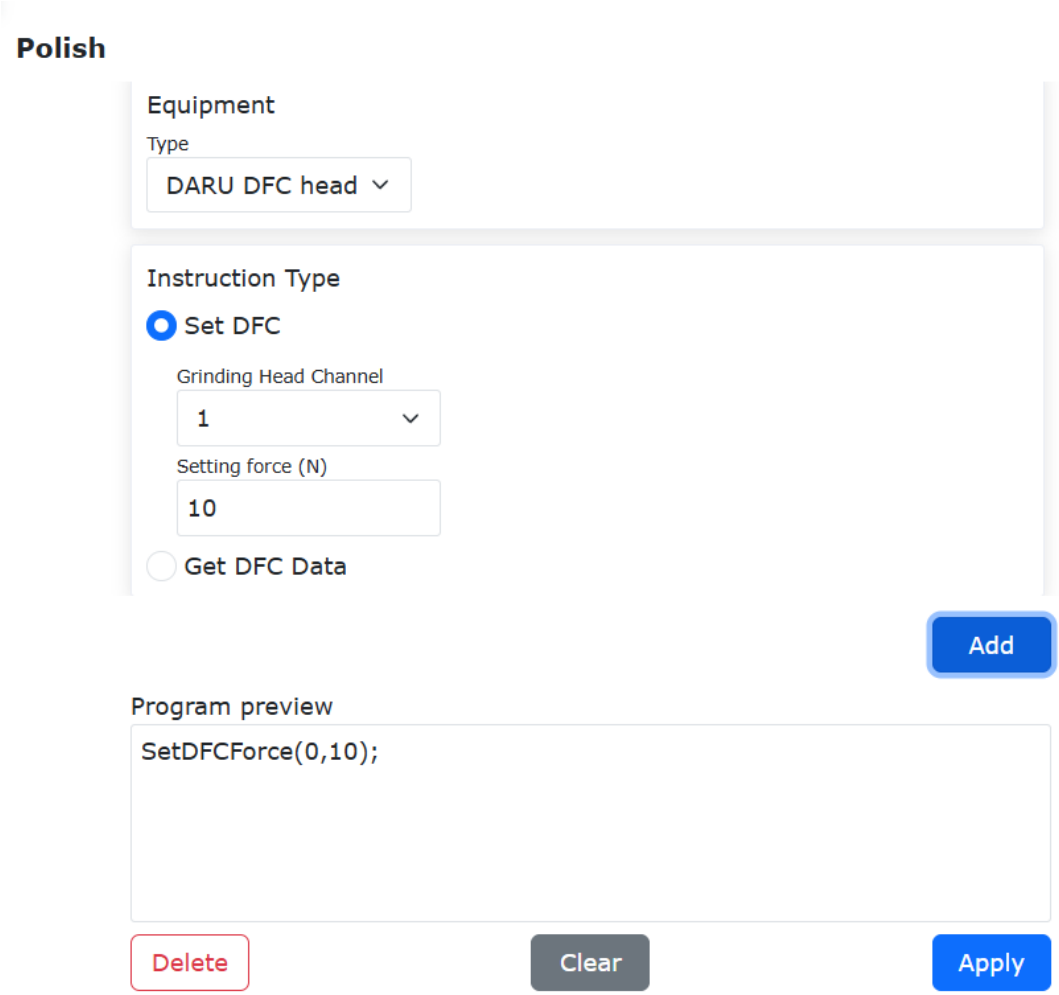


Figure 8.17-8 Add Polishing Head Control Instruction

1.3.8.17.3.2 Polishing Head Status Acquisition Instruction Addition

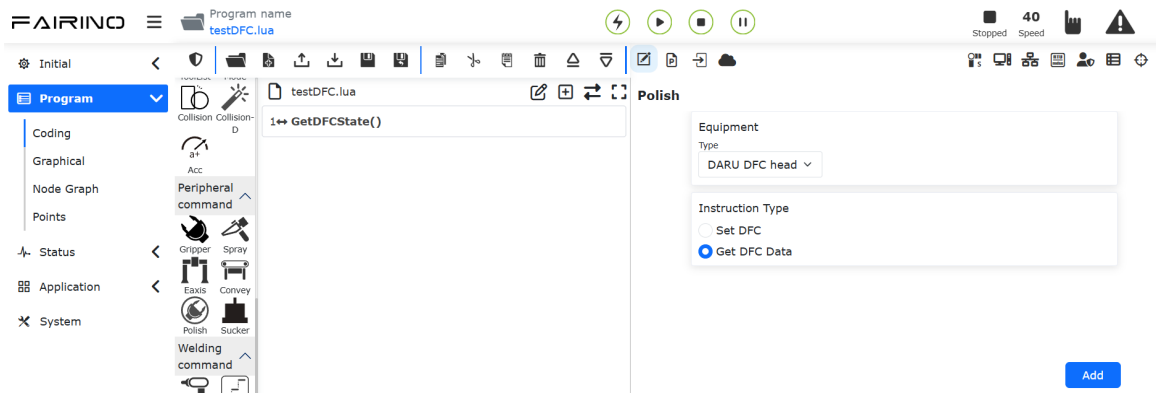
Click “Get DFC Data”, then click “Add” and “Apply” buttons sequentially. This adds an instruction “GetDFCState()” to “testDFC.lua” for acquiring polishing head data.

Figure 8.17-9 Add Get Polishing Head Status Instruction

The GetDFCState() instruction returns 2 values, as follows:

DFCwarn: Force control limit exceeded warning 0-Normal 1-Alarm;

force: Force feedback value.



In “testDFC.lua”, use three variables to receive the return values of the GetDFCState() function. Use Lua variable query to display the above information in the WebApp variable query display area.

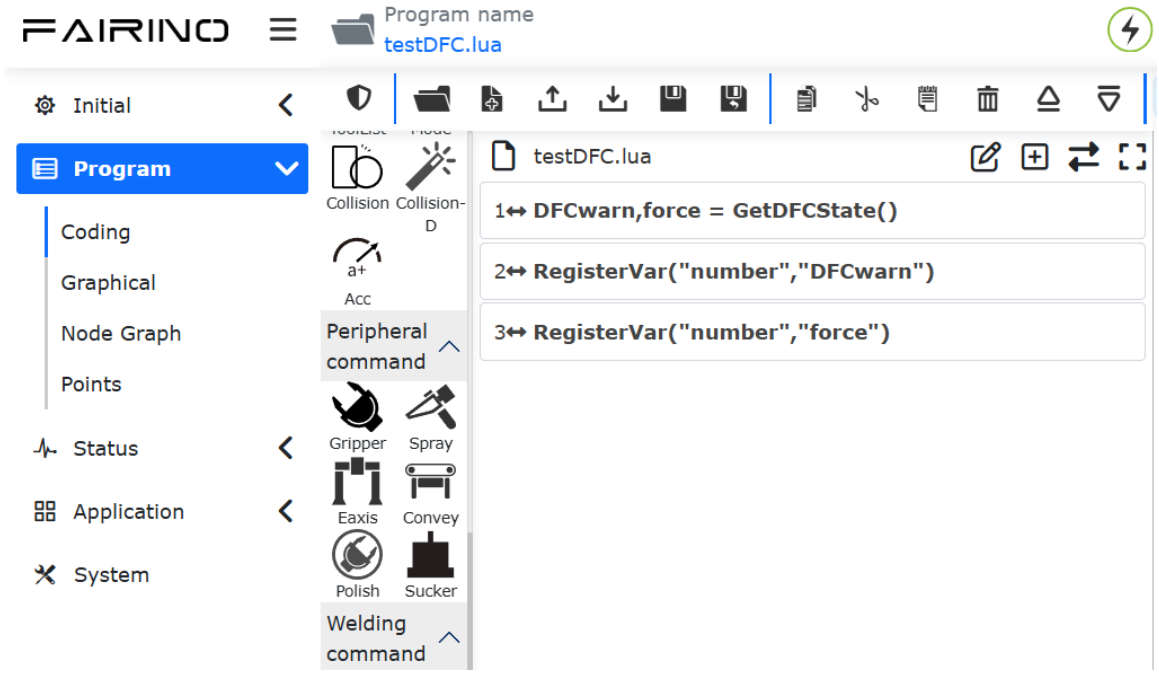


Figure 8.17-10 Get Polishing Head Status Program

1.3.8.17.3.3 Application Example

The following is a LUA program example for DFC polishing head control and monitoring:

```
SetDFCForce(0,25)
while (1) do
    PTP(c1,100,-1,0)
    SetDO(0,1,0,0)
    ARC(c2,0,0,0,0,0,0,0,0,c3,0,0,0,0,0,0,0,0,100,-1,0,100,200)
    DFCwarn,force = GetDFCState()
    RegisterVar("number","DFCwarn")
```

(continues on next page)

(continued from previous page)

```
    RegisterVar("number","force")
    if(DFCwarn == 1) then
        PTP(safe,100,-1,0)
        break
    else
        PTP(p6,100,-1,0)
    end
    SetDO(0,0,0,0)
end
```

1.3.9 Coding

1.3.9.1 Introduction

Click the command on the left to add a program node to the program tree. When the program is running, the currently executed program node is highlighted in green.

In manual mode, click the first icon on the right side of the node to make the robot execute the instruction alone, and the second icon is to edit the content of the node.

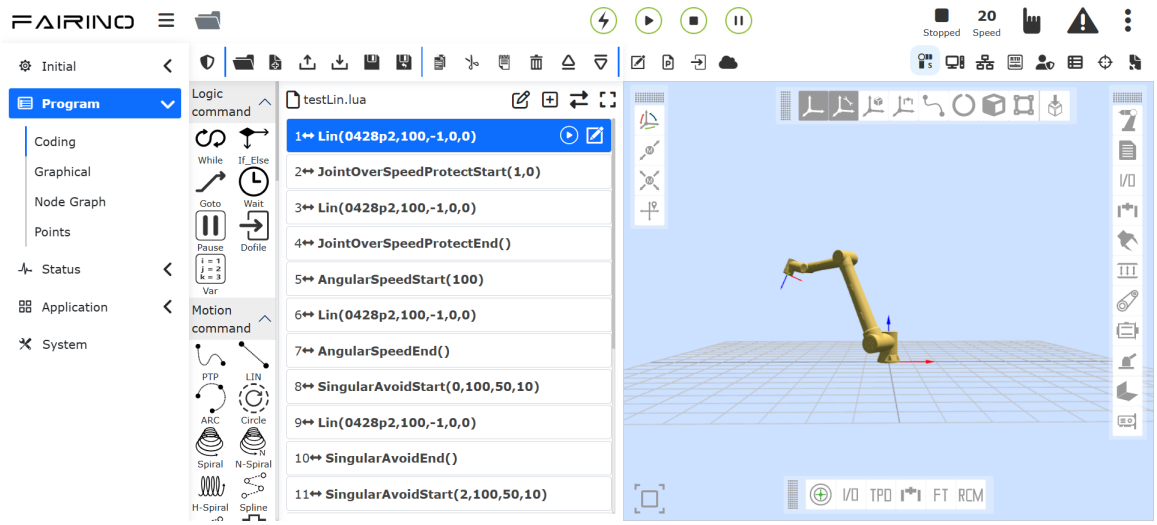


Figure 9.1-1 Program tree interface

Click “” to switch modes, and the teaching program text can be changed to the editing state.

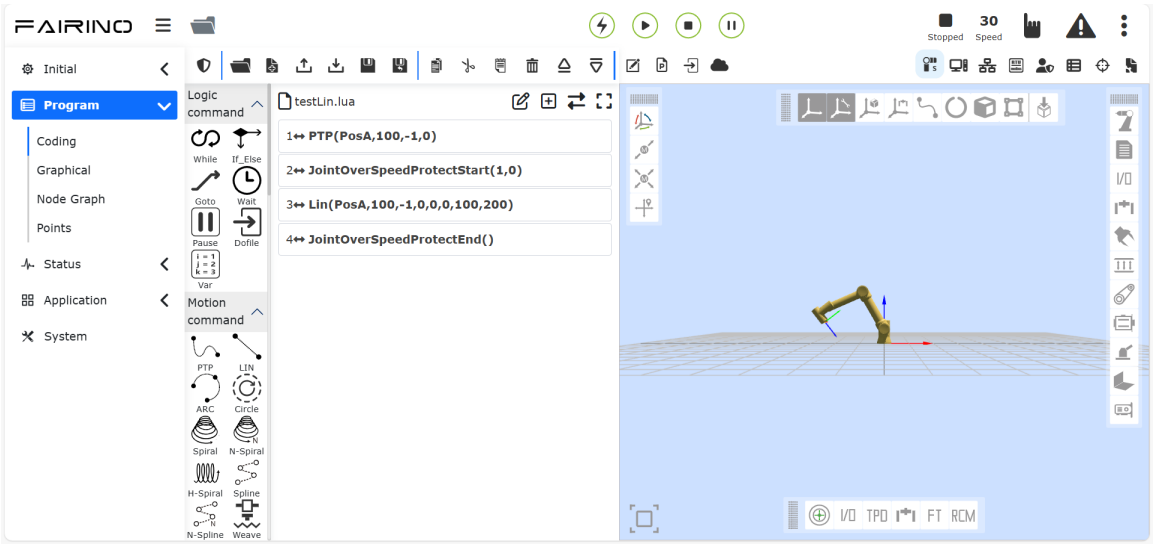


Figure 9.1-2 Teaching program editing status

The icons to the right of the program name are described as follows:



Note:

name: **Expand/Zoom**

effect: Expand/zoom the program tree interface



Note:

name: **Add teaching points**

effect: Add a local teaching point to the current program



Note:

name: **Rename**

effect: Rename the current program

1.3.9.2 Tool bar

Modify the program tree using the toolbar at the bottom of the program tree.



Note:

name: **Open**

effect: Open user program file



Note:

name: **New build**

effect: Select a template to create a new program file



Note:

name: **Import**

effect: Import the file into the user program folder



Note:

name: **Export**

effect: Export user program files to a local point.



Note:

name: **Save**

effect: Save file edits



Note:

name: **Save as**

effect: Rename the file and store it in the user program or template program folder.



Note:

name: **Copy**

effect: Duplicates a node and allows it to be used for other operations (eg: paste it elsewhere in the program tree).



Note:

name: **Paste**

effect: Allows you to paste previously cut or copied nodes.



Note:

name: **To cut**

effect: Cuts a node and allows it to be used for other operations (eg: paste it elsewhere in the program tree).



Note:

name: **Delete**

effect: Deletes a node from the program tree.



Note:

name: **Move up**

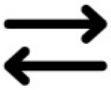
effect: Move the node up.



Note:

name: **Move down**

effect: Move the node down.



Note:

name: **Toggle edit mode**

effect:The program tree mode and lua editing mode switch each other.

The icons on the top right are described as follows:



Note:

name: **Programming add/edit**

effect: Add/edit the contents of the current program command



Note:

name: **Robot Model**

effect: Return to the robot 3D model interface



Note:

name: **NewDofile subroutine**

effect: When there is a NewDofile instruction in the current program command, click to enter and select the subroutine name to view the subroutine content.



Note:

name: **Modbus TCP Settings**

effect: Configure Modbus TCP communication parameters



Note:

name: **Current program backup**

effect: Record the changes made to the current program



Note:

name: **Local teaching point**

effect: Applies only to the teaching points of the current program

1.3.9.3 Program command

The left side is mainly for adding program commands. Click the icon above each keyword to enter the detailed interface of adding program commands on the right. There are two main operations for adding program commands to files:

- 1. Open the relevant command and click the Apply button to add the command to the program.
- 2. click the "Add" button first, at this time the command is not saved in the program file, and you need to click "Apply" again to save the command in the file.

The second method often occurs when multiple commands of the same type are issued. We add an add button and display the content of the added command to this type of command. Click the Add button to add a command, and the

added command displays all the added commands. , click “Apply” to save the added command to the opened file on the right.

1.3.9.4 Logic Command Interface

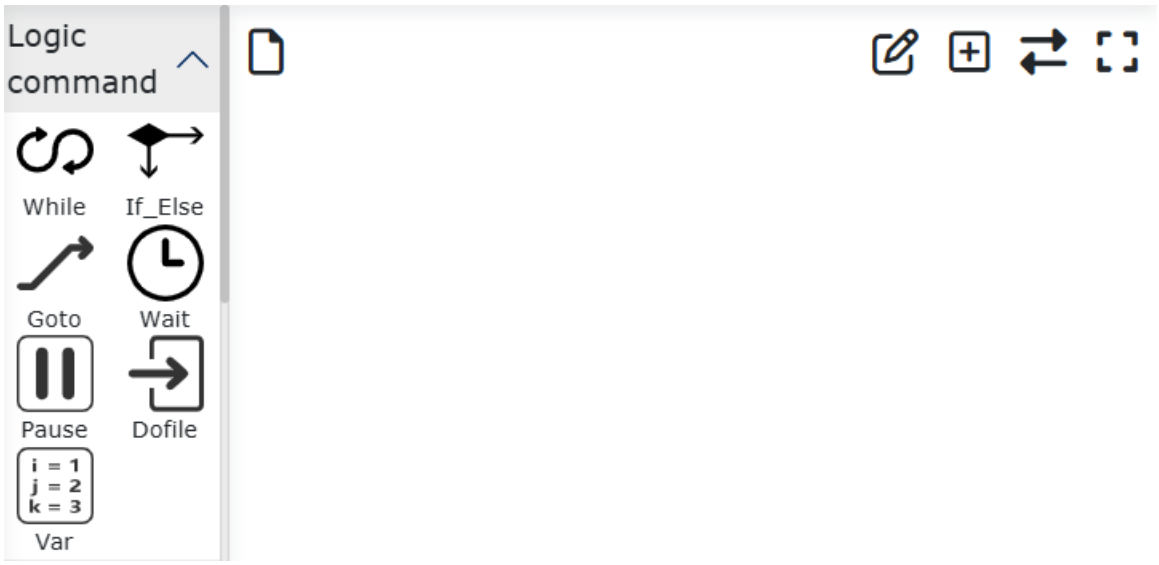


Figure 9.4 Logic Command Interface

1.3.9.4.1 While command

Select the loop scenario of the While command, the scenario is as follows:

- Always loop
- Limited number of cycles: Enter the number of loops and variable name
- Loop while expression is true: Click the input box to pop up the expression editor and select the corresponding expression according to the usage scenario

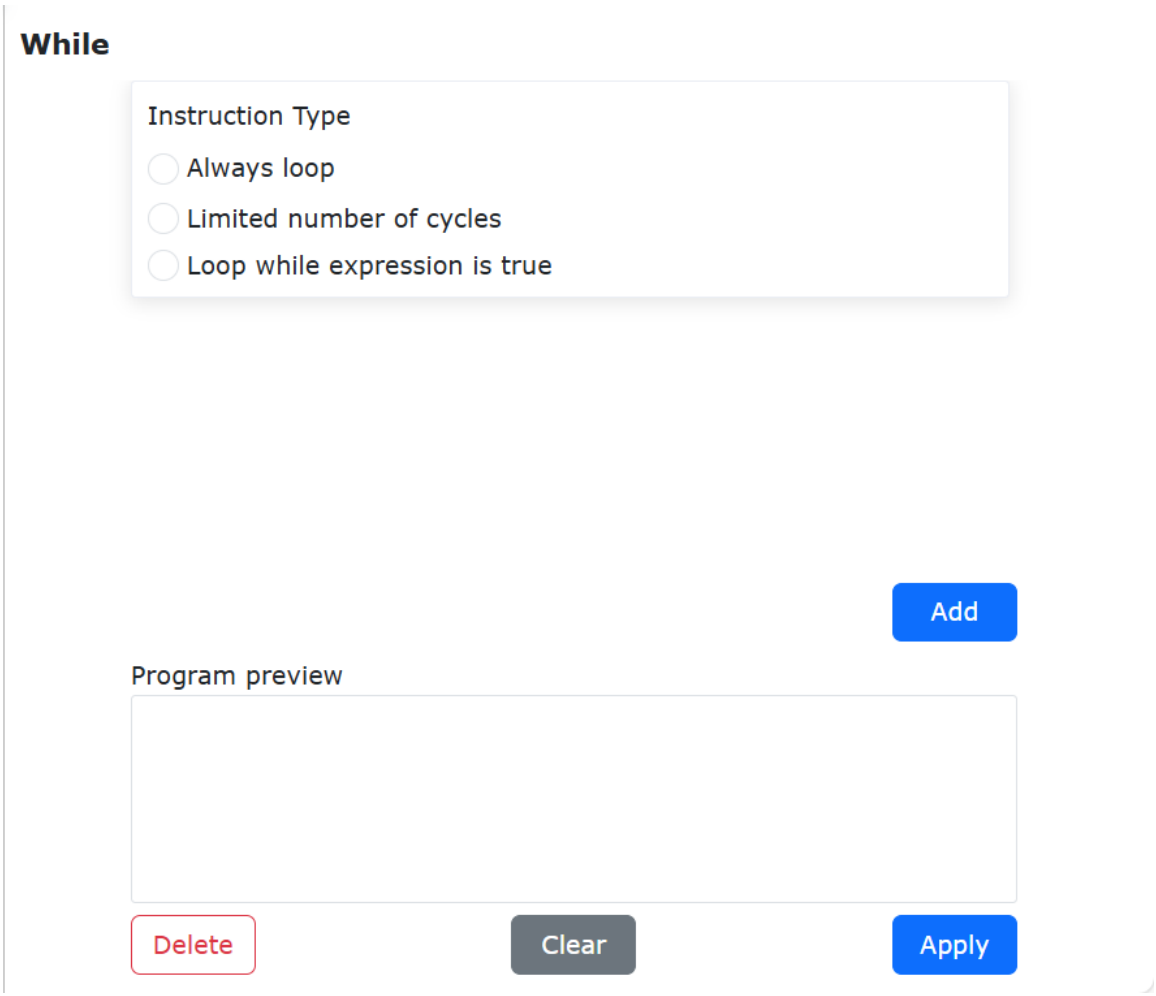


Figure 9.4-1-1 While command interface

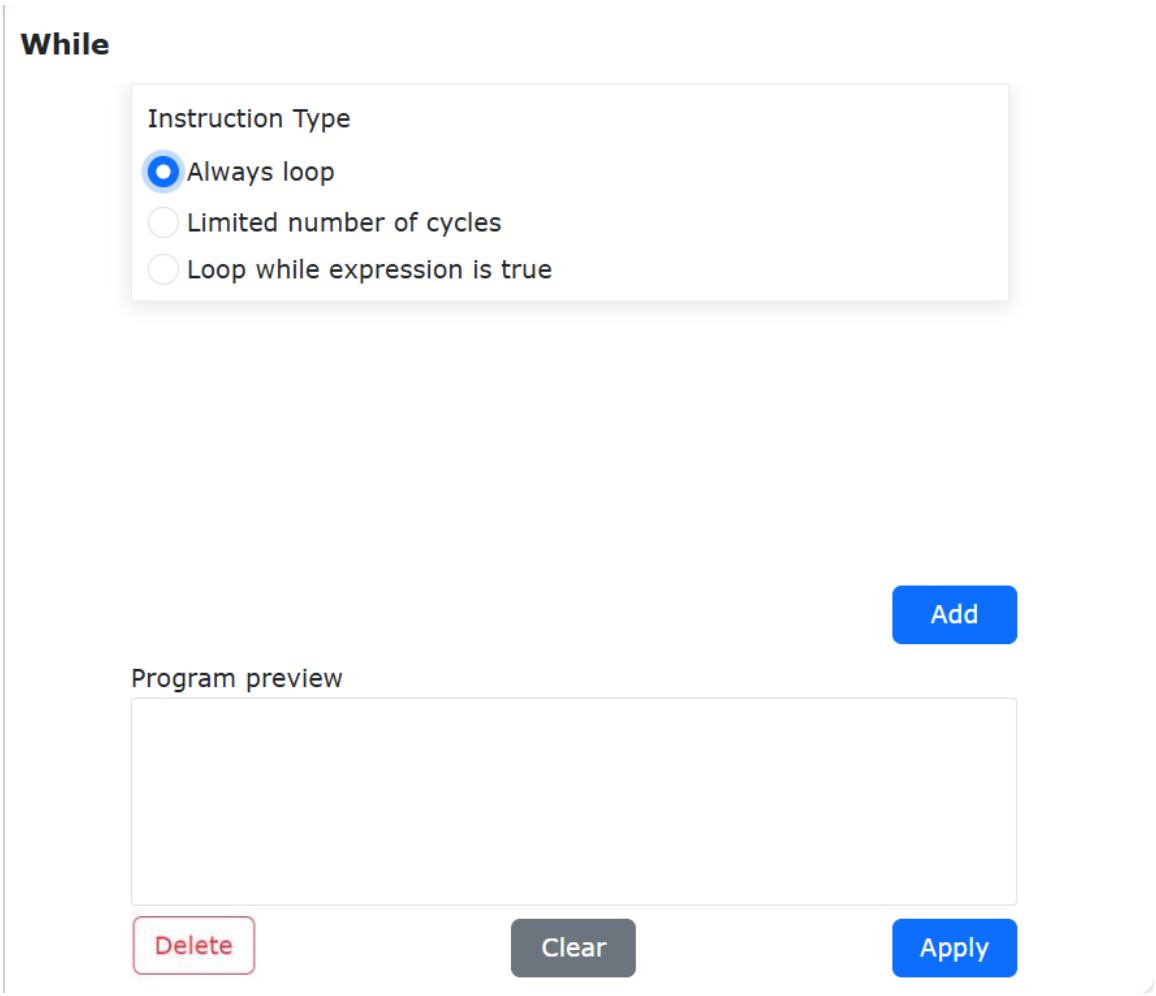


Figure 9.4-1-2 While command - Always loop

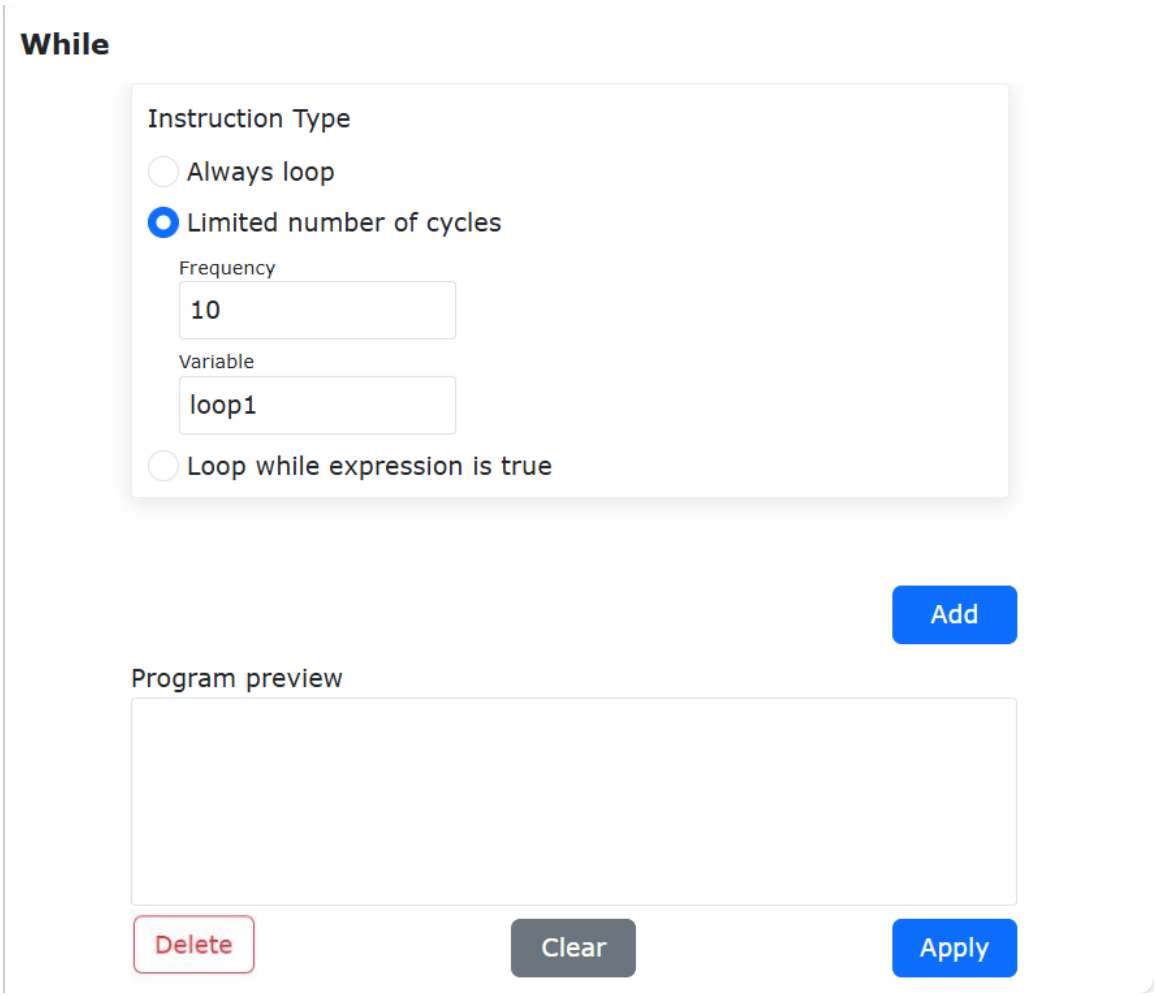


Figure 9.4-1-3 While command - Limited number of cycles

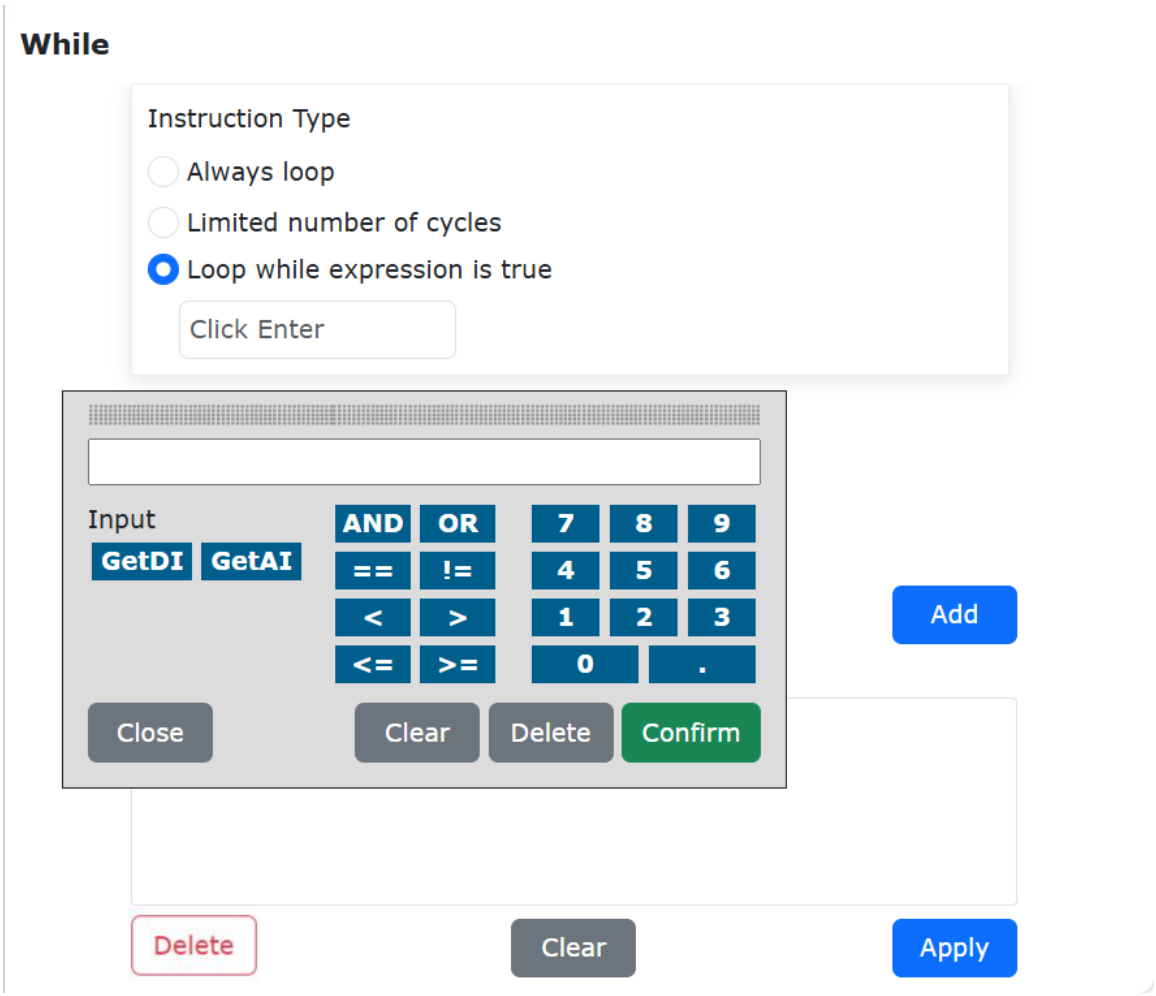


Figure 9.4-1-4 While command - Expression Editor

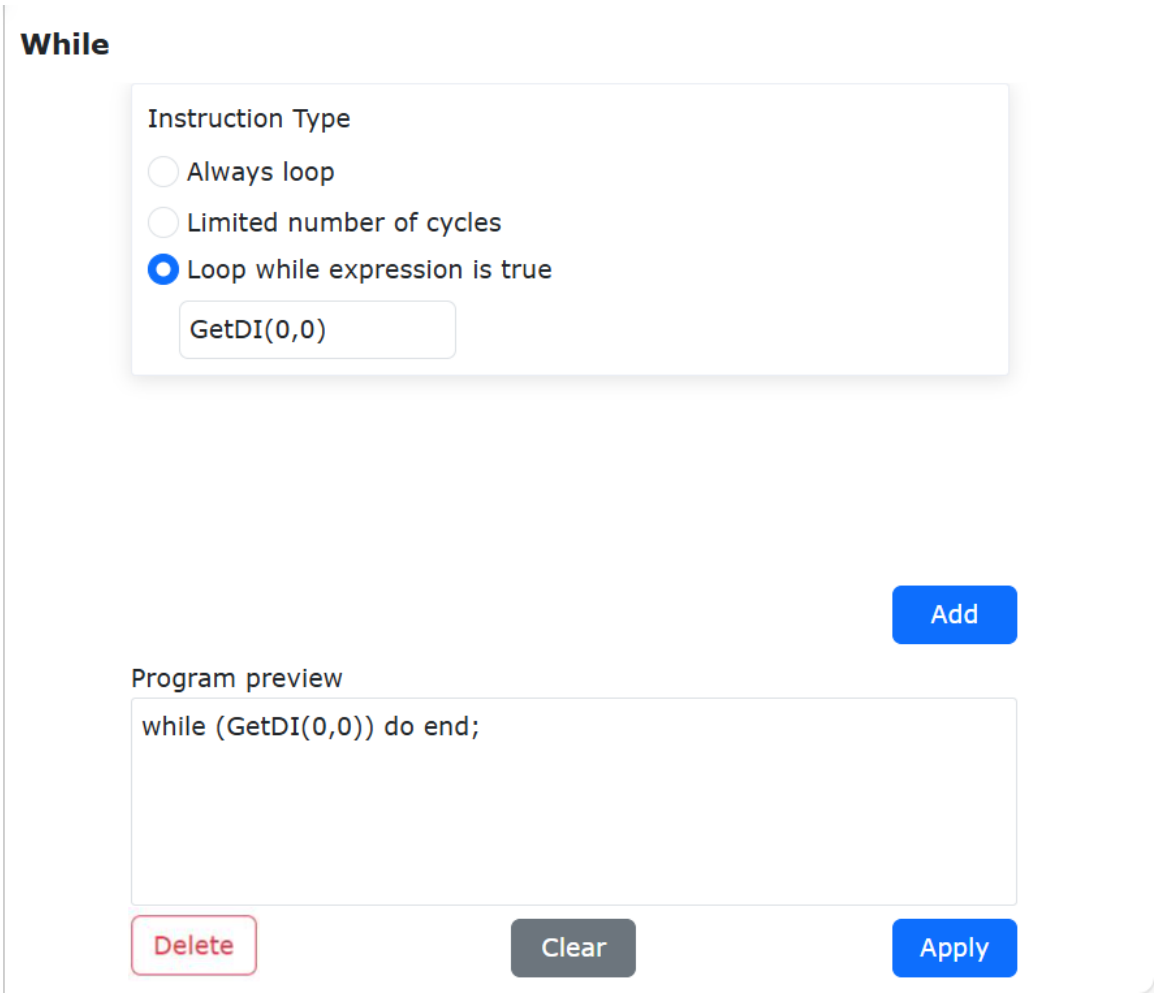


Figure 9.4-1-5 While command - Loop while expression is true

For ease of operation, you can enter any do content and edit other instructions in the program to insert them instead.

1.3.9.4.2 if...else command

Click the "if...else" button to enter the if...else command editing interface.

This command contains the following buttons:

- Add else if: When there is no "else" expression, click this button to add an "else if" expression
- Delete else if: When an "else if" expression exists, click this button to delete the "else if" expression.
- Add else: Click this button to add an "else" expression
- Delete else: Click this button to delete the "else" expression

After clicking the corresponding button to add, click the input box to pop up the expression editor and select the corresponding expression according to the usage scenario. After adding, click "Add" and "Apply".

This instruction requires a certain programming foundation. If you need help, please contact us.

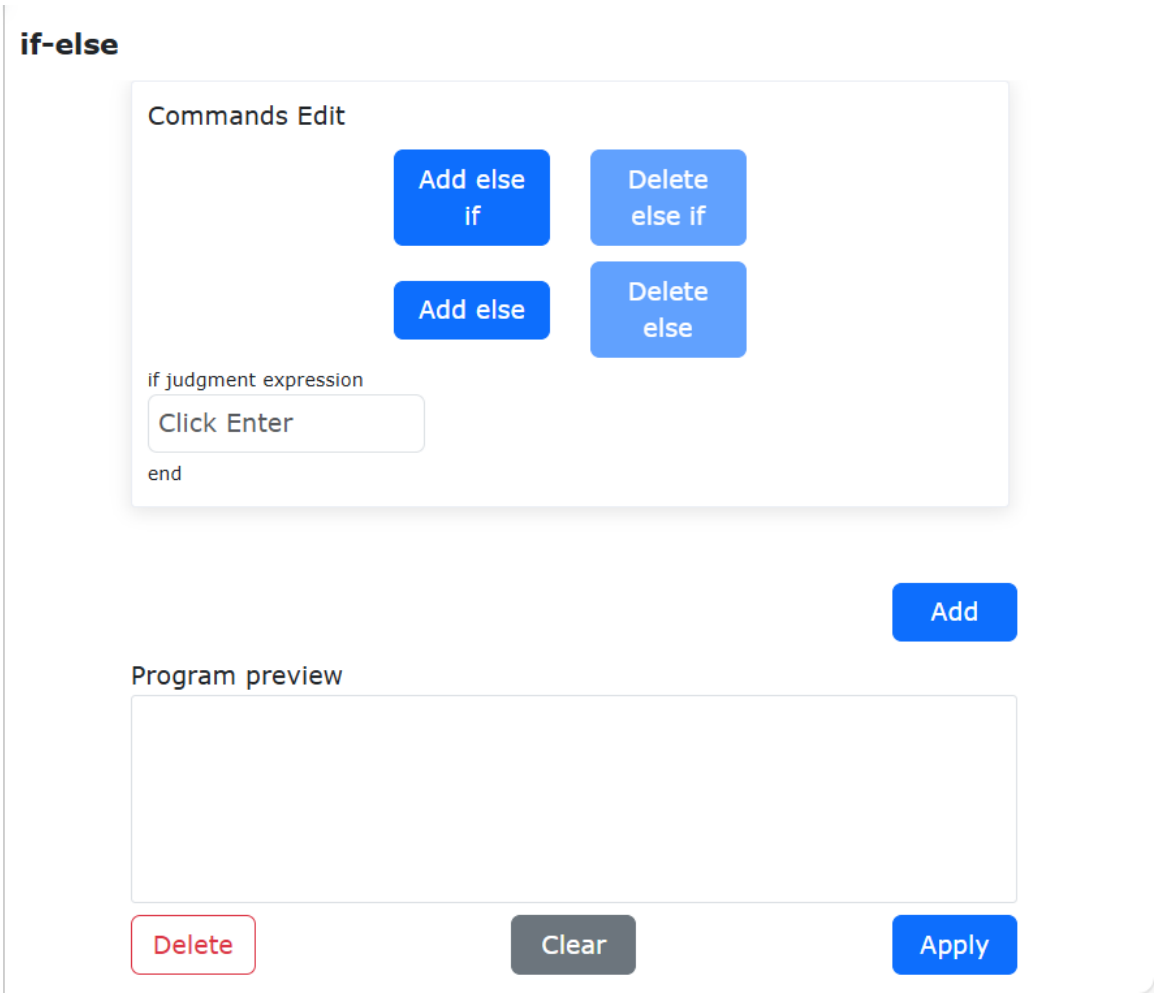


Figure 9.4-2 if…else command interface

1.3.9.4.3 Goto command

Click the “Goto” button to enter the Goto command editing interface.

The Goto instruction is a jump instruction, enter the statement in the input box on the right, and click “Add” and “Apply” after editing. (This instruction requires a certain programming foundation, if you need help, please contact us)

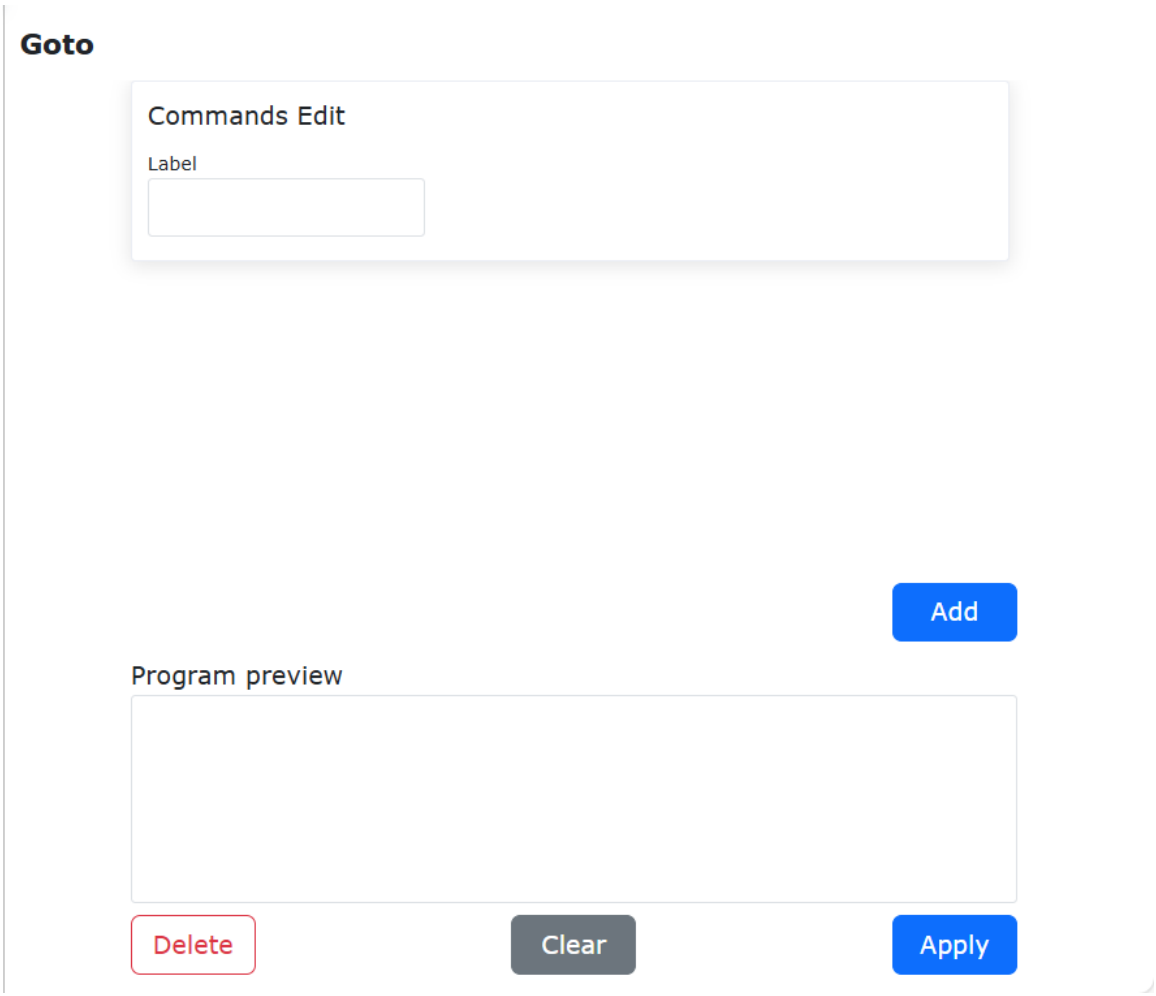


Figure 9.4-3 Goto command interface

1.3.9.4.4 Wait command

Click the “Wait” icon to enter the Wait command editing interface.

This instruction is a delay instruction, which is divided into three parts: “WaitMs”, “WaitDI” and “WaitAI”.

“WaitTime” command delay waiting time unit is milliseconds, input the number of milliseconds to wait, click “Add”, “Apply”.

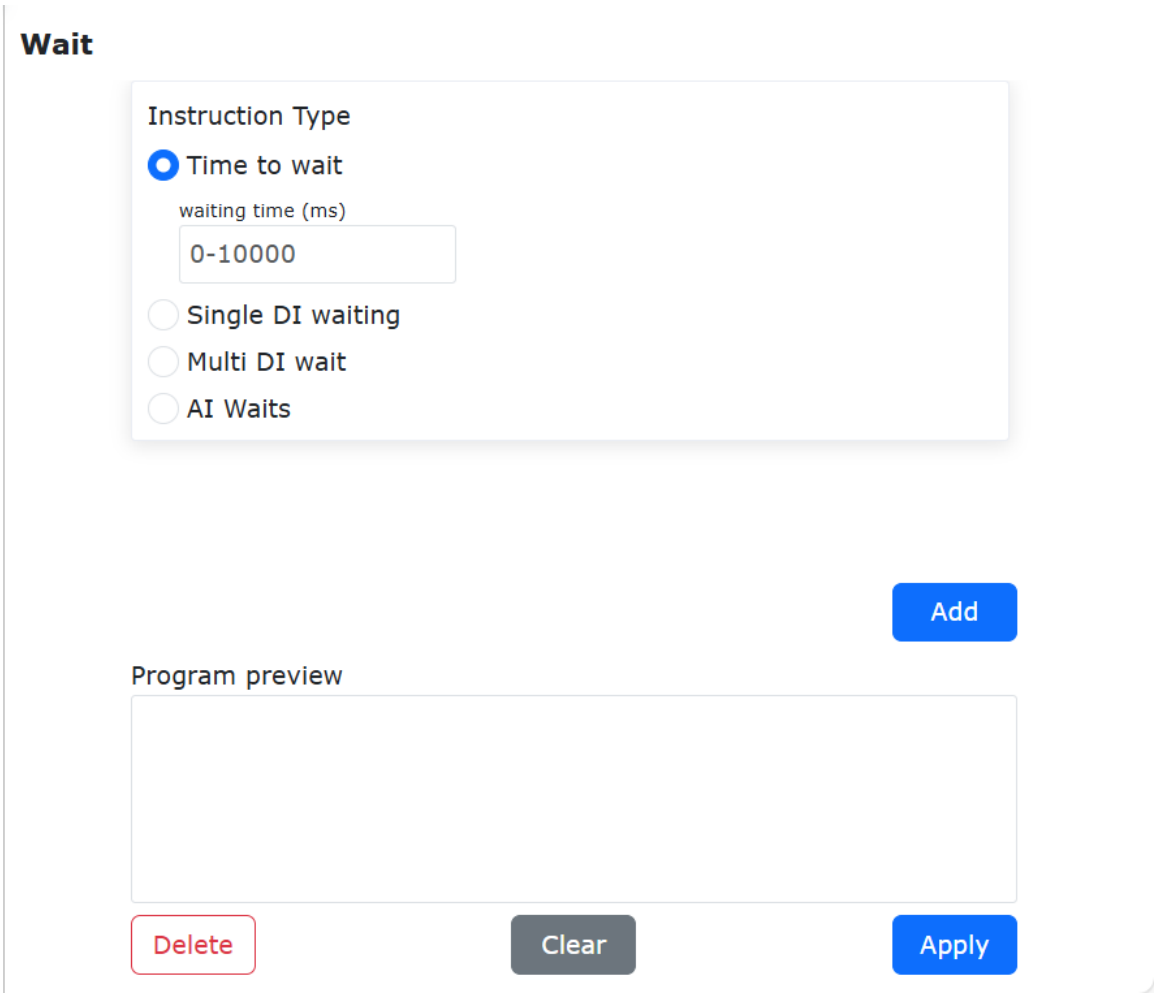


Figure 9.4-4 WaitTime command interface

“WaitDI” command, that is, single DI waiting, select the IO port number to be waited for, wait state, wait maximum time and wait timeout processing method, and click “Add” and “Apply”.

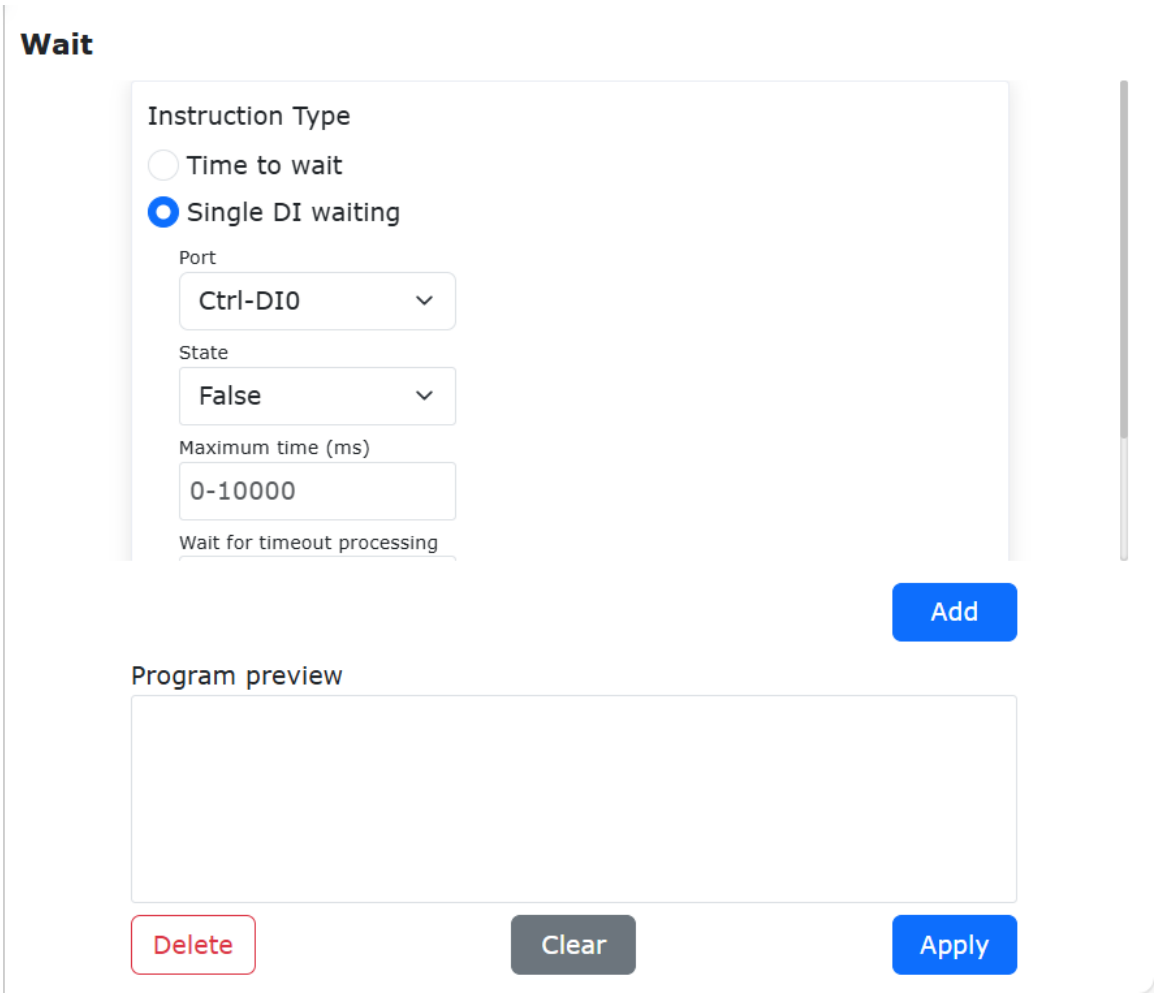


Figure 9.4-5 WaitDI command interface

"WaitMultiDI" command, that is, multi-DI waiting, first select the multi-DI establishment conditions, then check the DI port and status that need to wait, and finally set the maximum waiting time and waiting timeout processing method, click "Add" and "Apply".

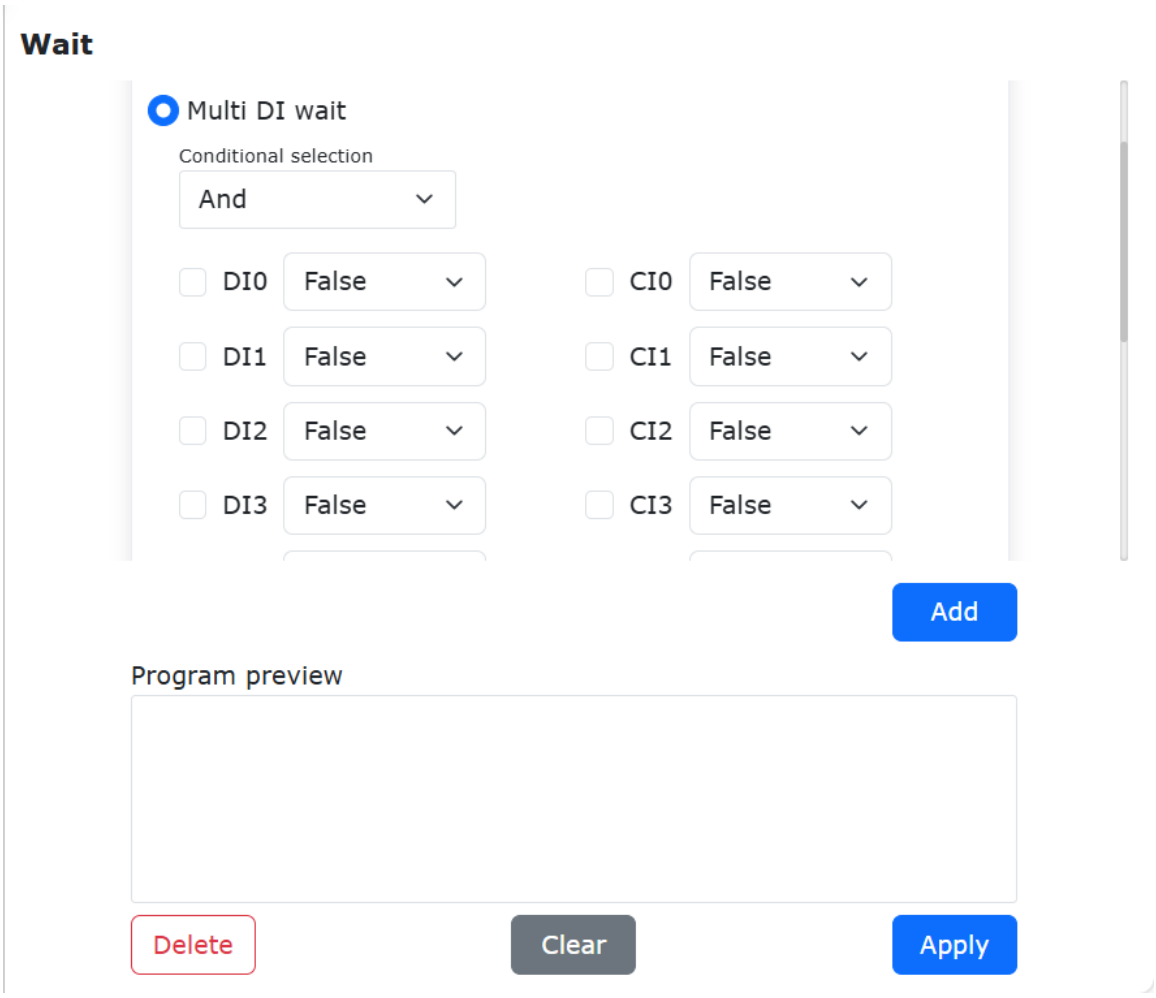


Figure 9.4-6 WaitMultiDI command interface

"WaitAI" command, select the analog quantity to be waited for, the value, the maximum waiting time and the waiting timeout processing method, and click "Add" and "Apply".

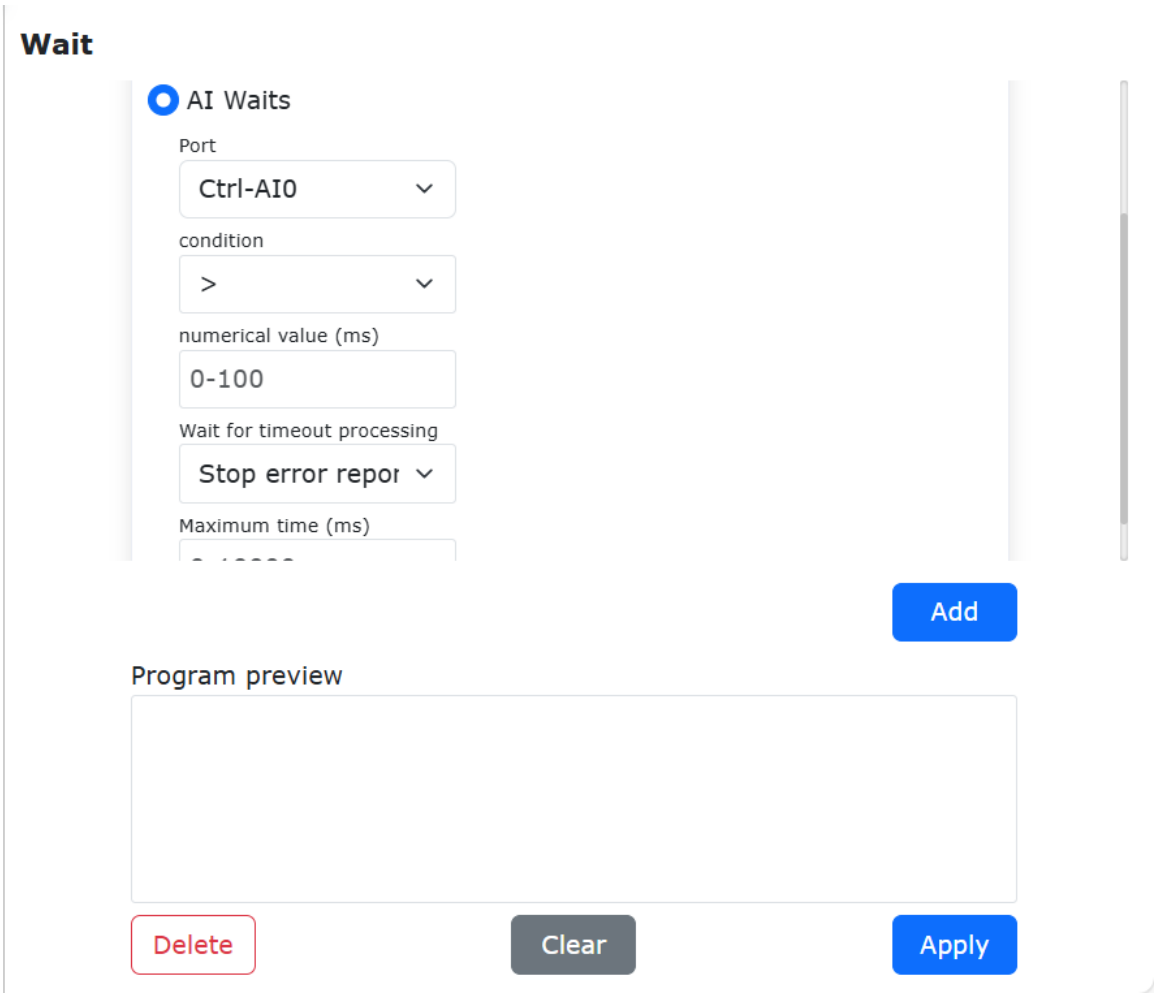


Figure 9.4-7 WaitAI command interface

1.3.9.4.5 Pause command

Click the “Pause” icon to enter the Pause command editing interface.

This instruction is a pause instruction. Insert this instruction into the program. When the program executes this instruction, the robot will be in a pause state. If you want to continue running, click the “Pause/Resume” button in the control area.

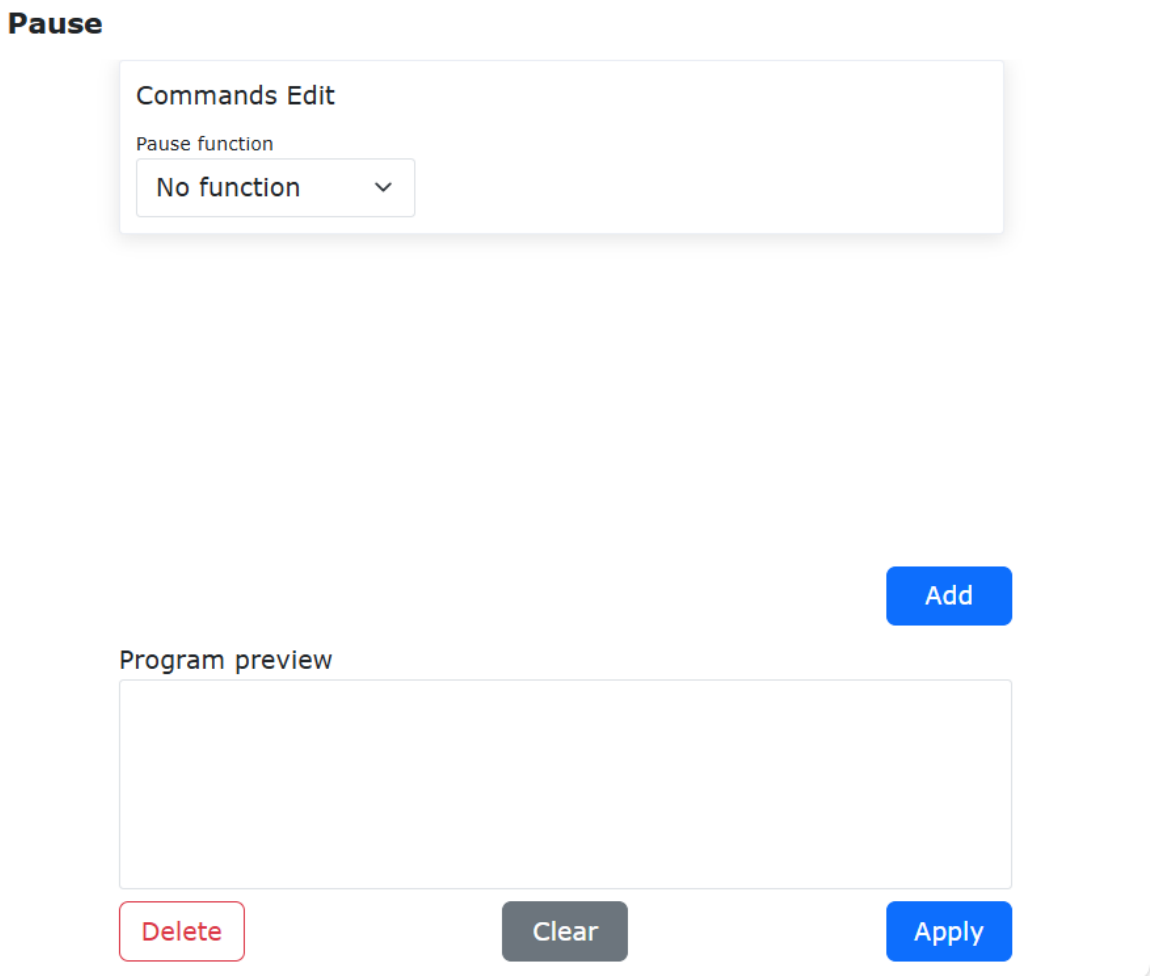


Figure 9.4-8 Pause command interface

1.3.9.4.6 Dofile command

Click the “Dofile” icon to enter the Dofile command editing interface.

The Dofile command calls the internal program of the controller. When using the Dofile command, the called subroutine needs to be saved, and the main program does not need to be saved again if it has not changed. The Dofile command supports two-level calls, and two parameter settings need to be paid attention to. One is the level of the call, and the other is the ID number of the call. In principle, the same ID number cannot appear in the same program.

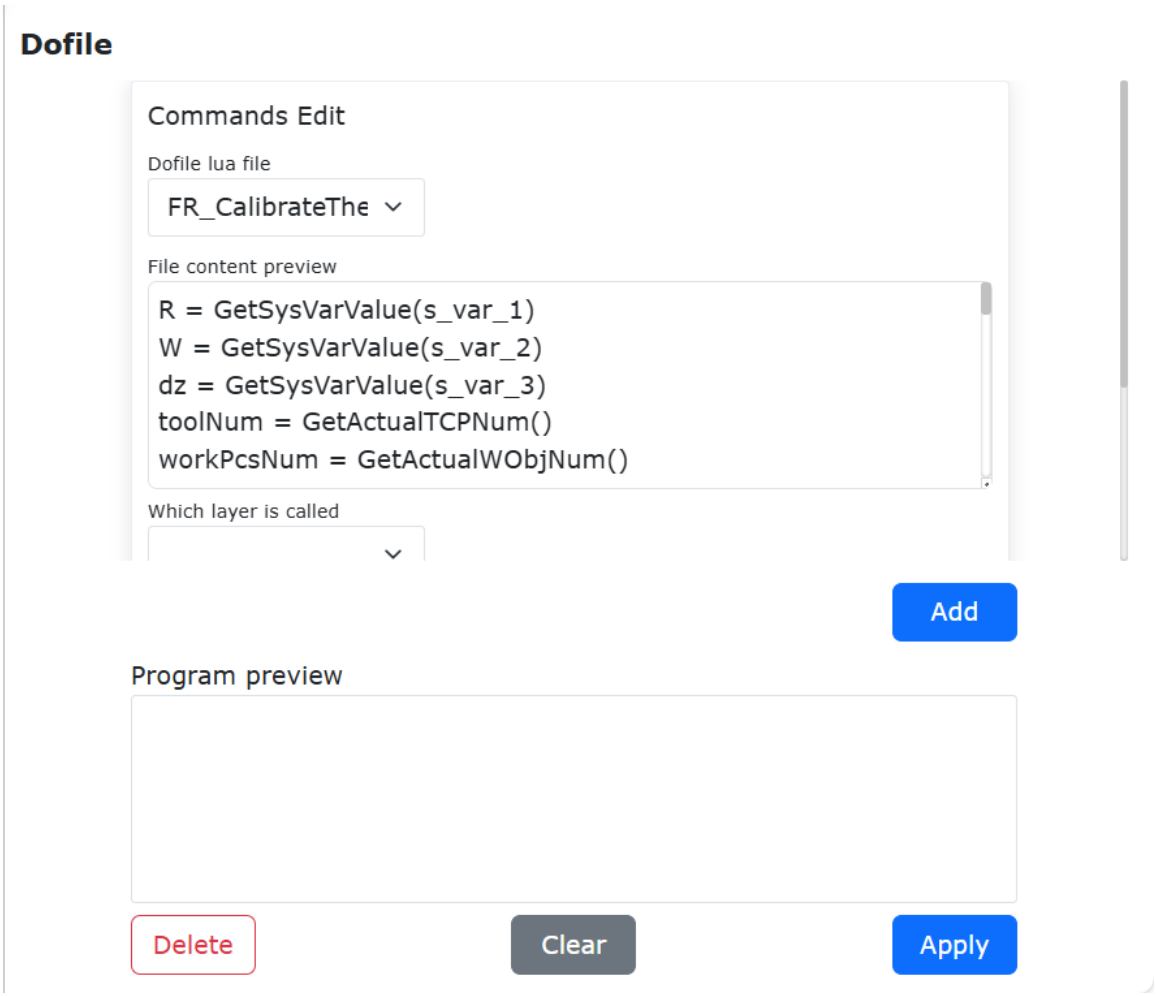


Figure 9.4-9 Dofile command interface

1.3.9.4.7 Var command

Click the “Var” icon to enter the Var command editing interface.

This command is a variable system command, which is divided into two parts: Lua variable definition, variable query and Sys variable renaming, getting value, and setting value. Lua variable definition can declare a variable and assign an initial value, and cooperate with while, if-else and other commands Use the Lua variable query command to query the value of the input variable name in real time and display it in the status bar. The number of Sys variables is fixed, and you can rename them, get variable values, and set variable values. The values stored in this variable will not be cleared when the system is turned off.

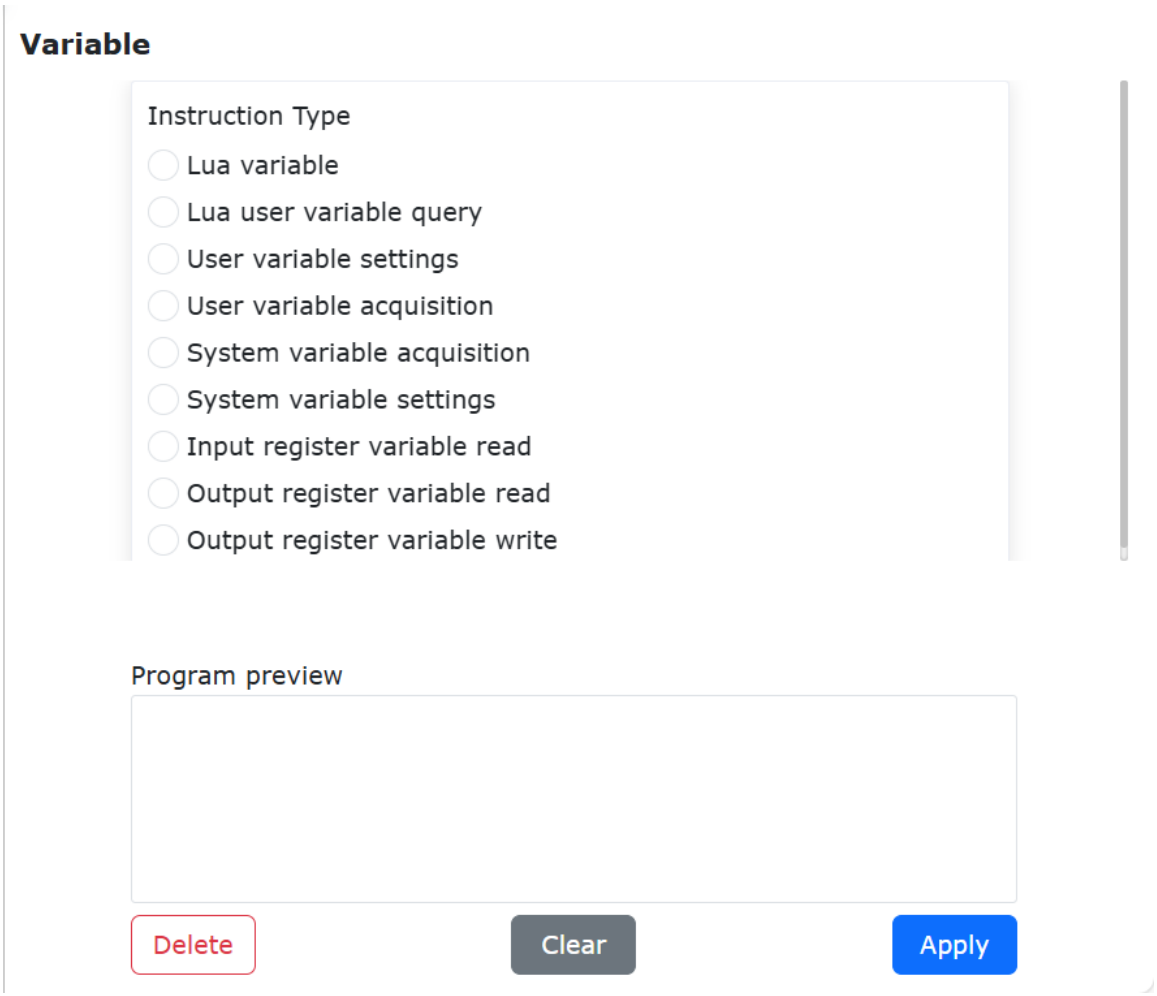


Figure 9.4-10 Var command interface

Important: Variable names must start with letters or underscores, not numbers or other special characters.

1.3.9.5 Motion command interface



Figure 9.5 Motion command interface

1.3.9.5.1 PTP command

Click the “PTP” icon to enter the PTP command editing interface.

You can choose the point to be reached, and the smooth transition time setting can realize that the movement from this point to the next point is continuous. Whether to set the offset, you can choose to offset based on the base coordinate system and based on the tool coordinates, and pop up x, y, z, rx, ry, rz offset settings, PTP specific path is the optimal path automatically planned by the motion controller, click “Add” and “Apply” to save this command.

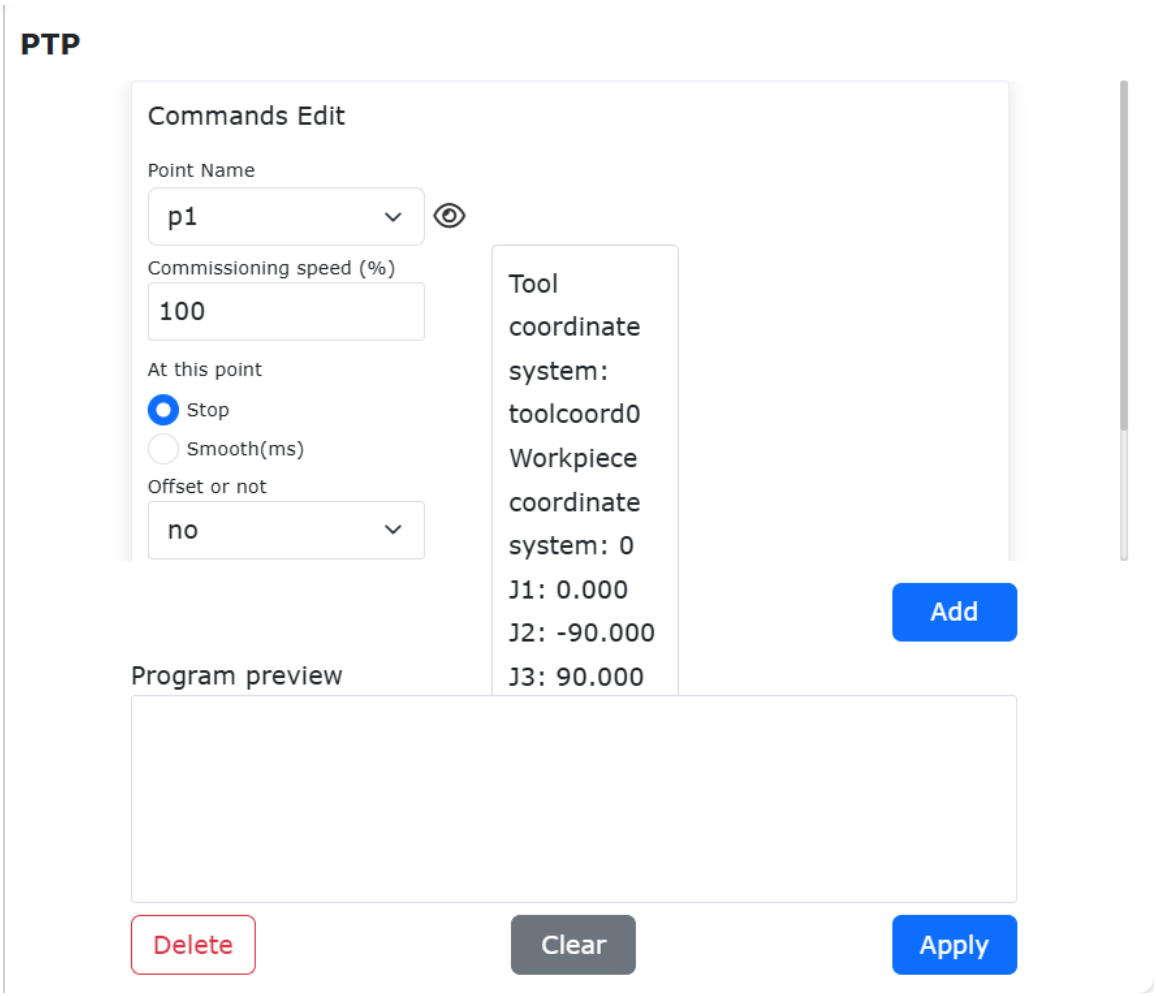


Figure 9.5-1 PTP command interface

1.3.9.5.1.1 Point-to-Point Relative Motion

The robot moves a relative distance from its current position. On the PTP command addition page, select the point name as “CurrentPos”. Choose the offset coordinate system as either the base coordinate system or the tool coordinate system according to the situation, and input the offset value. This means the robot will perform a certain offset motion along the set coordinate system based on its current position. (“CurrentPos” is a system point and does not require teaching.)

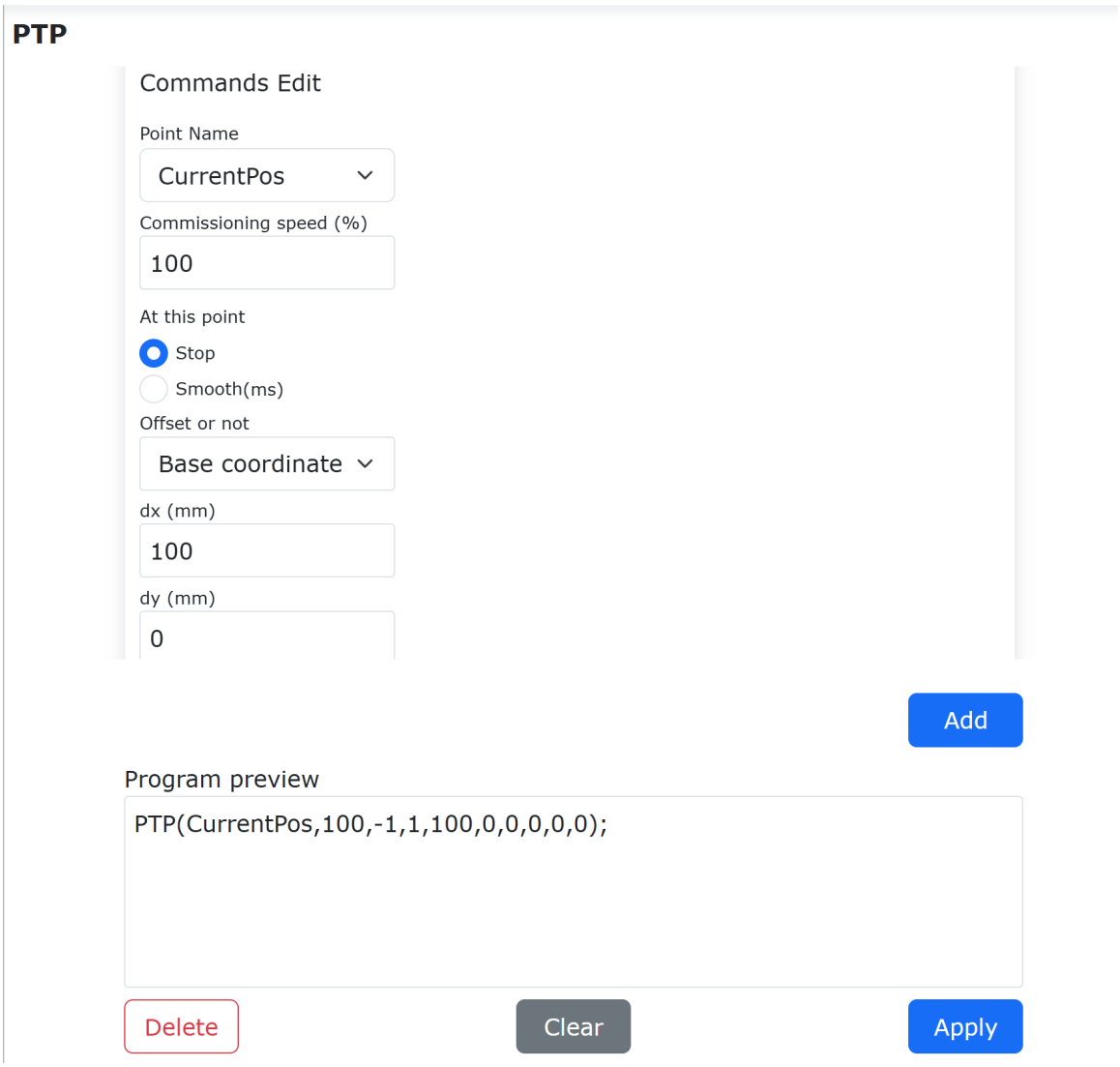


Figure 9.5-1-1 PTP Relative Motion Command

Click the “Add” and “Apply” buttons to add a robot PTP relative motion command to the Lua program. Switch the robot to automatic mode and click the start button. In the example program, the robot will move 100mm along the X+ direction of the base coordinate system from its current position.

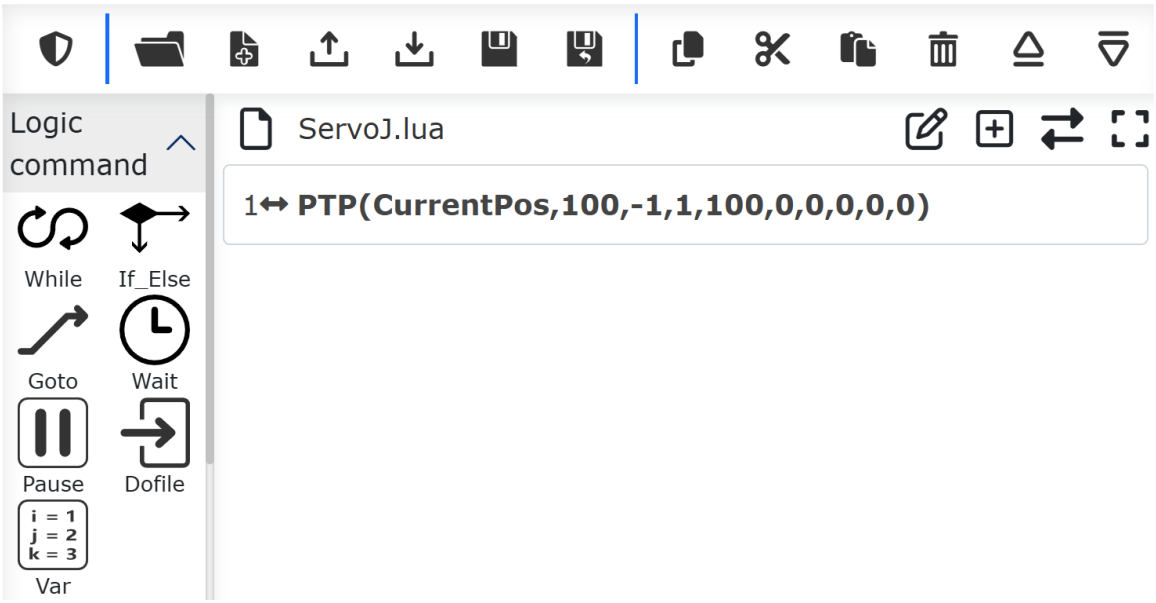


Figure 9.5-1-2 PTP Relative Motion Command Added

1.3.9.5.2 Lin command

Click the “Lin” icon to enter the Lin command editing interface.

The function of this command is similar to the “PTP” command, but the path of the point reached by this command is a straight line.

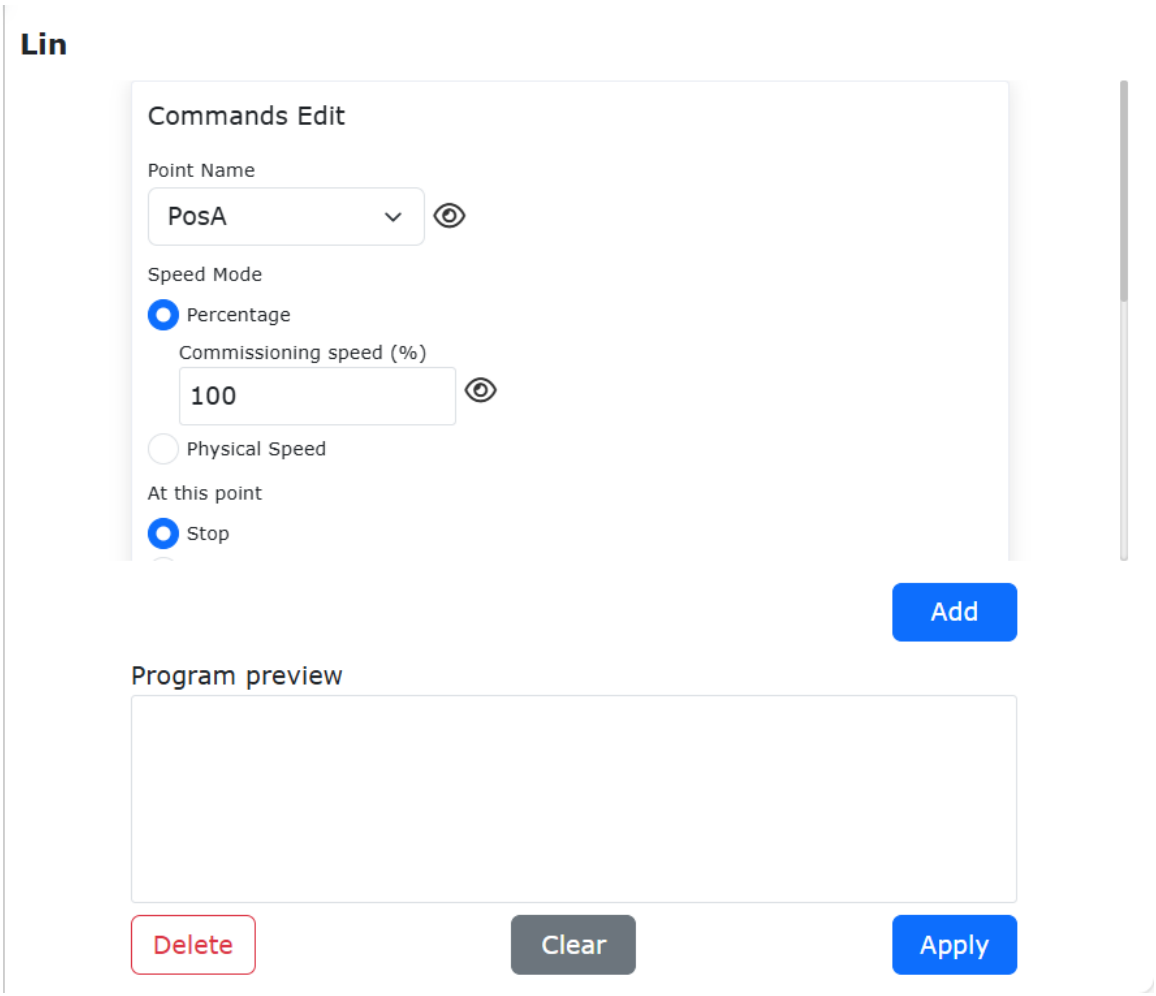


Figure 9.5-2 Lin command interface

1.3.9.5.2.1 Linear Relative Motion

The robot moves a relative distance along a straight line from its current position. On the LIN command addition page, select the point name as “CurrentPos”. Choose the offset coordinate system as either the base coordinate system or the tool coordinate system according to the situation, and input the offset value. This means the robot will perform a certain offset motion along the set coordinate system based on its current position. (“CurrentPos” is a system point and does not require teaching.)

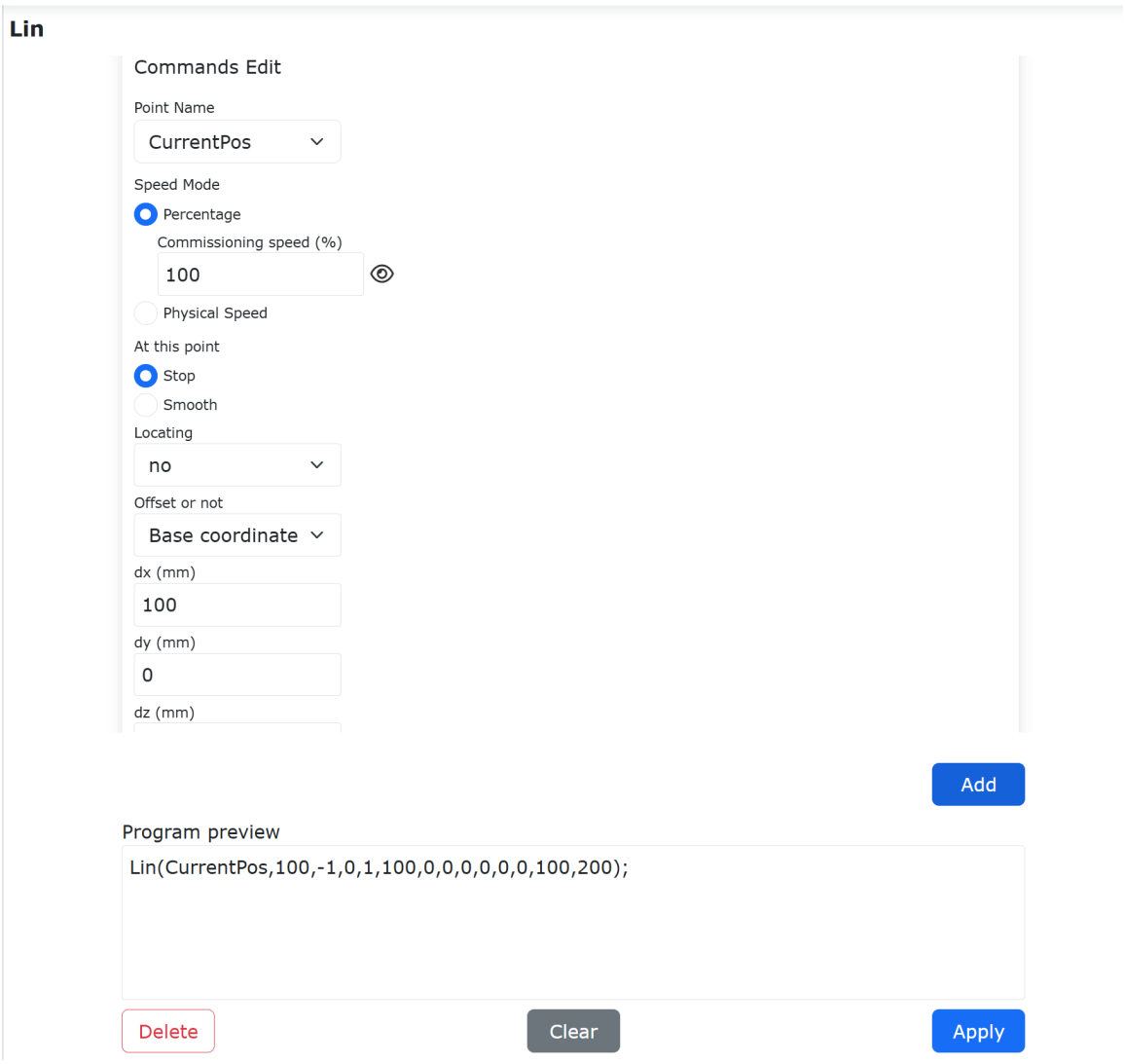


Figure 9.5-2-1 LIN Relative Motion Command

Click the “Add” and “Apply” buttons to add a robot linear relative motion command to the Lua program. Switch the robot to automatic mode and click the start button. In the example program, the robot will move 100mm along the X+ direction of the base coordinate system from its current position.

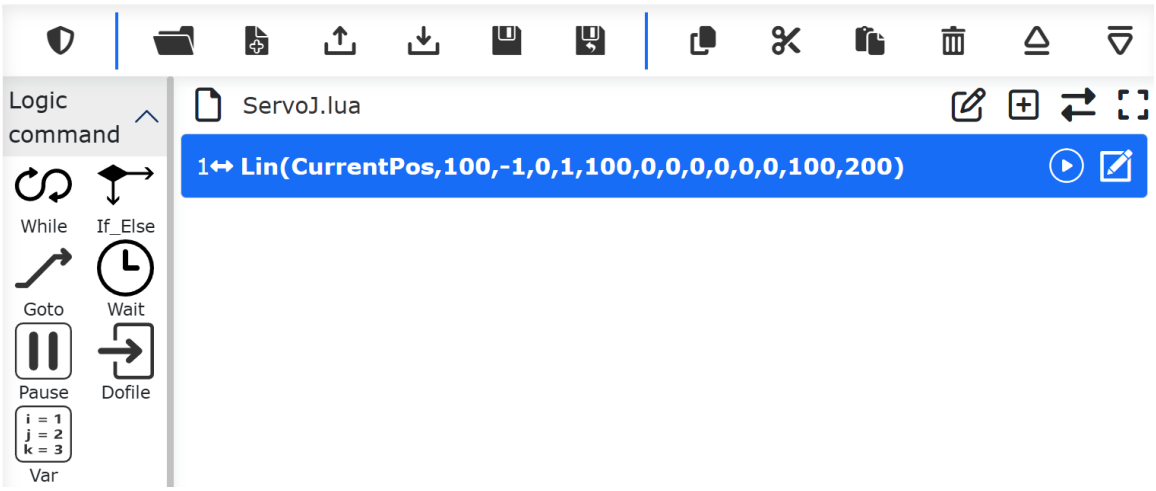


Figure 9.5-2-2 LIN Relative Motion Command Added

Robot linear relative motion supports functions such as smoothing, physical speed, and welding wire search.

Important: When the selection point name is “seamPos”, the straight line command is applied in the welding scene using the laser sensor. Due to the accumulated operating error during welding, “whether to offset” and “offset amount” are added.

Whether to offset: No, base coordinate system offset, tool coordinate system offset, laser original data offset;

Offset: x, y, z, rx, ry, rz, range: -300~300;

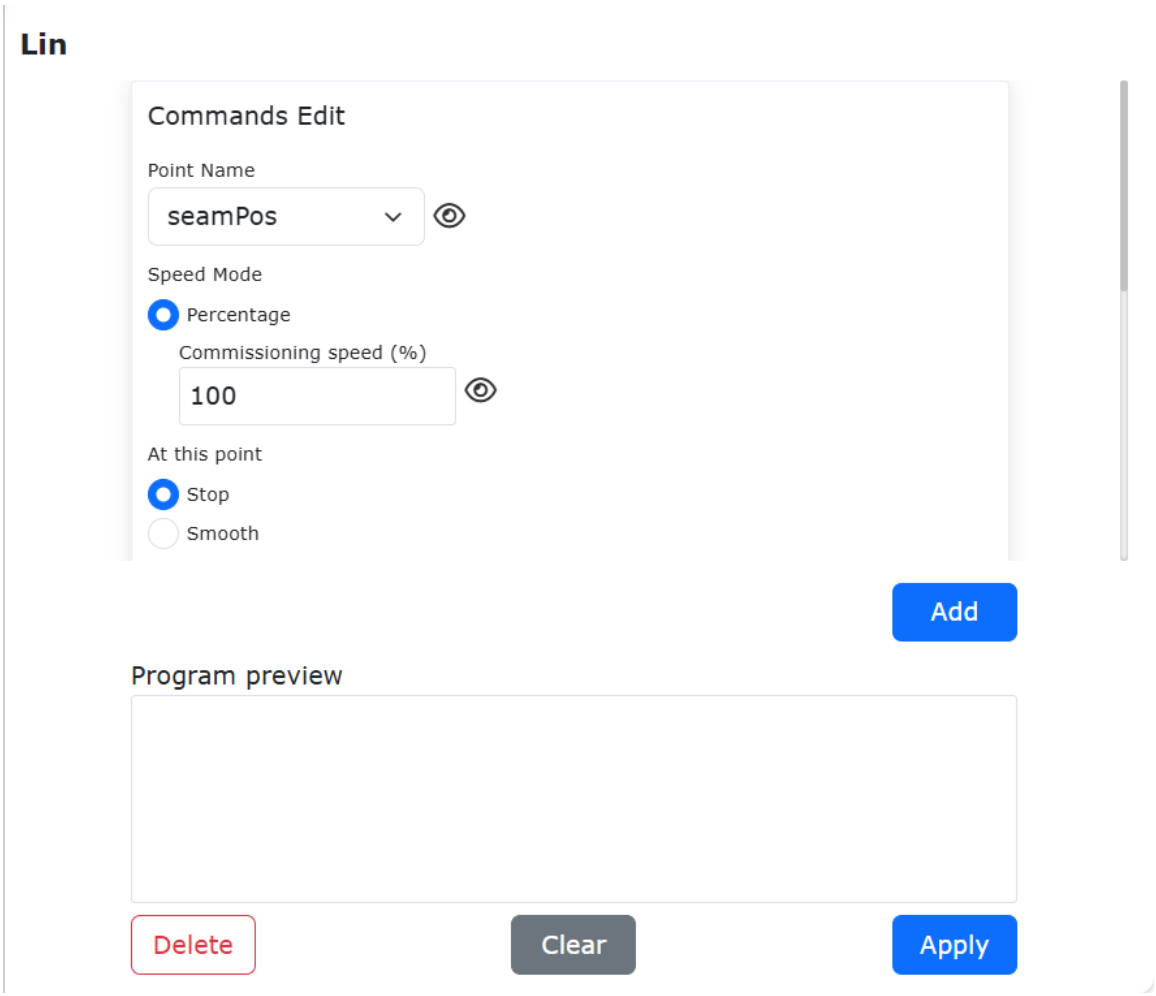


Figure 9.5-2-3 Lin command interface(Welding scene)

The LIN command allows selecting the motion speed mode as “Percentage” or “Physical Speed”:

- Percentage: Input the debugging speed percentage. The robot moves at a percentage of its maximum speed. The actual robot motion speed is calculated as: V = Robot Maximum Speed × Global Speed Percentage × Debugging Speed Percentage. Hover the mouse over the small eye icon to the right of the “Debugging Speed” input box to display the actual physical speed (unit: mm/s) of the robot in both manual and automatic modes under the current debugging speed setting.

Lin

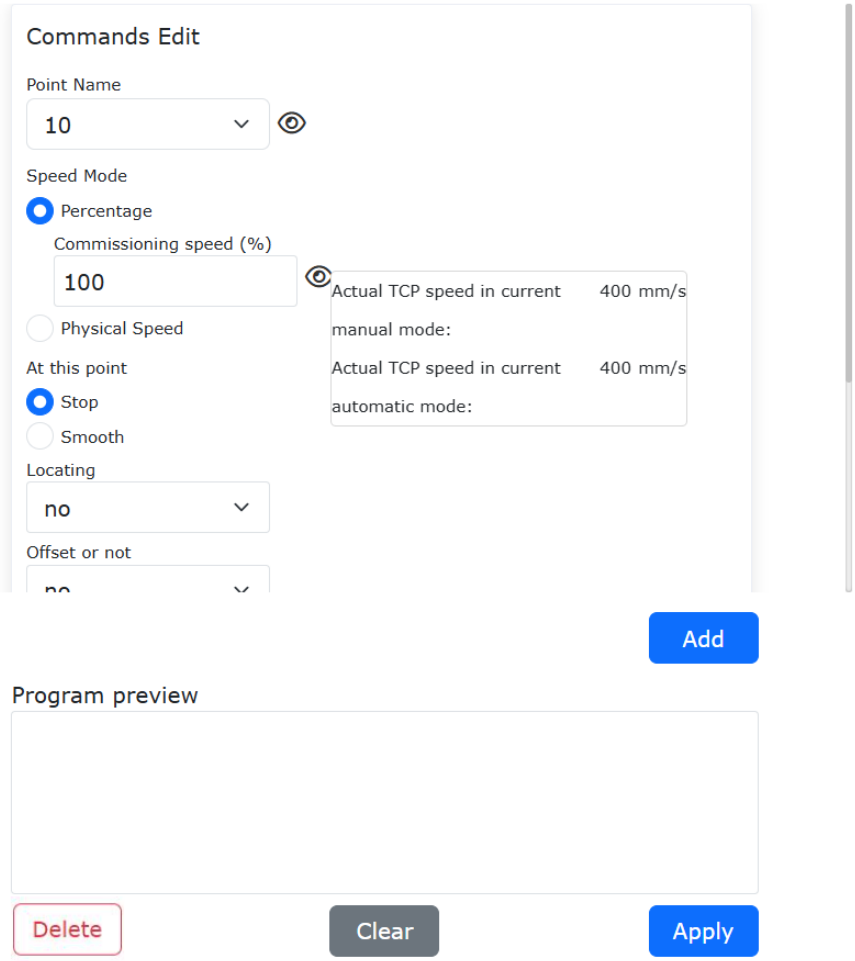


Figure 9.5-2-4 Input Percentage to Display Actual Physical Speed Value

- Physical Speed: The input speed is the actual operating speed of the robot, unit mm/s; the input acceleration is usually set to twice the speed. (The maximum physical speed of the LIN command is limited by the global speed percentage. If the robot's maximum operating speed is 1000mm/s and the global speed is 50%, then the maximum physical speed for the LIN command is 1000 × 50% = 500mm/s).

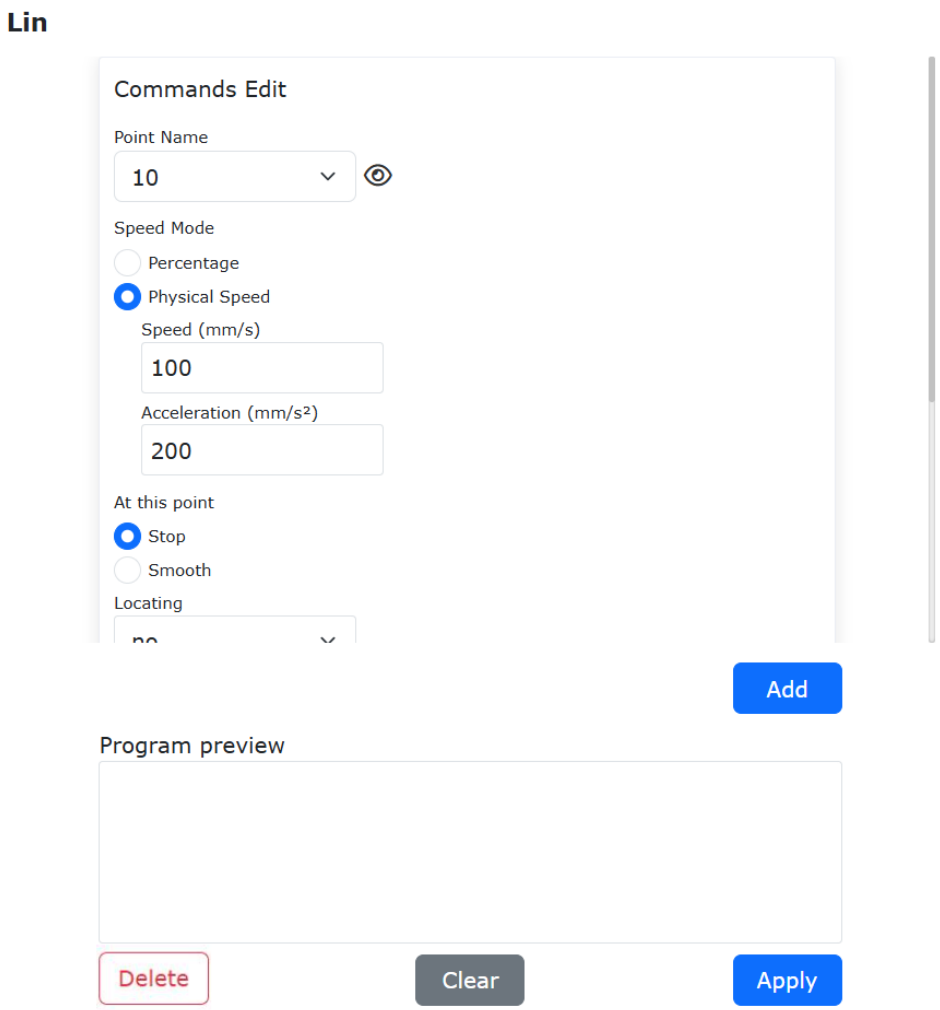


Figure 9.5-2-5 Input Actual Physical Speed

1.3.9.5.2.2 LIN command joint overspeed processing function

When using the Cartesian space linear motion command LIN, the constrained condition is the linear velocity, but the actual operation is affected by the workspace, and the angular velocity of the joint may have exceeded the limit when the linear velocity requirement is met. This function implements an optional handling strategy to deal with joint overspeed during LIN movements.

Step1: Click the Linear Motion Command button;



Figure 9.5-3-1 Click the Linear Motion Command button

Step2: Select the linear motion command target waypoint;

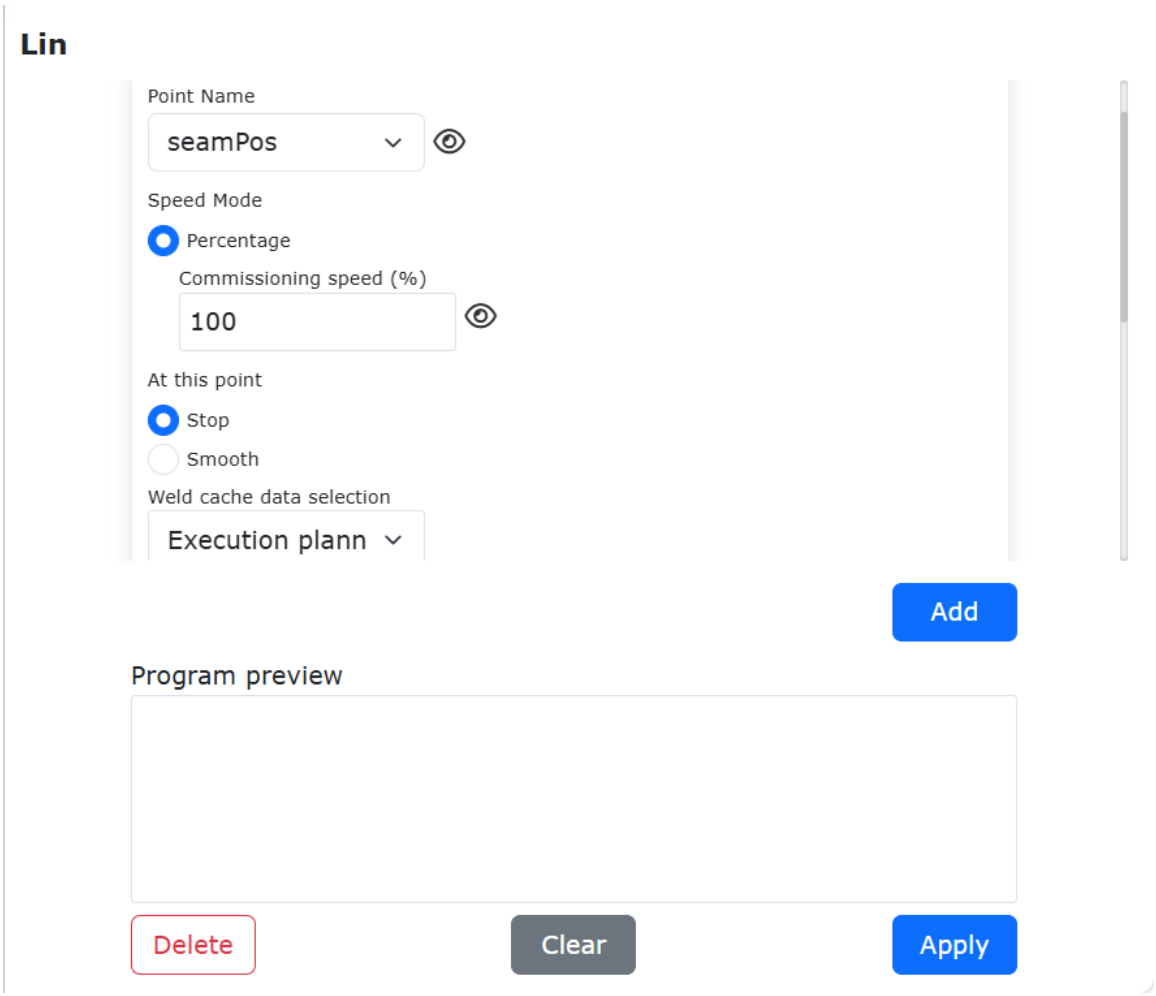


Figure 9.5-3-2 Select the linear motion target waypoint

Step3: Turn on the joint overspeed protection switch;

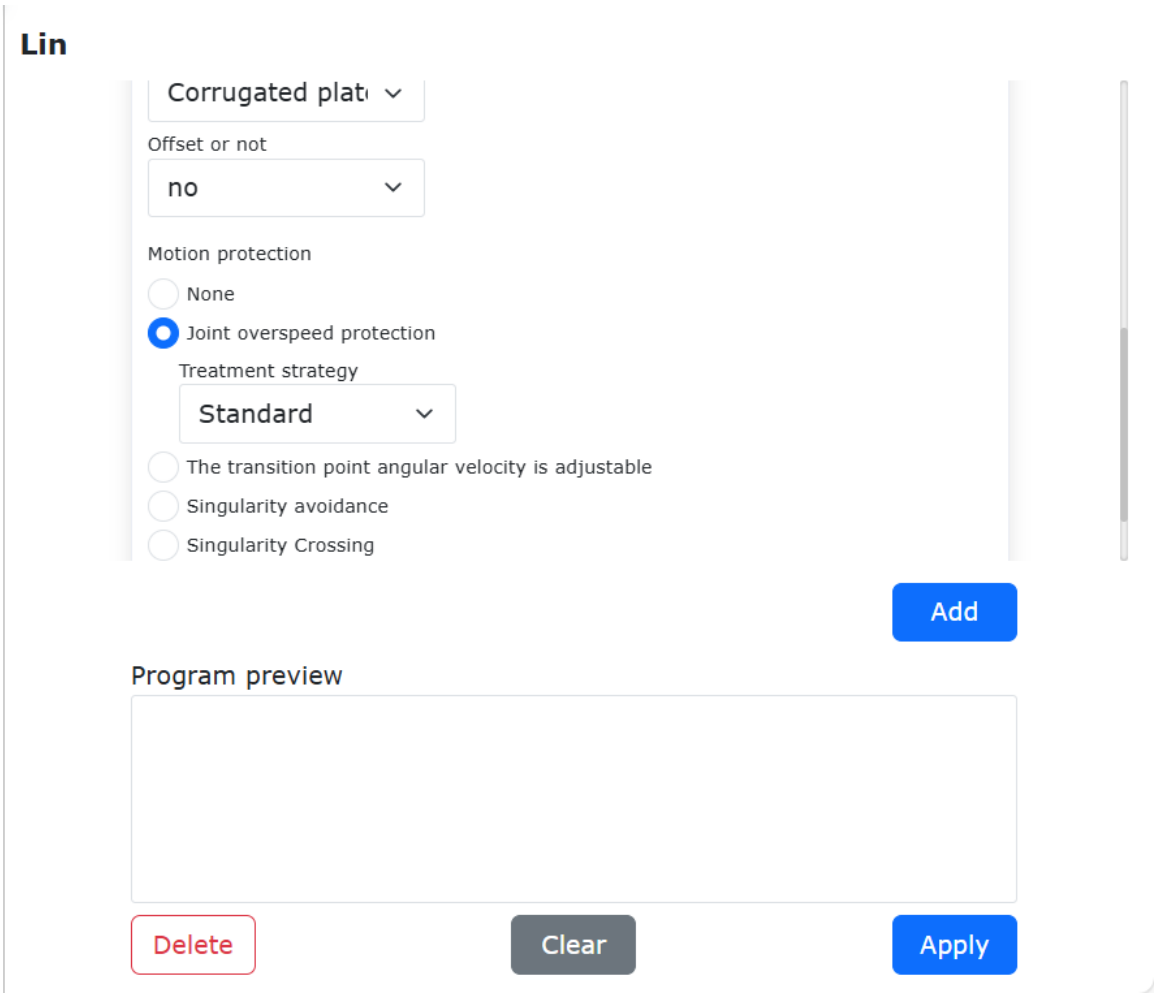


Figure 9.5-3-3 Turn on the joint overspeed protection switch button

Step4: Select the joint overspeed treatment strategy (first two options is not for joint overspeed treatment);

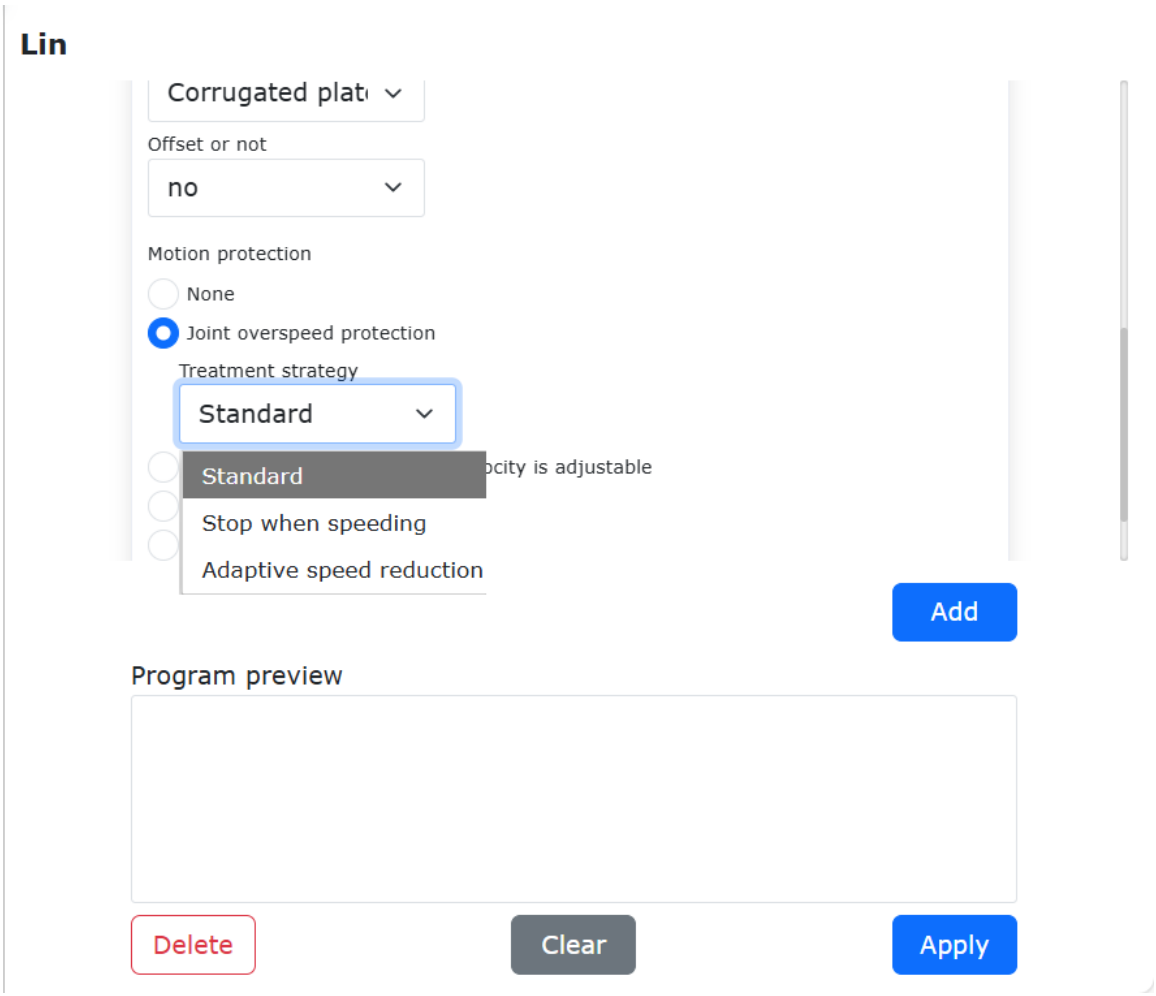


Figure 9.5-3-4 Joint overspeed treatment strategies

Step5: Set the processing option and processing parameters, then click the Add button to add the Lua command;

Under the adaptive speed reduction strategy, the deceleration threshold is the percentage of the linear velocity reduction value relative to the set linear velocity, and when the deceleration value exceeds the set threshold, the robot will report an error and stop.

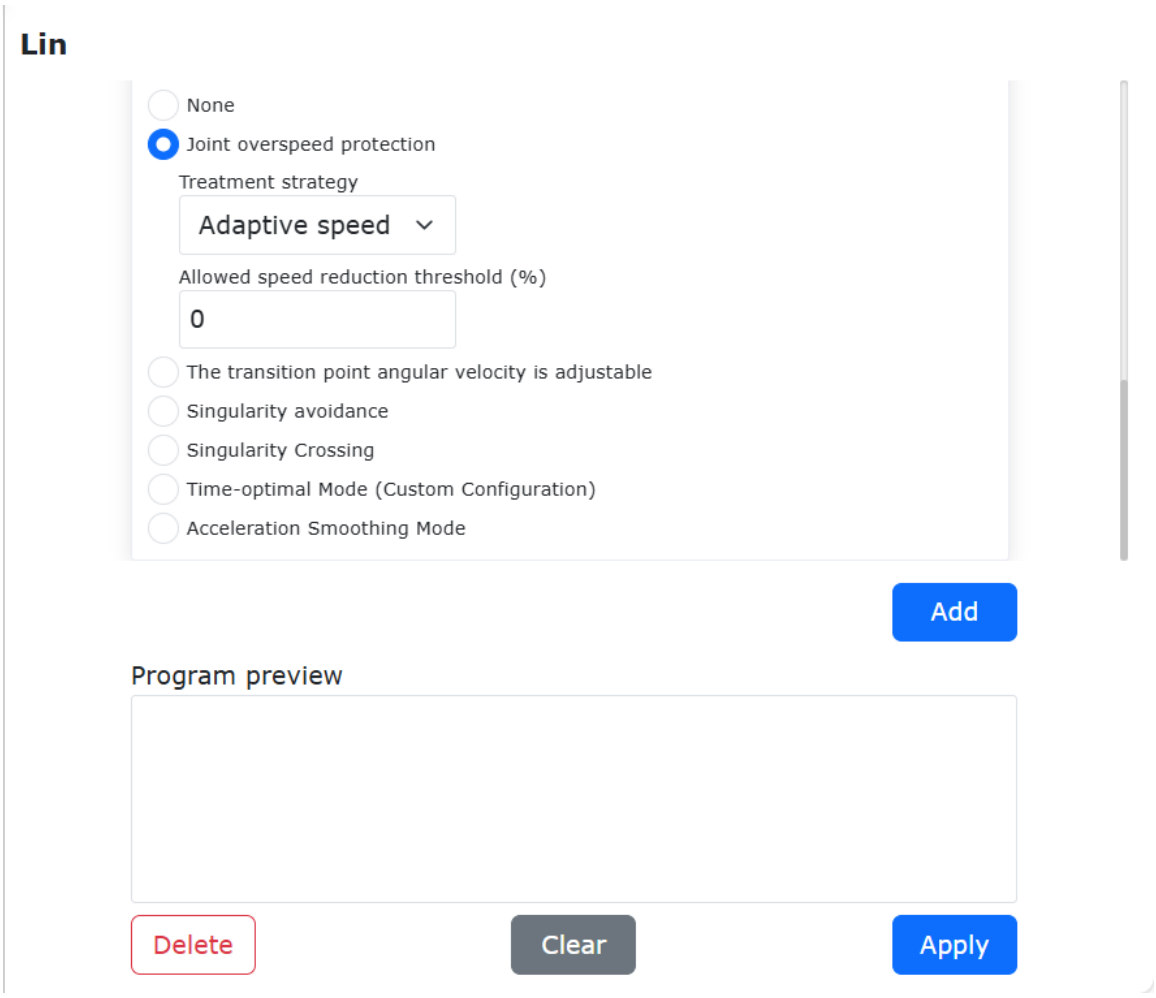


Figure 9.5-3-5 Joint overspeed treatment strategy selection and setting

Step6: The added Lua command is shown in the figure;

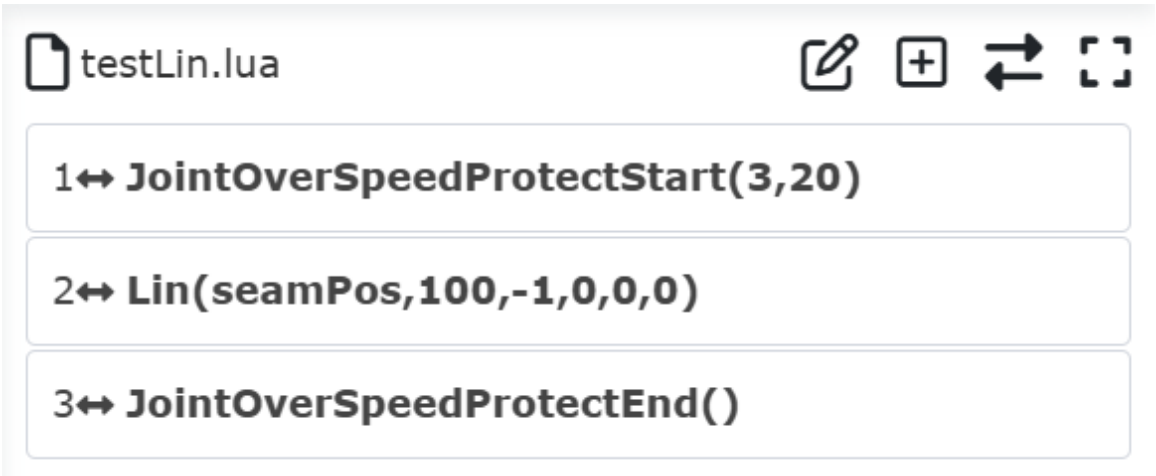


Figure 9.5-3-6 lua command

Joint overspeed protection begins:JointOverSpeedProtectStart(a,b);

a: strategy type number(same as the order of drop-down box)

b: threshold(0~100)

Joint overspeed protection ends:JointOverSpeedProtectEnd();

Note: For the "singularity crossing" motion protection, please refer to the description of the singularity crossing function in automatic mode.

1.3.9.5.2.3 Angular velocity adjustable function

This function can be used when encountering a workpiece that requires corner welding during the welding process, or when a specific linear line is planned (a quick transition is required when the attitude change is large and the position change is small, but the linear speed cannot be accelerated).

Step1:Set the tool coordinate system to calibrate the tool size and attitude of the welding gun.

Note: The values on the interface are examples only, and the actual tool status shall prevail.

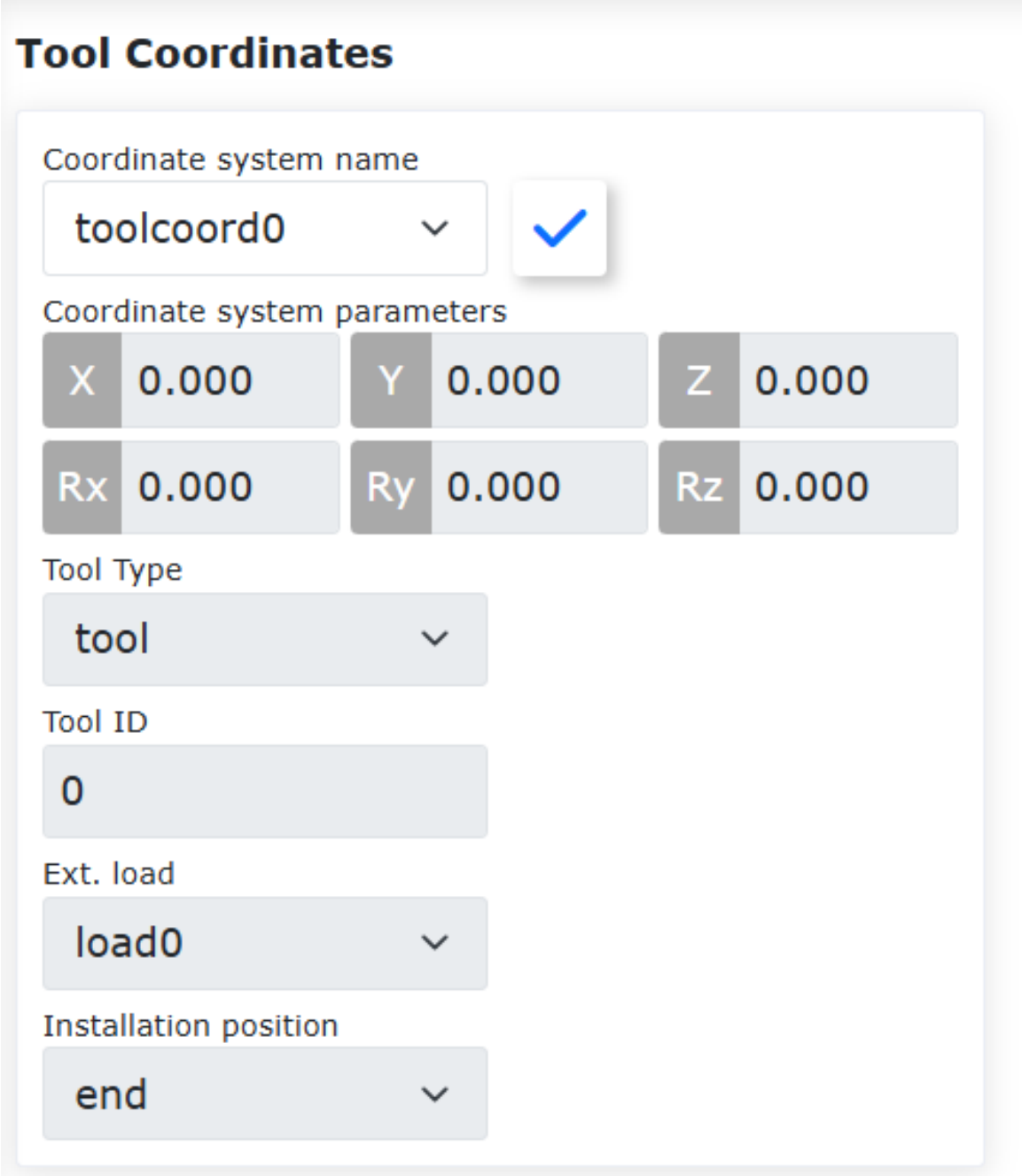


Figure 9.5-3-7 Sets the tool coordinate system

Step2:Click on “Program”, select “Coding”, and select ” LIN” in the “Motion Command” category.

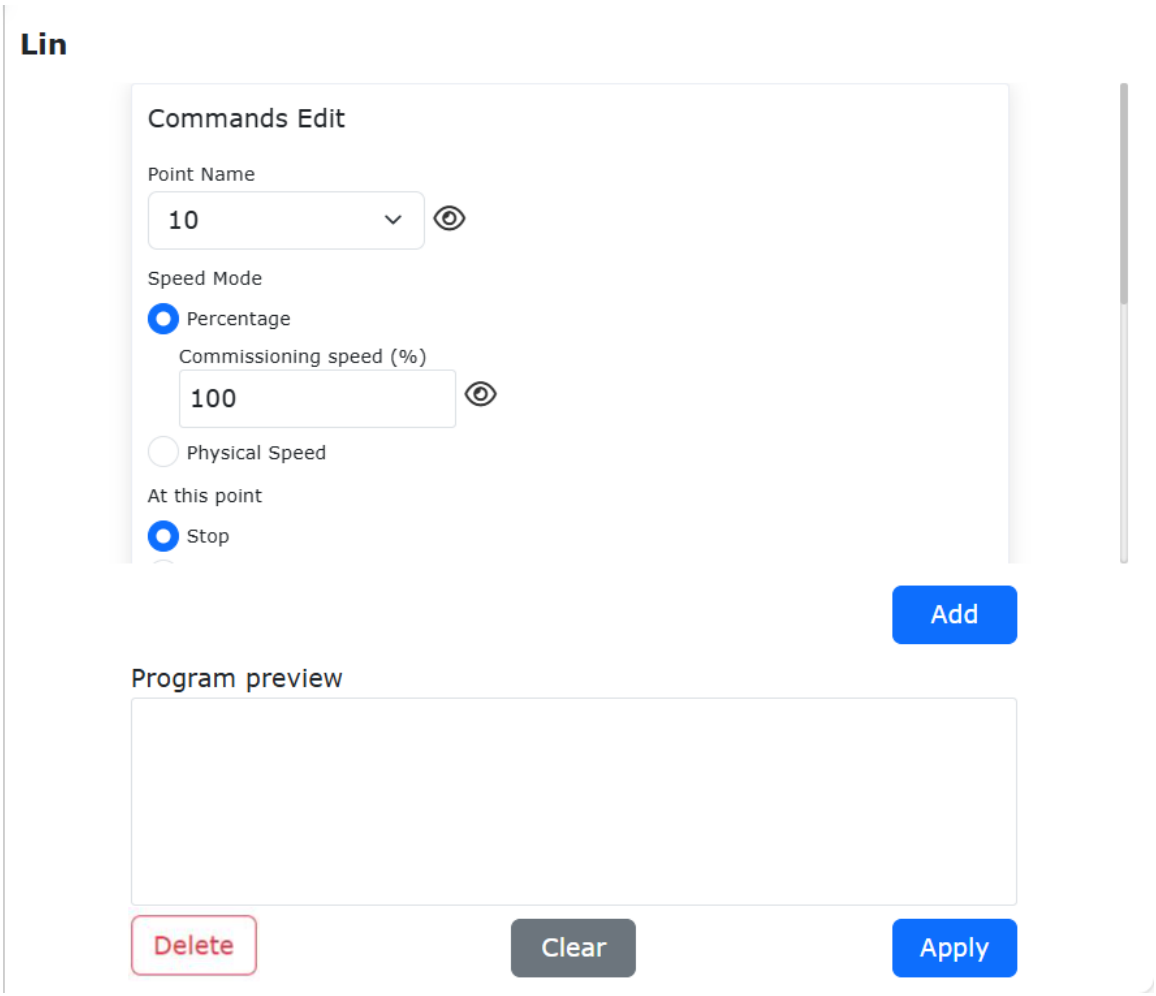


Figure 9.5-3-8 Straight line command setting interface

Step3:Set the starting point of each straight line of wrapping angle welding as the transition point, turn on the "The transition point angular velocity is adjustable" button, and set the maximum acceleration percentage (the default maximum angular velocity of 100% is 360°/s).

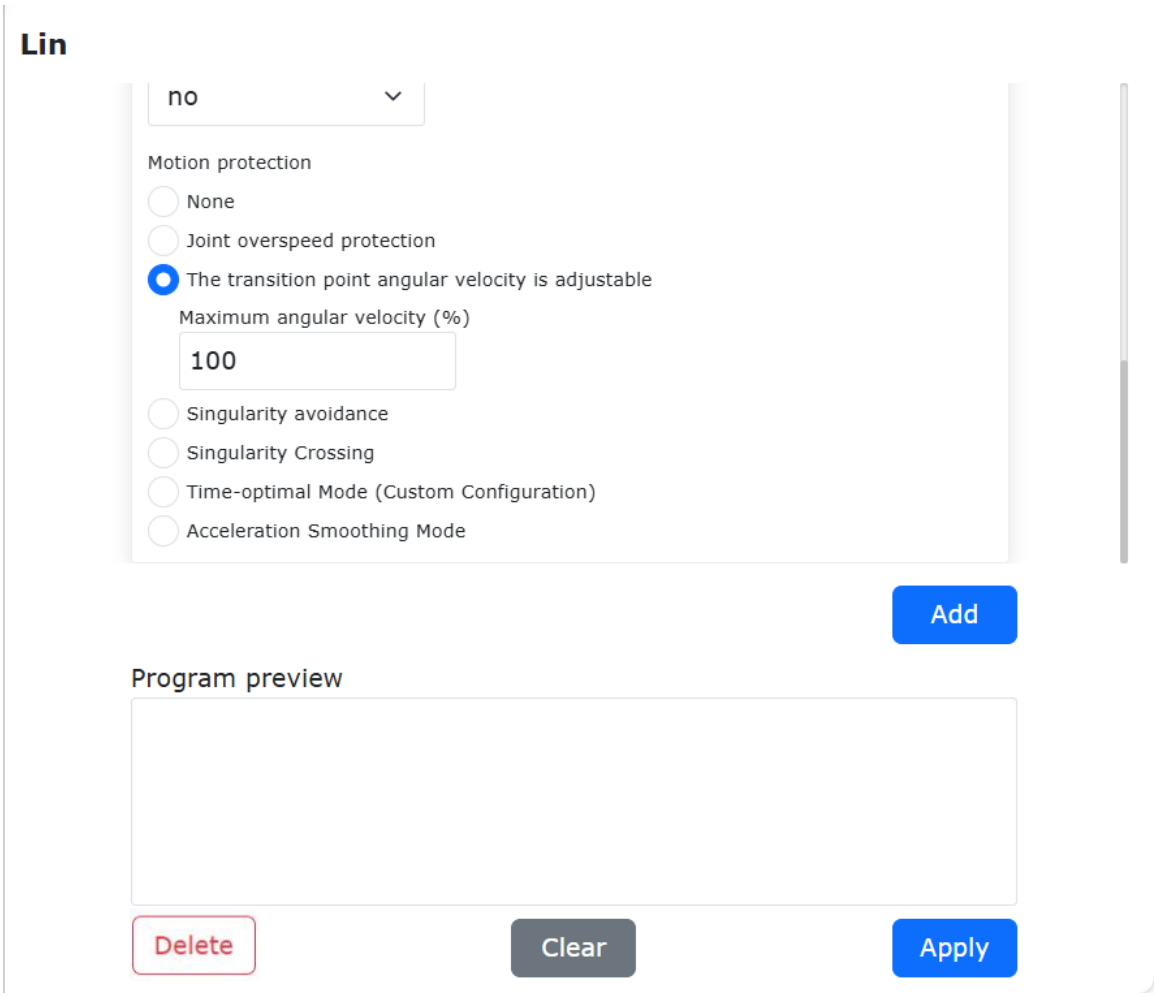


Figure 9.5-3-9 Transition point angular velocity adjustment parameter configuration interface

Step4:Click the “Add” button to generate a LIN command with the adjustment of the transition attitude angular velocity.

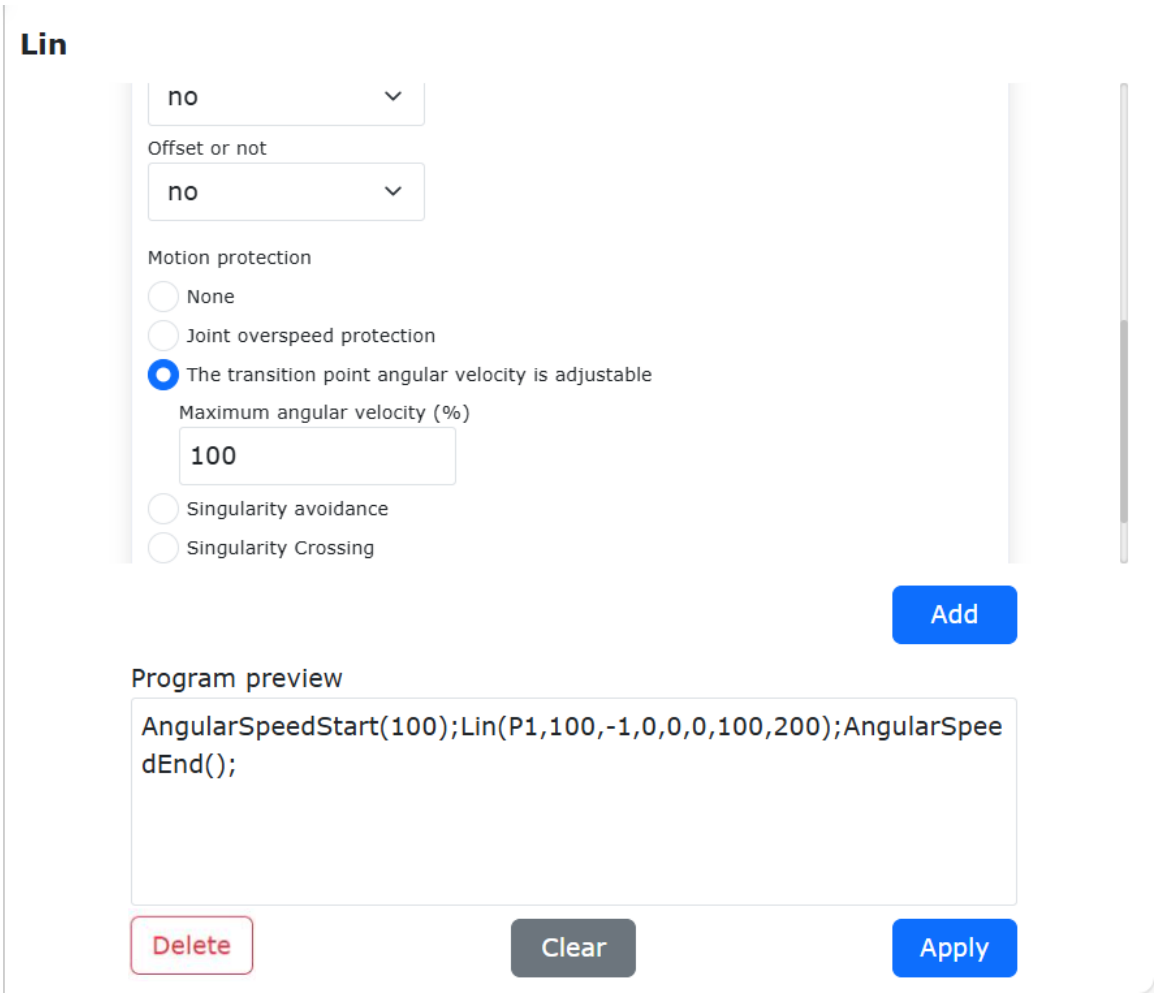


Figure 9.5-3-10 Add a transition point linear motion command

Step5:The robot completes the attitude transition at the starting point, normally executes the linear command movement to the end point of the section, closes the "The transition point angular velocity is adjustable" button, and adds a termination waypoint.

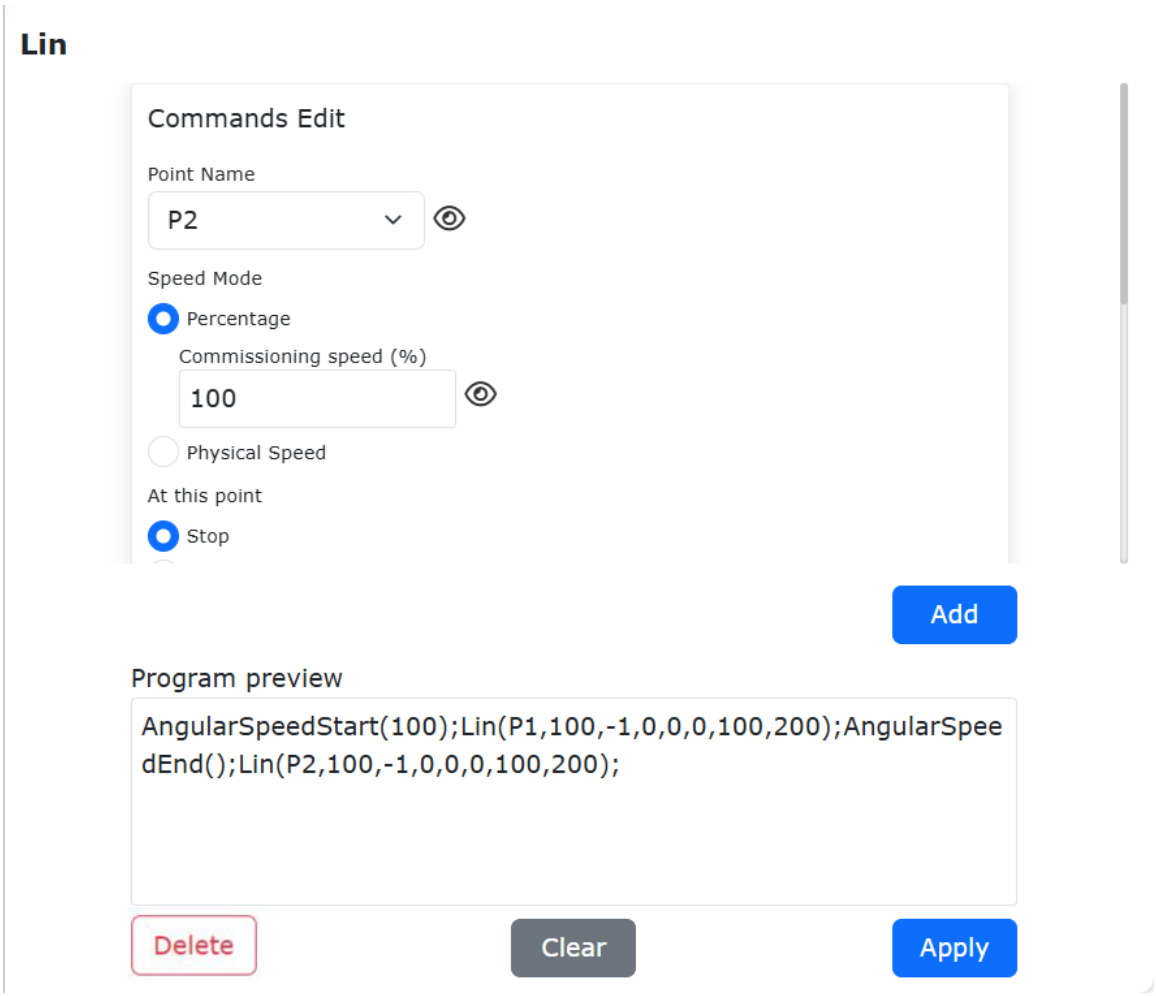


Figure 9.5-3-11 Inserts the end point of the line

Step6:Click the “Apply” button to generate the corresponding LUA command.

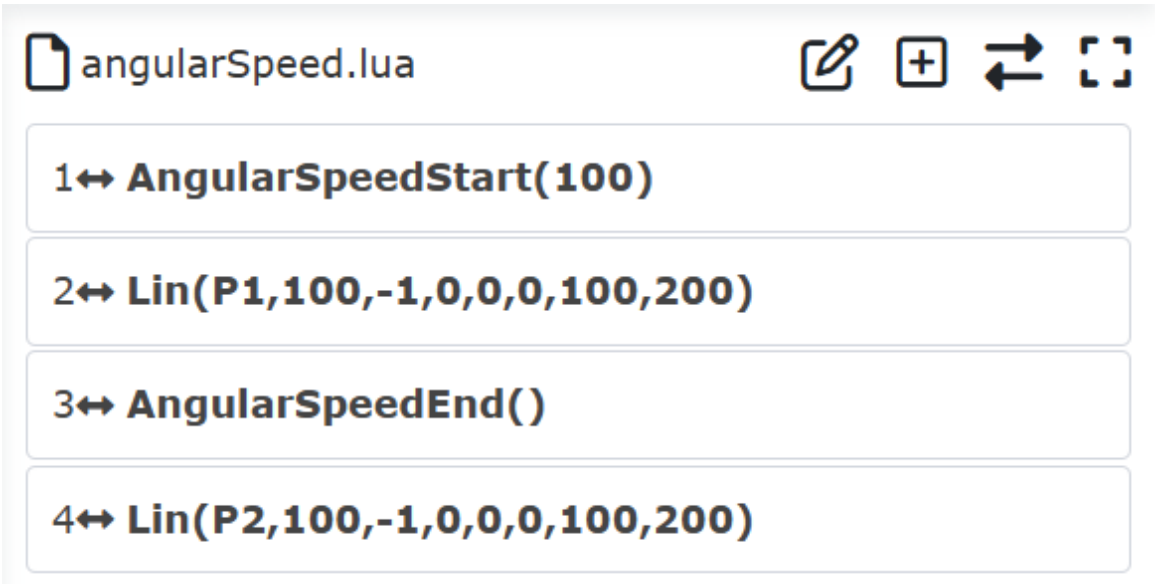


Figure 9.5-3-12 Generate a straight LUA instruction with transition points

A complete set of corner wrapping welds usually has more than one transition point, and in the case of corner wrapping shown in Figure 7, there are two attitude transition points with small position change and large attitude change during the welding process.

Point 1 is the starting point of the first section of welding, and point 2 is the end point of the first section of welding;

Point 3 is the starting point of the second section of welding, and point 4 is the end point of the second section of welding;

Point 5 is the starting point of the third section of welding, and point 6 is the end point of the third section of welding.

The attitude transition occurs from the end point of the previous section of welding to the starting point of the next section of welding, so it is necessary to add the attitude angular velocity adjustment instruction at the starting point of the next section of welding, so that the maximum linear velocity remains unchanged during the transition of the wrapping angle attitude, and the maximum angular velocity is increased, so that the process of wrapping angle welding process runs.

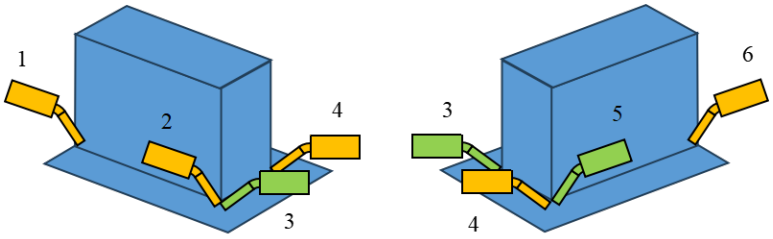


Figure 9.5-3-13 Example of a wrapping welding process

1.3.9.5.3 Arc command

Click the “Arc” icon to enter the Arc command editing interface.

The “Arc” command is an arc motion, which includes three points. The first point is the starting point of the arc, the second point is the middle transition point of the arc, and the third point is the end point.

Both the transition point and the end point can be set to offset, and you can choose to base coordinate system offset Shift and offset based on tool coordinates, and pop up x, y, z, rx, ry, rz offset settings, and the end point can set a smooth transition radius to achieve continuous motion effect.

Important: For circular motion, you need to add PTP or Lin command to move to the starting point first.

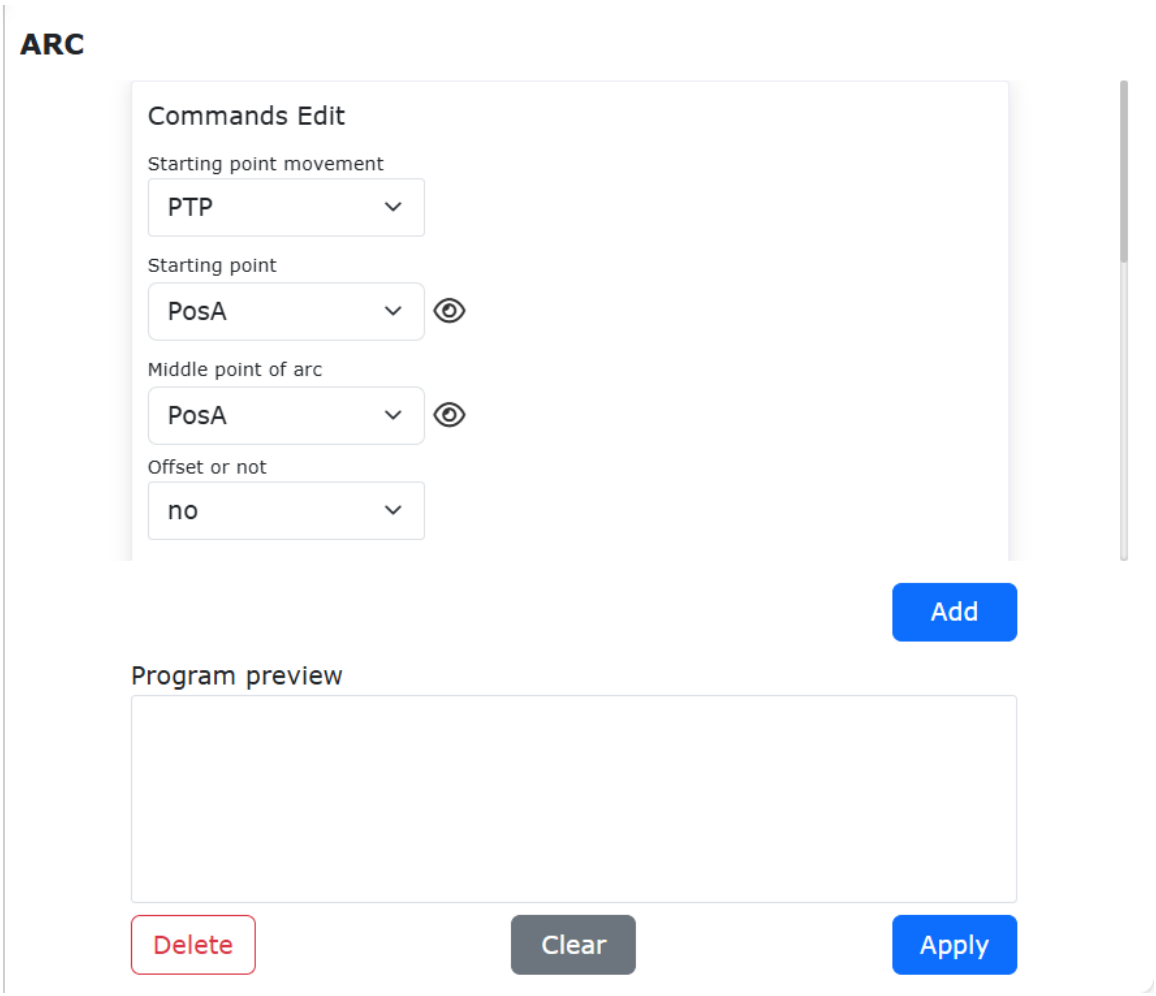


Figure 9.5-4 Arc command interface

The ARC command allows you to select the motion speed mode as either “Percentage” or “Physical Speed”:

- **Percentage:** Input a debugging speed percentage. The robot moves at a percentage of its maximum speed. The actual robot movement speed is calculated as: V = Robot Maximum Speed × Global Speed Percentage × Debugging Speed Percentage. Hover the mouse over the small eye icon to the right of the “Debugging Speed” input box to display the actual physical speed (in mm/s) of the robot in both manual and automatic modes under the current debugging speed settings.

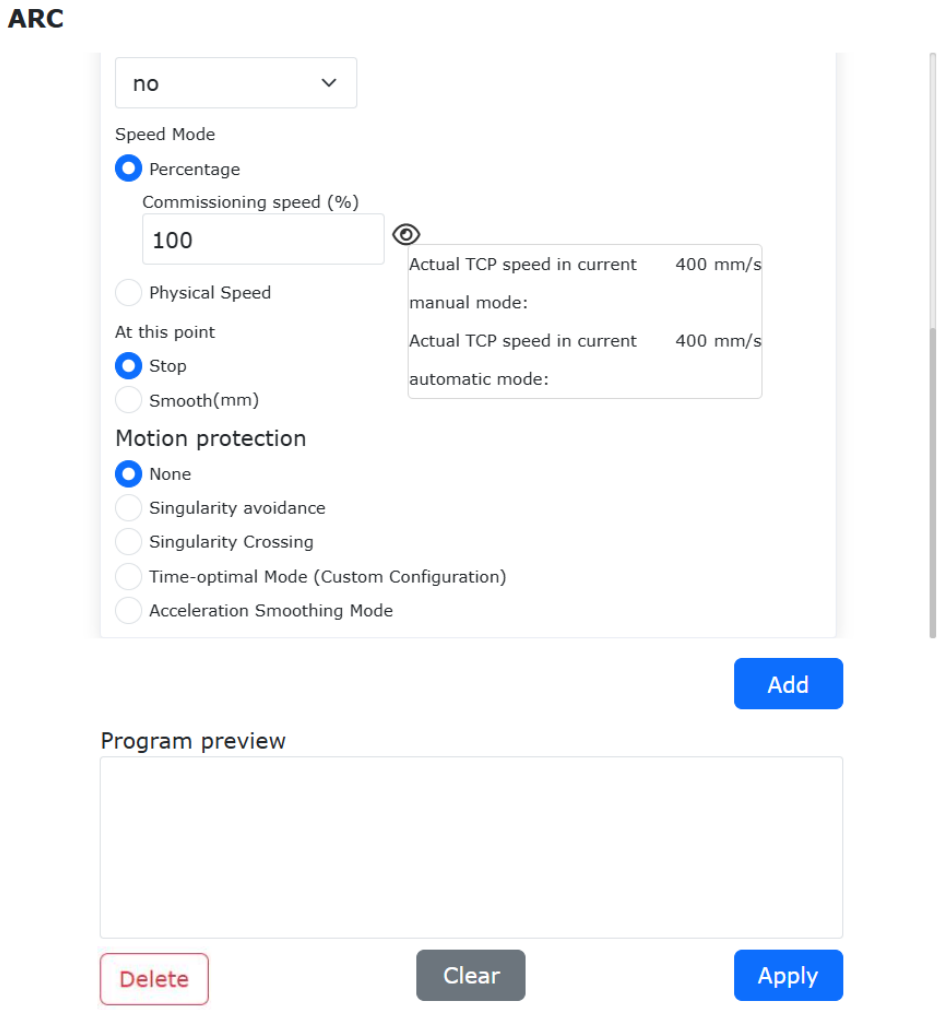


Figure 9.5-4-1 Display Actual Physical Speed Value When Inputting Percentage

- **Physical Speed:** The input speed is the actual operating speed of the robot, in mm/s. The input acceleration is typically set to twice the speed. (The maximum physical speed of the LIN command is limited by the global speed percentage. If the robot's maximum operating speed is 1000 mm/s and the global speed is 50%, then the maximum physical speed for the LIN command is 1000 × 50% = 500 mm/s).

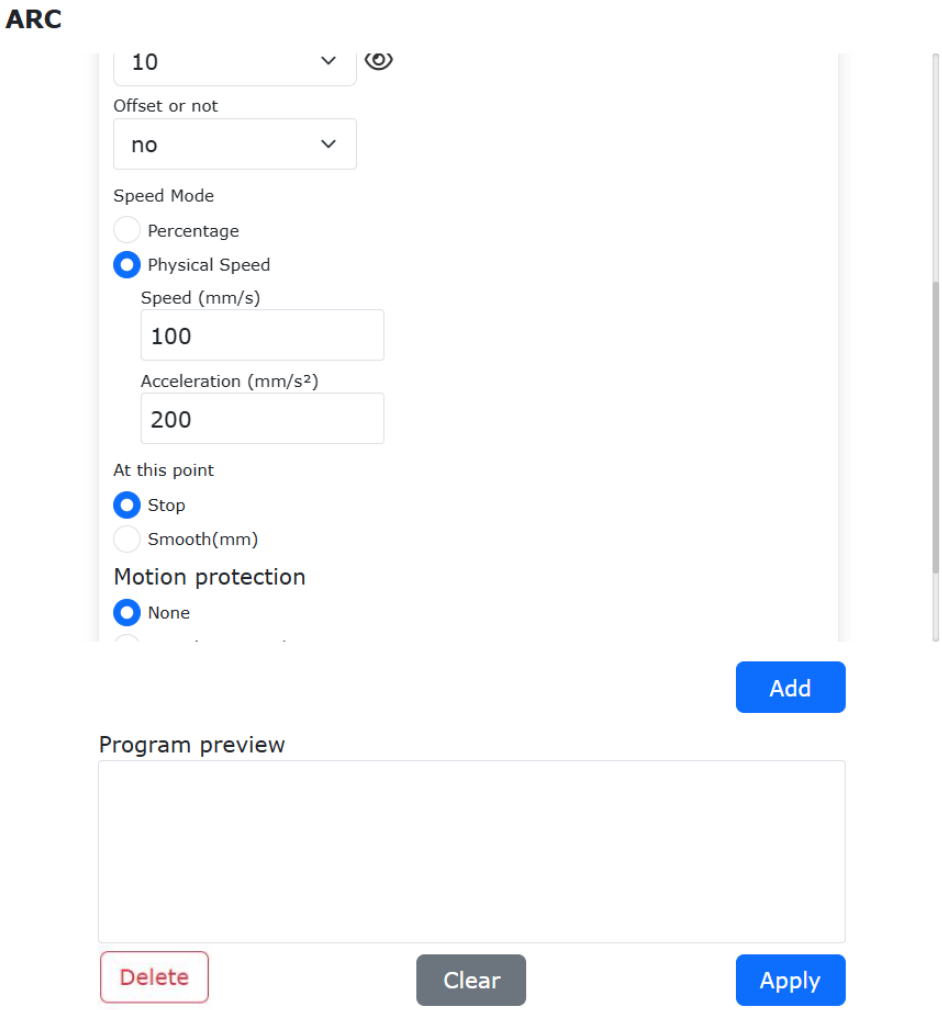


Figure 9.5-4-2 Input Actual Physical Speed

1.3.9.5.4 Circle command

Click the “Circle” icon to enter the Circle command editing interface.

The cooperative robot can carry out the circle trajectory movement by adding the circle command. Before adding the circle command, it is necessary to teach three path points on the circle trajectory. Suppose that the three path points on the circle trajectory are “P1”, “P2” and “P3” respectively, where “P1” is the starting point of the circle trajectory, “P2” and “P3” are the middle point 1 and 2 of the circle trajectory.Move the robot to the above three points and add the names of the teaching points as “P1”, “P2” and “P3” respectively.

Important: For full circle trajectory motion, you need to add PTP or Lin command to move to the starting point first.

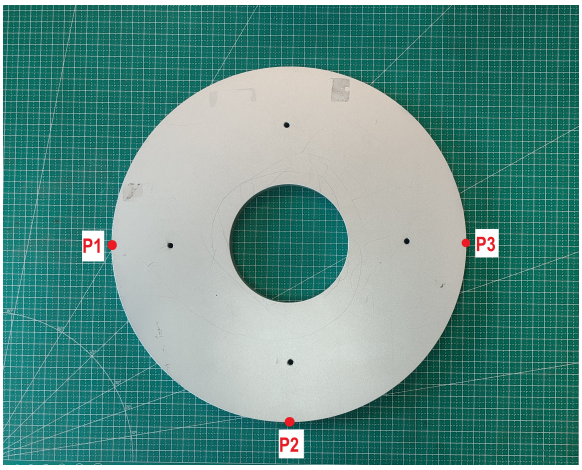
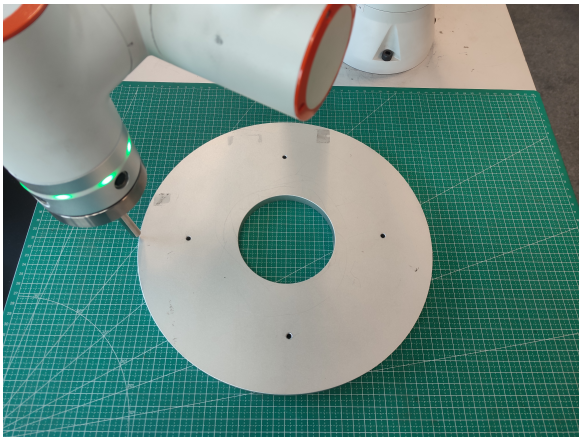
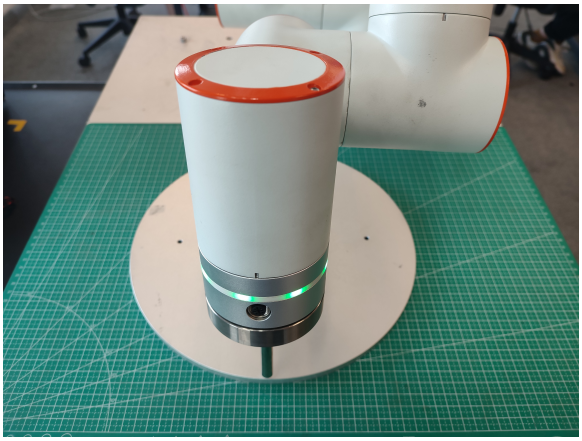


Figure 9.5-5 Circle trajectory





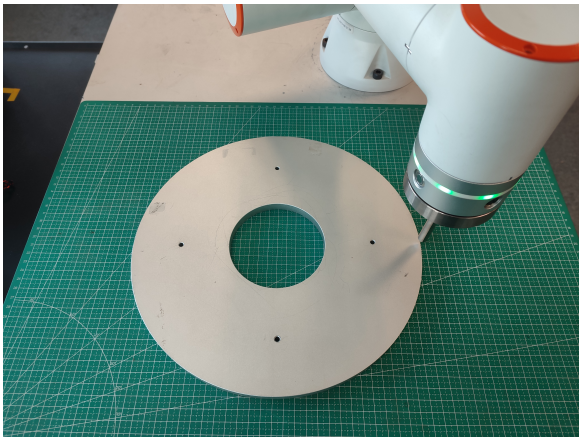


Figure 9.5-6 Teaching “P1”, “P2” and “P3”

1.3.9.5.4.1 Circle command addition

Step1: Create a new user program “testCircle.lua” and click the “Circle” button to open the page of adding circle commands.

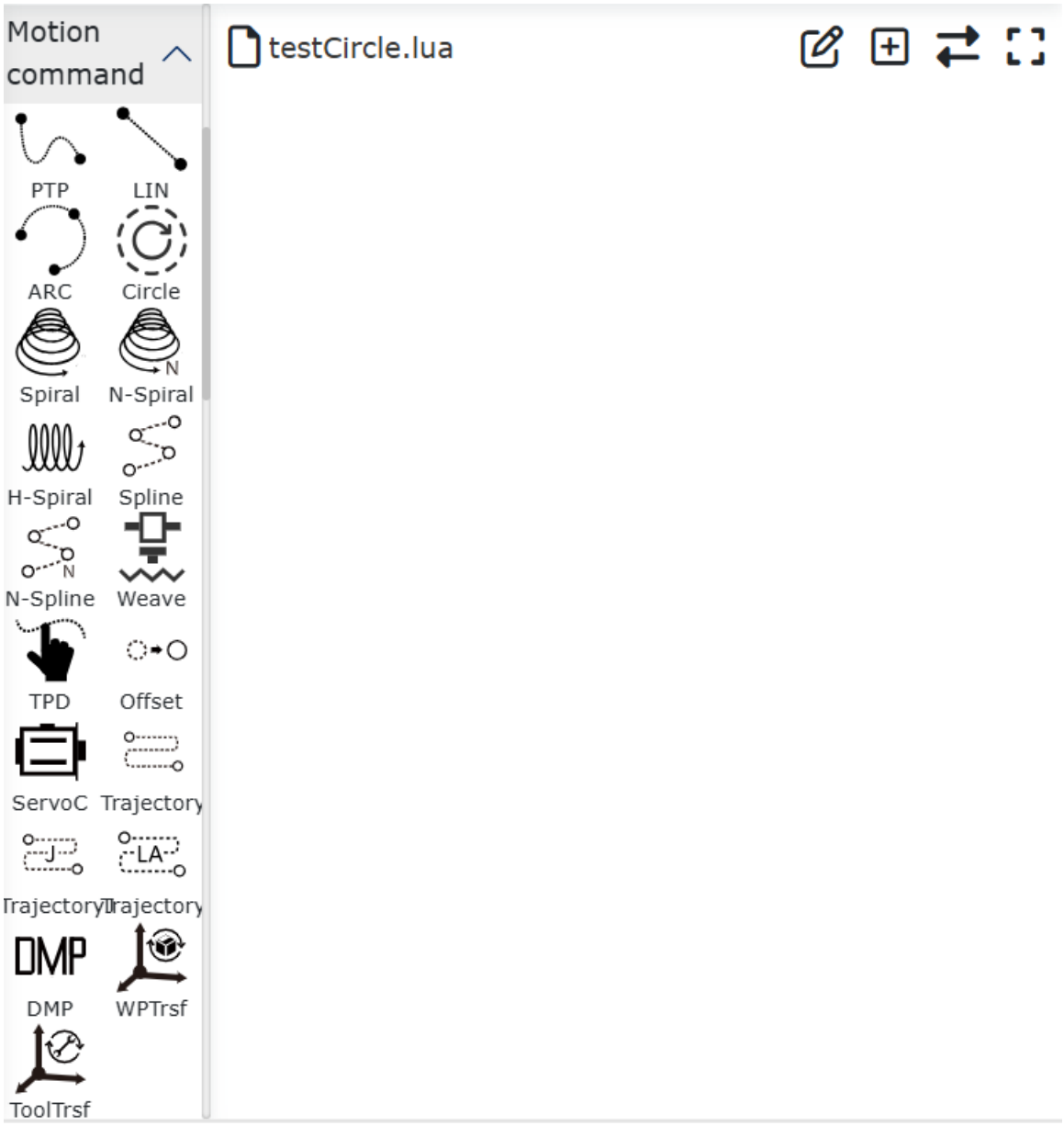


Figure 9.5-7 Add the circle command button

Step2: In the full circle instruction adding page, select the starting point motion mode and the starting point as “P1”.

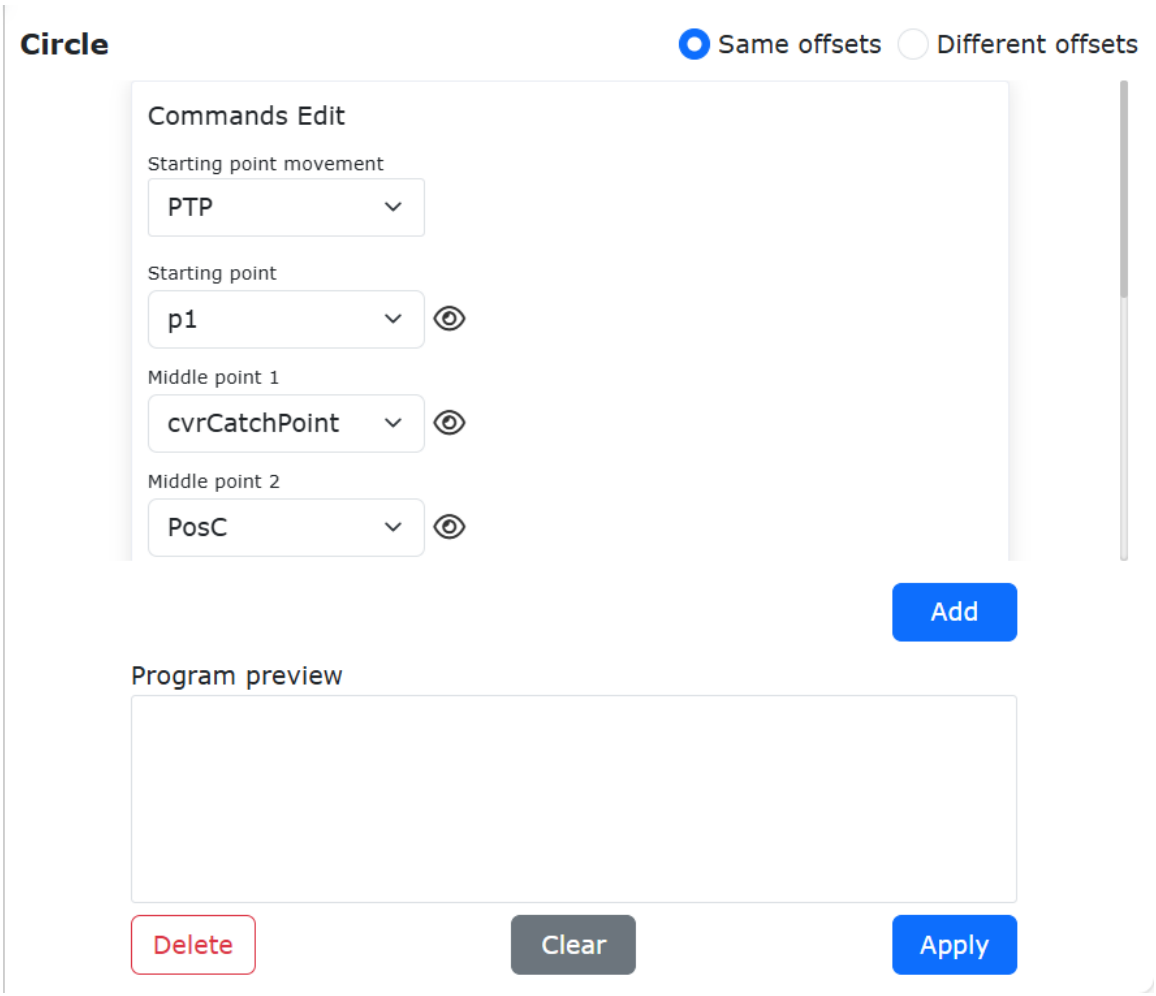


Figure 9.5-8 Starting point motion mode and starting point “P1”

Step3: On the Add Circle Command page, select “Circle Intermediate Point 1” as point “P2”, and “Circle Intermediate Point 2” as point “P3”.

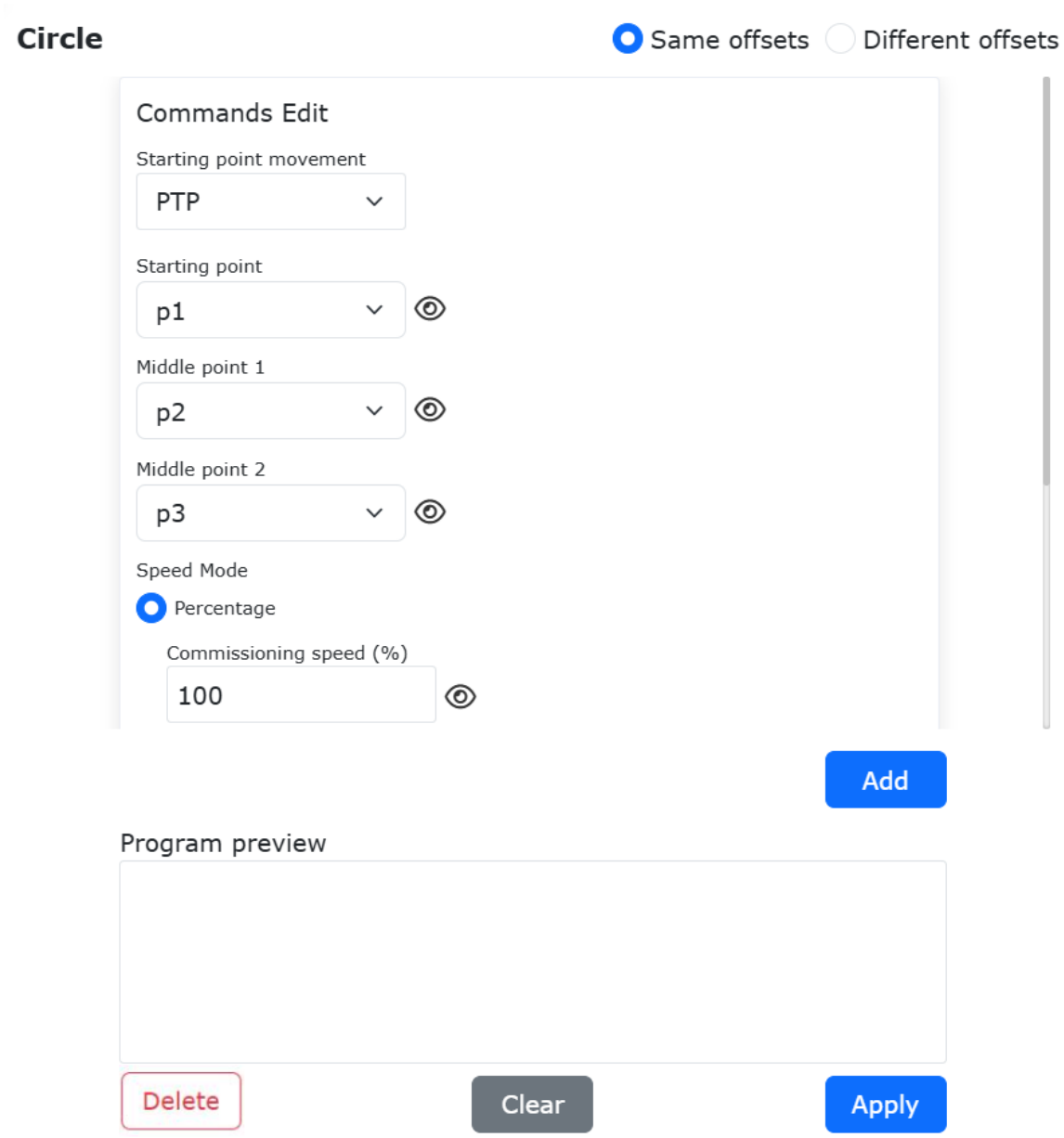


Figure 9.33-9 Selecting Arc Intermediate Points and Endpoint

Step4: Select the speed mode and input the speed value.

The Circle command allows you to select the motion speed mode as either “Percentage” or “Physical Speed”:

- **Percentage:** Input a debugging speed percentage. The robot moves at a percentage of its maximum speed. The actual robot movement speed is calculated as: V = Robot Maximum Speed × Global Speed Percentage × Debugging Speed Percentage. Hover the mouse over the small eye icon to the right of the “Debugging Speed” input box to display the actual physical speed (in mm/s) of the robot in both manual and automatic modes under the current debugging speed settings.

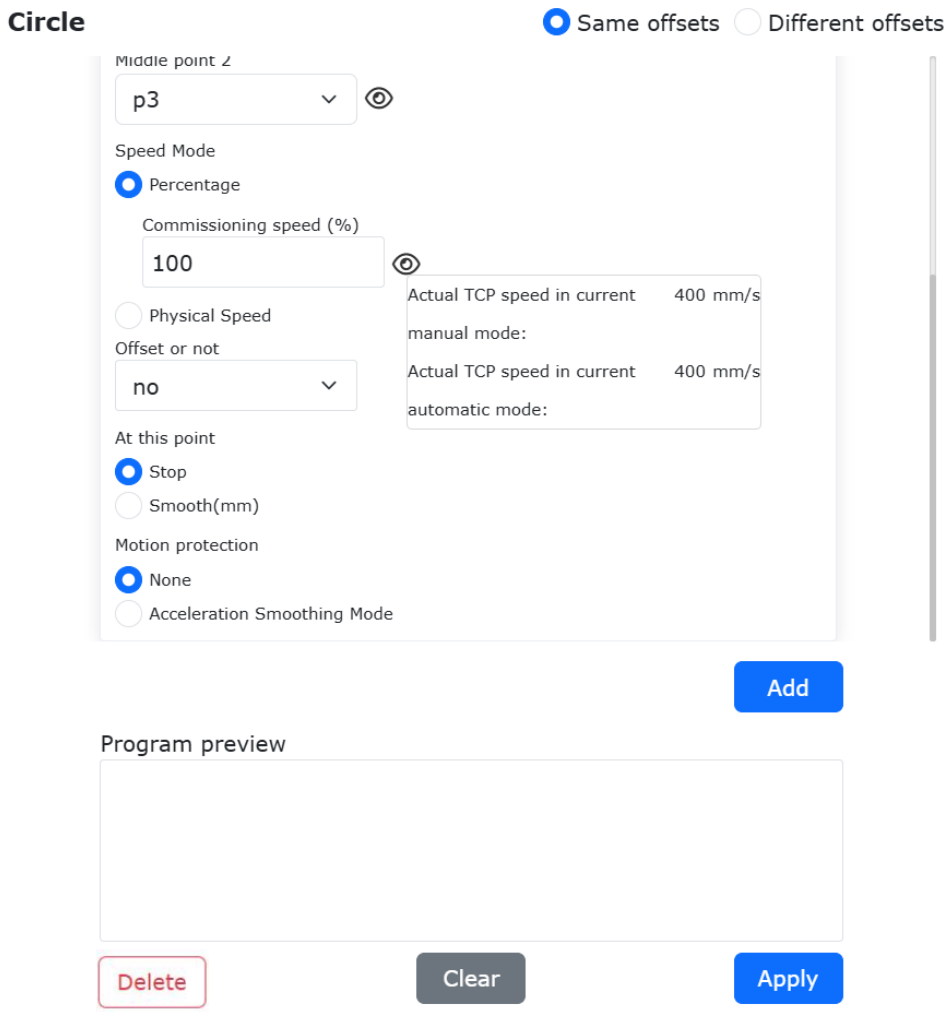


Figure 9.33-10 Display Actual Physical Speed Value When Inputting Percentage

- **Physical Speed:** The input speed is the actual operating speed of the robot, in mm/s. The input acceleration is typically set to twice the speed. (The maximum physical speed of the LIN command is limited by the global speed percentage. If the robot's maximum operating speed is 1000 mm/s and the global speed is 50%, then the maximum physical speed for the LIN command is 1000 × 50% = 500 mm/s).

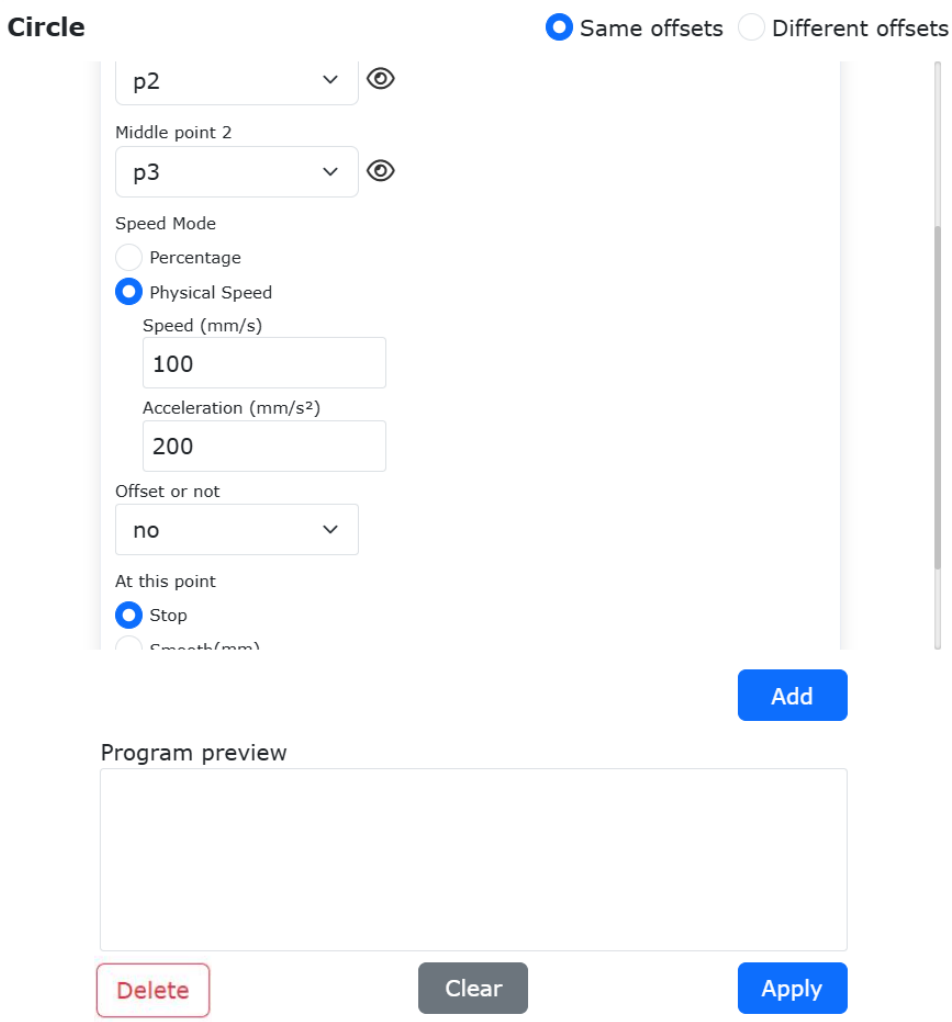


Figure 9.33-11 Input Actual Physical Speed

Step5: Click the "Add" button and then the "Apply" button in sequence. At this point, the circle motion command has been added to "testCircle.lua".

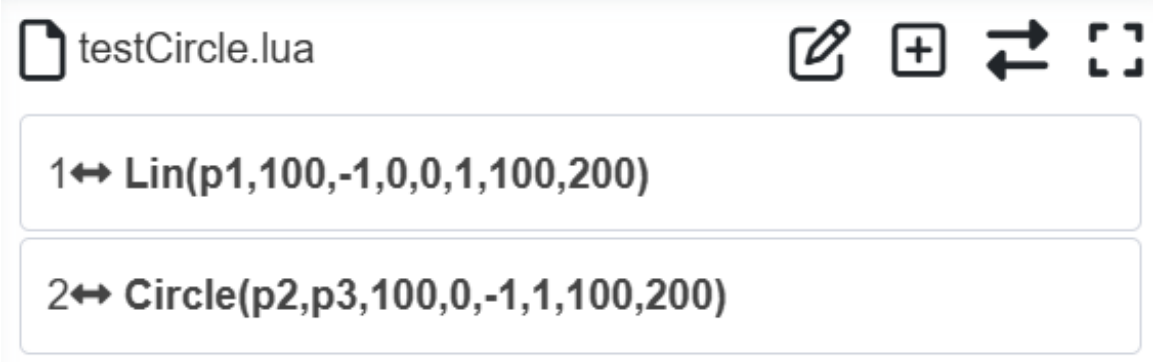


Figure 9.33-12 Addition of Circle Motion Command

Step5: At this point, the circle motion command has been added to "testCircle.lua".

Switch the robot to automatic mode, start the program on the premise of ensuring safety, and the robot will move according to the circle trajectory shown in Figure 1.

1.3.9.5.4.2 Circle trajectory offset

The circular motion of the cooperative robot supports the offset of the positions of the middle point 1 and the middle point 2 of the circular trajectory, and the offset types include the following two types:

The middle points of the two trajectories of the circle have the same offset: the middle point 1 (P2) and the middle point 2 (P3) of the circle are offset by the same offset (dx, dy, dz, drx, dry, drz).

The middle points of the two trajectories of the circle have different offsets: the middle point 1 (P2) and the middle point 2 (P3) of the circle adopt two different offsets 1 (dx1, dy1, dz1, drx1, dry1, drz1) and 2 (dx2, dy2, dz2, drx2, dry2, drz2) respectively.

The following demonstrates the usage of "same offset" and "different offset" respectively.

(1) Same offset

Open the circle command addition page, select "Same Offset" for Offset Type, and also select the starting point motion mode and starting point as "P1", and the middle point 1 of the full circle as "P2".

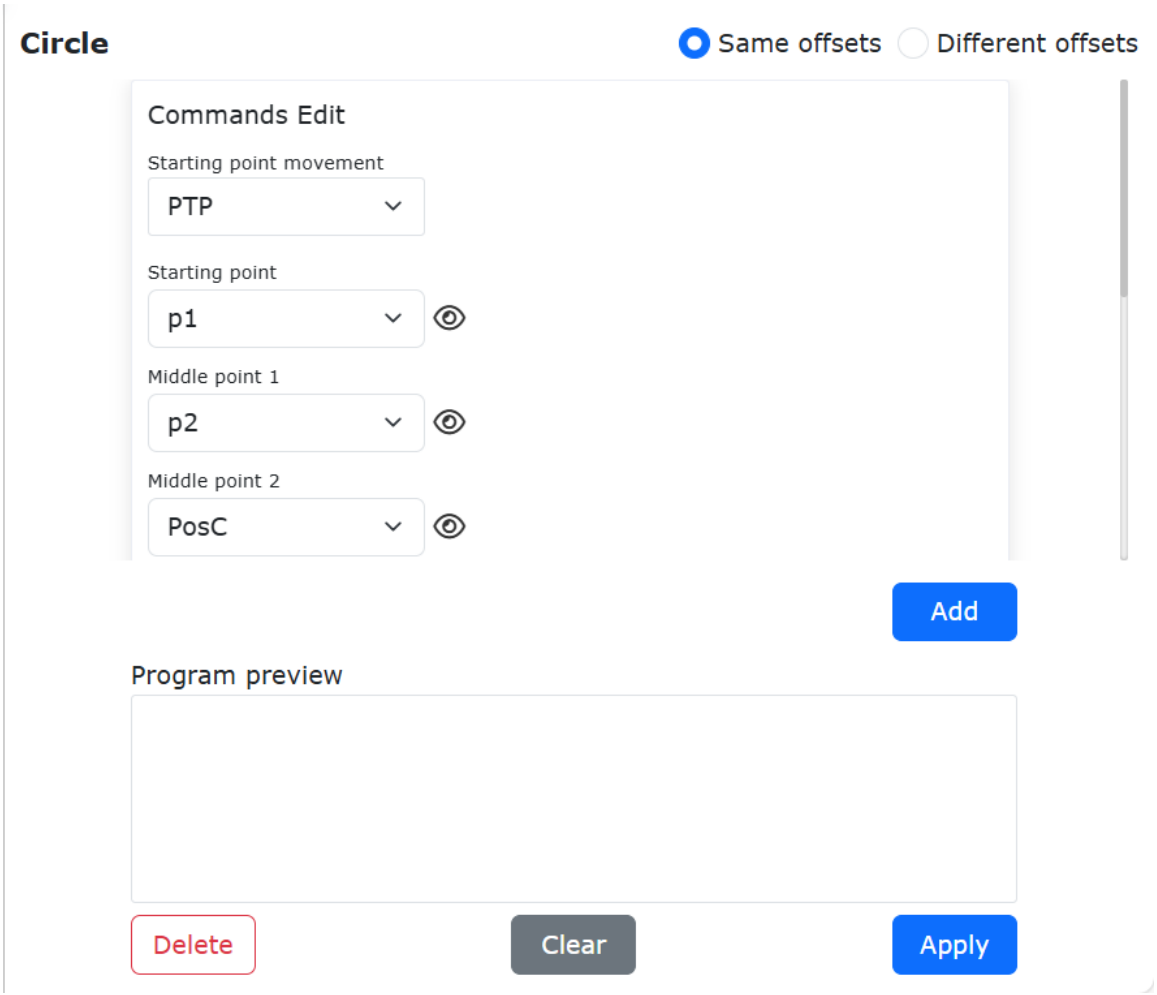


Figure 9.5-12 Same offset of the circle

Select P3 for the middle point 2 of the circle, and select Base Coordinate Offset for Offset.

Note: Note: you can select Tool Coordinate Offset according to the actual work situation.

Enter the offset dx as 10mm, and then click the Add button and the Apply button at the bottom of the page.

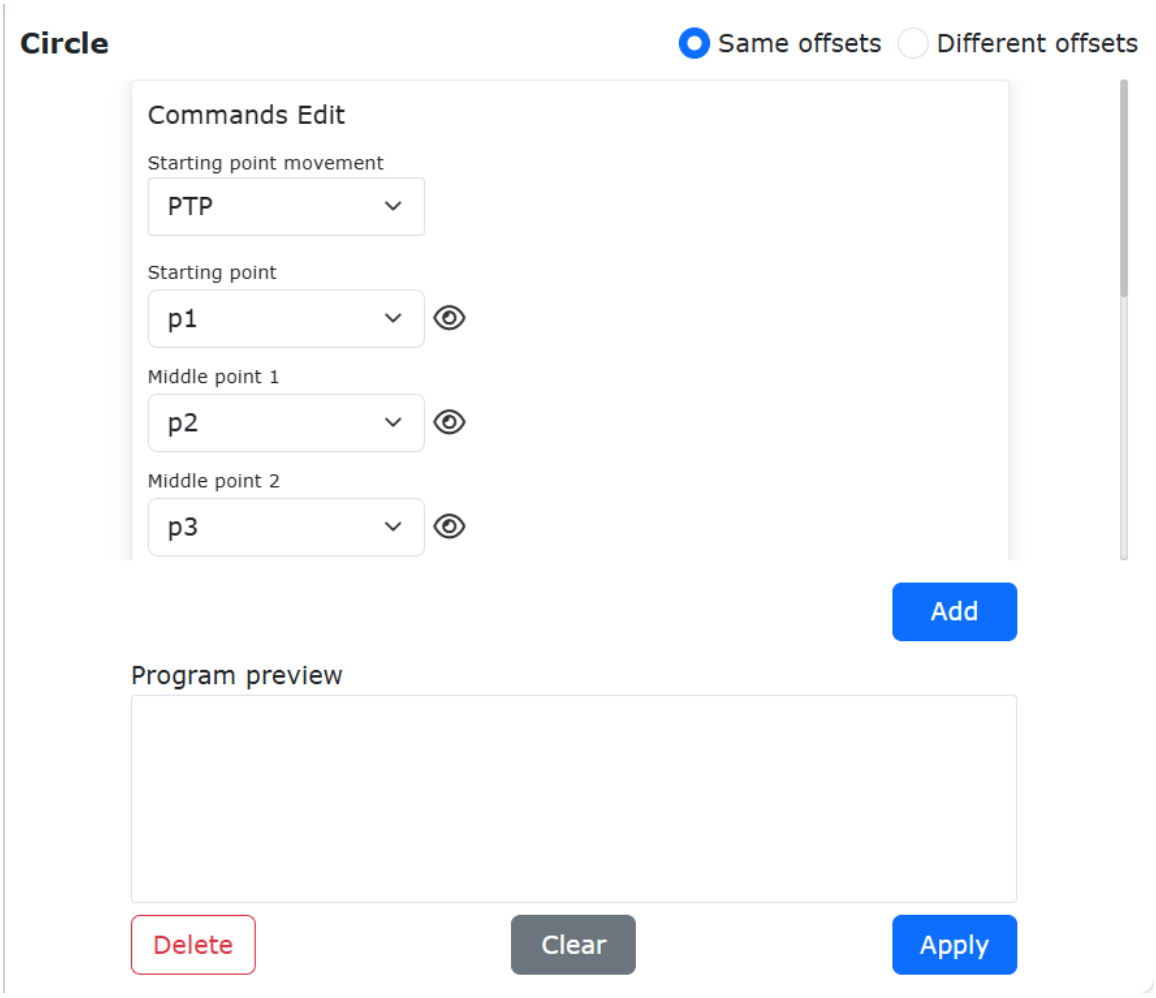


Figure 9.5-13 Setting Offset

At this time, a full circle starting point is “P1”, and the two intermediate points “P2” and “P3” are offset by 10mm along the X-axis direction of the base coordinate system. The full circle instruction has been added to the “testCircle.lua” program.

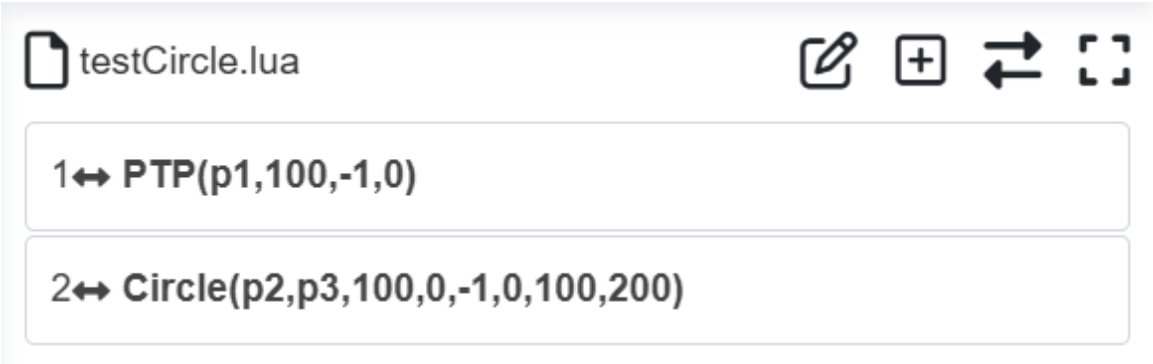


Figure 9.5-14 Same Offset Program for Circle

Switch the robot to the automatic mode, and start the program under the condition of ensuring safety. The actual motion trajectory of the robot passes through the circles of “P1”, “P2” and “P3”, where “P2” is the point where the original

"P2" is offset by 10mm in the X direction, and "P3" is the point where the original "P3" is offset by 10mm in the X direction.

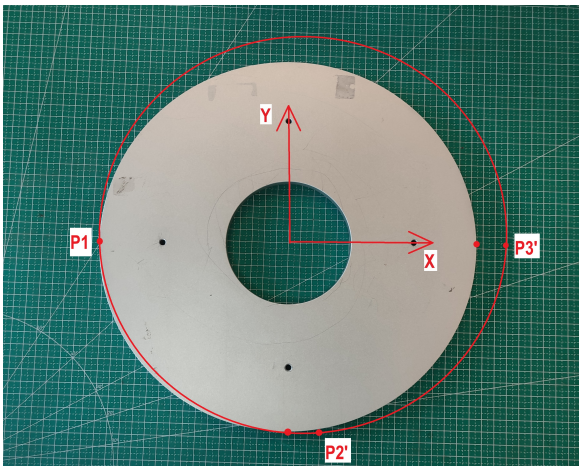


Figure 9.5-15 Track with the same offset X10mm

(2) Different offset

Open the circle command adding page, select "Different Offsets" for Offset Type, and also select the starting point motion mode and starting point as "P1", and the middle point 1 of the full circle as "P2", and select "Base coordinate offset" for "Offset".

Note: Note: you can select "Tool coordinate offset" according to the actual work situation.

Enter the offset dy as 10mm.

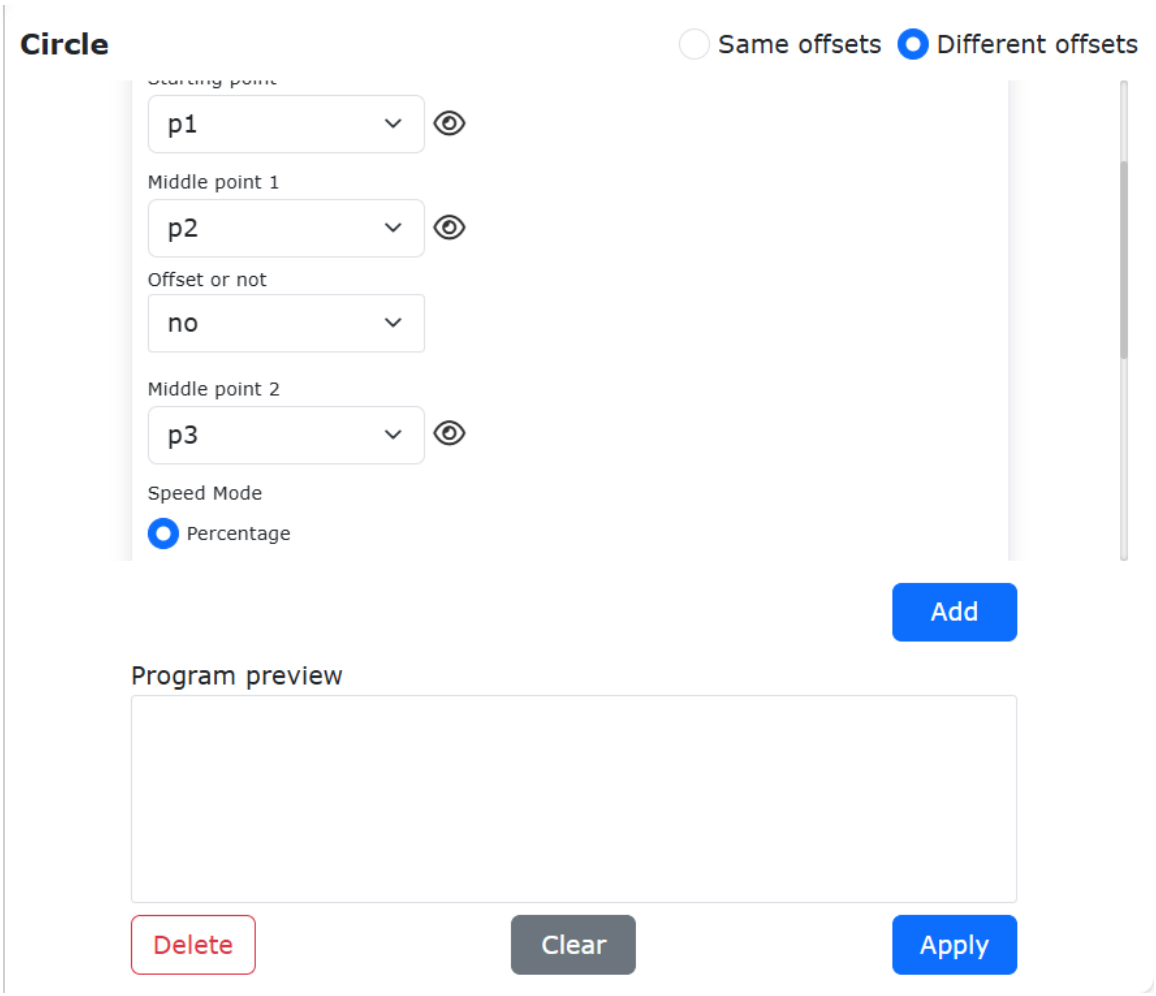


Figure 9.5-16 Different offsets

Select P3 for the middle point 2 of the circle, and select Base Coordinate Offset for Offset

Note: Note: you can select Tool Coordinate Offset according to the actual work situation.

Enter the offset dx as 10mm, and then click the Add button and the Apply button at the bottom of the page.

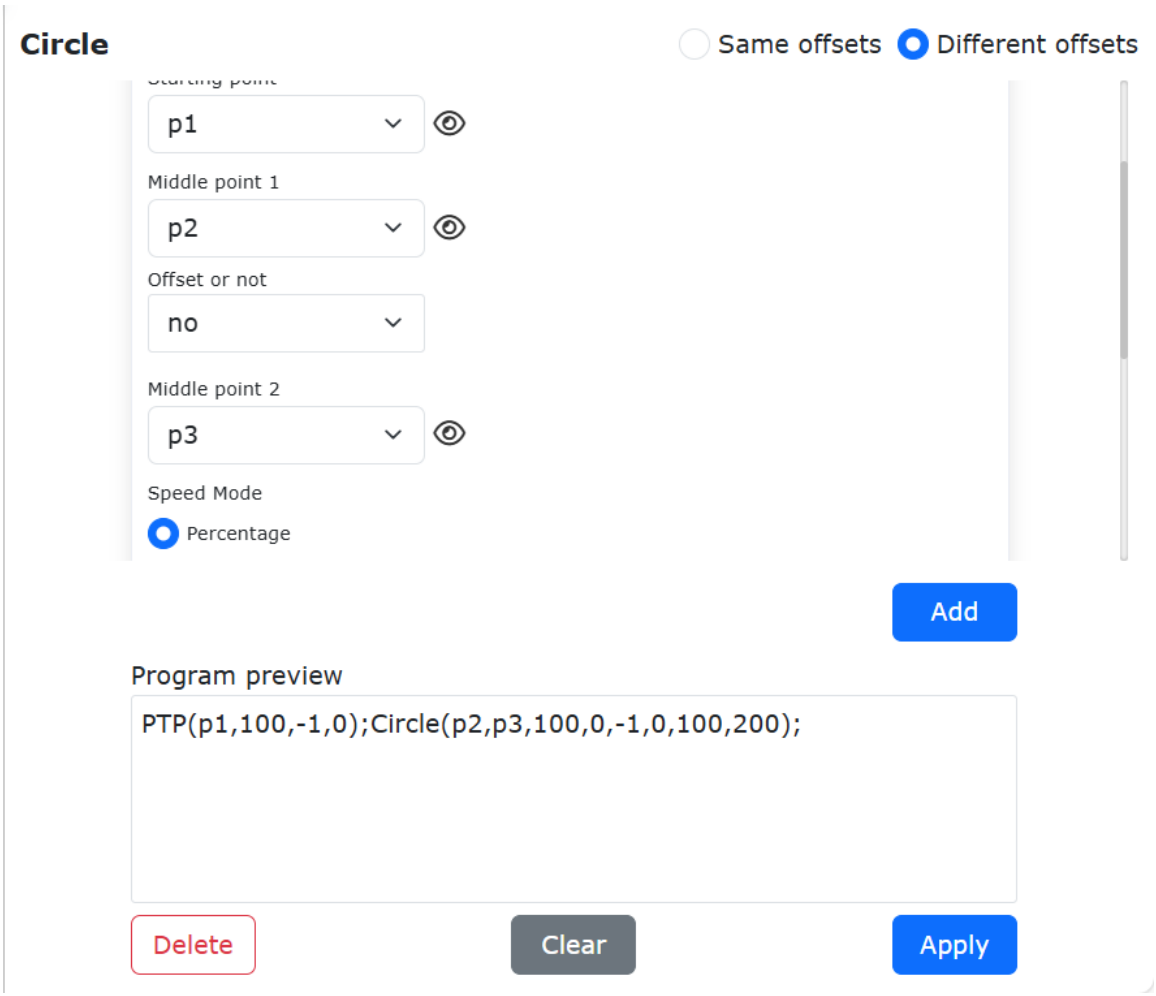


Figure 9.5-17 Offset of middle point 2 is set with different offsets

At this time, an command that the middle point "P2" of the circle is shifted by 10mm along the Y direction of the base coordinate system and "P3" is shifted by 10mm along the X axis direction of the base coordinate system has been added to the "testCircle.lua" program; Of course, a linear motion command needs to be added before the circle motion command to make the robot move to the starting point of the circle.

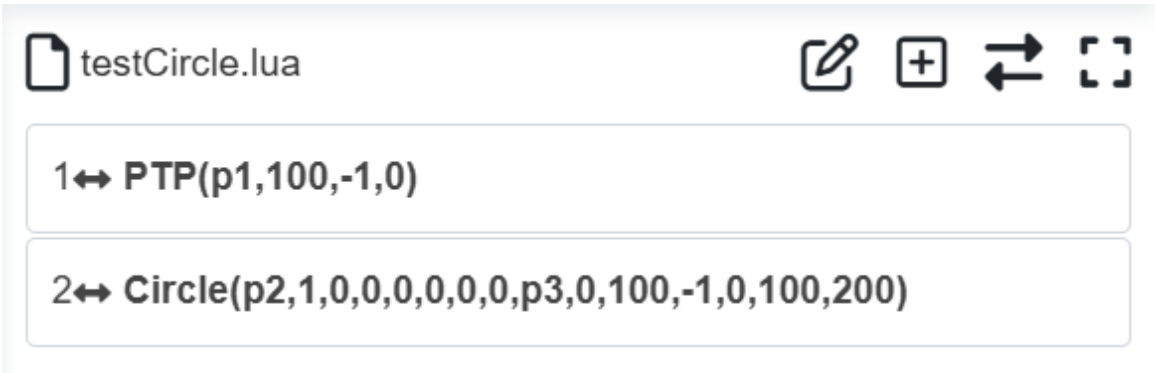


Figure 9.5-18 Program for Two Different Offset Points of Circle

Switch the robot to the automatic mode, and start the program under the condition of ensuring safety. The actual motion trajectory of the robot passes through the circles of "P1", "P2" and "P3", where "P2" is the point where the original

"P2" is offset by 10mm in the Y direction, and "P3" is the point where the original P3 is offset by 10mm in the X direction.

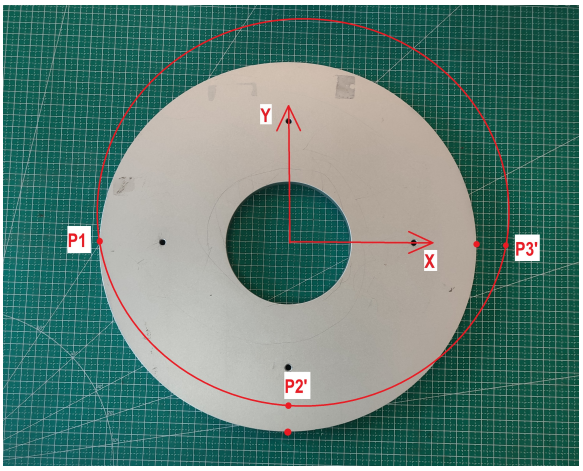


Figure 9.5-19 Two trajectory points of the circle are offset from the trajectory respectively

1.3.9.5.5 Spiral command

Click the "Spiral" icon to enter the Spiral command editing interface.

The "Spiral" command is a spiral motion, which includes three points, which form a circle. On the third point setting page, there are several parameters including the number of spiral turns, attitude correction angle, radius increment and rotation axis direction increment. Setting, the number of helix circles is the number of motion circles of the helix, the attitude correction angle is the attitude at the end of the helix and the attitude of the first point of the helix, the radius increment is the increment of the radius of each circle, and the direction of the rotation axis is increased. The amount is the increment in the direction of the screw axis. set up Whether to offset, the offset takes effect on the trajectory of the entire helix.

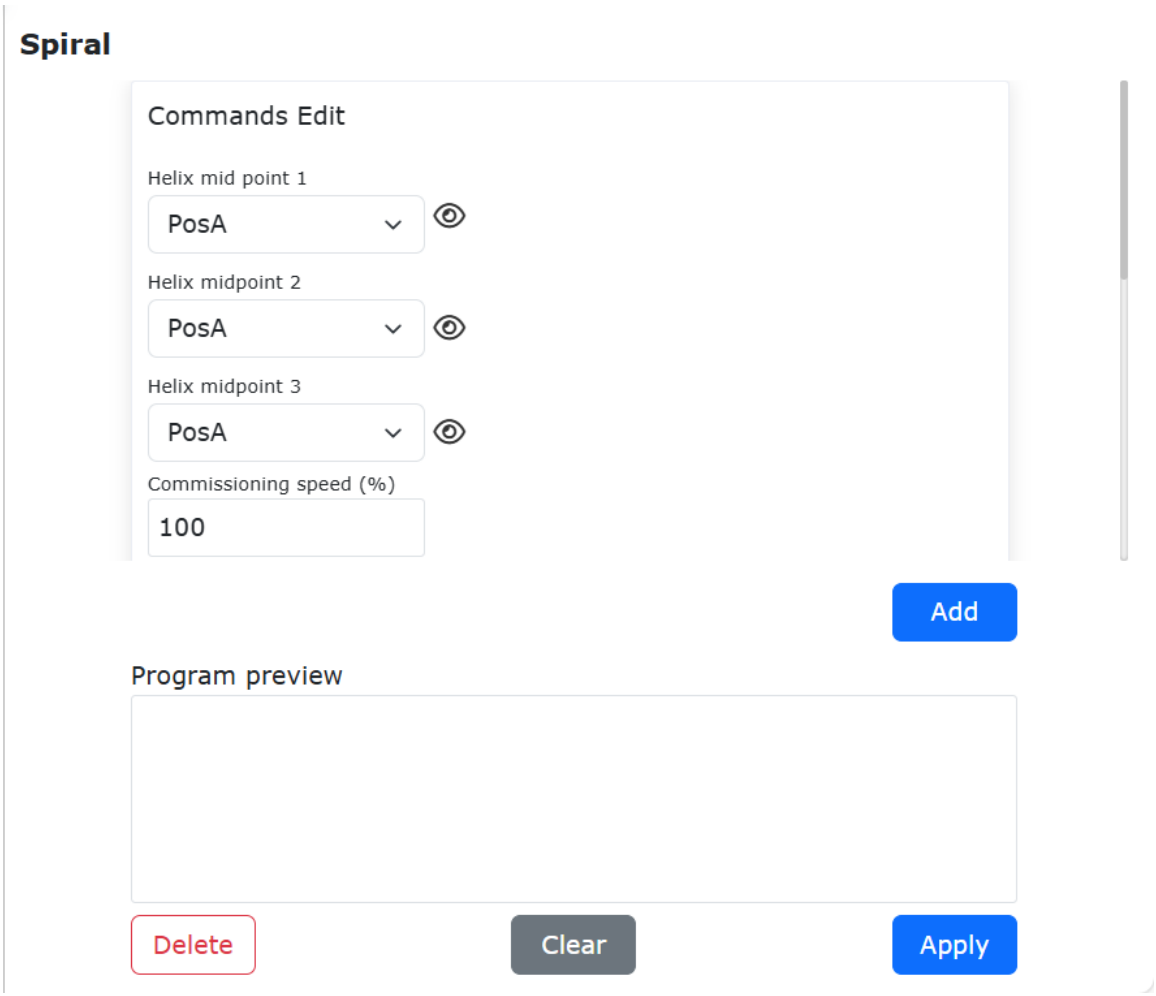


Figure 9.5-20 Spiral command interface

1.3.9.5.6 N-Spiral command

Click the “N-Spiral” icon to enter the N-Spiral command editing interface.

The “N-Spiral” command is an optimized version of the spiral motion. This command only needs one point plus the configuration of various parameters to realize the spiral motion. The robot takes the current position as the starting point, and the user sets the debugging speed, whether to offset, the number of spiral turns, the spiral inclination, the initial radius, the radius increment, the rotation axis direction increment and the rotation direction. The number of spiral turns is the helix. The number of motion circles, the helix inclination is the angle between the Z axis of the tool and the horizontal direction, the attitude correction angle is the attitude at the end of the helix and the attitude of the first point of the helix, the initial radius is the radius of the first circle, and the radius increment That is, the increment of the radius of each circle, the increment in the direction of the rotation axis is the increment in the direction of the screw axis, and the direction of rotation is clockwise and counterclockwise.

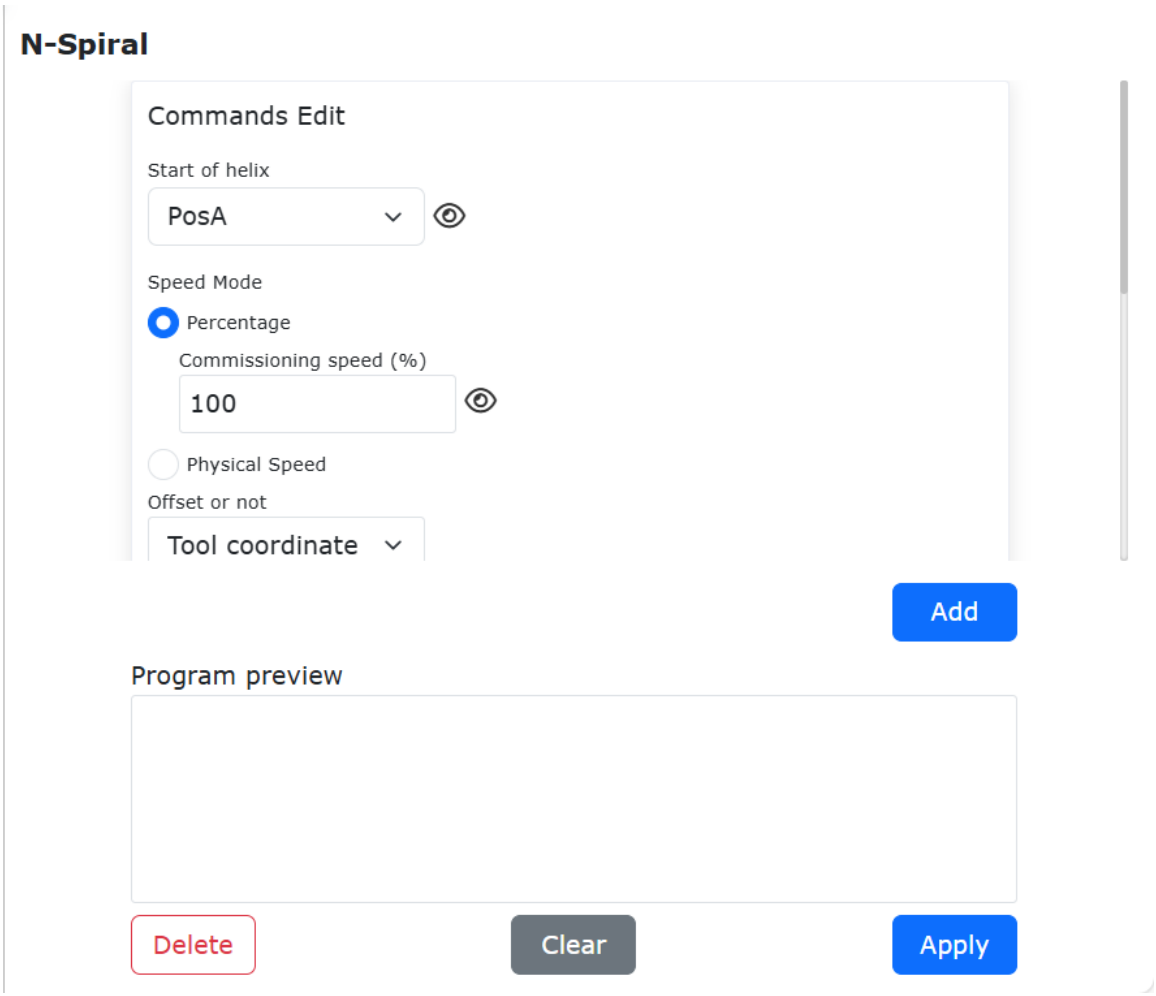


Figure 9.5-21 N-Spiral command interface

1.3.9.5.6.1 Uniform Speed Per Helix Circle Setting Function

1.3.9.5.6.2 Overview

When using the helical motion command, the operating speed of the helical motion can be set, ensuring the speed of each circle maintains the set operating speed.

1.3.9.5.6.3 Operation Process

Step1: Select the teaching point that executes the helical motion. This manual uses “P0” as the name of the teaching point.

Step2: Click the “Teach Program” -> “Program Programming” button, select the “New Helix” command. In the “Speed Mode”, select “Physical Speed”, and set the speed value and acceleration value. This speed value is the actual operating speed of the helix. Set parameters such as “Number of Helix Circles”, “Helix Inclination Angle”, “Initial Radius”, “Radius Increment”, “Rotation Axis Direction Increment”, and “Rotation Direction” according to actual needs, as shown in Figure 2-1.

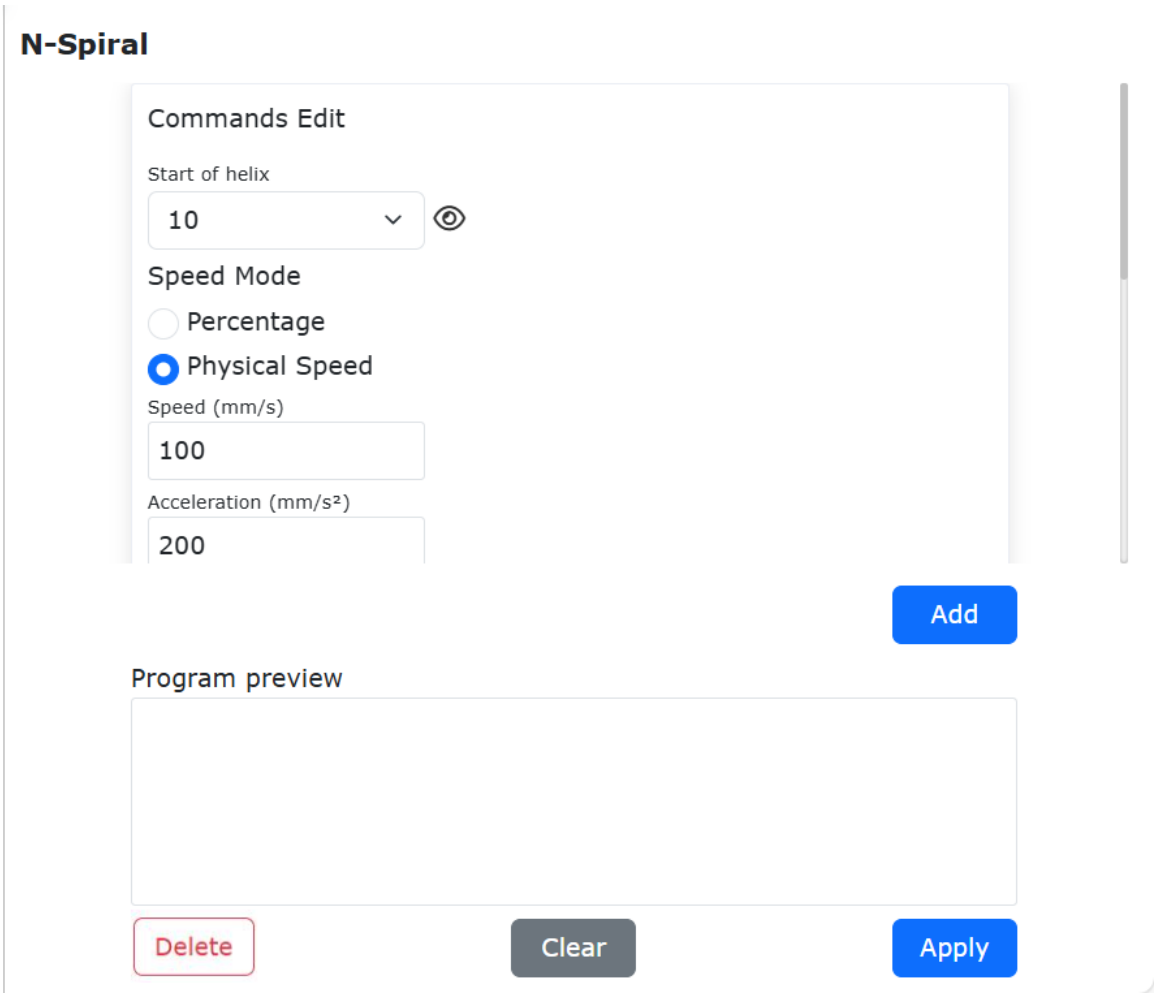


Figure 9.5-21-1 New Helix Parameter Settings

Step3: Add the motion command, generate the Lua program and run it to achieve the helical motion function at the set speed, as shown in Figure 2-2.

```
1↔ PTP(P0,100,0,2,50,0,0,- 30,0,0)
2↔ NewSpiral(P0,100,2,50,0,0,- 30,0,0,5,30,50,10,10,0,1,100,200)
```

Figure 9.5-21-2 Typical Program for Running Helix at Set Speed

1.3.9.5.7 H-Spiral command

Click the “H-Spiral” icon to enter the H-Spiral command editing interface.

The “H-Spiral” command is a horizontal space spiral motion. This command is set after the single-segment motion (straight line) command.

- Spiral radius: 0~100mm
- Helix angular speed: 0~2rev/s
- Direction of rotation: spiral clockwise/counterclockwise
- Helix inclination angle: 0~40°

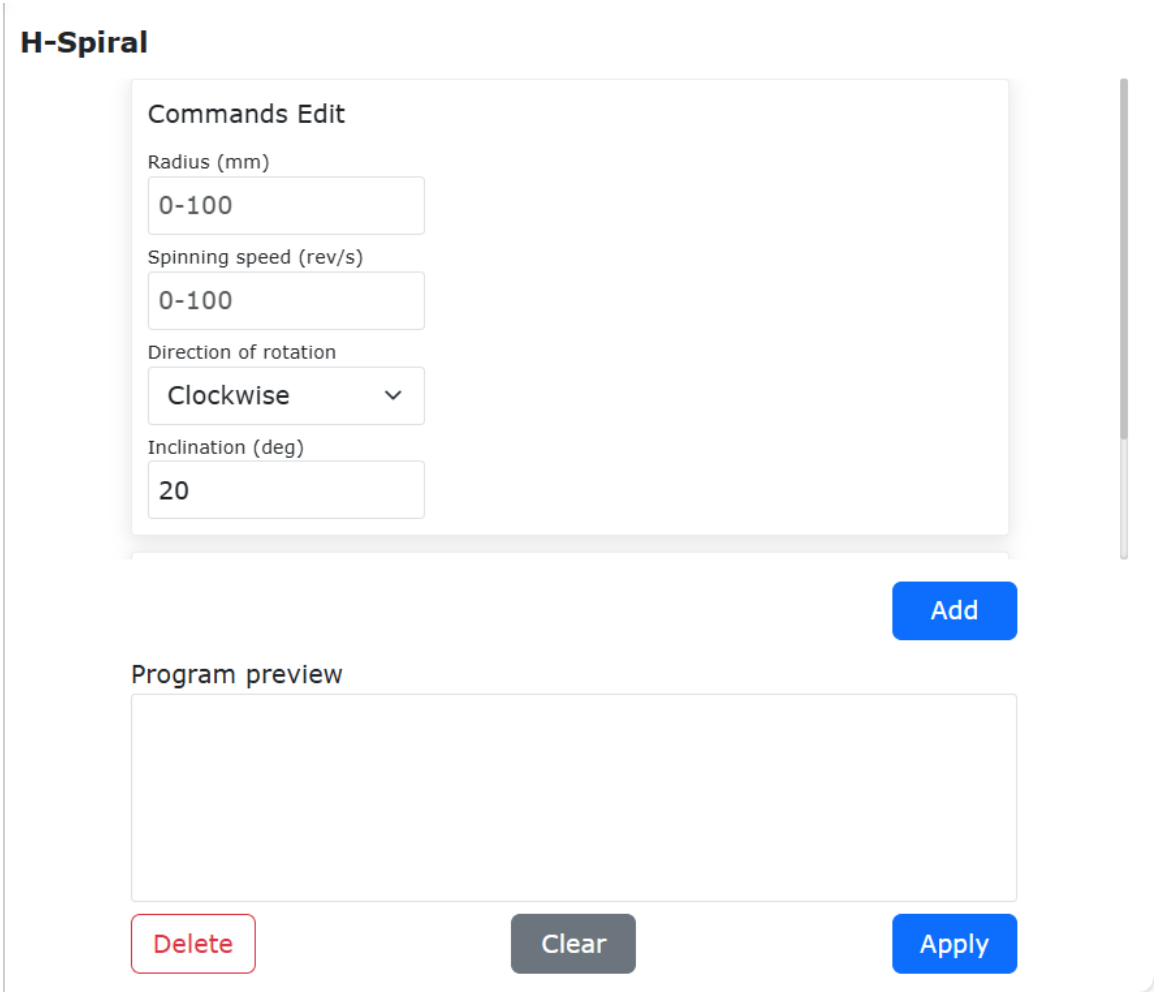


Figure 9.5-22 H-Spiral command interface

1.3.9.5.8 Spline command

Click the “Spline” icon to enter the Spline command editing interface.

The command is divided into three parts: the start of the spline group, the spline segment and the end of the spline group. The start of the spline group is the start mark of the spline movement. The spline segment includes SPL, SLIN and SCIRC segments. Click the corresponding icon to enter the command Add interface, the end of spline group is the end sign of spline movement.

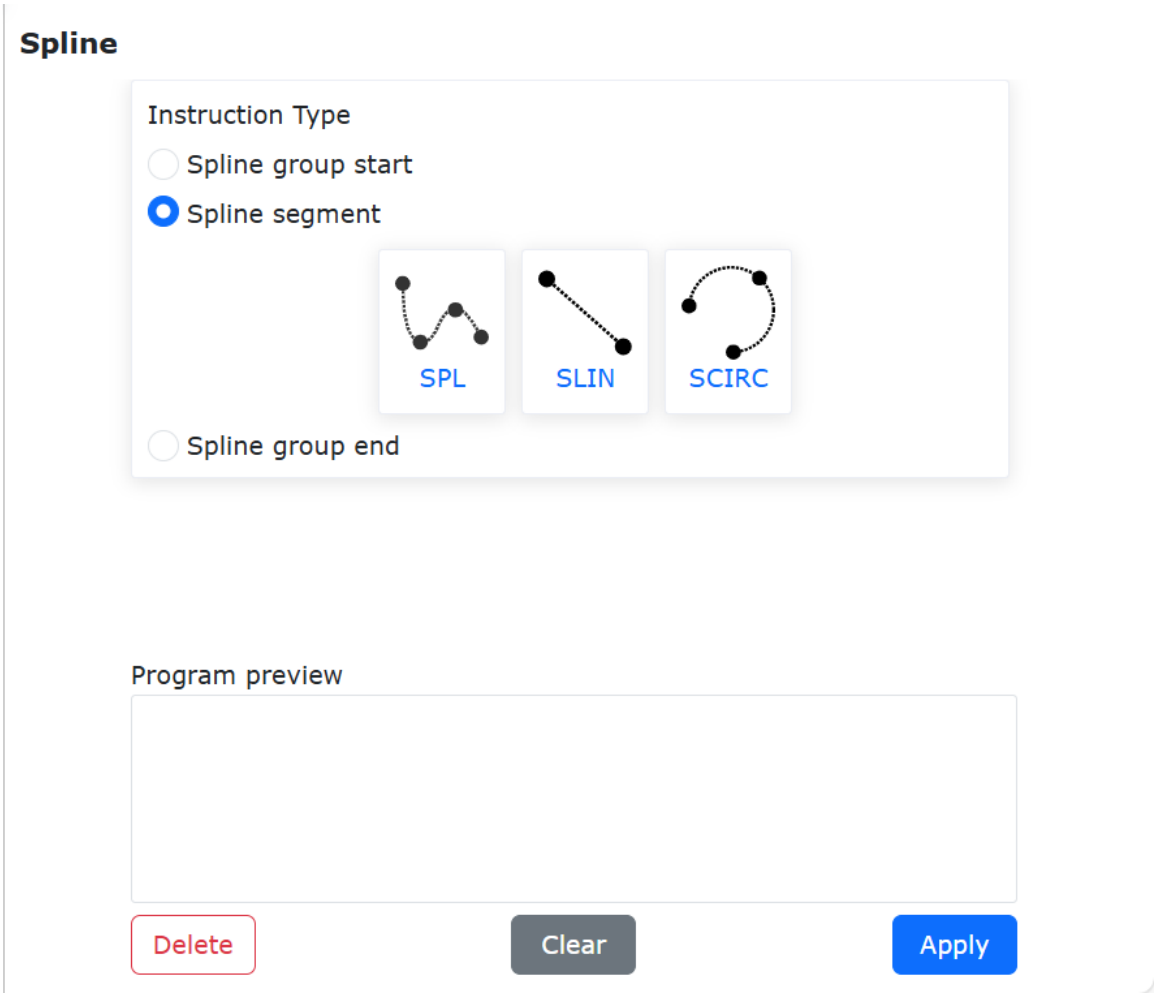


Figure 9.5-23 Spline command interface

1.3.9.5.9 N-Spline command

Click the “N-Spline” icon to enter the N-Spline command editing interface.

This instruction is an optimization instruction for the Spline instruction algorithm, and will replace the existing Spline instruction in the future.

This instruction is divided into three parts: the start of the multi-point trajectory, the segment of the multi-point trajectory and the end of the multi-point trajectory. The start mark, the multi-point track segment is to set each track point.

Click the icon to enter the point adding interface, the end of the multi-point track is the end mark of the multi-point track movement, here you can set the control mode and debugging speed.

- Control mode: arc transition point/given path point
- Global average connection time: integer, greater than 10, default value is 2000

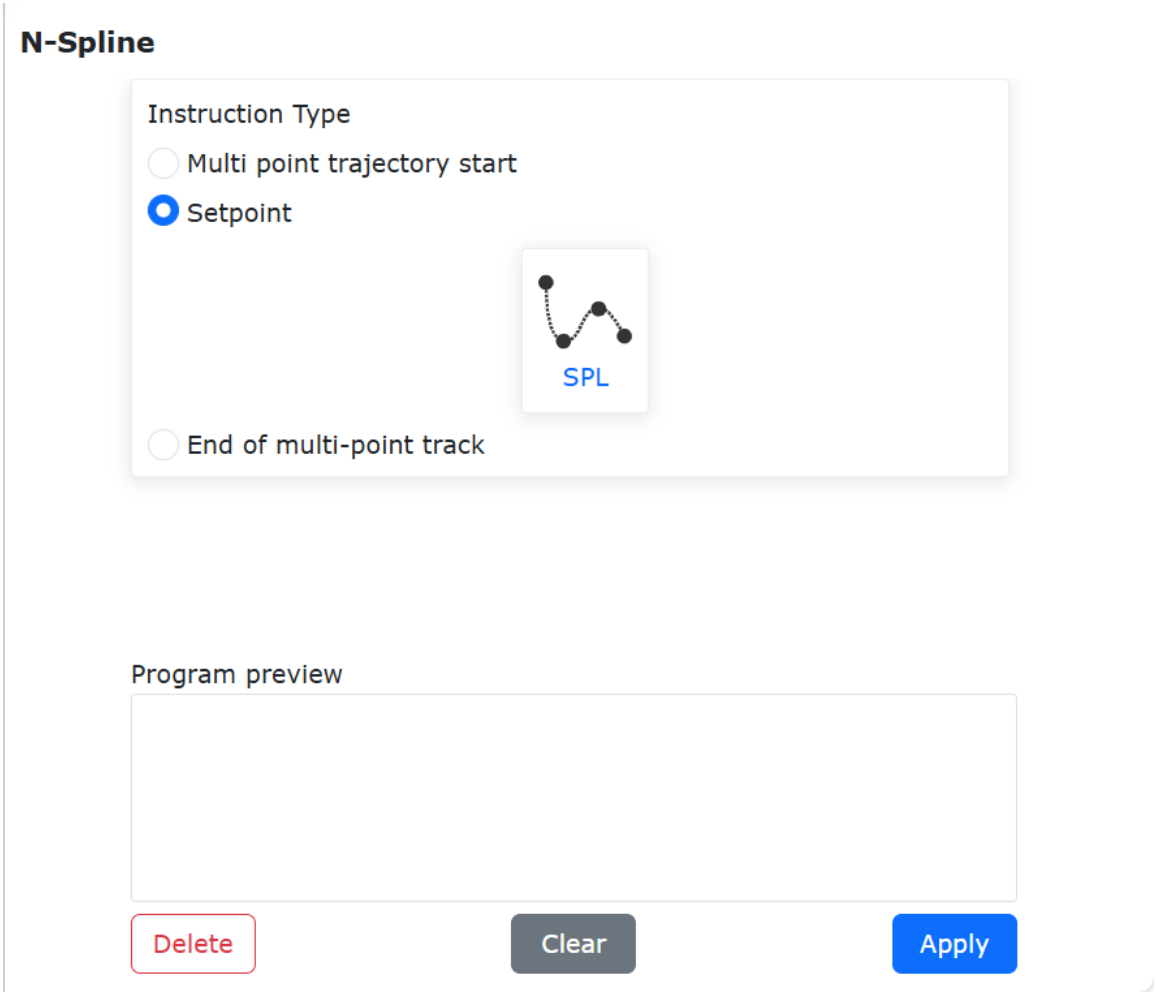


Figure 9.5-24 N-Spline command interface

1.3.9.5.10 Weave command

Click the "Weave" icon to enter the Weave command editing interface. The "Weave" command consists of two parts.

- Select the weaving number with configured parameters, click "Start Weaving" and "Stop Weaving" and apply to add related commands to the program.

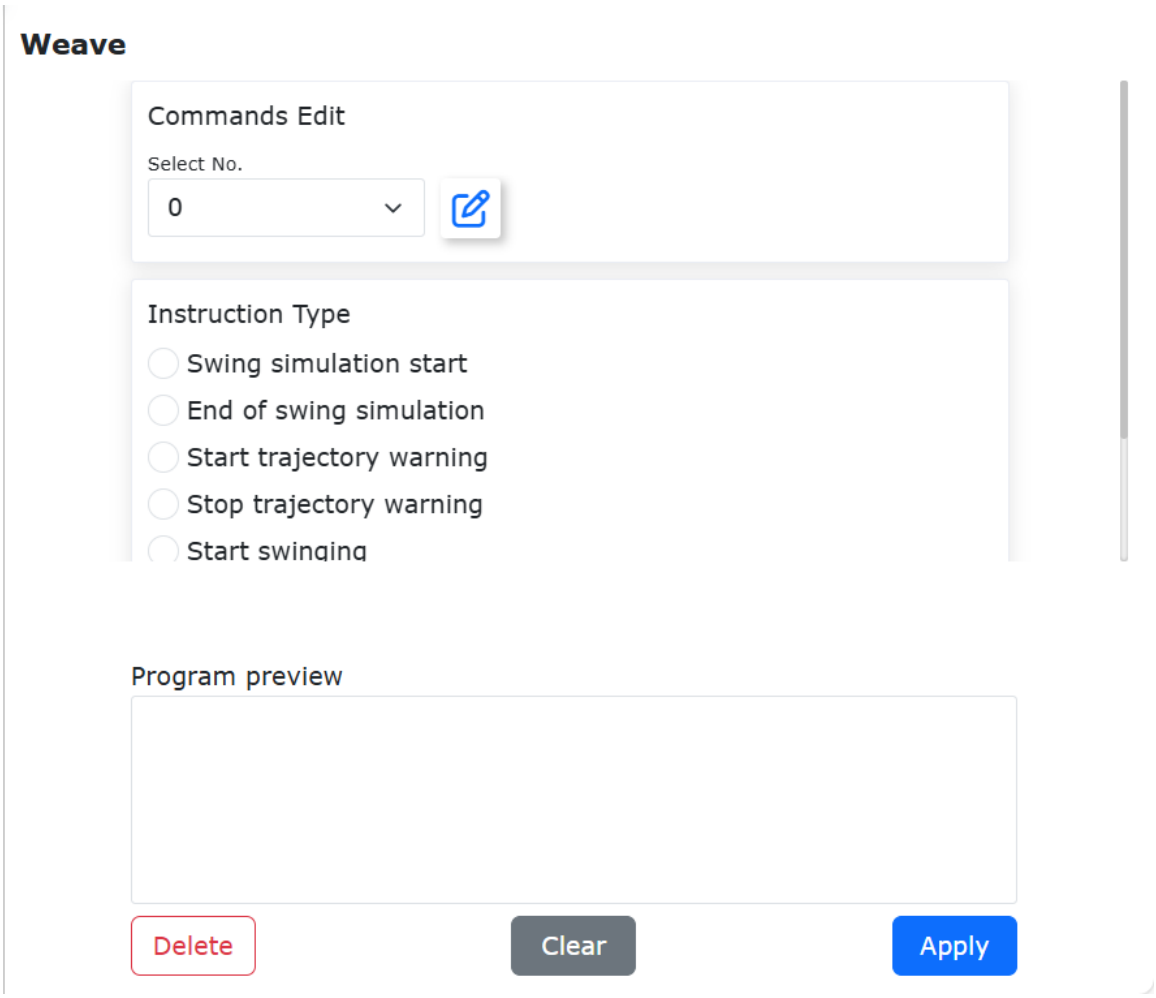


Figure 9.5-25 Weave command interface

- Click "Configuration and Test" to select the weaving type according to the usage scenario and configure the weaving parameters. After the configuration is completed, the weaving trajectory can be tested by pressing the start weaving test and stop weaving test buttons. The current swing types are:
  - Triangular wave swing (LIN/ARC)
  - Vertical L-shaped triangular wave swing (LIN/ARC)
  - Circular Oscillation - Clockwise (LIN)
  - Circular Oscillation - Counterclockwise (LIN)
  - Sine wave swing (LIN/ARC)
  - Vertical L-shaped sine wave swing (LIN/ARC)
  - Vertical welding triangle swing

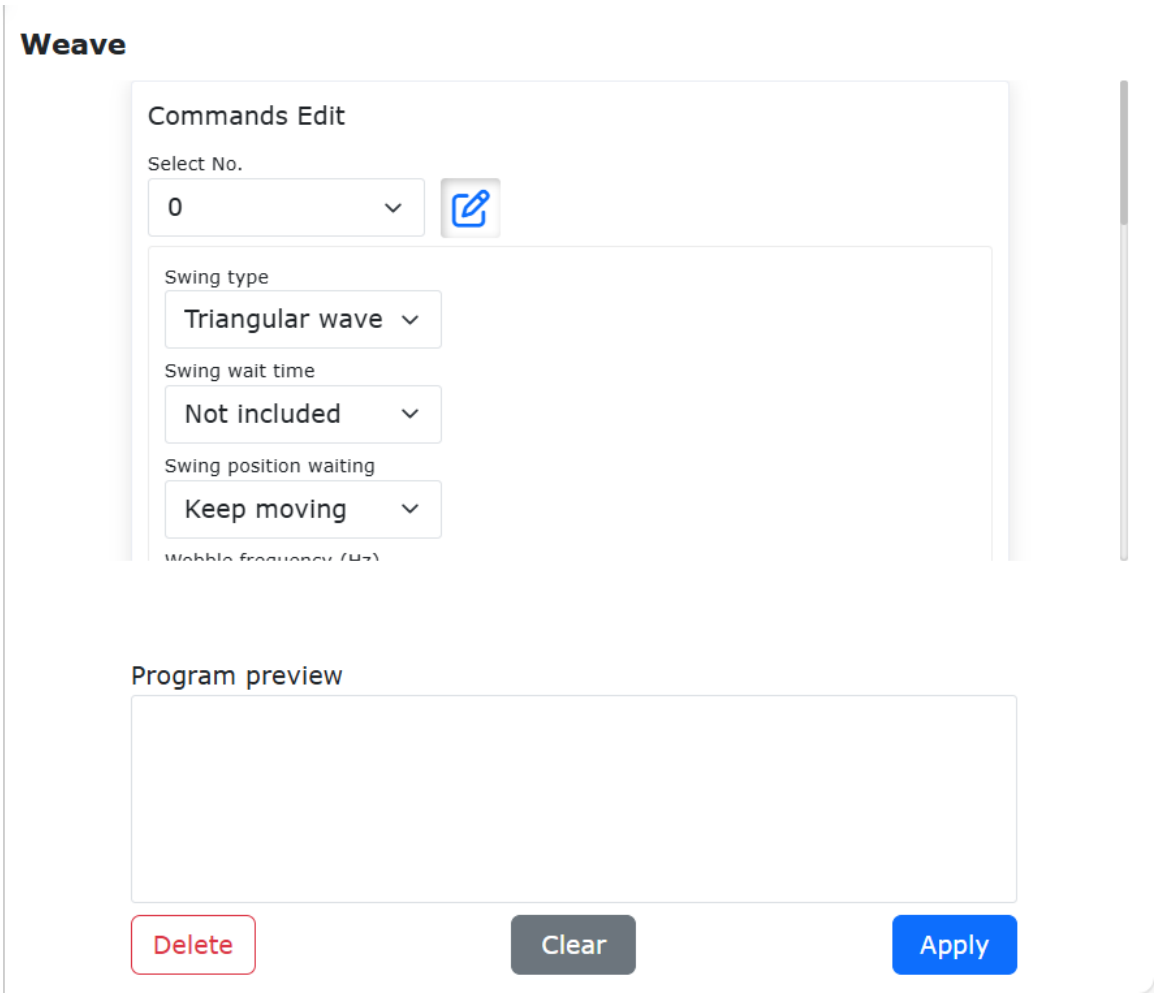


Figure 9.5-26 Weave configuration and testing command interface

1.3.9.5.10.1 Slope serration weave function

This function allows the end of the robot tool to generate slope serration weave trajectory in Cartesian space. The weave is superimposed on the linear trajectory, and the azimuth parameter (unit deg) defines the azimuth angle of the swing welding on the specified welding plane;

When the value is positive, the left endpoint is skewed in the forward direction, and when it is negative, the right endpoint is skewed in the forward direction; If it is 90deg or -90deg, it can weave in the forward direction.

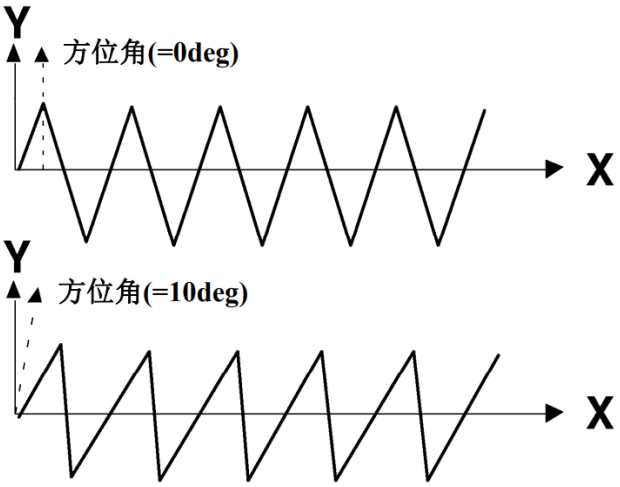


Figure 9.5-26-1 Swing azimuth effect

Step1:Edit to set up basic linear motion.

```
1  PTP(weave0,100,-1,0)
2  Lin(weave1,10,-1,0,0)
```

Figure 9.5-26-2 Example of a basic linear motion LUA program

Step2:Add weave command in motion command block.

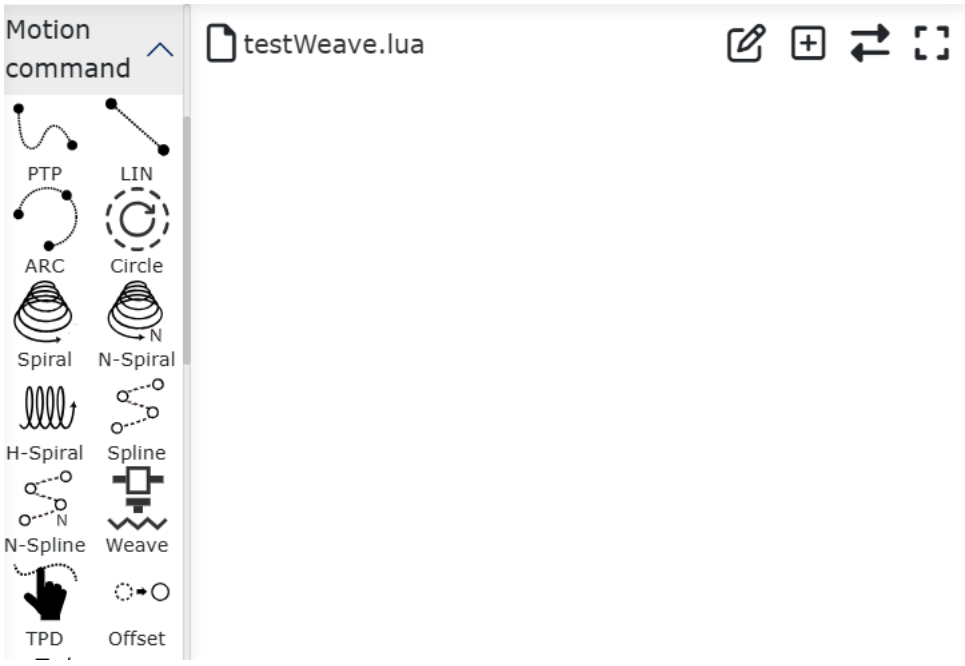


Figure 9.5-26-3 Add weave command

Step3:Click the “Configure” button, select “Triangular wave swing” or “Sine wave swing” from the drop-down box, imput desired azimuth angle in “Swing direction azimuth” box and click “Apply”.

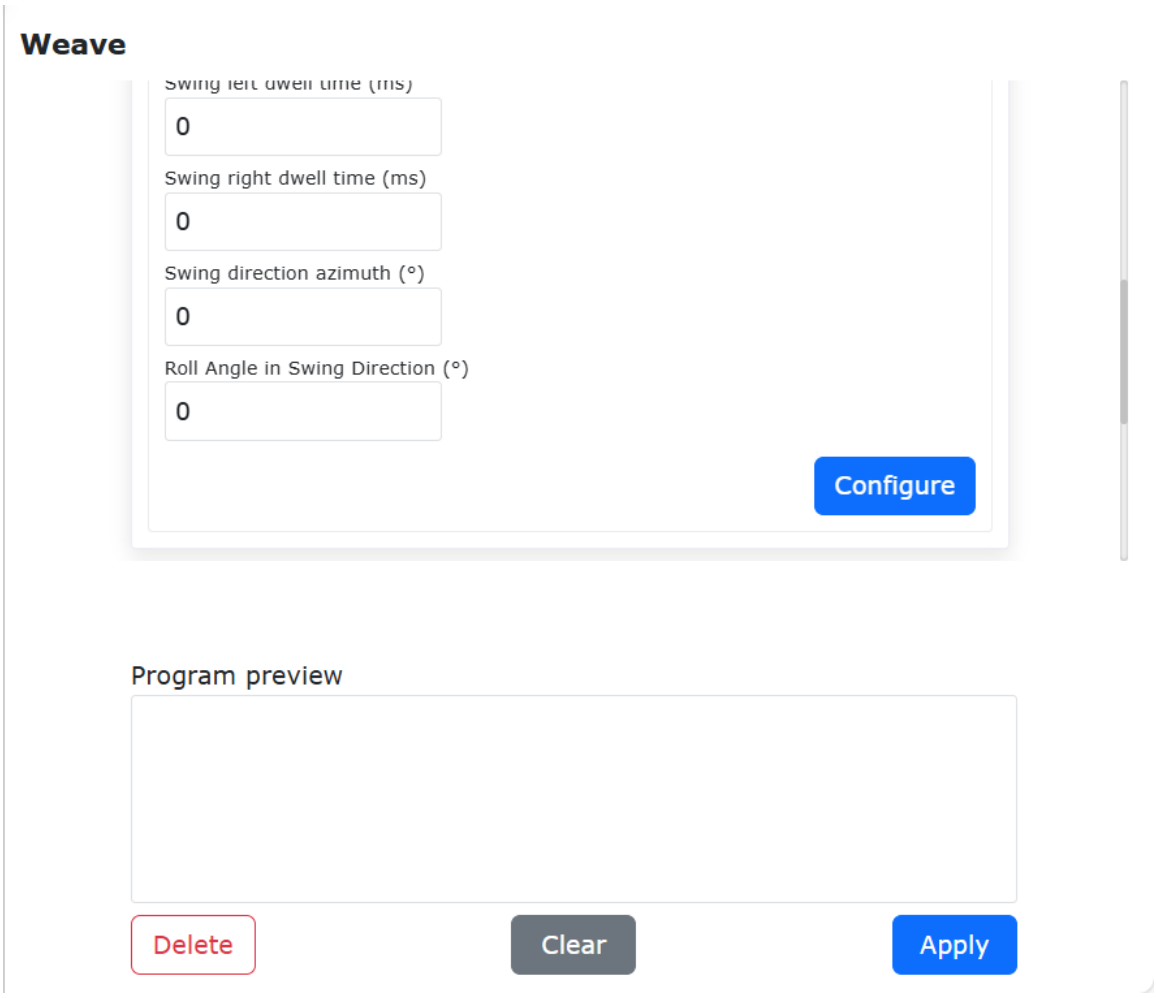


Figure 9.5-26-4 Weave parameter configuration

Step4:Click the “Start Swing” button to add the swing command above the linear motion; Click the “Stop Swing” button to add the swing command below the linear motion.

```
1  PTP(weave0,100,-1,0)
2  WeaveStart(0)
3  Lin(weave1,10,-1,0,0)
4  WeaveEnd(0)
```

Figure 9.5-26-5 Add the lua program after the swing command

Step5:Click “Start Running”buttom, and the end trajectory of the robot is shown in the figure.

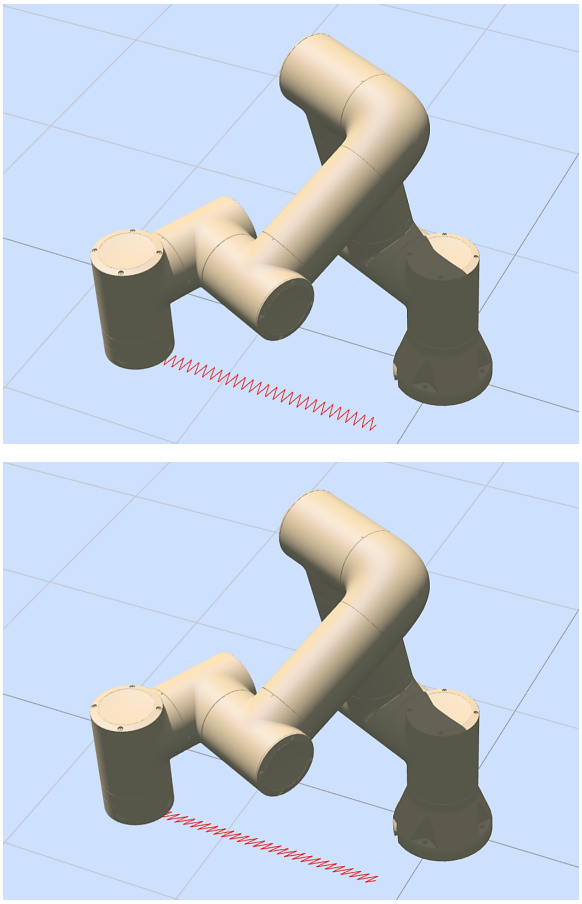


Figure 9.5-26-6 Zigzag weave (left) slope serration weave (right)

1.3.9.5.11 TPD command

Click the "TPD" button to enter the TPD command editing interface.

In this command, the user first needs to have a recorded track.

About track recording: Before preparing to record the track, first save the starting point of the track. When the robot is in the dragging mode, input the file name, select the period (assuming the value is x, that is, record a point every x milliseconds, it is recommended to record a point every 4 milliseconds), the point starts recording, and the user can drag the robot to specify Movement, after the recording is completed, click to stop recording, and the previous movement track of the robot can be saved. When a movement cannot be fully recorded, a A reminder that the number of recording points exceeds the limit is displayed, and the user needs to record the exercise in several times.

When programming, first use the PTP instruction to reach the starting point of the corresponding trajectory, then select the trajectory in the TPD trajectory reproduction instruction, select whether it is smooth, set the debugging speed, and click "Add" and "Apply" in sequence to insert the program. The trajectory loading command is mainly used to pre-read the trajectory file and extract it into a trajectory command, which is better applied to the conveyor belt tracking scene.

Note: For the detailed operation of TPD, please refer to the teaching programming (TPD) function operation instruction module.

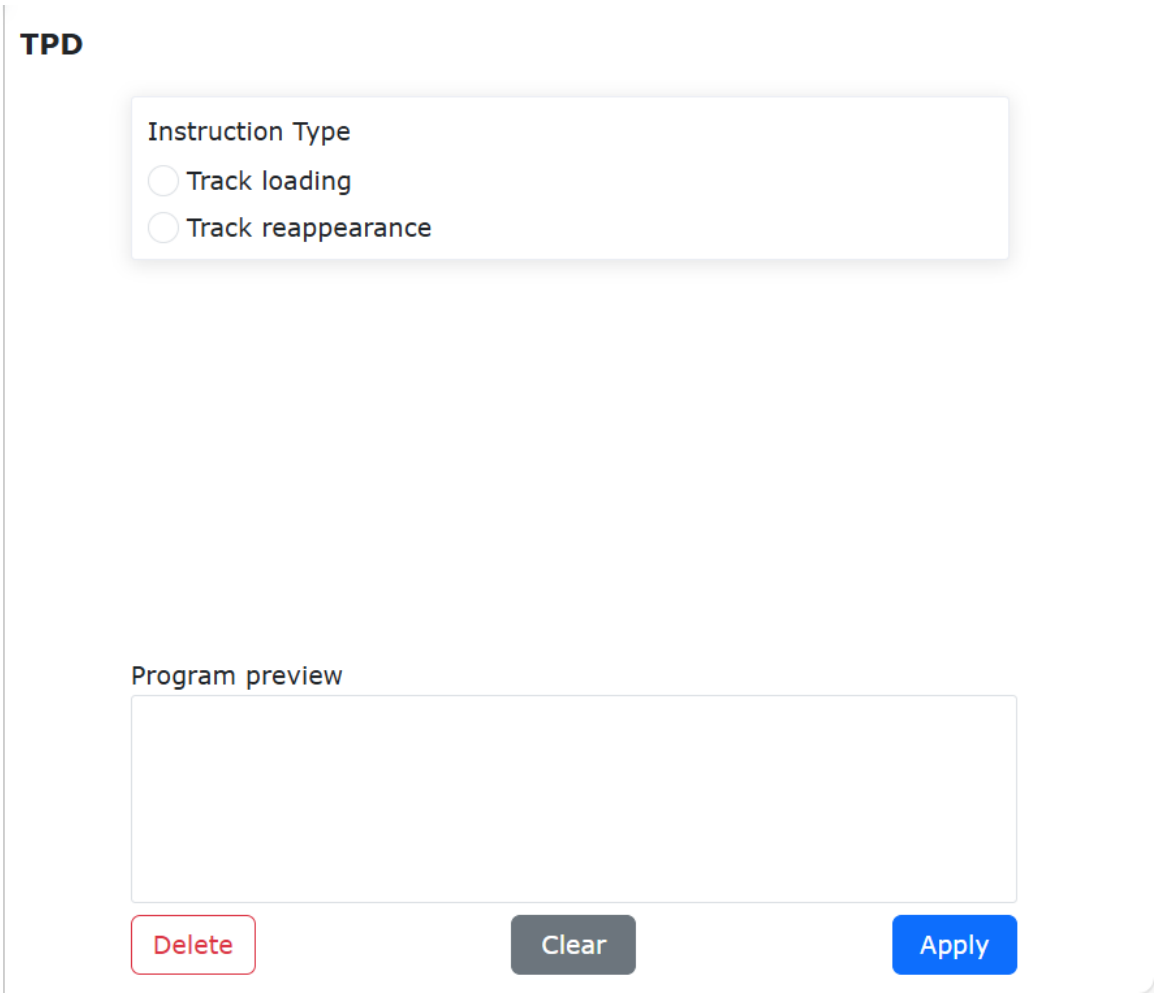


Figure 9.5-27 TPD command interface

1.3.9.5.12 Offset command

Click the “Offset” icon to enter the Offset command editing interface.

This command is an overall offset command. Input each offset, add the opening command and closing command to the program, and the motion command between the start and close will be offset based on the base coordinates (or workpiece coordinates).

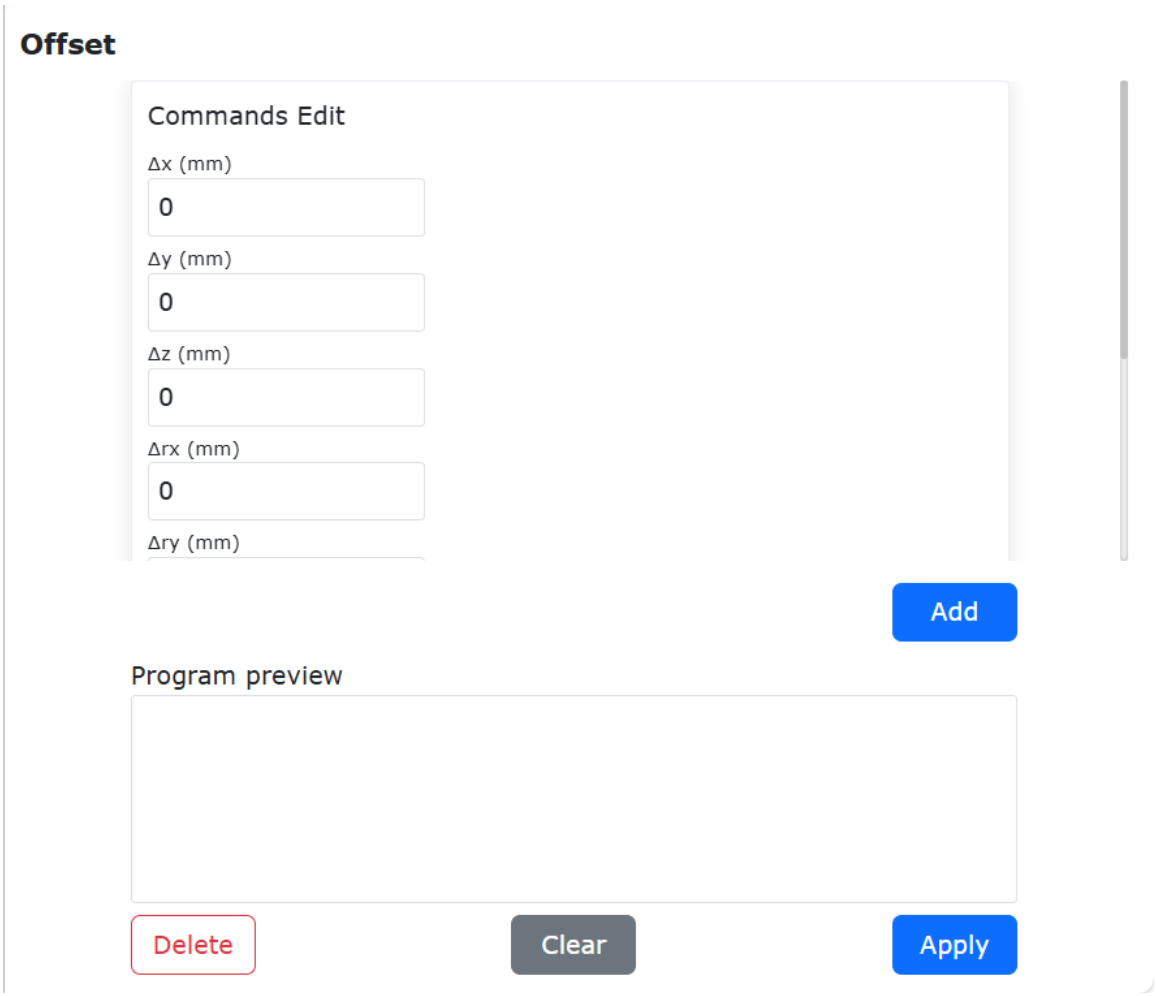


Figure 9.5-28 Offset command interface

1.3.9.5.13 Servo Command

Click the “Servo” icon to enter the servoMotion command editing interface. Servo motion includes servo motion in Cartesian space and servo motion in joint space.

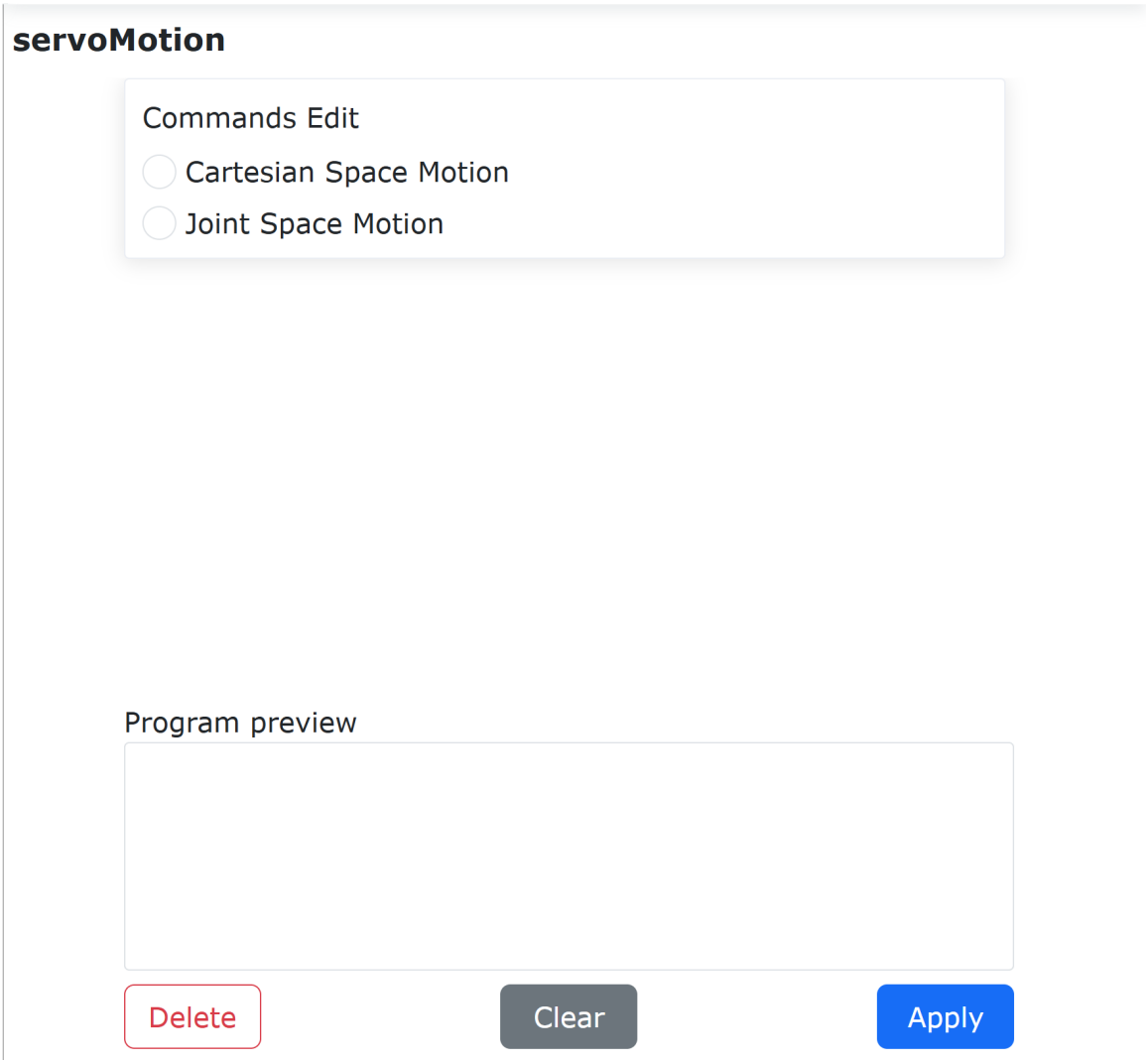


Figure 9.5-29-1 Servo Motion Command Interface

1.3.9.5.13.1 Servo Motion in Cartesian Space

ServoCart servo control (Cartesian space motion) command. This command can control robot motion through absolute pose control or offset control based on the current pose.

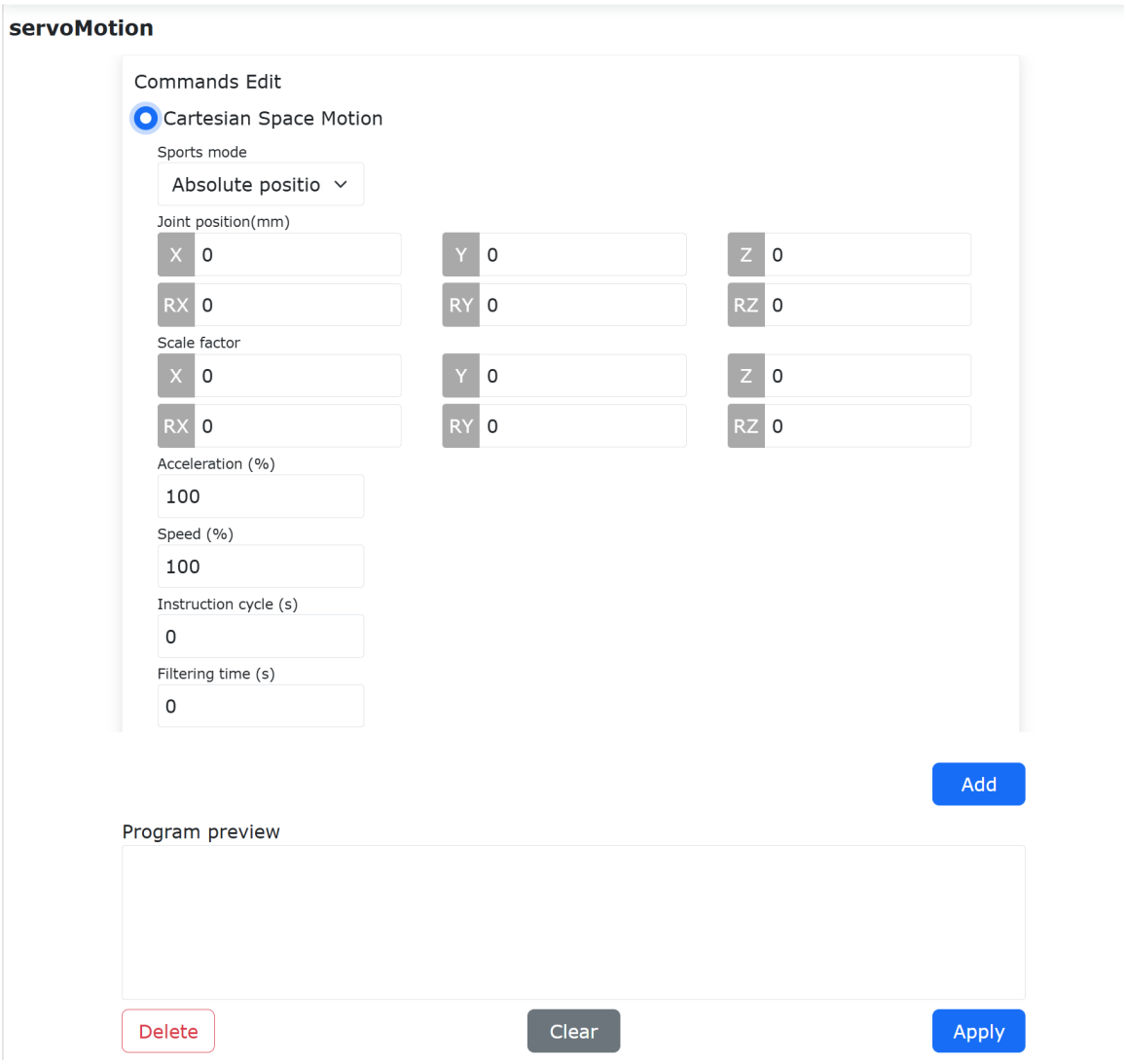


Figure 9.5-29-2 ServoCart Command Interface

Example of absolute pose control program:

```
PTP(p1,100,-1,0)
x,y,z,rx,ry,rz = GetActualTCPPose()
while 1 do
    if type(x) == "number" then
    x = x + 1
    WaitMs(8)
    ServoCart(0,x,y,z,rx,ry,rz,0,0,0,0,0,0,100,100,0.008,0,0)
    end
end
```

Figure 9.5-29-3 ServoCart Absolute Motion

In this example, x, y, z, rx, ry, rz (Cartesian position) are the acquired current robot position. Additionally, users can control robot motion by reading trajectory data files, sending trajectory data via socket communication, etc.

Example of control program based on offset from current pose (base coordinate offset):

```
PTP(p1,100,-1,0)
while 1 do
    WaitMs(8)
    ServoCart(1,0,1,-1,0,0,0,0.4,0.4,0.4,0.2,0.2,0.2,100,100,0.008,0,0)
    end
end
```

Figure 9.5-29-4 ServoCart Relative Motion

1.3.9.5.13.2 Servo Motion in Joint Space

ServoJ servo control (joint space motion) command. This command can control robot motion through absolute robot joint positions.

Click "Teaching Program", "Program Programming", "Servo Motion" in sequence. On the servoMotion command page, select "Joint Space Motion".

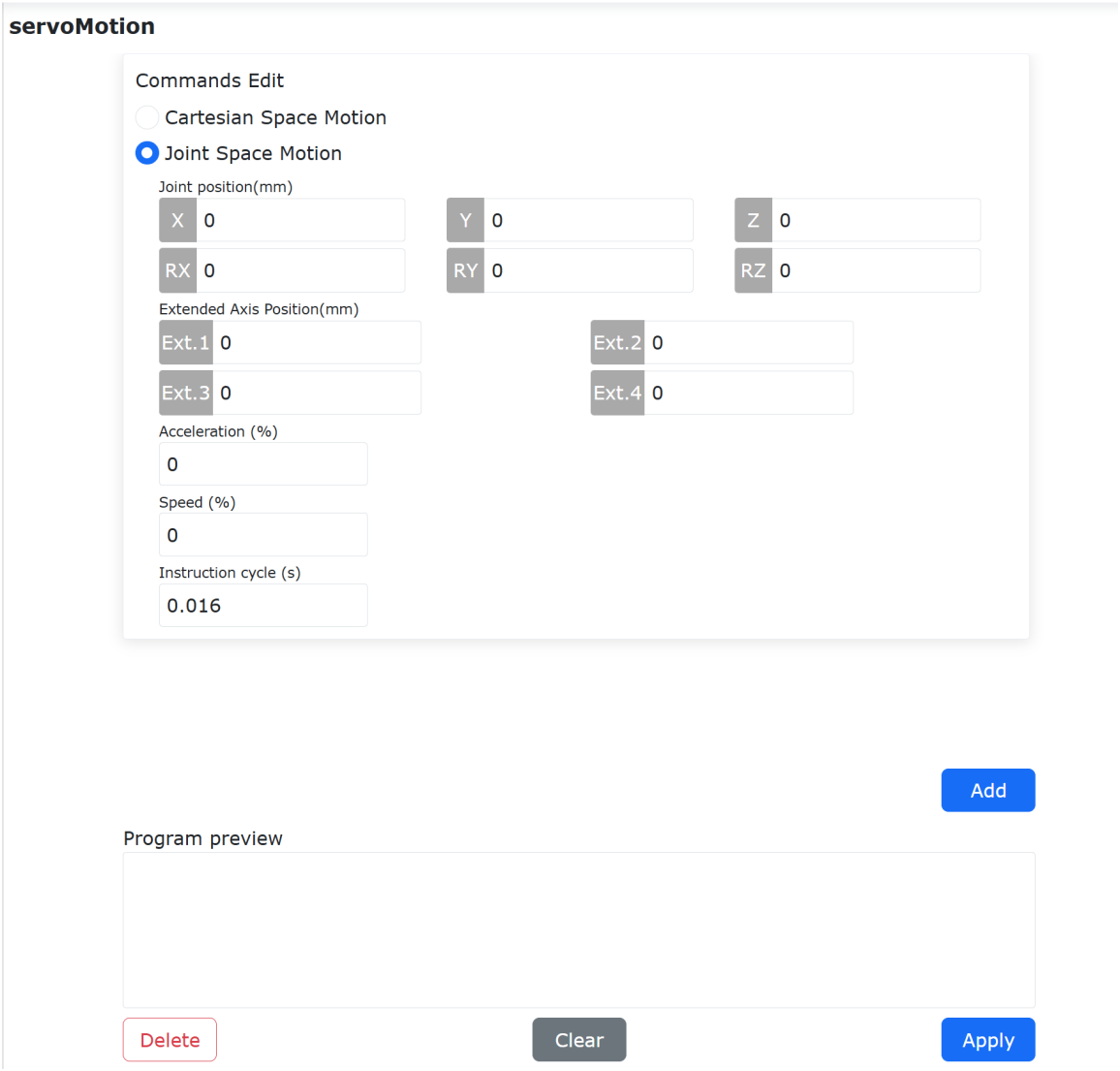


Figure 9.5-29-5 ServoJ Command Editing

The parameters in the command are explained as follows:

- **Joint Position**: The target joint position for ServoJ motion. The motion from the current position to the target position needs to be completed within the set command cycle. If the deviation between the target position and the current position command is too large, the robot may report errors such as joint overspeed.
- **Extended Axis Position**: The target extended axis position for ServoJ motion.
- **Acceleration**: Acceleration percentage for ServoJ motion (currently not available).
- **Speed**: Speed percentage for ServoJ motion (currently not available; the actual robot operating speed currently depends on the position difference between two ServoJ commands and the command cycle).

- **Command Cycle**: Execution time interval between two ServoJ commands.

Input the corresponding target position, speed, acceleration, and command cycle. Click the "Add" and "Apply" buttons to add a ServoJ command to the LUA program.

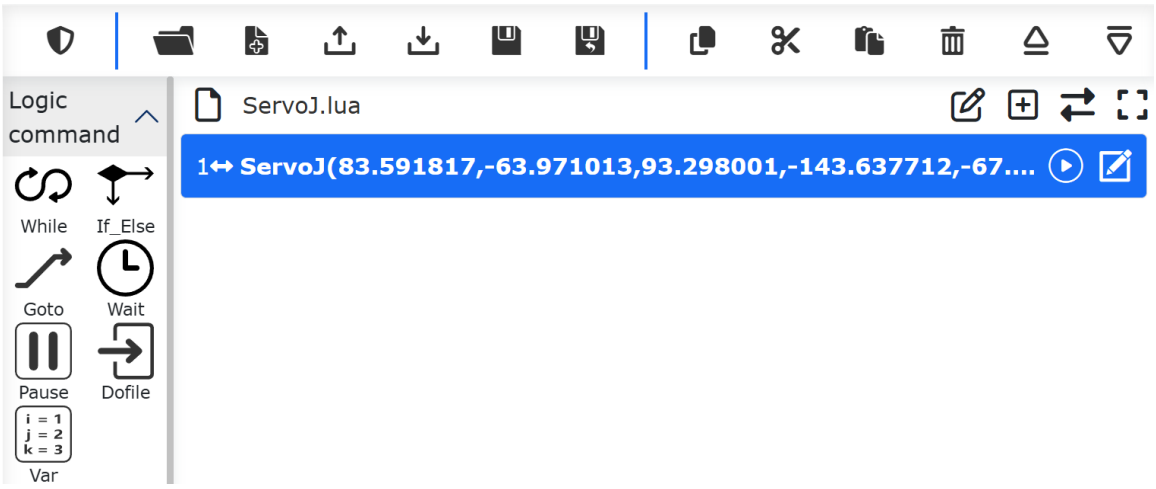


Figure 9.5-29-6 ServoJ Command Added to Lua Program

In practice, it is often necessary to continuously send multiple ServoJ commands according to the set command cycle. The target joint positions of these ServoJ commands form a continuous robot motion curve, enabling flexible robot motion control. The sending cycle of the commands must match the set command cycle.

ServoJ motion control can be implemented in the LUA program through loops or by adding multiple commands consecutively.

```
1  j5 = 89.379
2  i = 0
3  while i < 100 do
4      i = i + 1
5      j5 = 89.379 + i * 0.2
6      ServoJ(-8.804,-70.906,-125.749,-73.132,j5,-114.28,0,0,0,0,100,100,0.016,0,0)
7      WaitMs(16)
8  end
9  i = 0
10 while i < 100 do
11     i = i + 1
12     j5 = 109.379 - i * 0.2
13     ServoJ(-8.804,-70.906,-125.749,-73.132,j5,-114.28,0,0,0,0,100,100,0.016,0,0)
14     WaitMs(16)
15 end
```

Figure 9.5-29-7 Example of Continuous ServoJ Motion

```
ServoJ(83.592000,-63.971000,93.298000,-143.638000,-67.888000,-57.305000,0,0,0.008,0,0)
ServoJ(83.591817,-63.971013,93.298001,-143.637712,-67.888005,-57.304858,0,0,0.008,0,0)
ServoJ(83.591266,-63.971051,93.298005,-143.636848,-67.888018,-57.304431,0,0,0.008,0,0)
ServoJ(83.590349,-63.971114,93.298011,-143.635408,-67.888042,-57.303719,0,0,0.008,0,0)
ServoJ(83.589065,-63.971203,93.298019,-143.633392,-67.888074,-57.302723,0,0,0.008,0,0)
ServoJ(83.587414,-63.971318,93.298030,-143.630800,-67.888115,-57.301443,0,0,0.008,0,0)
ServoJ(83.585396,-63.971457,93.298043,-143.627632,-67.888166,-57.299878,0,0,0.008,0,0)
ServoJ(83.583012,-63.971623,93.298058,-143.623888,-67.888226,-57.298028,0,0,0.008,0,0)
ServoJ(83.580260,-63.971813,93.298076,-143.619568,-67.888296,-57.295894,0,0,0.008,0,0)
ServoJ(83.577142,-63.972029,93.298096,-143.614672,-67.888374,-57.293475,0,0,0.008,0,0)
ServoJ(83.573657,-63.972271,93.298119,-143.609200,-67.888462,-57.290771,0,0,0.008,0,0)
ServoJ(83.569805,-63.972538,93.298144,-143.603152,-67.888559,-57.287783,0,0,0.008,0,0)
ServoJ(83.565586,-63.972830,93.298171,-143.596528,-67.888665,-57.284511,0,0,0.008,0,0)
ServoJ(83.561000,-63.973148,93.298201,-143.589328,-67.888780,-57.280953,0,0,0.008,0,0)
ServoJ(83.556047,-63.973491,93.298233,-143.581552,-67.888905,-57.277112,0,0,0.008,0,0)
ServoJ(83.550728,-63.973859,93.298267,-143.573200,-67.889039,-57.272985,0,0,0.008,0,0)
ServoJ(83.545041,-63.974253,93.298304,-143.564272,-67.889182,-57.268574,0,0,0.008,0,0)
ServoJ(83.538988,-63.974673,93.298343,-143.554768,-67.889335,-57.263879,0,0,0.008,0,0)
```

Figure 9.5-29-8 Example of Continuous ServoJ Motion 1

1.3.9.5.13.3 Extended Axis Debugging Instructions

1.3.9.5.13.4 Overview

The GetInverseKinExaxis instruction interface for the inverse kinematics of extended axis target positions and the ServoCart instruction interface with extended axis positions support various scenarios where extended axes and robots are used simultaneously.

1.3.9.5.13.5 Operation Process

Step1: The parameters and return value of the inverse kinematics GetInverseKinExaxis instruction interface are shown in the tables below.

Table 9.5-1 GetInverseKinExaxis Parameters

No.	Data Type	Variable Name	Detailed Description
1	uint8_t	posMode	0: Absolute pose, 1: Relative pose - Base coordinate system, 2: Relative pose - Tool coordinate system
2	float	desePos[6]	Robot Cartesian position
3	float	exaxis[4]	Extended axis position
4	int	toolNum	Tool number [0-14]
5	int	workPieceNum	Workpiece number [0-14]

Table 9.5-2 GetInverseKinExaxis Return Value

No.	Data Type	Variable Name	Detailed Description
1	float	jointPos[6]	Joint position

Step2: The calling format of the inverse kinematics GetInverseKinExaxis instruction in a Lua program is shown in the figure below. By simply inputting the parameters listed in the table, the corresponding joint values can be obtained. For calls in the SDK, refer to the corresponding SDK documentation.

```
j1,j2,j3,j4,j5,j6 = GetInverseKinExaxis(0,{x,y,z,a,b,c},{e1,e2,e3,e4},tool,work)
```

Figure 9.5-29-9 GetInverseKinExaxis Call in Lua

Step3: The ServoCart instruction interface with extended axis positions is shown in the table below, with no return value.

Table 9.5-3 ServoCart Parameters

No.	Data Type	Variable Name	Detailed Description
1	uint8_t	posMode	0: Absolute pose, 1: Relative pose - Base coordinate system, 2: Relative pose - Tool coordinate system
2	float	desePos[6]	Robot Cartesian position
3	float	gain[6]	Pose proportionality coefficient, used in relative pose cases
4	float	exaxis[4]	Extended axis position
5	float	acc	Acceleration proportion, 0~100, default 0
6	float	vel	Velocity proportion, 0~100, default 0
7	float	interval	Instruction period [s]
8	float	filterTime	Filter time [s], temporarily unavailable
9	float	posGain	Proportional amplifier for target position, temporarily unavailable

Step4: The calling format of the ServoCart instruction with extended axis positions in a Lua program is shown in the figure below. By simply inputting the parameters listed in the table, the robot can execute ServoCart motion with extended axis positions. For calls in the SDK, refer to the corresponding SDK documentation.

```
ServoCart(0,{x,y,z,a,b,c}, {0,0,0,0,0,0},{e1,e2,e3,e4},0,0,0.001,0,0)
```

Figure 9.5-29-10 ServoCart Call in Lua

1.3.9.5.14 Trajctory command

Click the “Trajctory” icon to enter the Trajctory command editing interface.

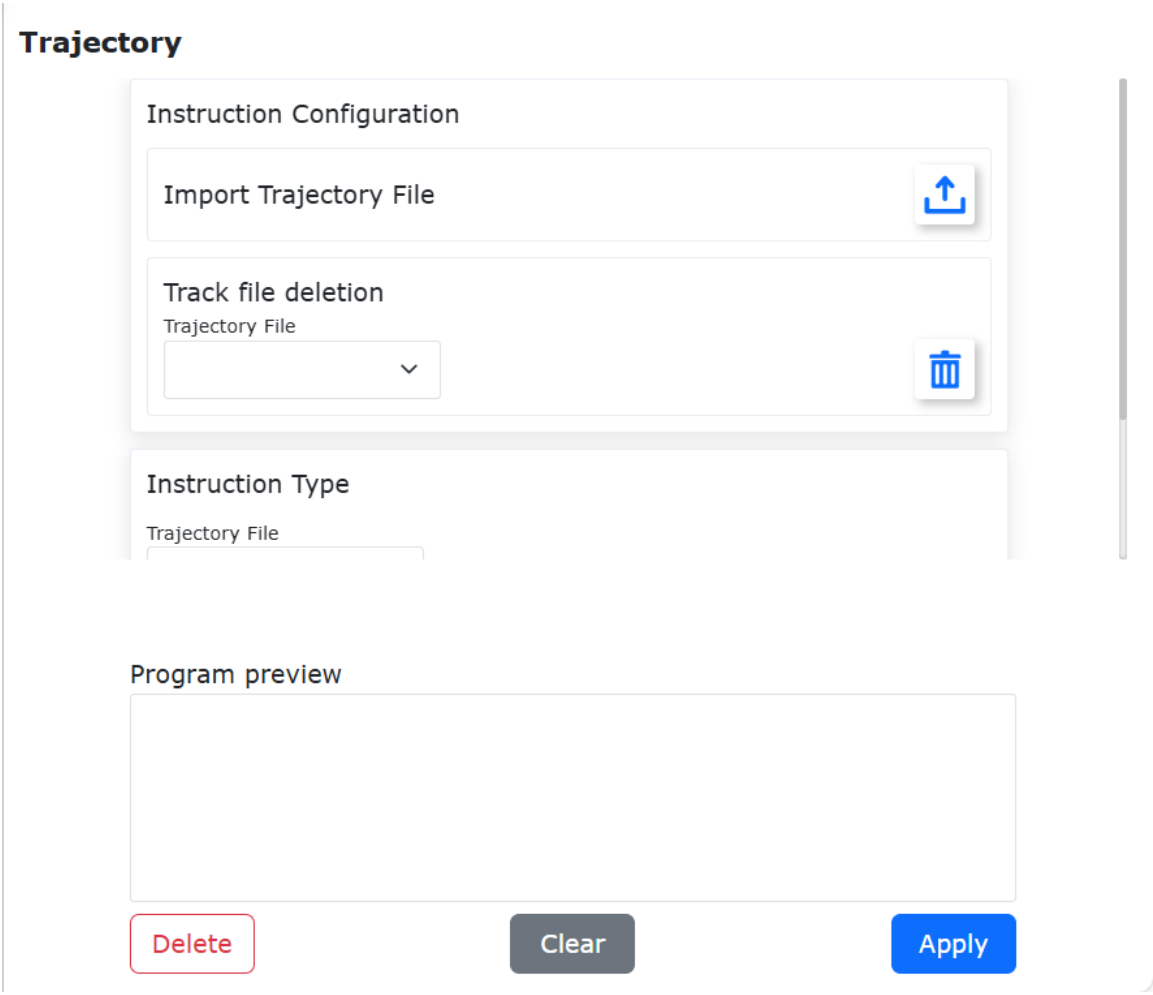


Figure 9.5-30 Trajctory command interface

1.3.9.5.15 TrajctoryJ command

Click the “TrajctoryJ” icon to enter the TrajctoryJ command editing interface.

1. Trajectory file import function: select a local computer file to import to the robot control system.
2. Track preloading: select the imported track file and load it by command.
3. Trajectory movement: Send the robot movement through the combination command of the preloaded trajectory file and the selected debugging speed.
4. Print the track point number: print the track point number during the robot running track, so as to check the progress of the current movement.

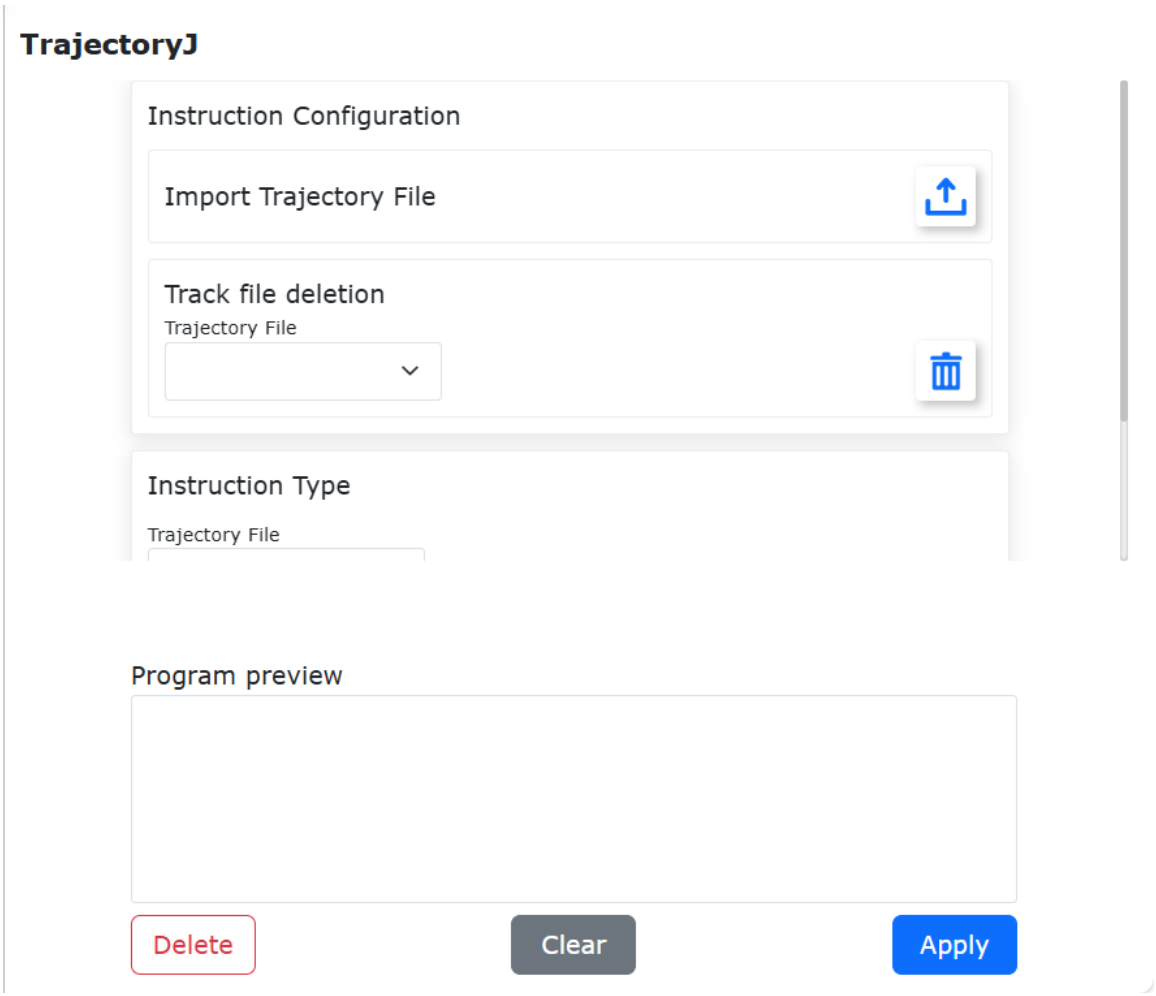


Figure 9.5-31 TrajctoryJ command interface

1.3.9.5.16 DMP command

Click the “DMP” icon to enter the DMP command editing interface.

DMP is a trajectory imitation learning method that requires prior planning of reference trajectories. In the command editing interface. , select the teaching point as the new starting point, click “Add” and “Apply” to save the command. The DMP specific path is a new trajectory that mimics the reference trajectory with a new starting point.

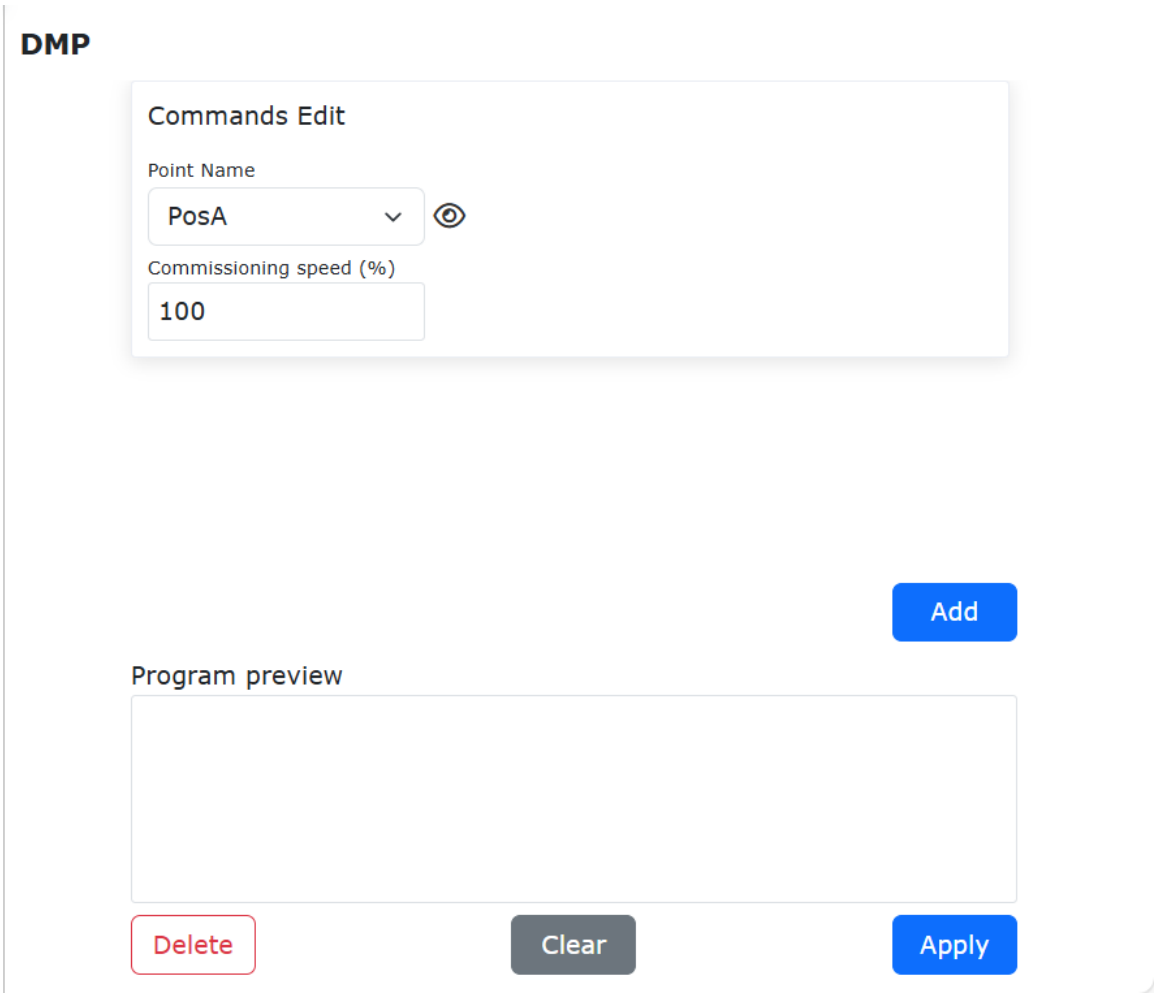


Figure 9.5-32 DMP command interface

1.3.9.5.17 WPTrsf command

Click the “WPTrsf” icon to enter the WPTrsf command editing interface.

Select the workpiece coordinate system to be automatically converted, and click “Add” and “Apply” to save the instruction. This instruction realizes automatic conversion of points in the workpiece coordinate system when executing internal PTP and LIN instructions. Use the example area to show and prompt the correct combination of instructions. After adding specific instructions, you can adjust the combination according to the actual scene.

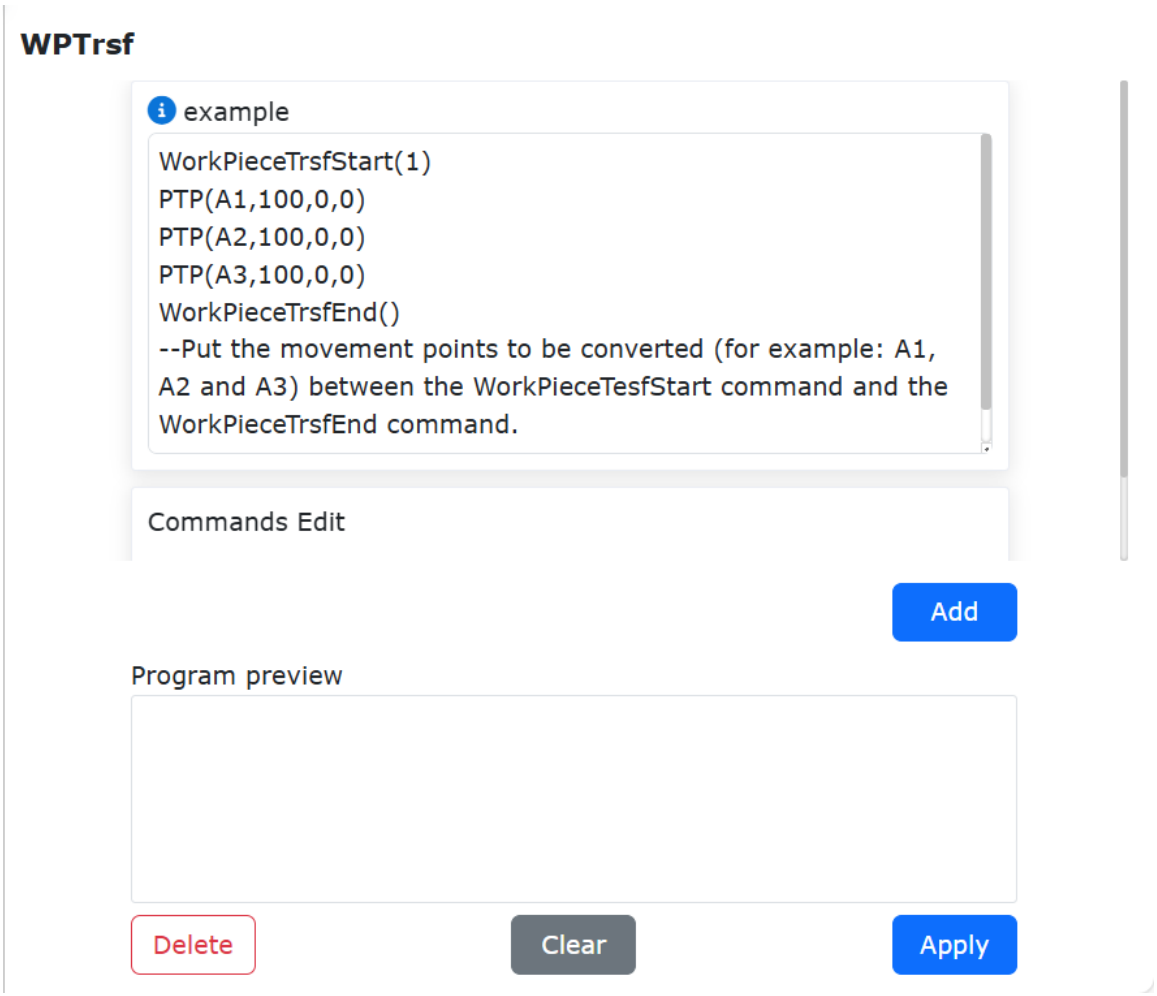


Figure 9.5-33 WPTrsf command interface

1.3.9.5.18 Tool conversion command

Click the “Tool conversion” icon to enter the ToolTrsf command editing interface.

After adding PTP and Lin instructions, select the tool coordinate system to be automatically converted, click “Add” and “Apply” to save the instruction, and the Cartesian coordinates of the points in the instruction will be automatically converted according to the currently set workpiece coordinate system.

Note: The usage example area shows and prompts the correct combination of instructions. After adding specific instructions, you can adjust the combination according to the actual scenario.

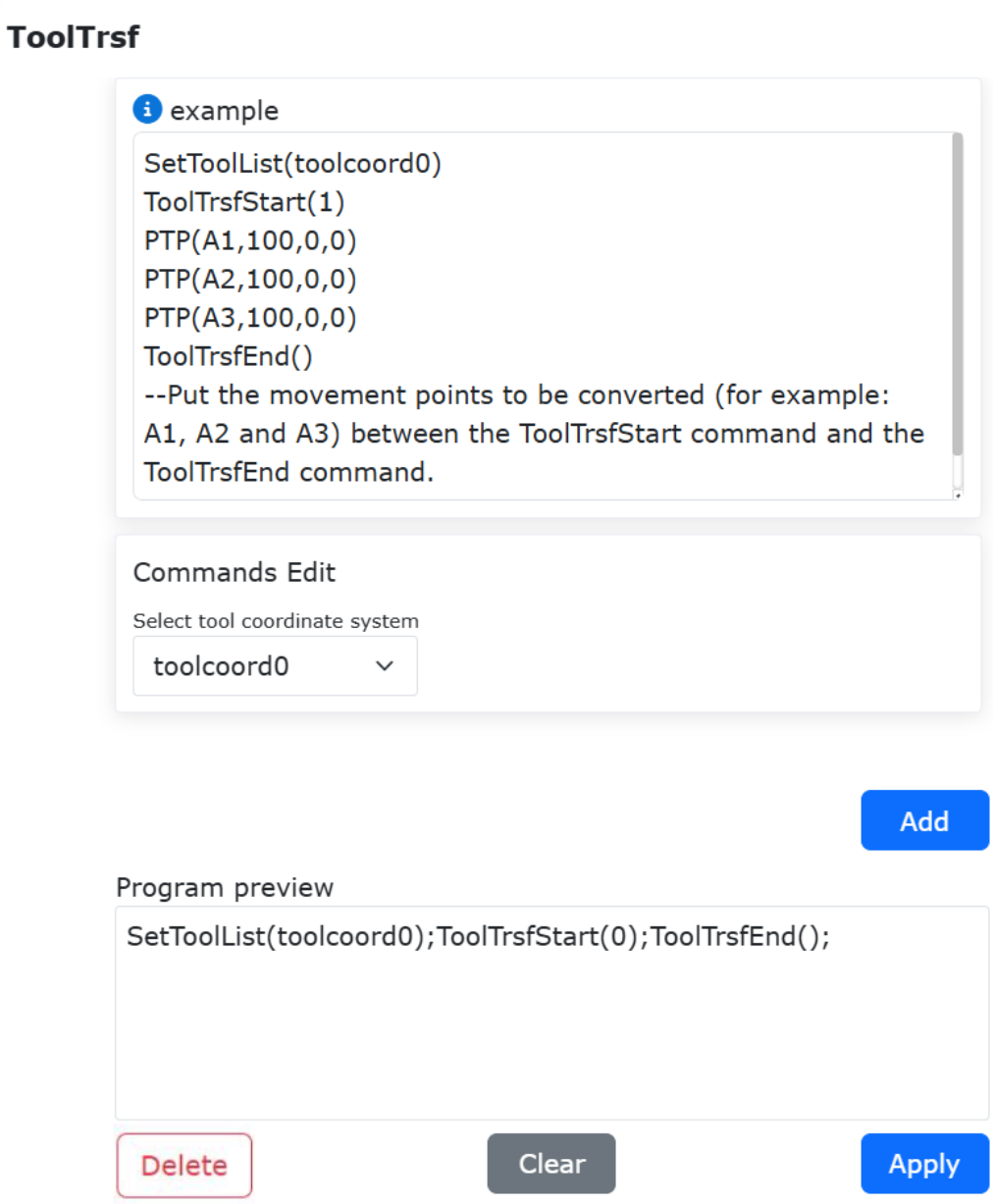


Figure 9.5-34 ToolTrsf command interface

1.3.9.6 Control command interface

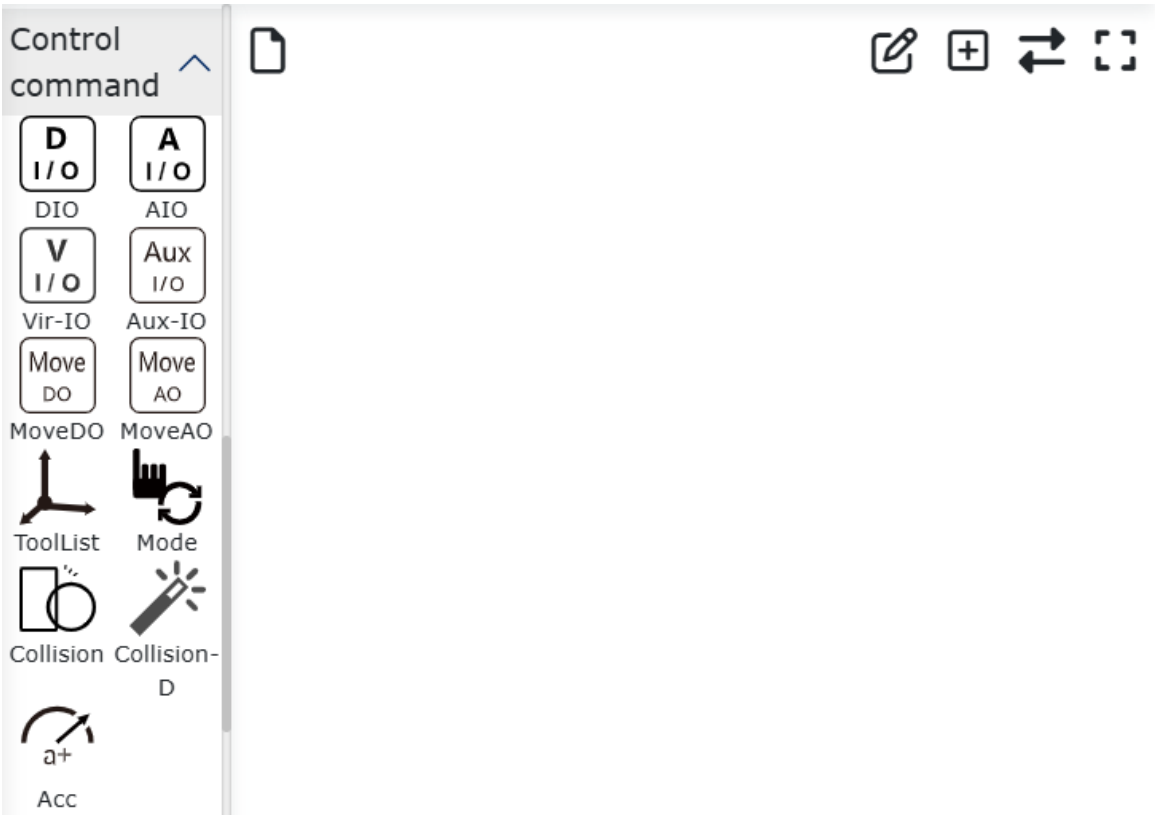


Figure 9.6 Control command interface

1.3.9.6.1 IO command

Click the “IO” icon to enter the IO command editing interface.

The “IO” command is divided into two parts: setting IO (SetDO/SPLCSetDO) and getting IO (GetDI/SPLCGetDI).

“SetDO/SPLCSetDO” This command can set the specified output DO state, including 16 digital outputs of the control box and 2 digital outputs of the tool. The state option “False” is closed, “True” is open, and whether to block the option selects “blocked”. “Indicates that the DO state is set after the movement stops, and selecting the “non-blocking” option means that the DO state is set during the last movement. Selecting “Break” for the smooth trajectory option means setting the DO state after the smooth transition radius ends, and selecting “Serious” means setting the DO state during the smooth transition radius movement. When this instruction is added in the auxiliary thread, whether the application thread needs to select yes, and other places use this instruction to select no. Click “Add”, “Apply”.

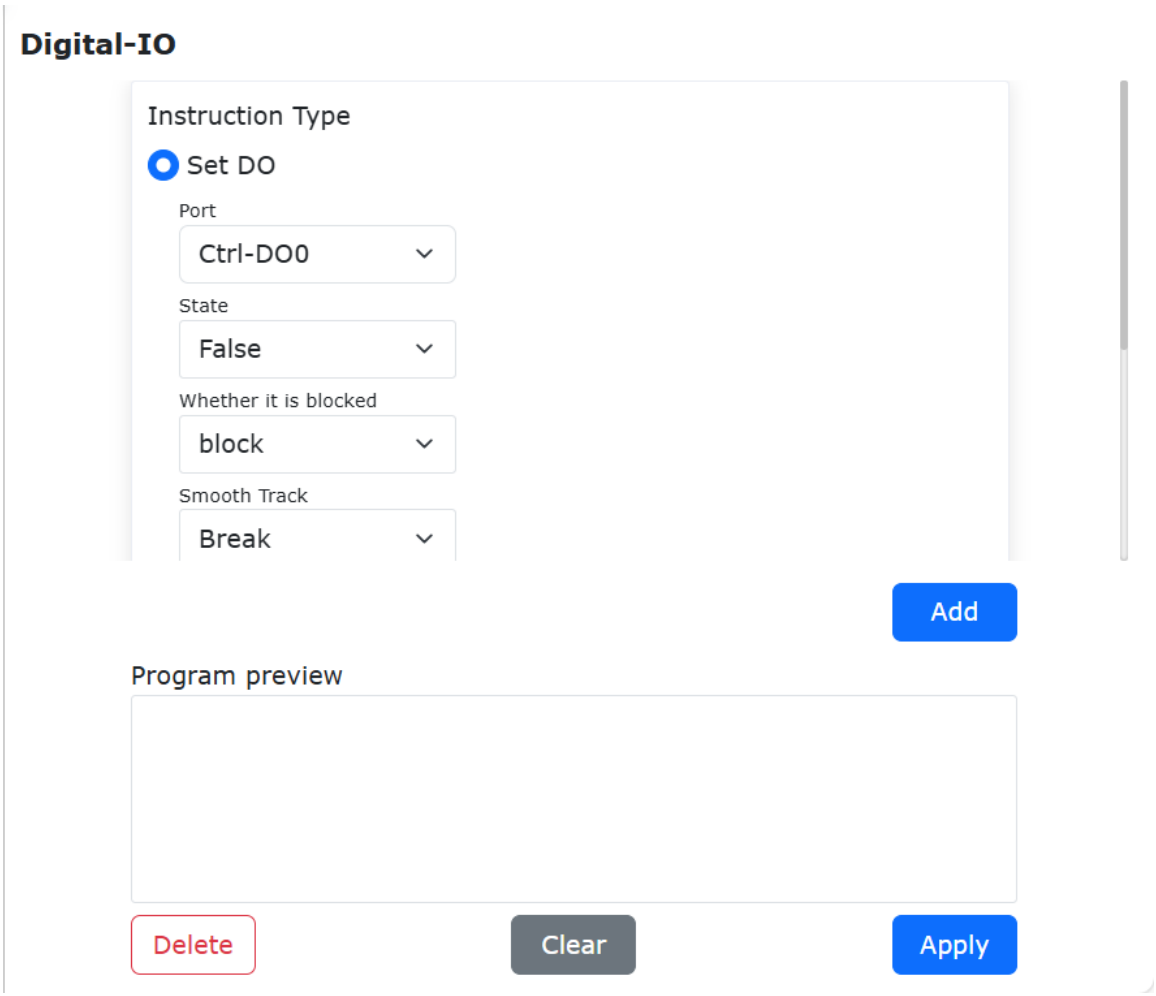


Figure 9.6-1 SetDO command interface

In the “GetDI/SPLCGetDI” command, select the value of the port number you want to get, whether to block or not, select “block” to get the DI status after the movement stops, and select the “non-blocking” option to get the DI state during the last movement. When this instruction is added in the auxiliary thread, whether the application thread needs to select yes, and other places use this instruction to select no. After selection, click the “Add” and “Apply” buttons.

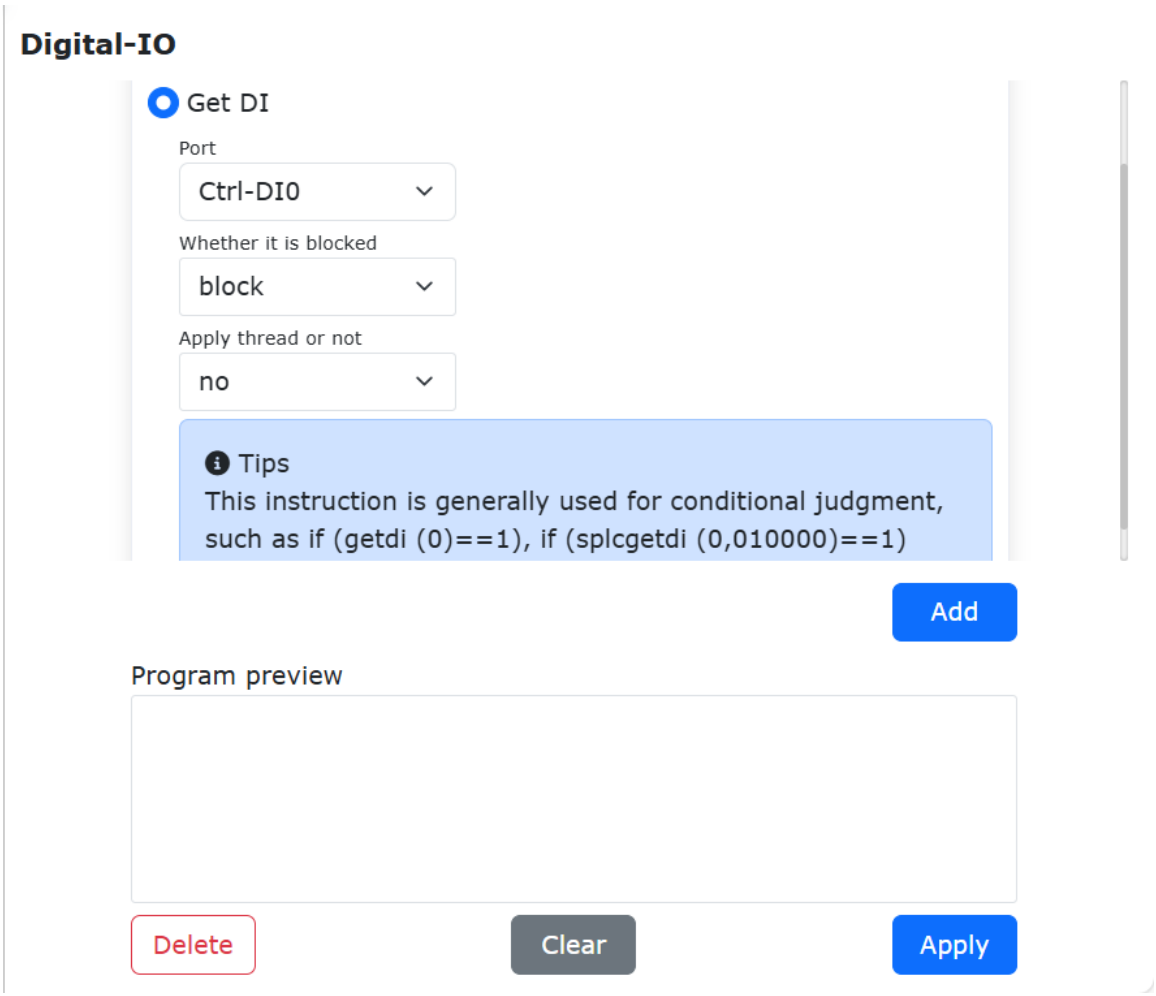


Figure 9.6-2 GetDI command interface

1.3.9.6.2 AI command

Click the “AI” icon to enter the AI command editing interface.

In this instruction, it is divided into two functions: setting analog output (SetAO/SPLCSetAO) and obtaining analog input (GetAI/SPLCGetAI).

“SetAO/SPLCSetAO” select the analog output that needs to be set, input the value that needs to be set, the range is 0-10, whether to block or not select “block” means to set the AO state after the movement stops, select “non-block” means to set the AO state after the last movement Set the AO state in the process. When this instruction is added in the auxiliary thread, whether the application thread needs to select yes, and other places use this instruction to select no. Click “Add”, “Apply”.

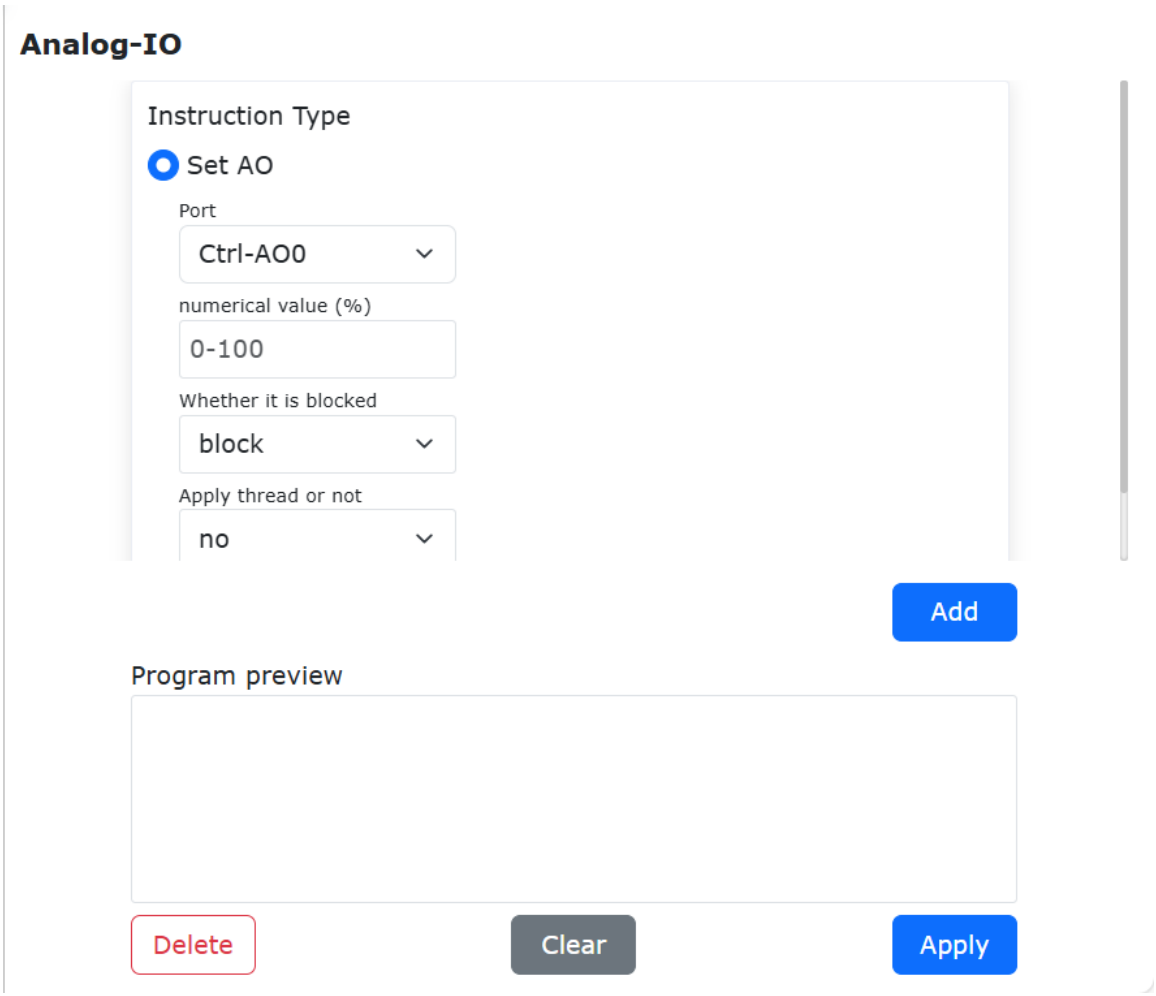


Figure 9.6-3 SetAO command interface

"GetAI/SPLCGetAI" selects the analog input that needs to be obtained, whether to block or not selects "blocked" to obtain the AI state after the movement stops, and selects the "non-blocked" option to obtain the AI state during the last movement. When this instruction is added in the auxiliary thread, whether the application thread needs to select yes, and other places use this instruction to select no. Click "Add", "Apply".

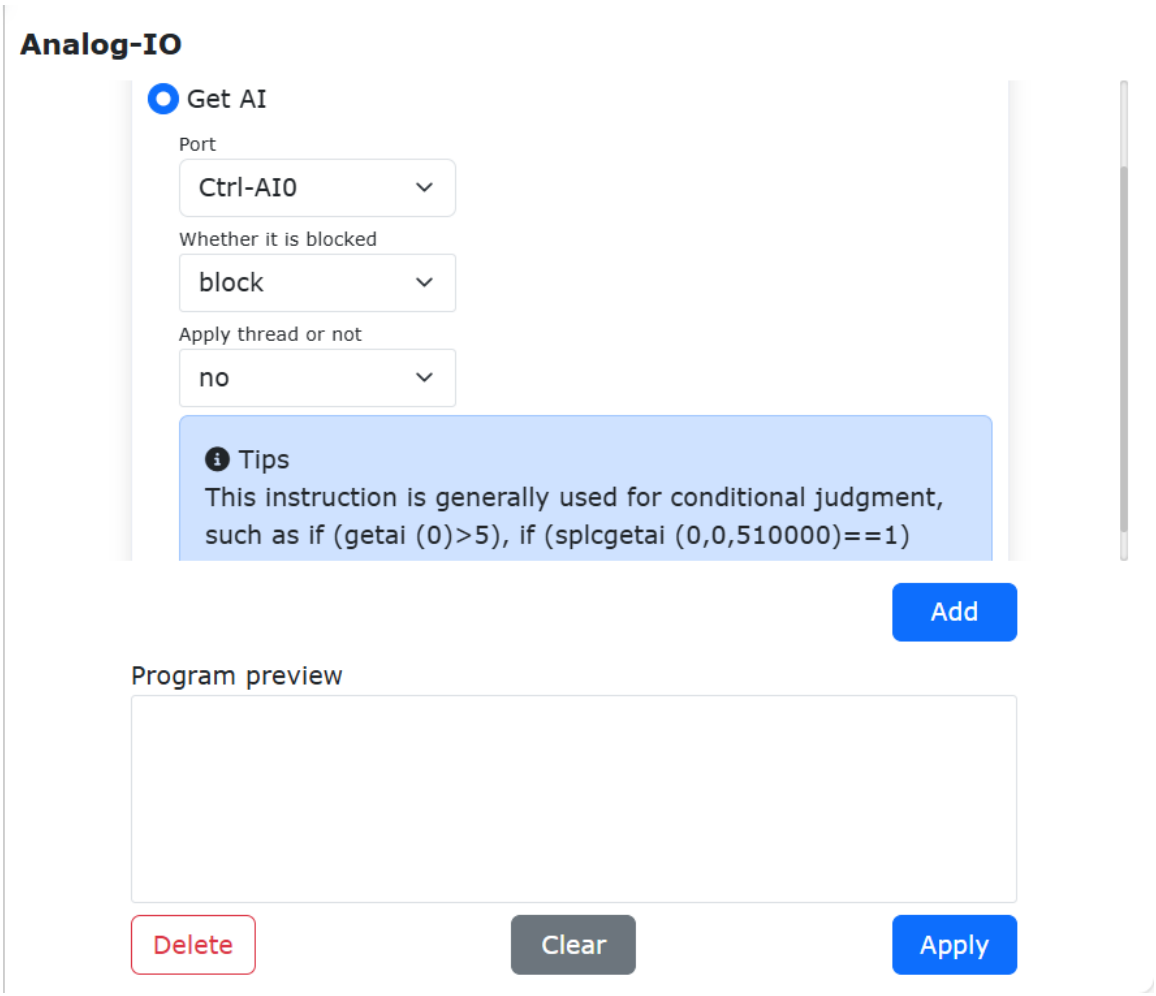


Figure 9.6-4 GetAI command interface

1.3.9.6.3 Vir-IO command

Click the “Vir-IO” icon to enter the Vir-IO command editing interface.

This command is a virtual IO control command, which can realize the setting of the simulated external DI and AI status, and obtain the simulated DI and AI status.

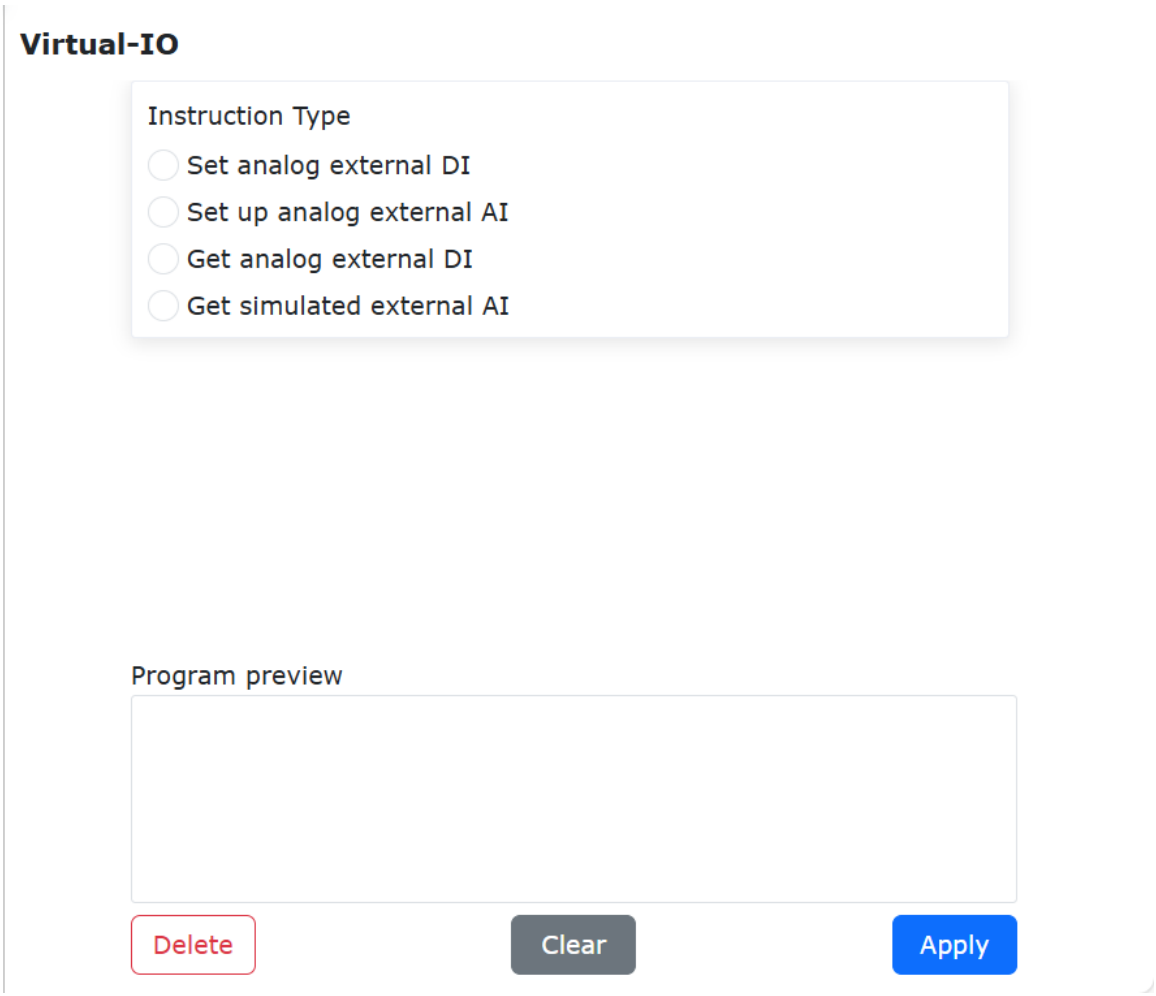


Figure 9.6-5 Vir-IO command interface

1.3.9.6.4 Aux-IO command

Click the “Aux-IO” icon to enter the Aux-IO command editing interface.

Aux-IO is the instruction function for the robot to communicate with the PLC to control the external expansion IO. It is necessary for the robot to establish UDP communication with the PLC. On the basis of the original 16-channel input and output, 128-channel input and output can be expanded. The usage of this command is the same as that mentioned above. IO usage is similar. There are certain technical difficulties in using this function, please contact us for consultation beforehand.

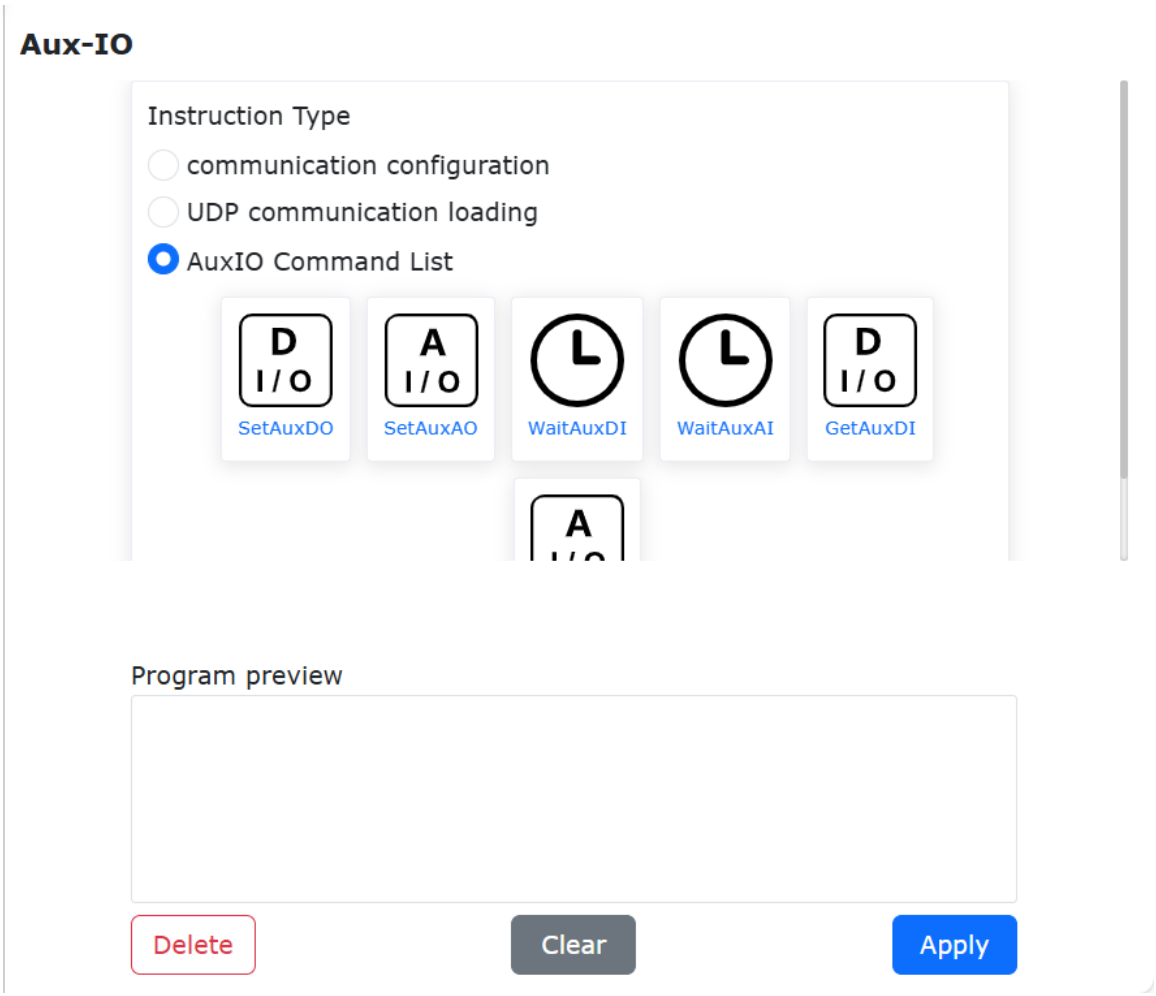


Figure 9.6-6 Aux-IO command interface

1.3.9.6.5 MoveDO command

Click the “MoveDO” icon to enter the MoveDO command editing interface.

This instruction is divided into continuous output mode and single output mode.

- Continuous output mode: During linear motion, the DO signal function is continuously output according to the set interval.

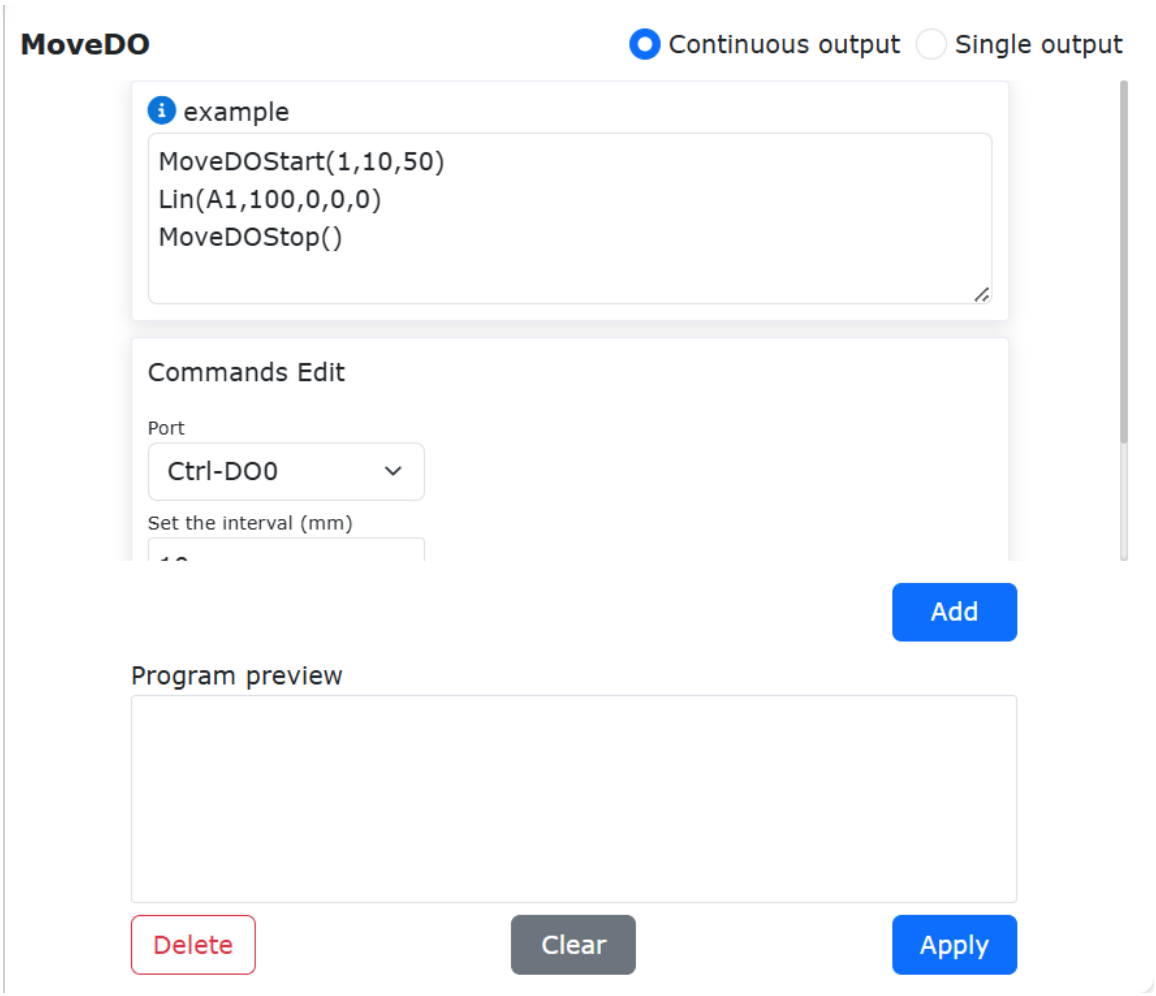


Figure 9.6-7 MoveDO command continuous output interface

- Single output mode: There are two options: constant speed segment output and free configuration. The setting time is output after the movement starts, and the reset time is output before the movement ends, ranging from [0, 1000].

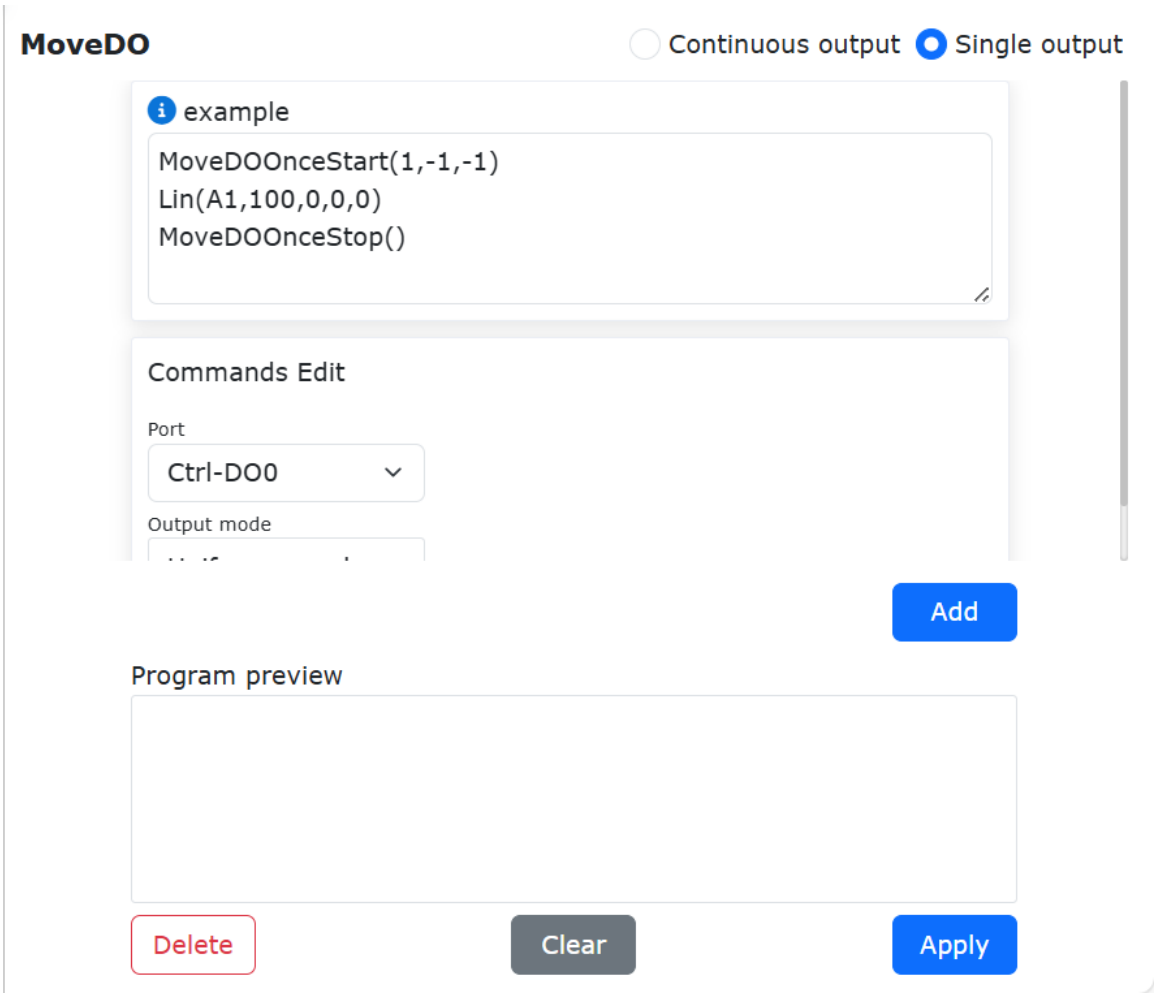


Figure 9.6-8 MoveDO instruction single output interface

1.3.9.6.6 MoveAO command

Click the “MoveAO” icon to enter the MoveAO command editing interface.

1. Summary

When used in conjunction with motion commands, this instruction can output AO signals proportionally based on real-time TCP speed during the motion process.

2. Description of Motion AO Command

The motion AO command is located in the teaching simulation program teaching command editing area, and the icon is Control Command-Motion AO.



Figure 9.6-9 Motion AO Instruction

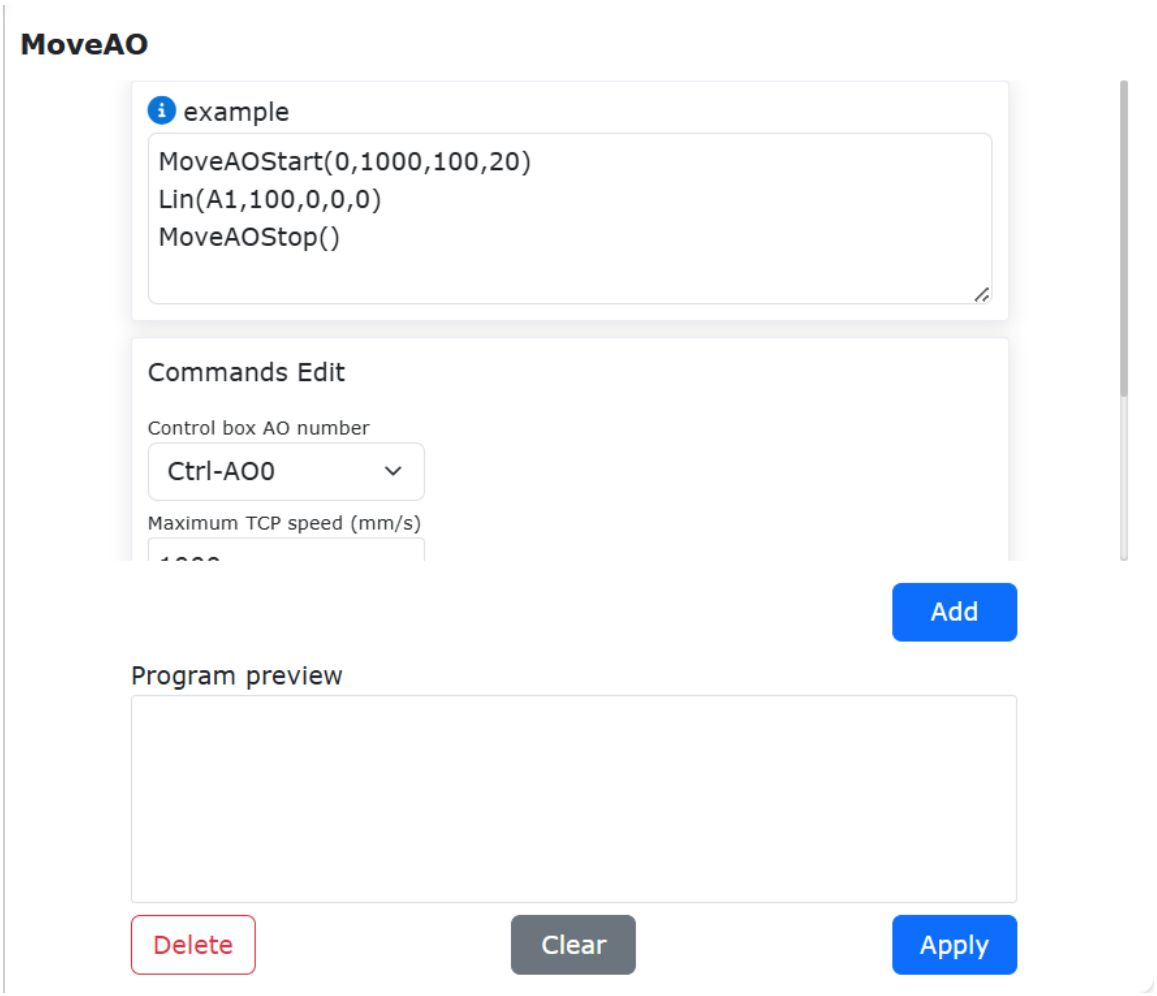


Figure 9.6-10 Details of Motion AO Instructions

- AO number: dropdown list selection, Ctrl-AO0 corresponds to control box AO0, Ctrl-AO1 corresponds to control box AO1, and End AO0 corresponds to end AO0.
- Maximum TCP speed: The maximum TCP speed value of the robot; Function: Proportional to real-time TCP speed.
- Maximum TCP speed AO percentage: The AO percentage corresponding to the maximum TCP speed value of the robot; Function: Set the upper limit value of AO output.
- Dead zone compensation value AO percentage: When there is a dead zone in the proportional valve, this parameter can be set to ensure AO output; Function: Set the lower limit value of AO output.

Important: Calculation formula:Output AO percentage=Real time TCP speed/Set maximum TCP speed * Set maximum TCP speed AO percentage.

The accompanying motion commands for this command are as follows:PTP/LIN/ARC/CIRCLE/SPLINE/NSPLINE/SERVOJ.

1.3.9.6.7 ToolList command

Click the “ToolList” icon to enter the ToolList command editing interface.

Select the tool coordinate system name and click “Apply” to add this command to the program. When the program runs this statement, the tool coordinate system of the robot will be set.

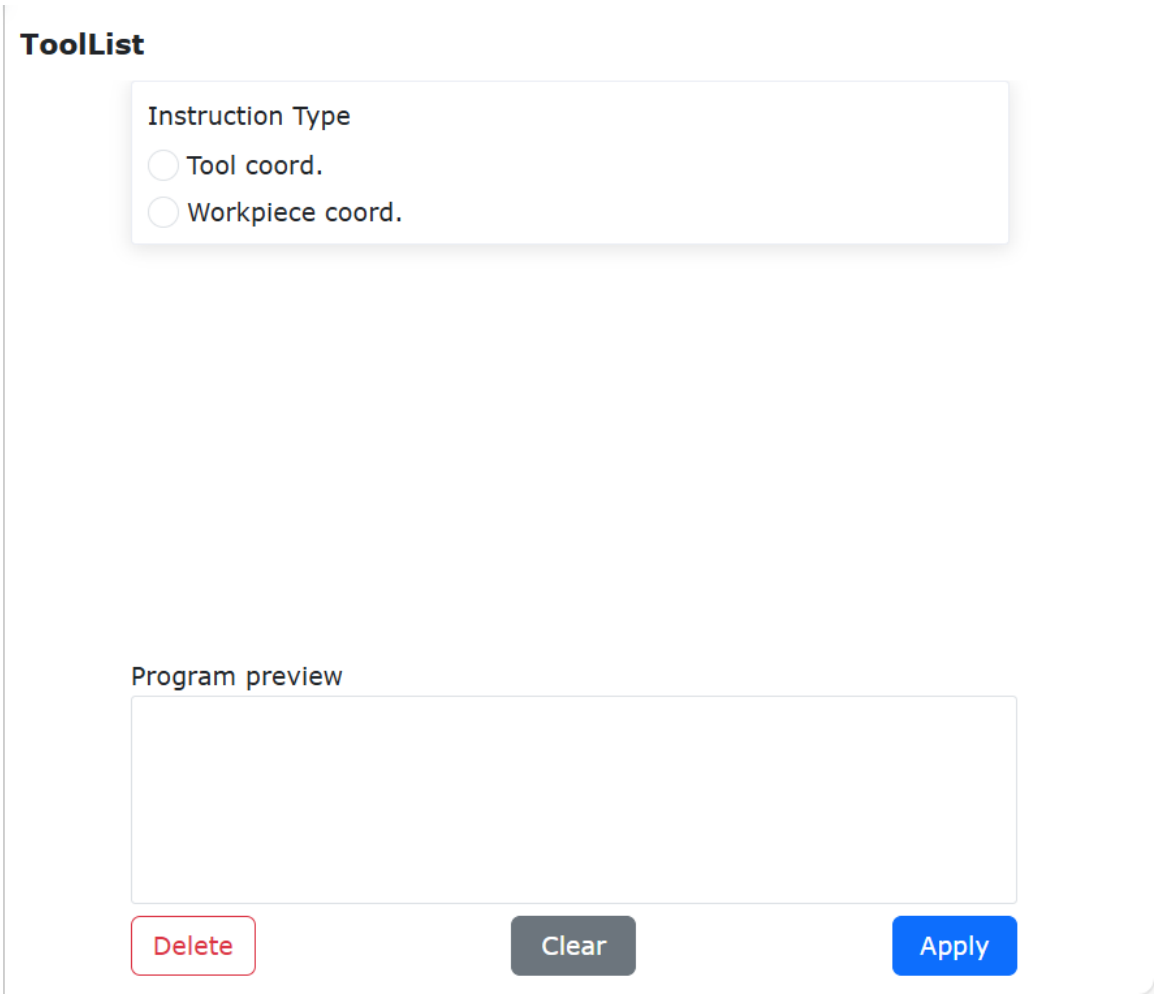


Figure 9.6-11 ToolList command interface

1.3.9.6.8 Mode command

Click the “Mode” icon to enter the Mode command editing interface.

This command can switch the robot to manual mode, and is usually added at the end of a program so that the user can automatically switch the robot to manual mode and drag the robot after the program runs.

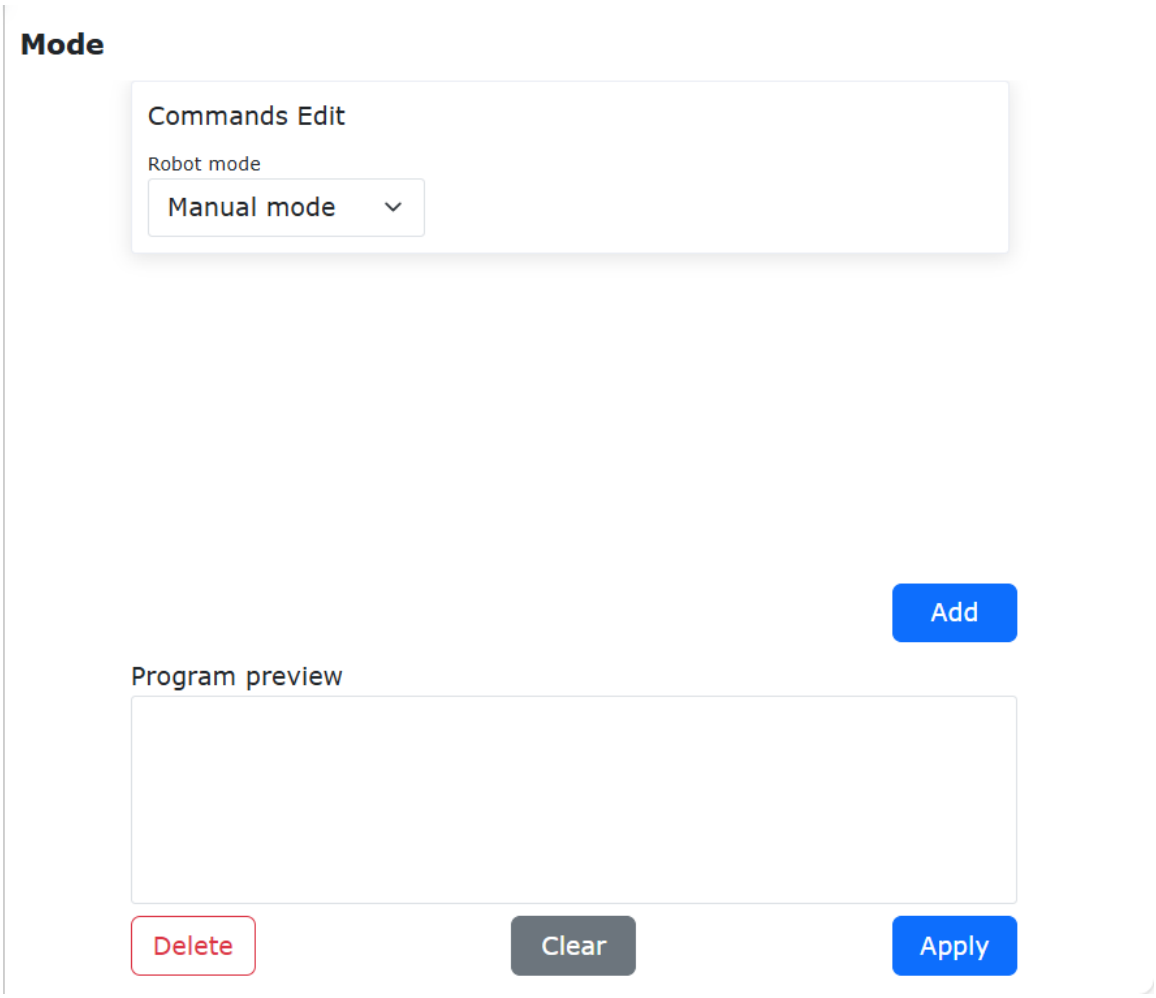


Figure 9.6-12 Mode command interface

1.3.9.6.9 Collision command

Click the “Collision” icon to enter the Collision command editing interface.

This command is used to set the collision level. Through this command, the collision level of each axis can be adjusted in real time during program operation, and the application scenario can be deployed more flexibly.

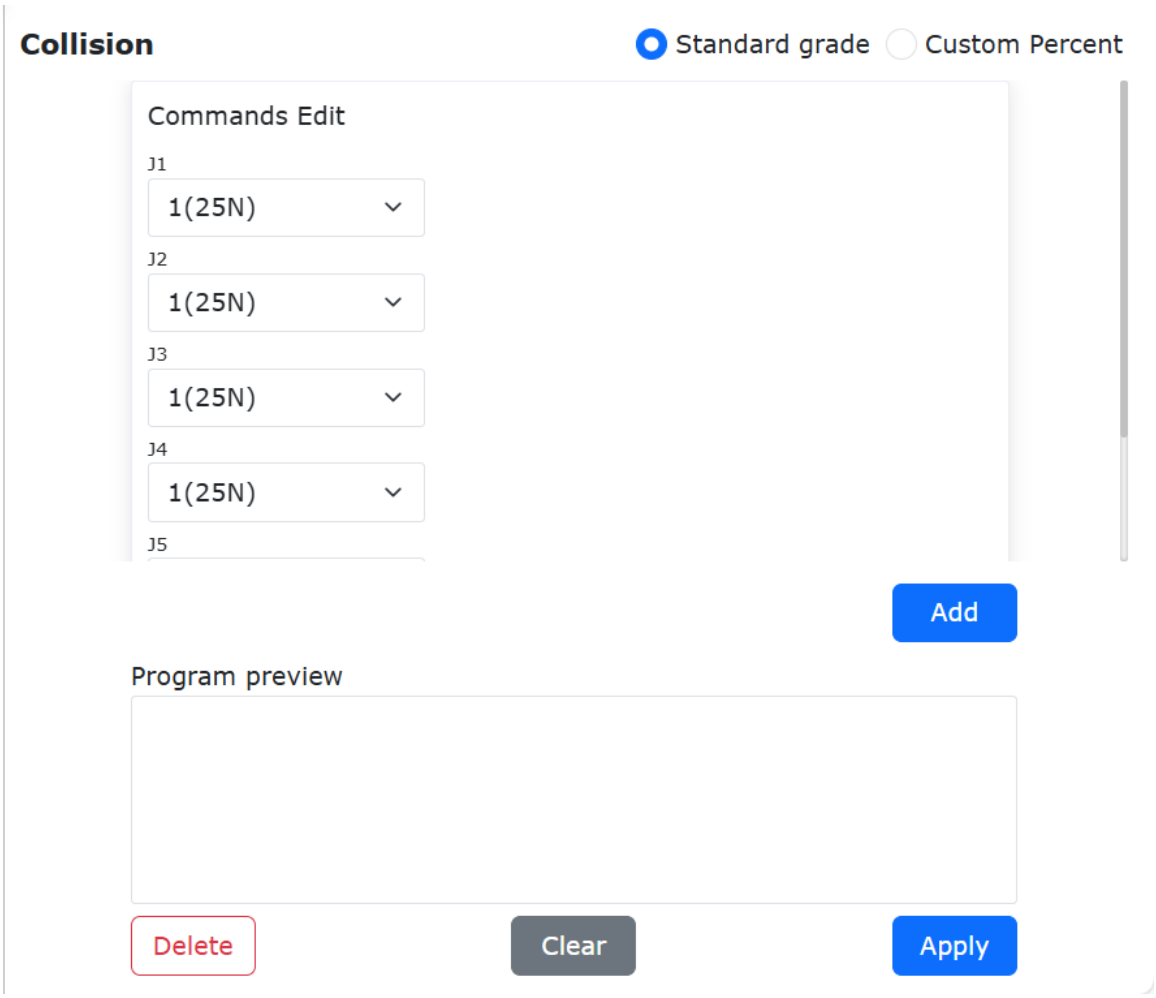


Figure 9.6-13 Collision command interface

1.3.9.6.10 Acc command

Click the “Acc” icon to enter the Acc command editing interface.

The Acc command is to realize the function that the acceleration of the robot can be set separately. By adjusting the acceleration scaling factor of the motion command, the acceleration and deceleration time can be increased or decreased, and the takt time of the robot action can be adjusted.

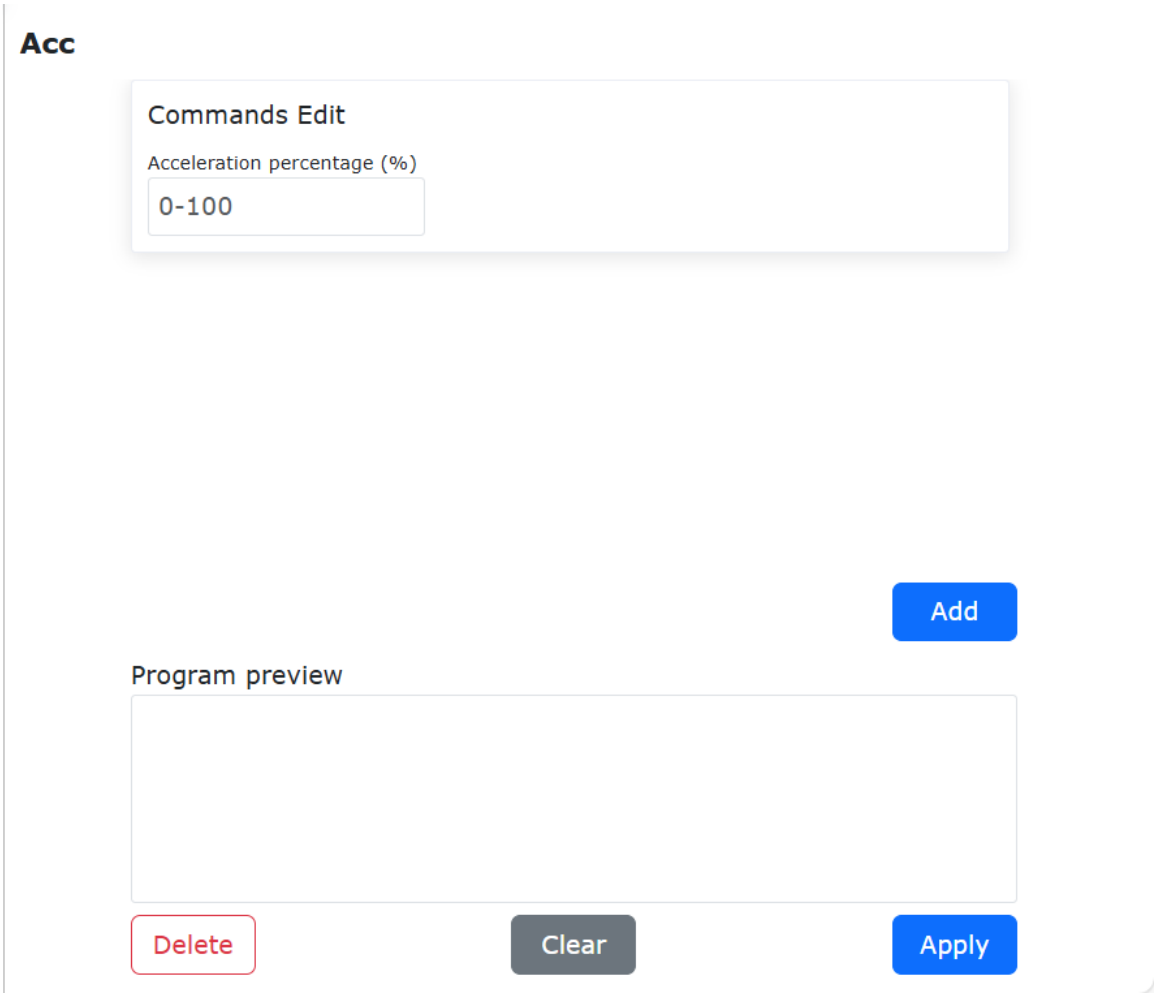


Figure 9.6-14 Acc command interface

1.3.9.7 Peripheral Command Interface



Figure 9.7 Peripheral Command Interface

1.3.9.7.1 Gripper command

Click the “Gripper” icon to enter the Gripper command editing interface.

In this command, it is divided into the gripper motion control command and the gripper activation/reset command. In the gripper control command, the number of the gripper that has been configured and activated is displayed. The user can edit through the edit box, or slide the slider to The required value is used to complete the setting of jaw opening and closing, opening and closing speed and opening and closing torque. Blocking means that the gripper moves in parallel with the previous movement command. Click the “Add” and “Apply” buttons to save the set value to the teaching file. The gripper reset/activation command displays the number of the grippers that have been configured, and the reset/activation command can be added to the program.

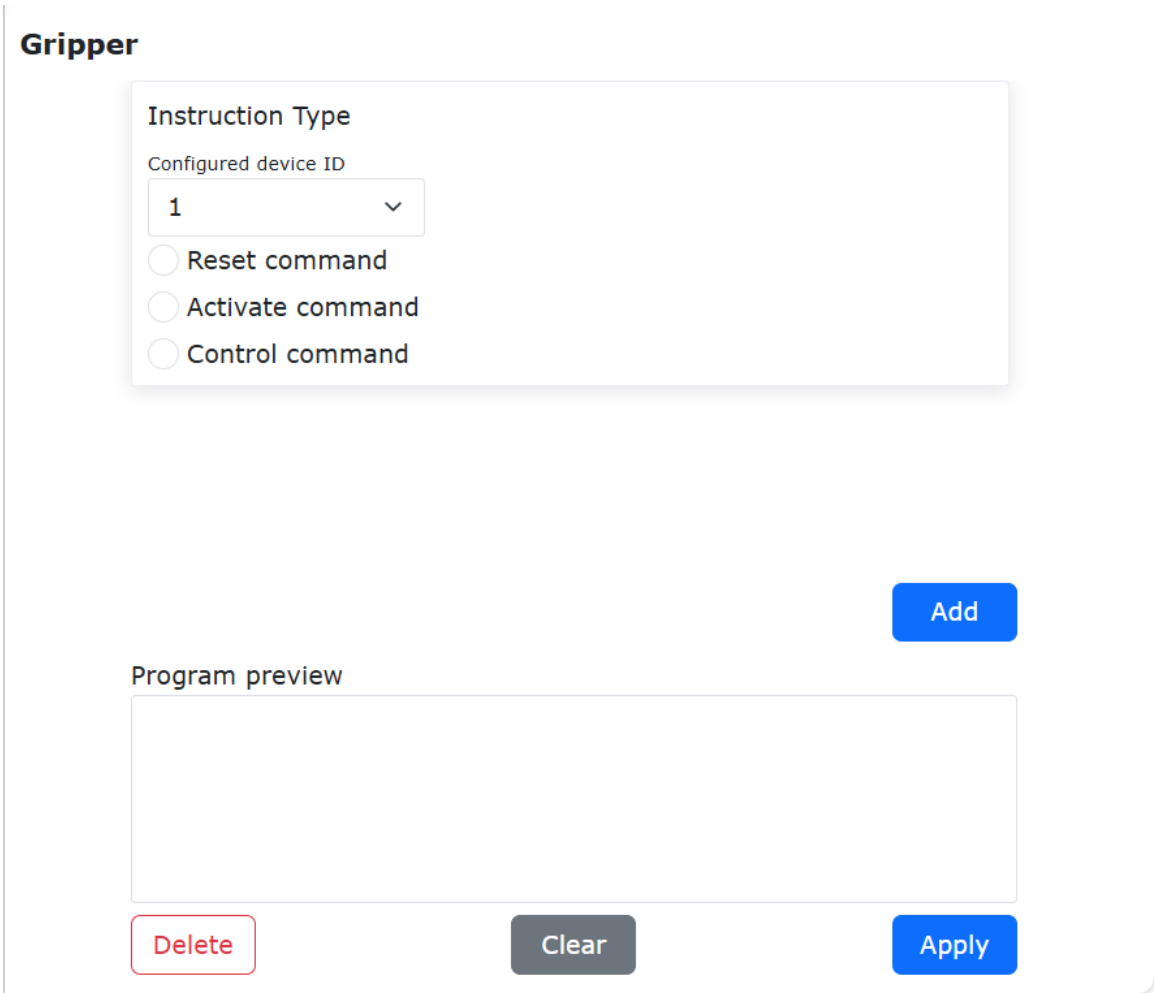


Figure 9.7-1 Gripper command interface

1.3.9.7.2 Spray command

Click the “Spray” icon to enter the Spray command editing interface.

This command is a spraying-related command, which controls the spray gun to “start spraying”, “stop spraying”, “start cleaning the gun” and “stop the light gun”. When editing the program command, it is necessary to confirm that the peripherals of the spray gun have been configured. For details, see the chapter on peripherals of the robot.

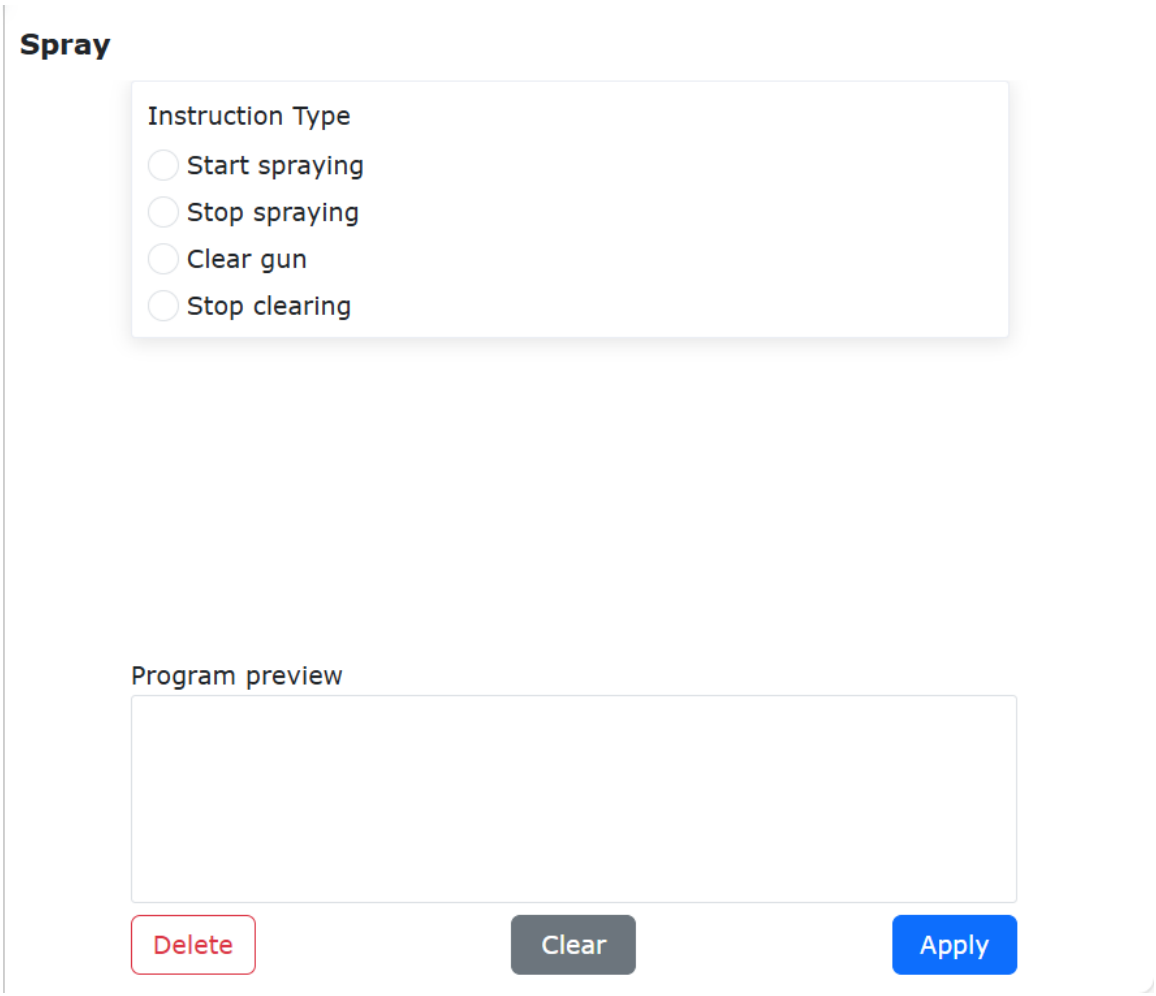


Figure 9.7-2 Spray command interface

1.3.9.7.3 EAxis command

Click the “EAxis” icon to enter the EAxis command editing interface., Select the combination mode:

- Controller + servo drive (485)
- Controller + PLC (UDP)

Select controller + PLC(UDP),this command is used in combination with the PTP command for scenarios using external axes, and can decompose the movement of a point in space in the X-axis direction to the movement of external axes. Select the external axis number, select synchronous motion mode, select the point to be reached, and click “Add” and “Apply” to save the command.

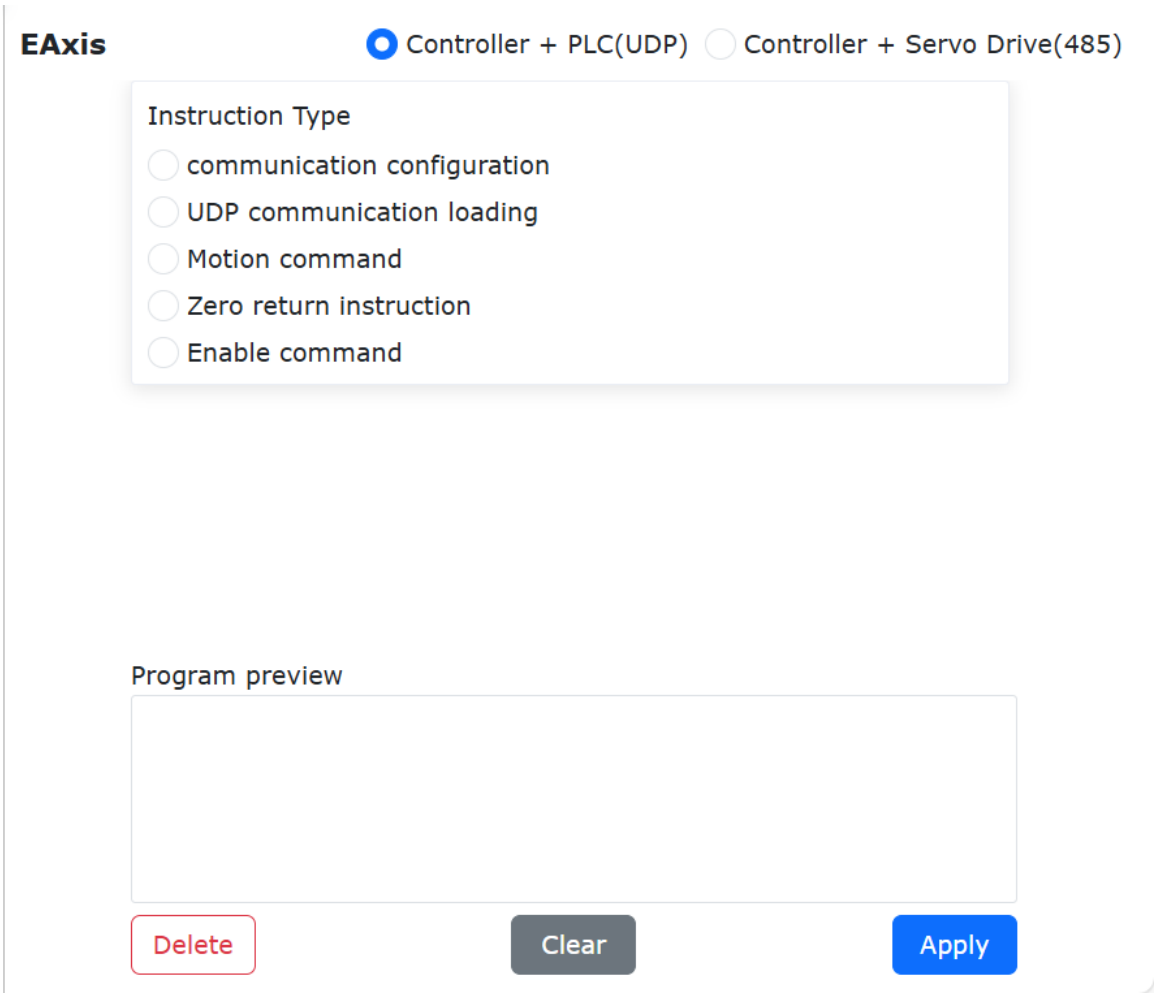


Figure 9.7-3 EAxis command interface

Select controller + servo drive(485),This command can configure the extended axis parameters. Set different parameters according to different control modes. The zero point of the configured expansion axis can be set.

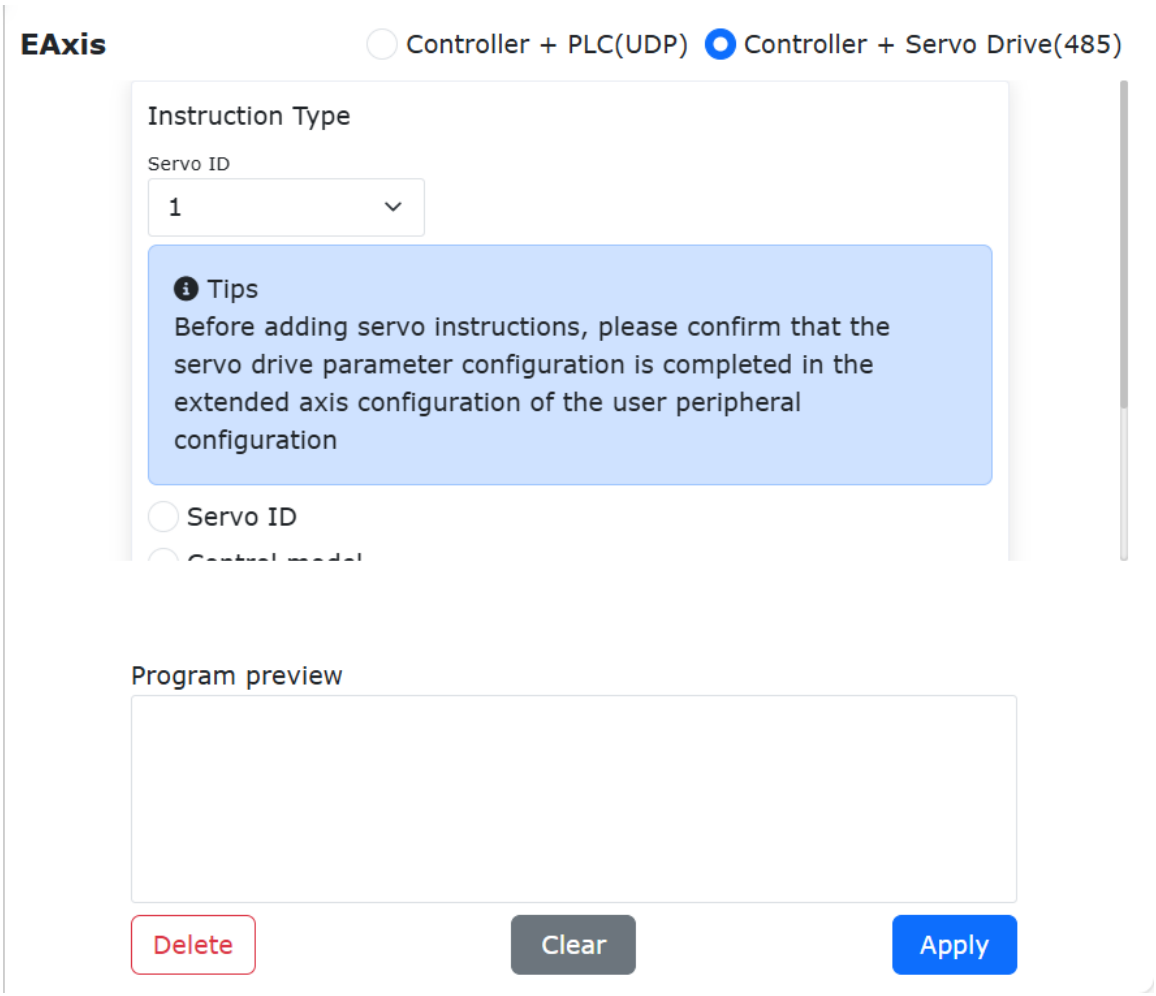


Figure 3.7-4 Servo command interface

1.3.9.7.4 Convey command

Click the “Convey” icon to enter the Convey command editing interface.

This command includes four commands: position real-time detection, IO real-time detection, tracking on and tracking off. See Robot Peripherals chapter for details.

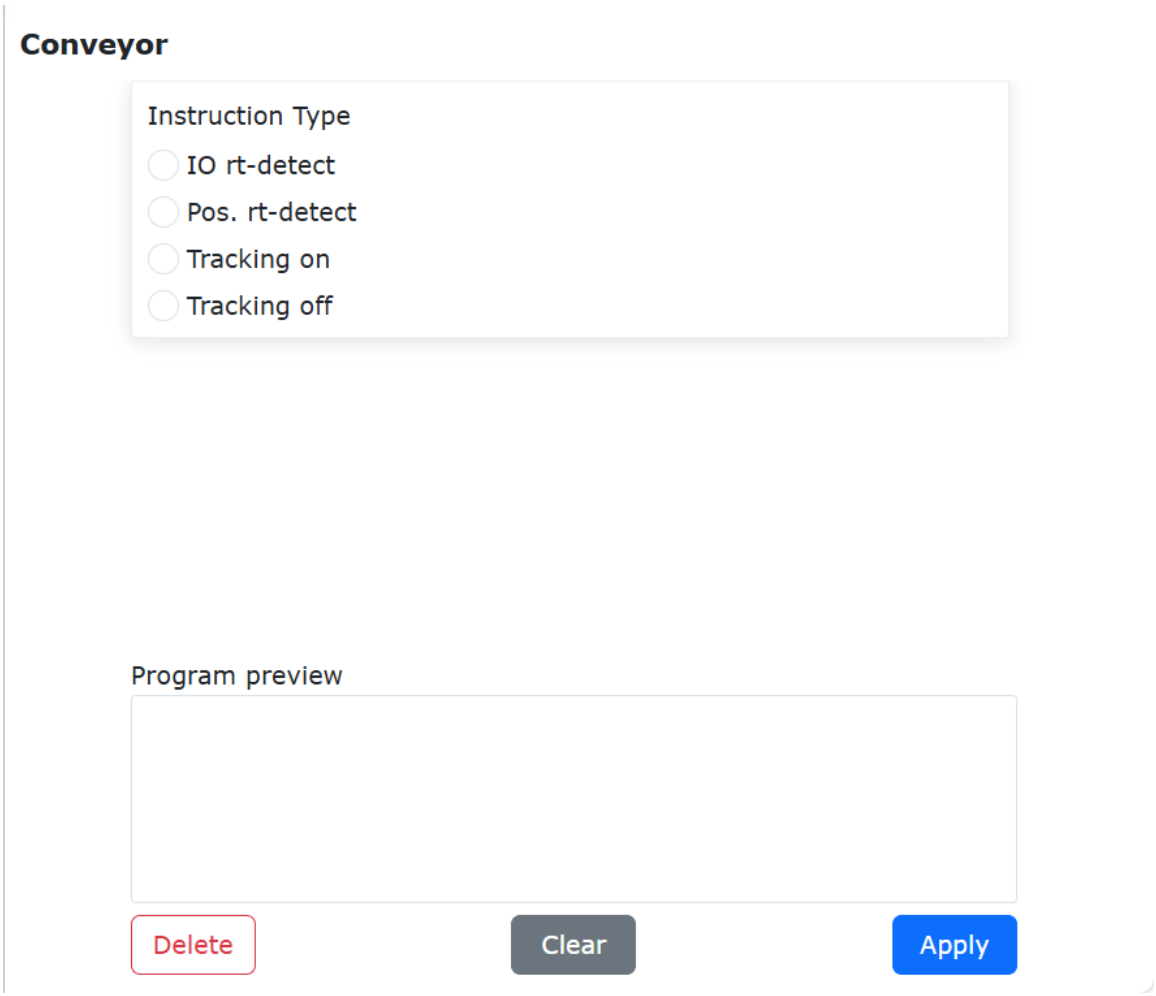


Figure 9.7-5 Conveyor command interface

1.3.9.7.5 Polish command

Click the “Polishing” icon to enter the Polish command editing interface.

This command can set the rotation speed, contact force, extension distance and control mode of the grinding equipment.

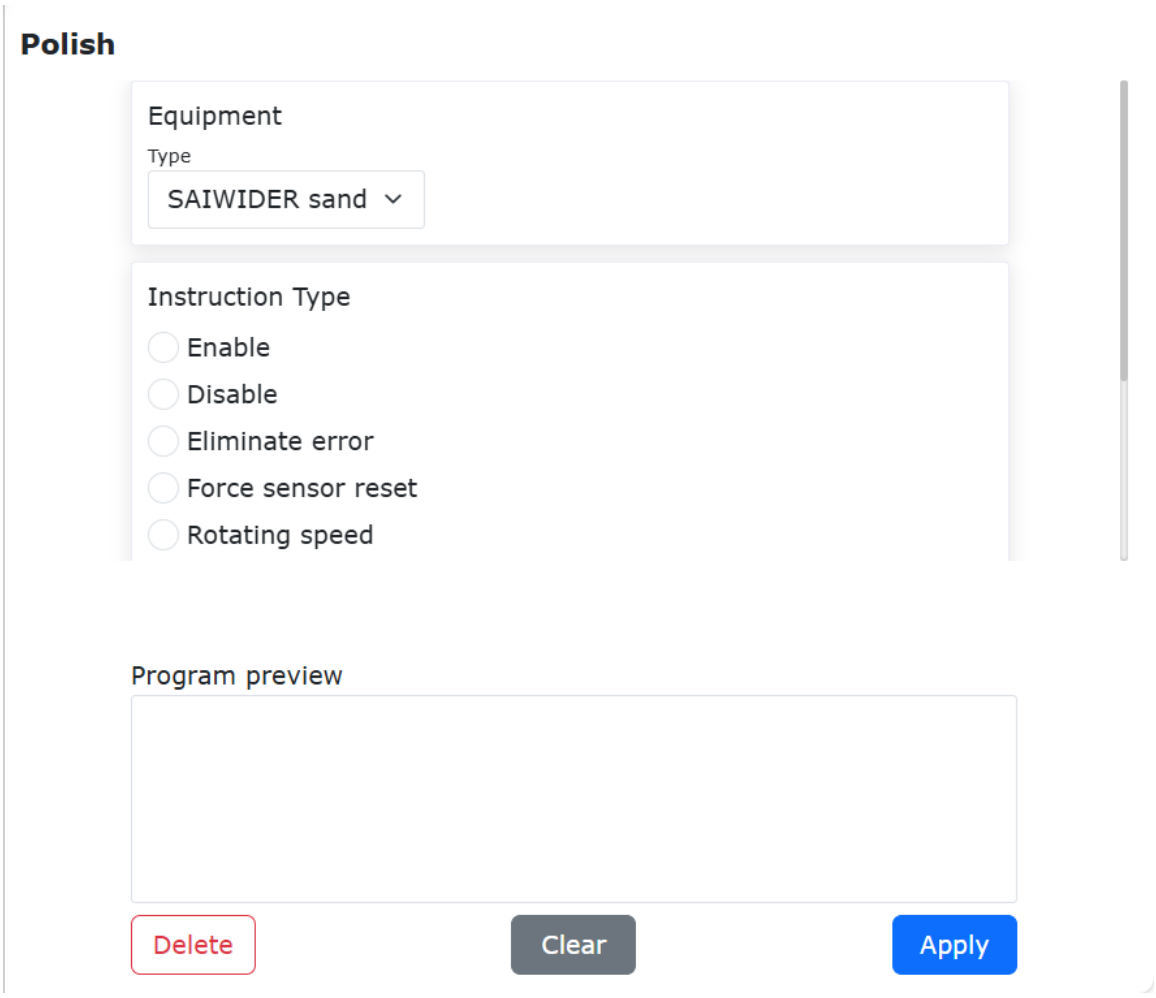


Figure 9.7-6 Polish command interface

1.3.9.8 Welding command interface



Figure 9.8 Welding command interface

1.3.9.8.1 Weld command

Click the “Weld” icon to enter the Weld command editing interface.

This command is mainly used for welding machine peripherals. Before adding this command, please confirm whether the welding machine configuration is completed in the user peripherals. For details, see the chapter on robot peripherals.

- welding voltage range: 0~700V
- welding current range: 0~1000A

Important: When configuring the output AO, welding current, and welding voltage, you need to select the I/O type. If you select controller I/O, you need to select the corresponding output AO.

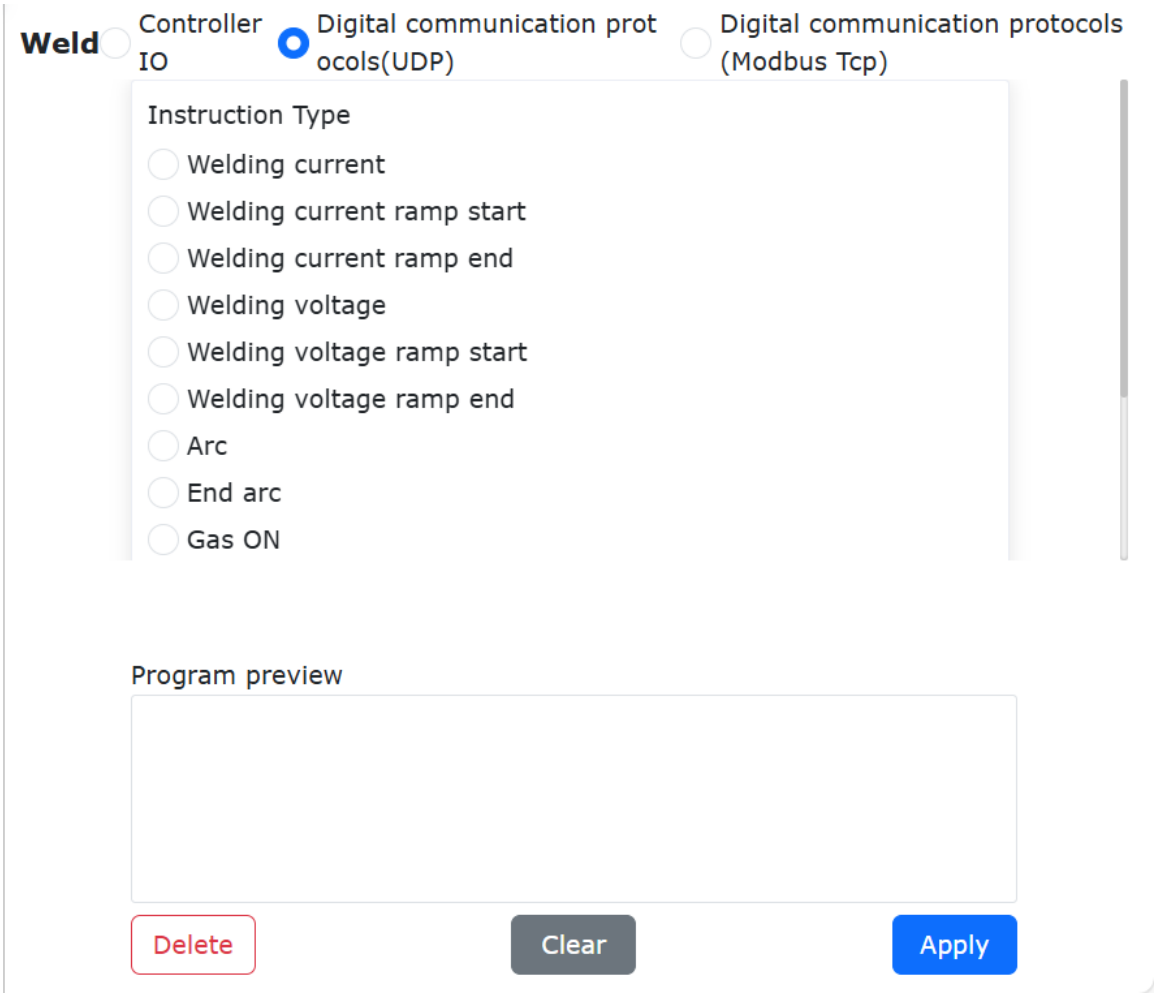


Figure 9.8-1 Weld command interface

1.3.9.8.2 Segment command

Click the “Segment” icon to enter the Segment command editing interface.

The collaborative robot can perform segment welding operations by adding segment welding instructions. Before adding segment welding instructions, you need to select the segment welding mode and teach the starting point and end point. The segment welding mode is divided into unchanged posture and changing posture. The robot considers whether to change the posture during the welding trajectory according to the selected segment welding mode.

Teach the starting point “segment01” and the end point “segment02”, and confirm the starting point and end point of the welding trajectory, as shown in the figure below.

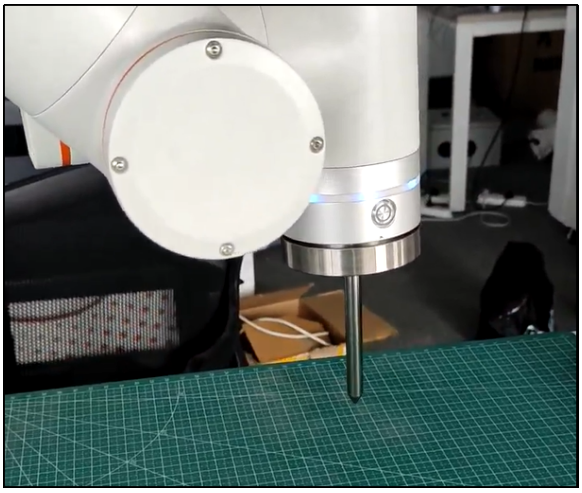


Figure 9.8-2-1 Starting point “segment01”

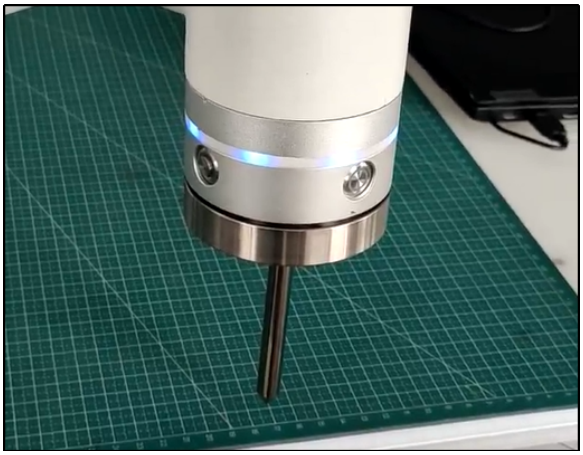


Figure 9.8-2-2 Starting point “segment02”

1.3.9.8.2.1 Segment welding command added

Step1: Create a new user program "testSegment1.lua", click the "Segment Welding" button, and open the segment welding instruction adding page.

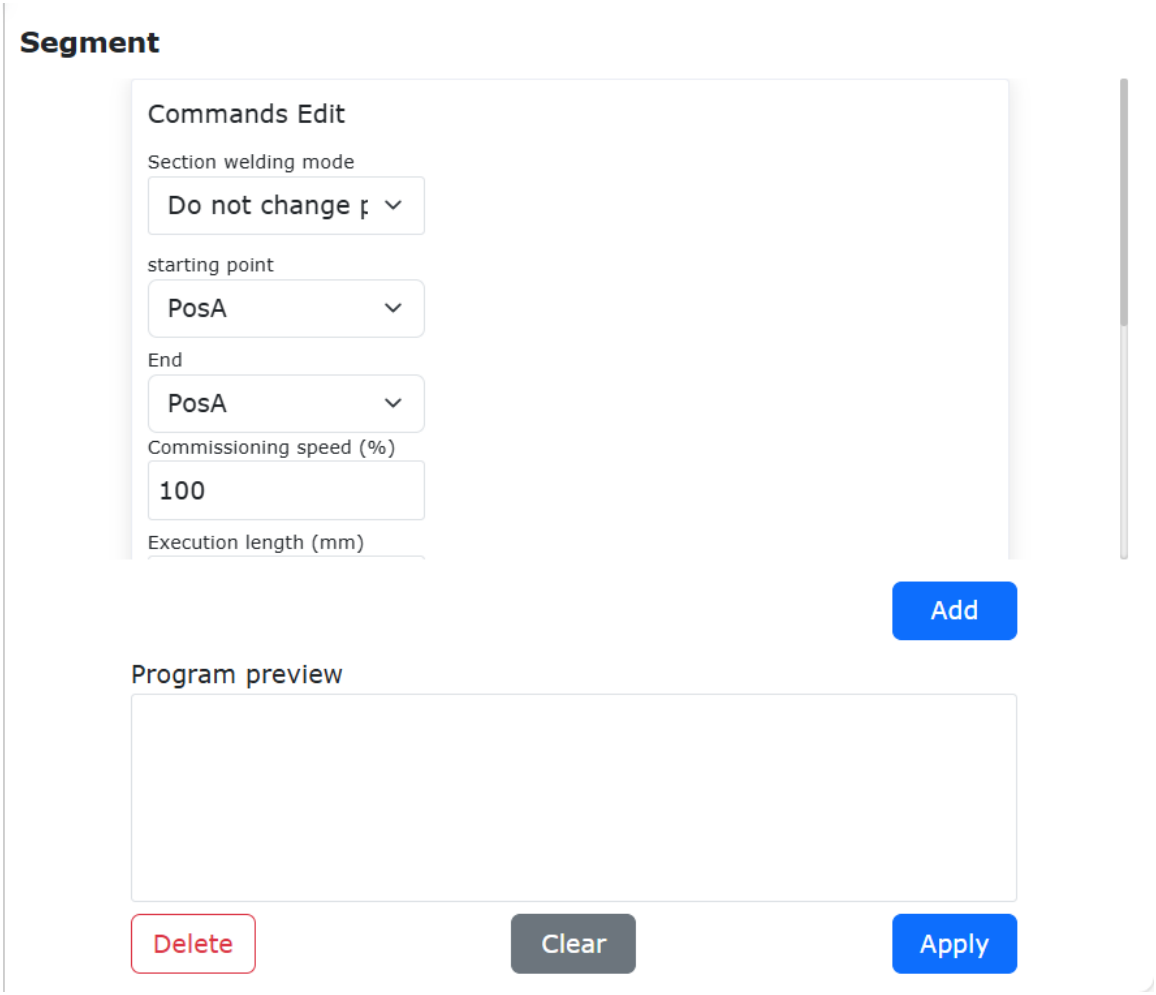


Figure 9.8-2-3 Add segment welding command button

Step2: On the segment welding instruction adding page, select "segment01" as the "start point" and "segment02" as the "end point".

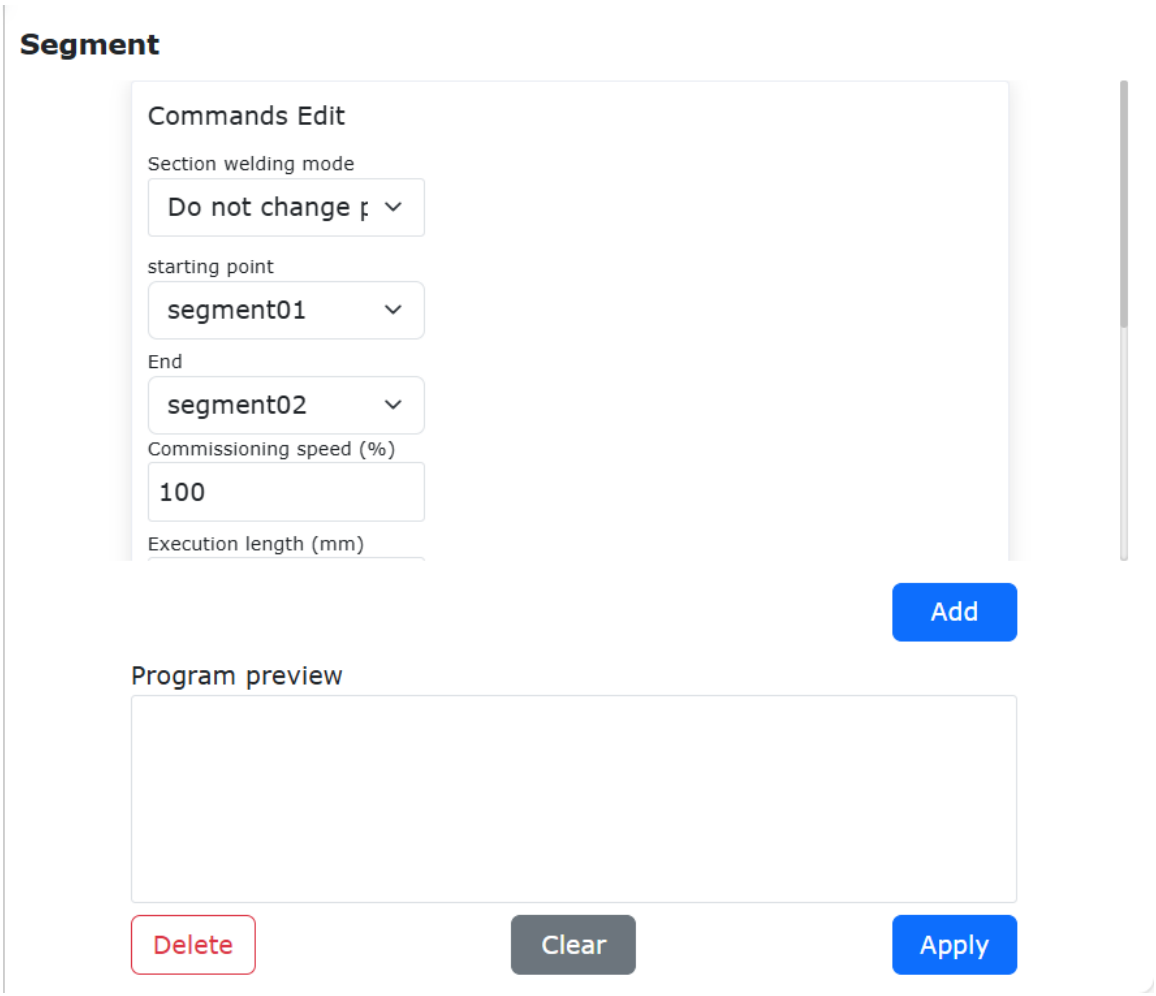


Figure 9.8-2-4 Starting point and end point of segment welding

Step3: Configure the debugging speed, execution length, non-execution length, functional mode, swing selection and rounding rules, and click the “Add” button and the “Apply” button in sequence.

Step4: At this time, “testSegment1.lua” has added segment welding motion instructions, as shown in Figure5.



Figure 9.8-2-5 Addition of segment welding motion instructions

1.3.9.8.2.2 Changes in segment welding motion trajectory and attitude

The segment welding mode of the collaborative robot can be selected for the segment welding movement. The mode types include the following two types;

No change in posture:The robot always maintains the posture of the starting point of the welding trajectory during the welding trajectory.

Changing posture:During the welding trajectory process, the robot calculates the Cartesian pose and joint position of each segment of the trajectory, and changes its posture during the segment welding operation.

The following demonstrates the usage of “no change posture” and “change posture” respectively.

1. Do not change posture

Open the segment welding instruction adding page, select “No change in attitude” for “segment welding mode”, also select “start point” as “segment01”, “end point” as “segment02”, set the execution length to 100, non-execution Set the length to 50, select other relevant configurations and save the program.

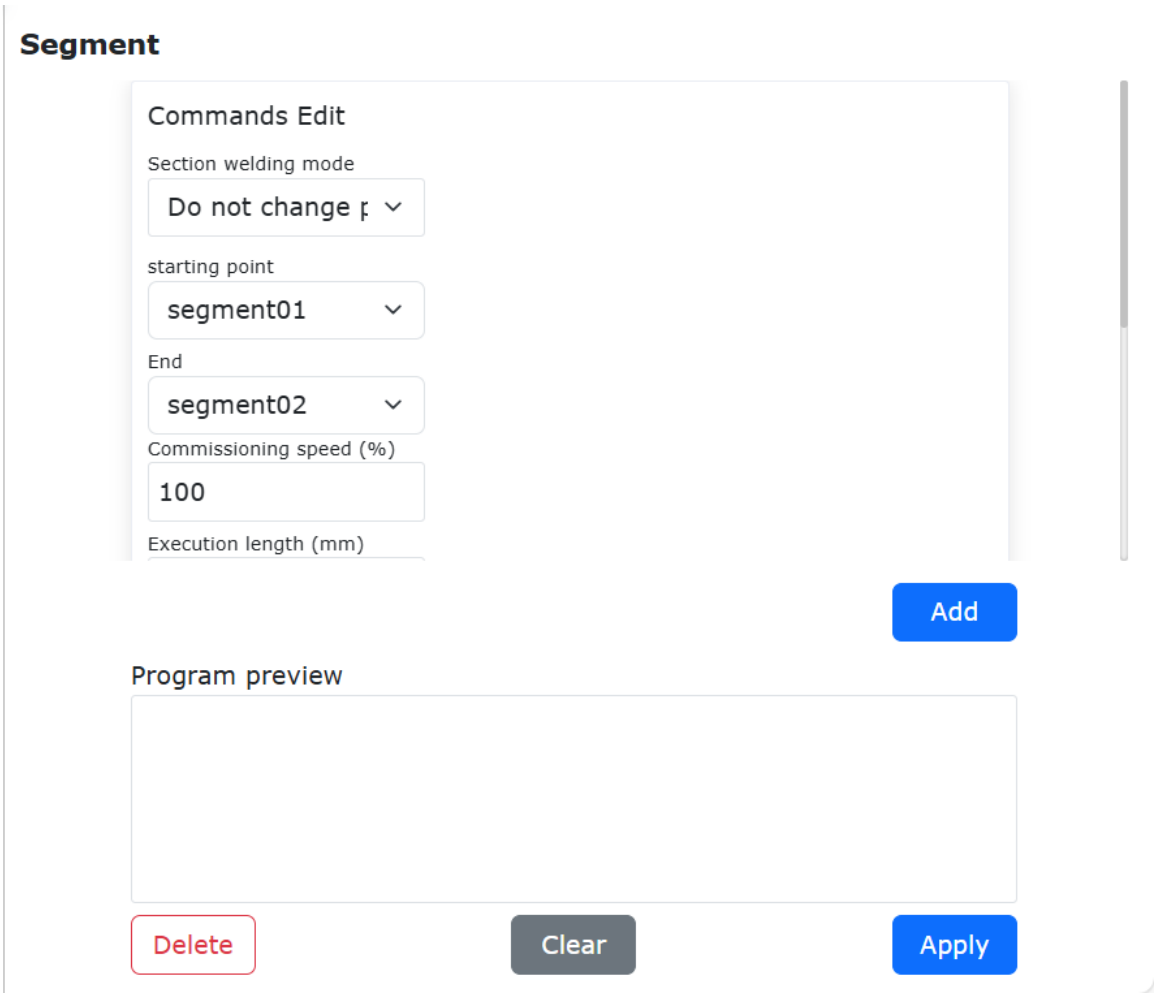


Figure 9.8-2-6 Segment welding mode without changing attitude

2. Change posture

Open the segment welding instruction adding page, select “Change Attitude” for “Segment Welding Mode”, also select “Start Point” as “segment01”, “End Point” as “segment02”, set the execution length to 100, and the non-execution length. Set it to 50, select other relevant configurations and save the program.

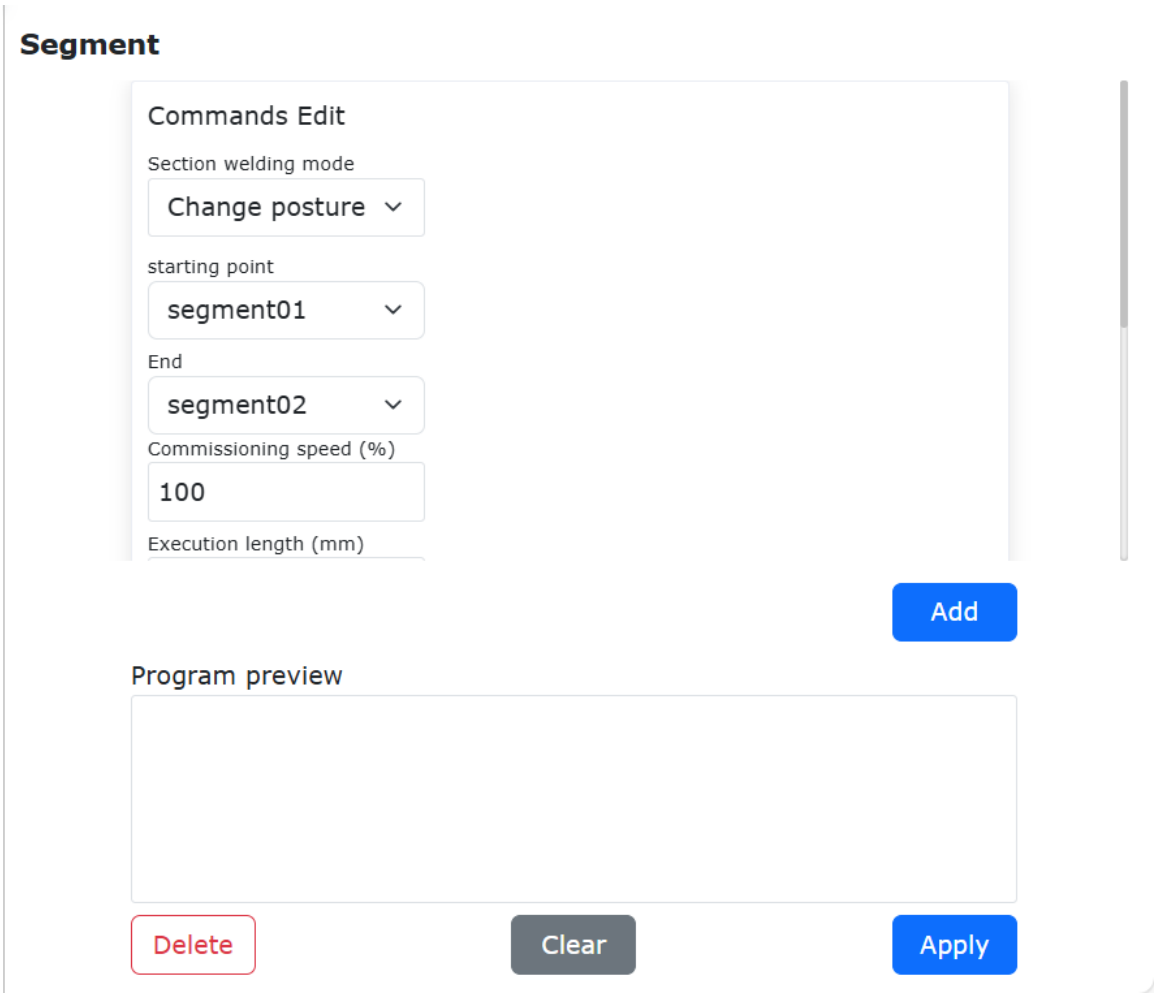


Figure 9.8-2-7 Changing attitude segment welding mode

3. Section welding operation type

Running program, robot segment welding operation conditions are divided into the following types:

1) If the function mode selects the first segment to execute the function, and the swing selects the execution segment to swing, the rounding rule will not round. Then the robot performs swing motion at 100mm and linear motion at 50mm alternately, and stops when it reaches the end point;

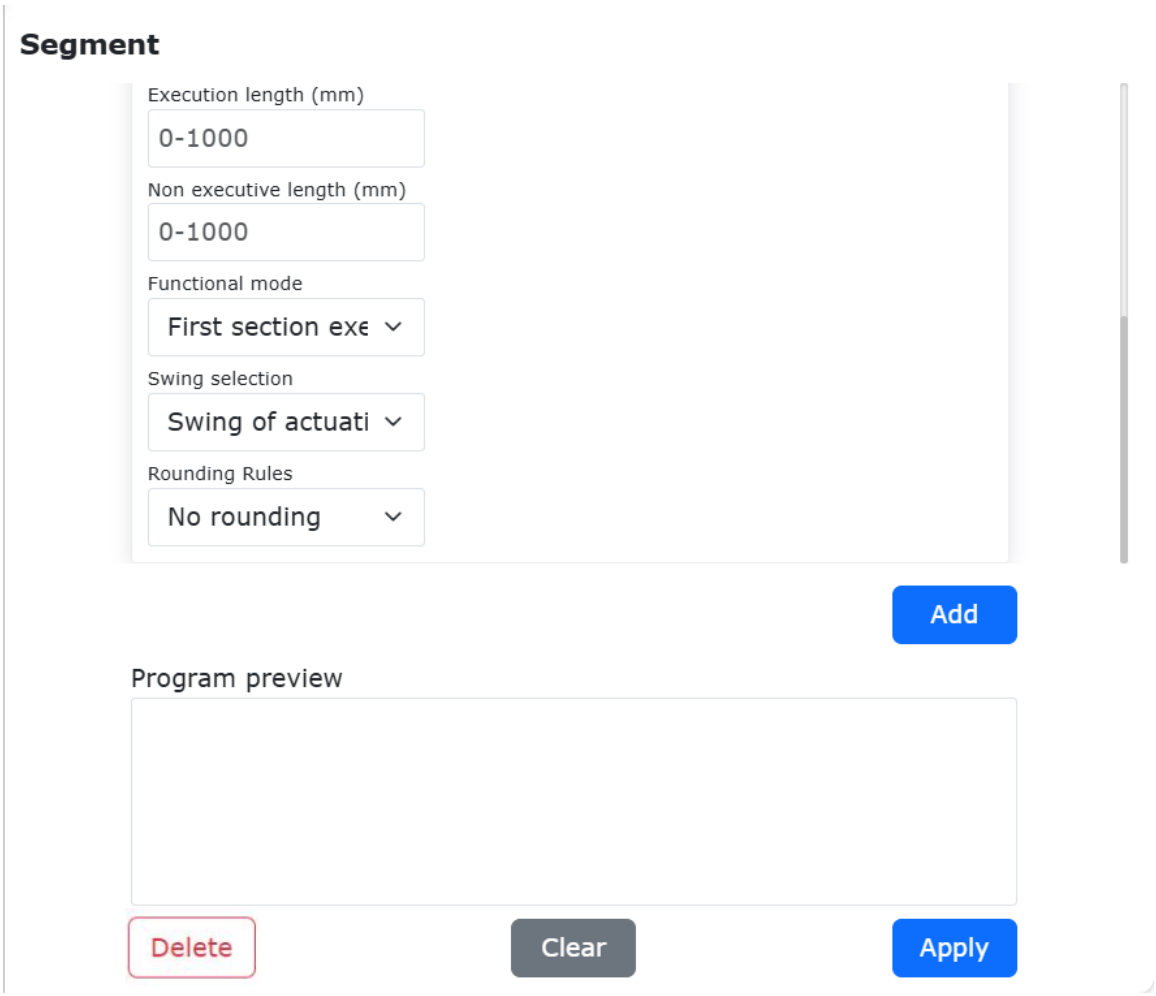


Figure 9.8-2-8 The first section executes the swing function without rounding

2) If the function mode selects the first segment to not execute the function, the swing selection does not execute the segment swing, and the rounding rules do not round. Then the robot performs swing motion for 50mm and linear motion for 100mm alternately, and stops when it reaches the end point;

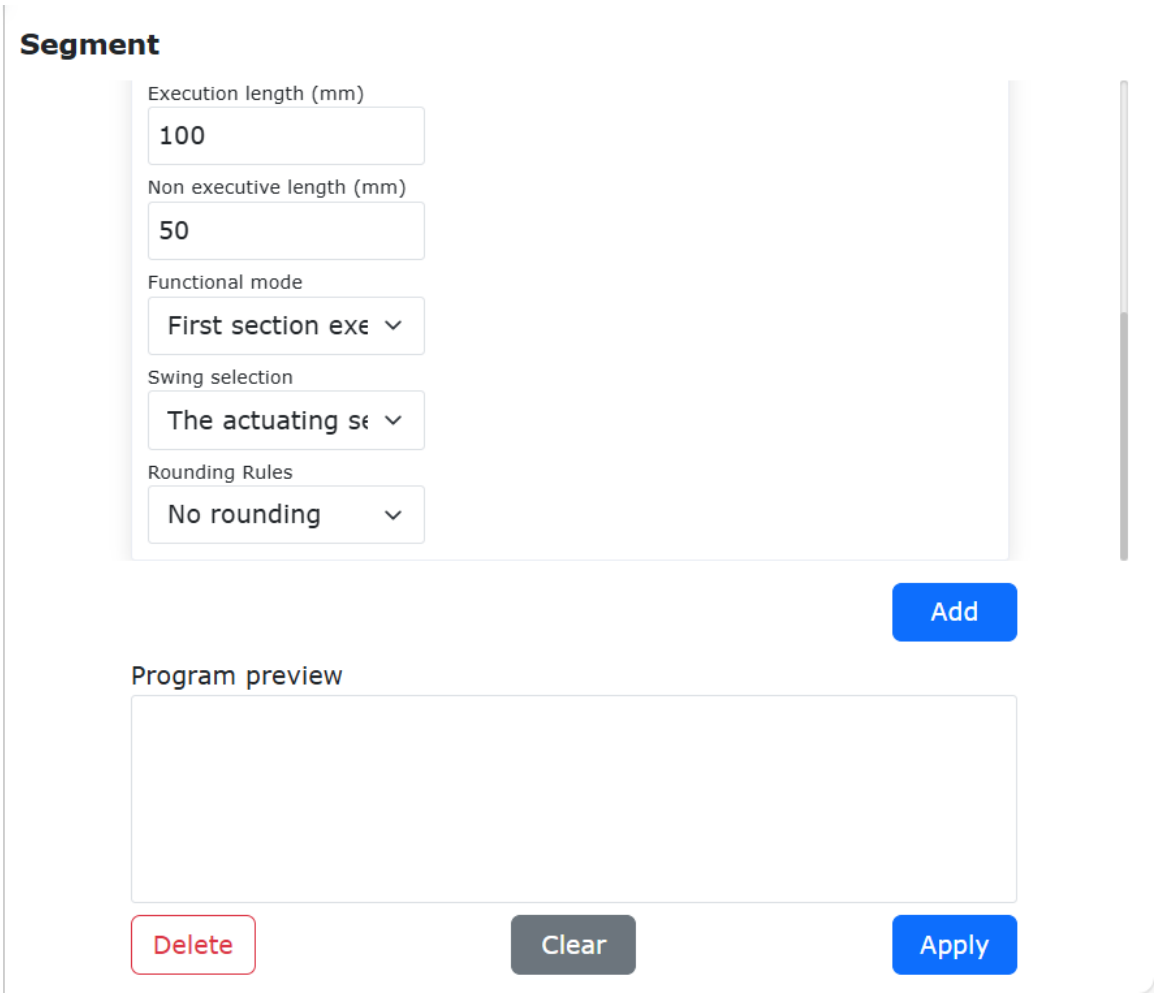


Figure 9.8-2-9 The first section does not execute the swing function and does not round

3) If the function mode selects the first segment to perform the function, the swing selects the execution segment to swing, and the rounding rules are rounded. Then the robot performs swing motion at 100mm and linear motion at 50mm alternately. After the last period of the overall cycle, if the remaining distance is less than 150mm, it will stop swinging;

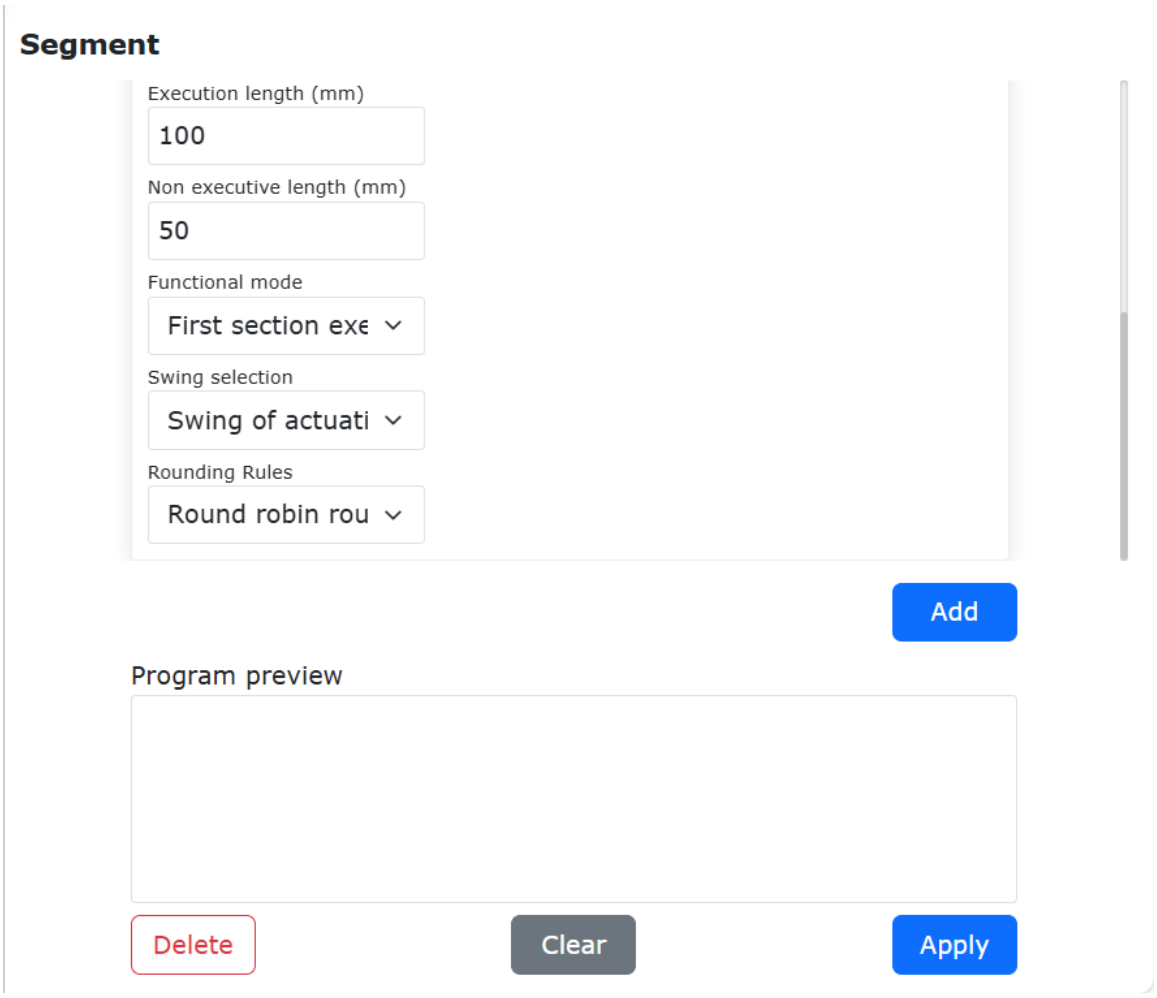


Figure 9.8-2-10 The first section performs circular rounding of the swing function

4) If the function mode selects the first segment to perform the function, and the swing selects not to execute the segment swing, the rounding rules are rounded. Then the robot performs swing motion at 50mm and linear motion at 100mm alternately. After the last period of the overall cycle, if the remaining distance is less than 150mm, it will stop swinging;

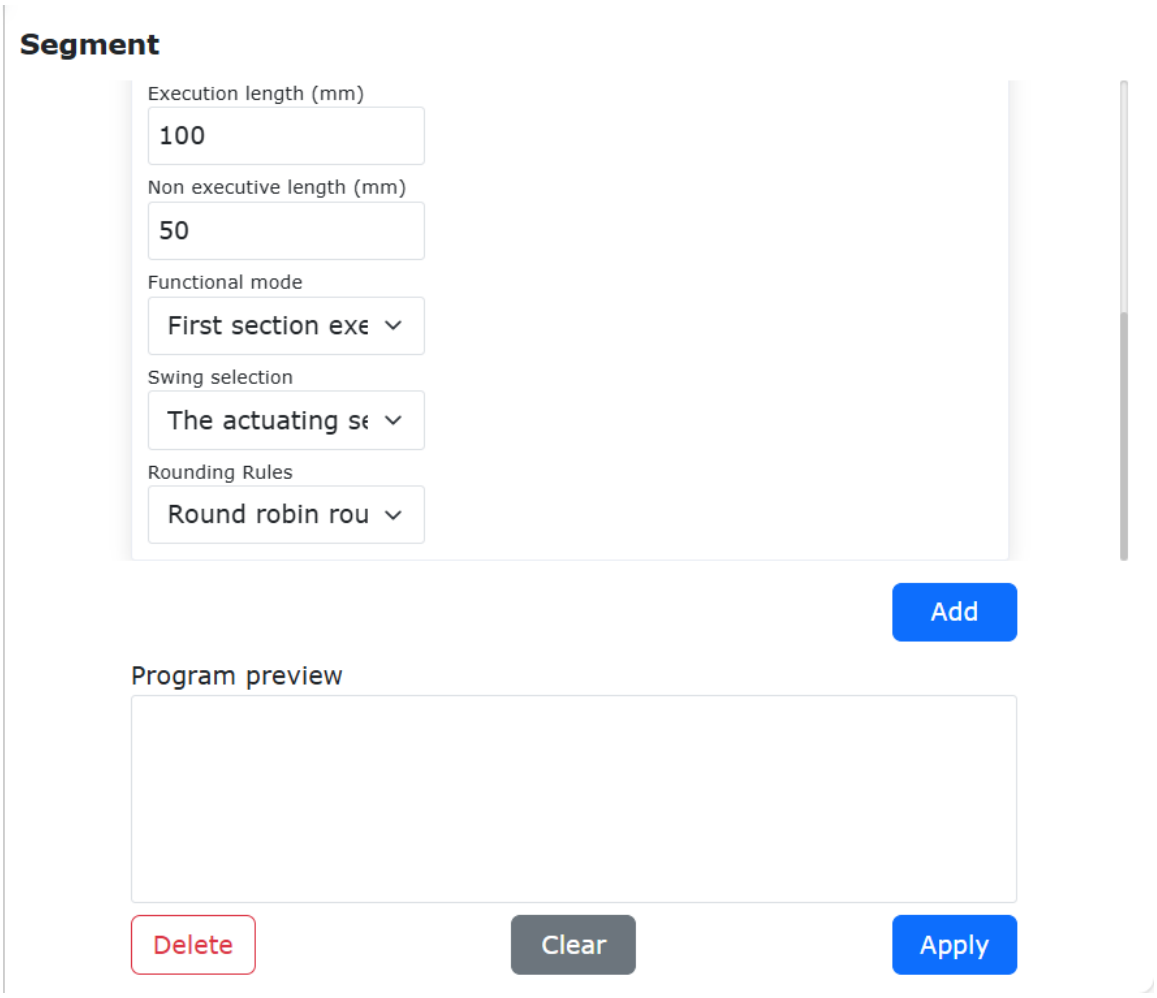


Figure 9.8-2-11 The first section does not perform circular rounding of the swing function

5) If the function mode selects the first segment to execute the function, the swing selects the execution segment to swing, and the rounding rule is single segment rounding. Then the robot performs swing motion at 100mm and linear motion at 50mm alternately. After the last cycle, if the next segment is 100mm, swing planning is performed and the remaining distance is less than 100mm, the swing will stop; if the next segment is 50mm, linear motion planning is performed and the remaining distance is If it is less than 50mm, the movement will stop;

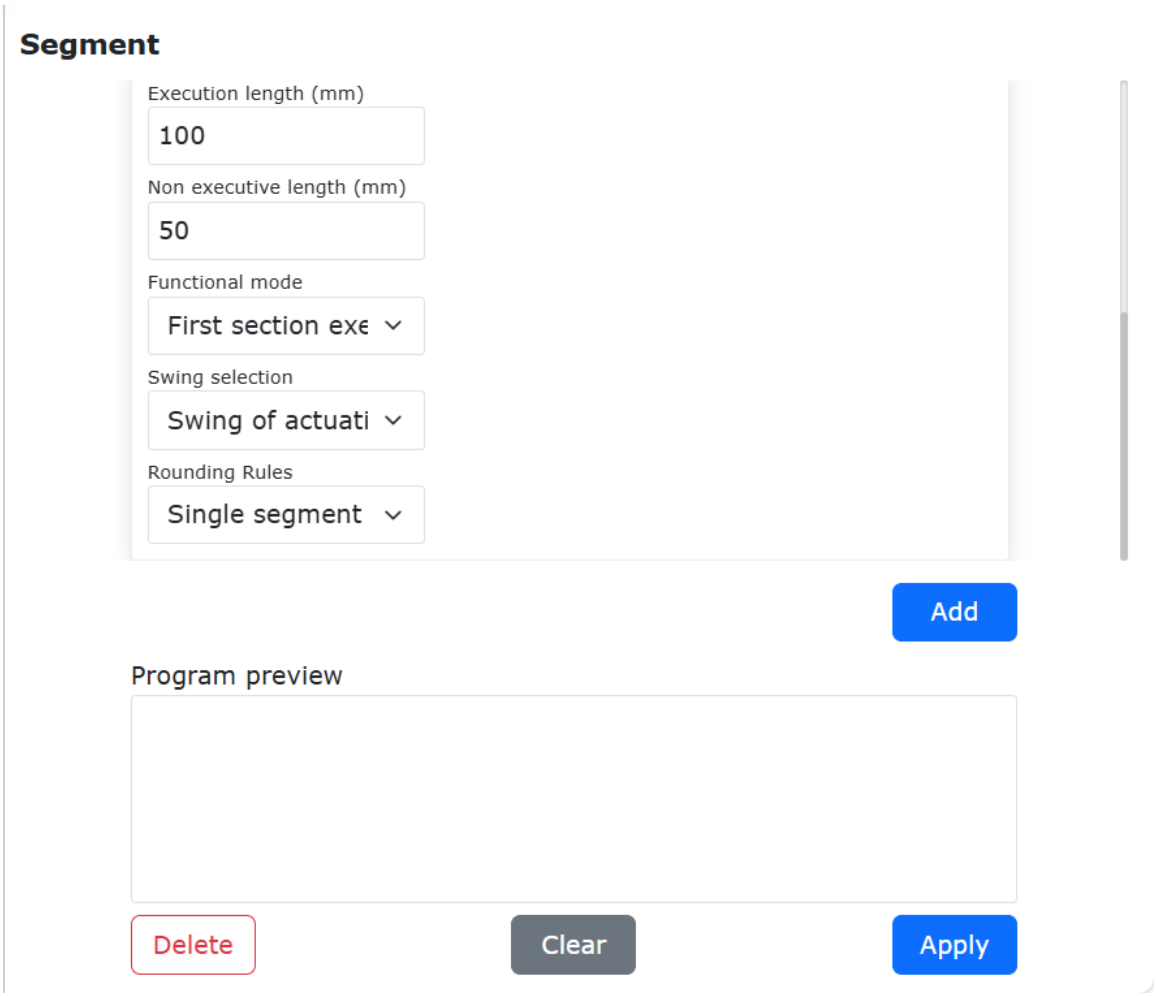


Figure 9.8-2-12 The first section performs single-section rounding of the swing function

6) If the function mode selects the first segment to perform the function, the swing selects not to execute segment swing, and the rounding rule is single segment rounding. Then the robot performs swing motion at 50mm and linear motion at 100mm alternately. After the last cycle, if the next segment is 50mm, swing planning is performed and the remaining distance is less than 50mm, then the swing is stopped; if the next segment is 100mm, linear motion planning is performed and the remaining distance is Less than 100mm, the movement stops.

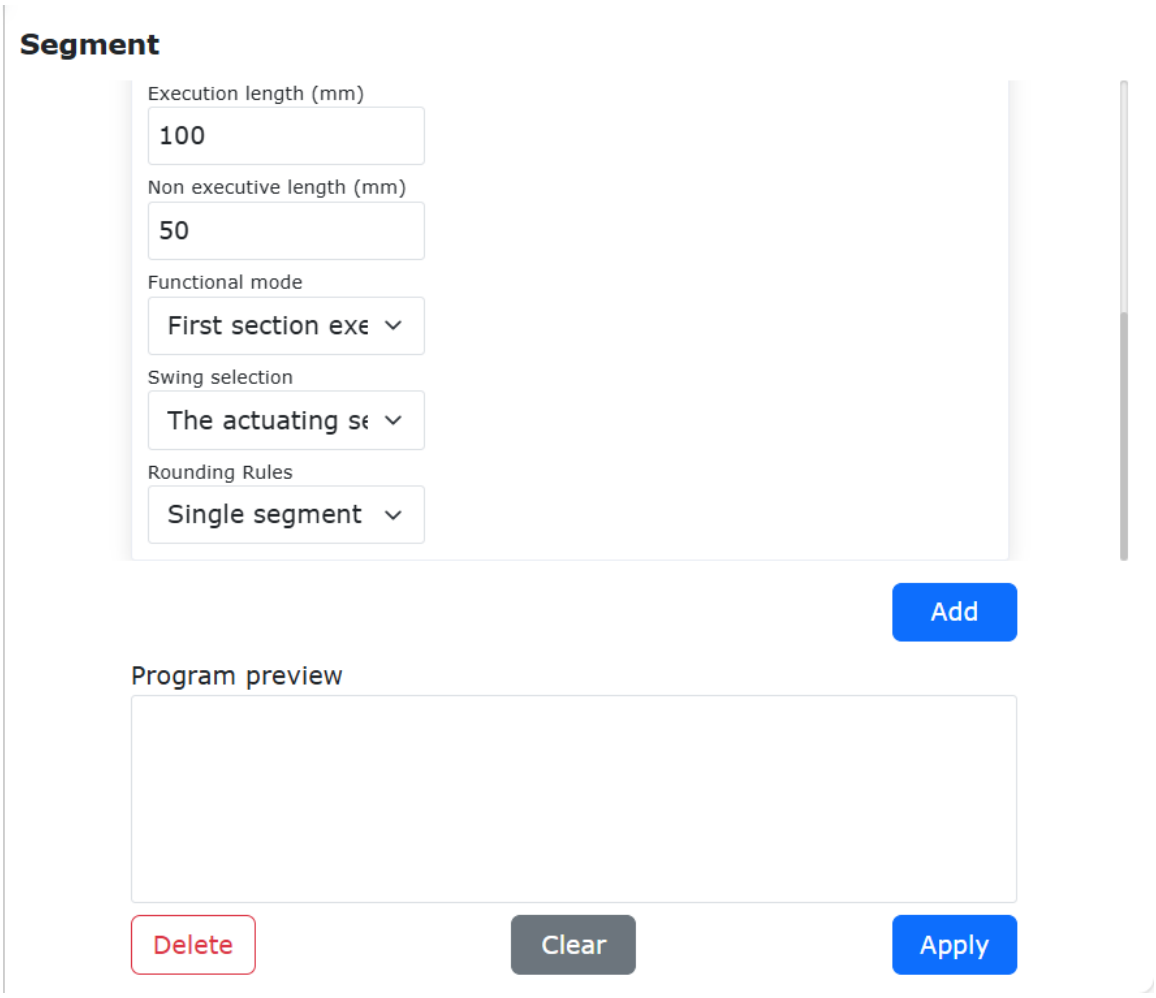


Figure 9.8-2-13 The first section does not perform single-section rounding of the swing function

4. Posture contrast

When configuring different segment welding modes, the posture of the robot during welding trajectory operation will also be different. The posture comparison during operation is as follows:

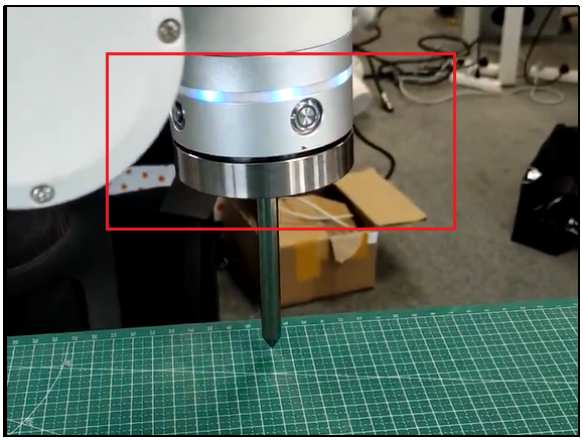


Figure 9.8-2-14 Initial posture of welding trajectory

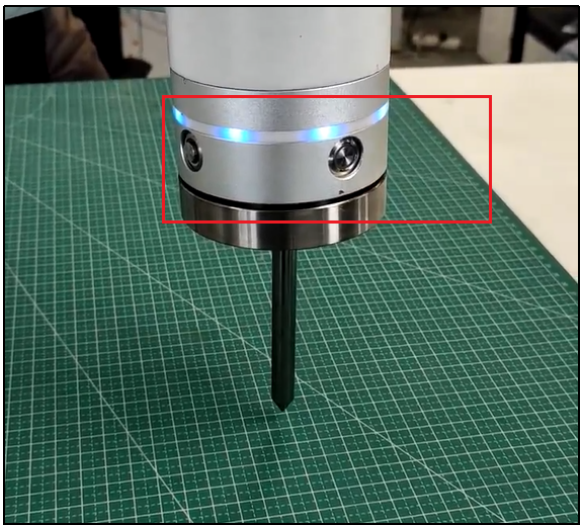


Figure 9.8-2-15 The posture does not change during operation

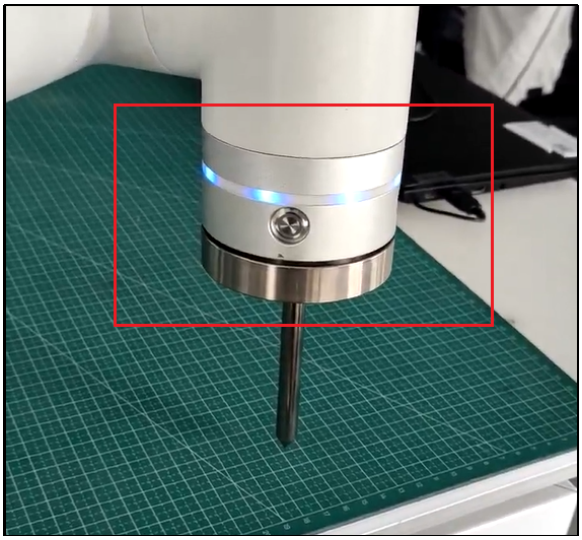


Figure 9.8-2-16 Change of attitude during operation

1.3.9.8.2.3 Actual scene of segment welding

In the actual test environment, the robot needs to be equipped with a welding gun and other configurations, and perform welding operations on the welding plate according to the created segment welding instructions. The actual scene diagram is as follows:



Figure 9.8-2-17 Actual scene of segment welding

1.3.9.8.3 Laser command

Click the “Laser” icon to enter the Laser command editing interface.

This command includes three parts: laser command, tracking command and positioning command. Before adding this command, please confirm whether the laser tracking sensor in the user peripheral has been configured successfully. See Robot Peripherals chapter for details.

In the sensor loading module, after displaying the corresponding “sensor command” interface according to the function selection, configure the sensor command:

Ruiniu/CXZK: Enter the weld type, range: 0~49 integer

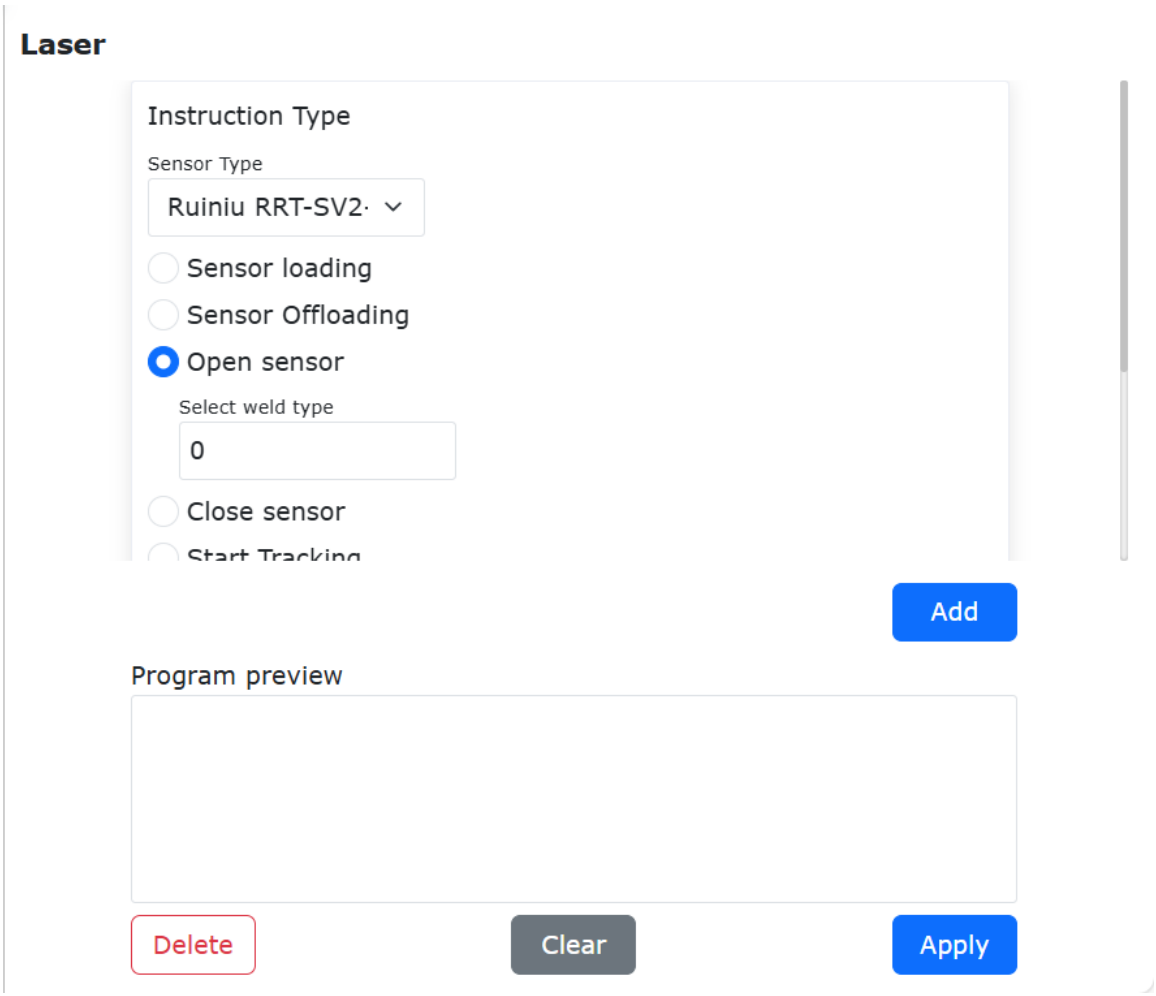


Figure 9.8-3-1 Laser command interface

Quanshi: Enter the task number, range: 0~255 integer

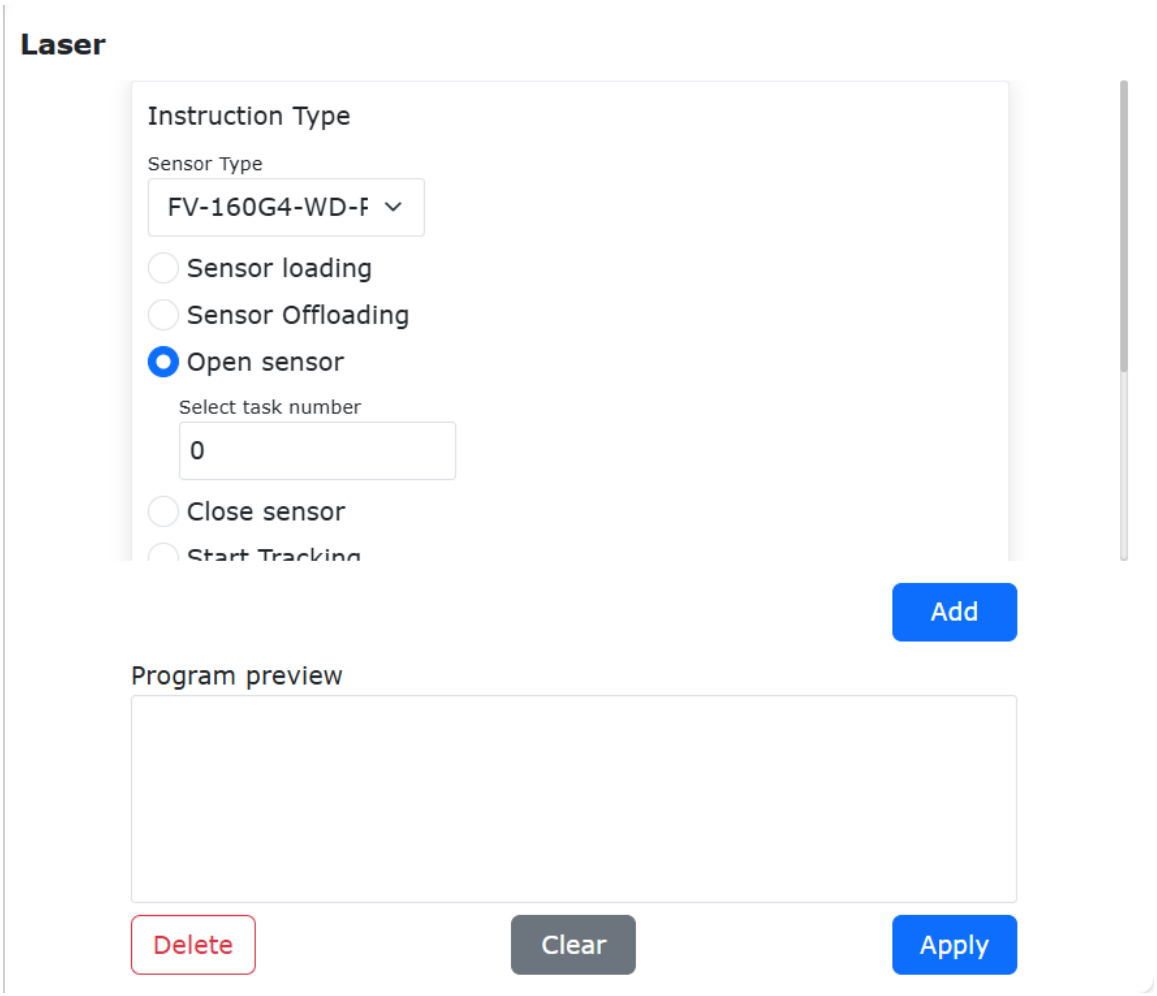


Figure 9.8-3-2 Laser command interface

1.3.9.8.3.1 Laser Sensor Fixed-Point Tracking Function

1.3.9.8.3.2 Overview

The current laser fixed-point tracking is implemented based on the extended axis method. Newly added custom tracking time tracking or IO trigger tracking methods are now available to adapt to various application scenarios. When selecting the custom tracking time tracking method, the tracking time needs to be set. Laser tracking starts when the program begins running and exits tracking when the set time is reached. When selecting the IO trigger tracking method, the Lua program or SDK program runs, tracking starts when the IO is triggered, and laser tracking stops when the IO signal is removed.

1.3.9.8.3.3 Custom Tracking Time Tracking Operation Process

Step1: Click “Initial Setup” - “Peripherals” - “Line Laser Sensor” - “Adapted Devices” to enter the configuration page. The configuration page includes “Sensor Configuration”, “Communication Configuration and Loading”, “Reference Calculation”, etc. Click “Sensor Configuration” to set sensor input filter parameters, set the maximum difference according to actual conditions, select data processing as “Raw Data (No Transformation)”, set the sensitivity coefficient X direction to 0, set Y direction and Z direction according to actual conditions, recommended to set to 1; click “Communication Configuration and Loading” to input corresponding communication parameters to connect the laser sensor. For detailed configuration, refer to the corresponding section of the user manual.

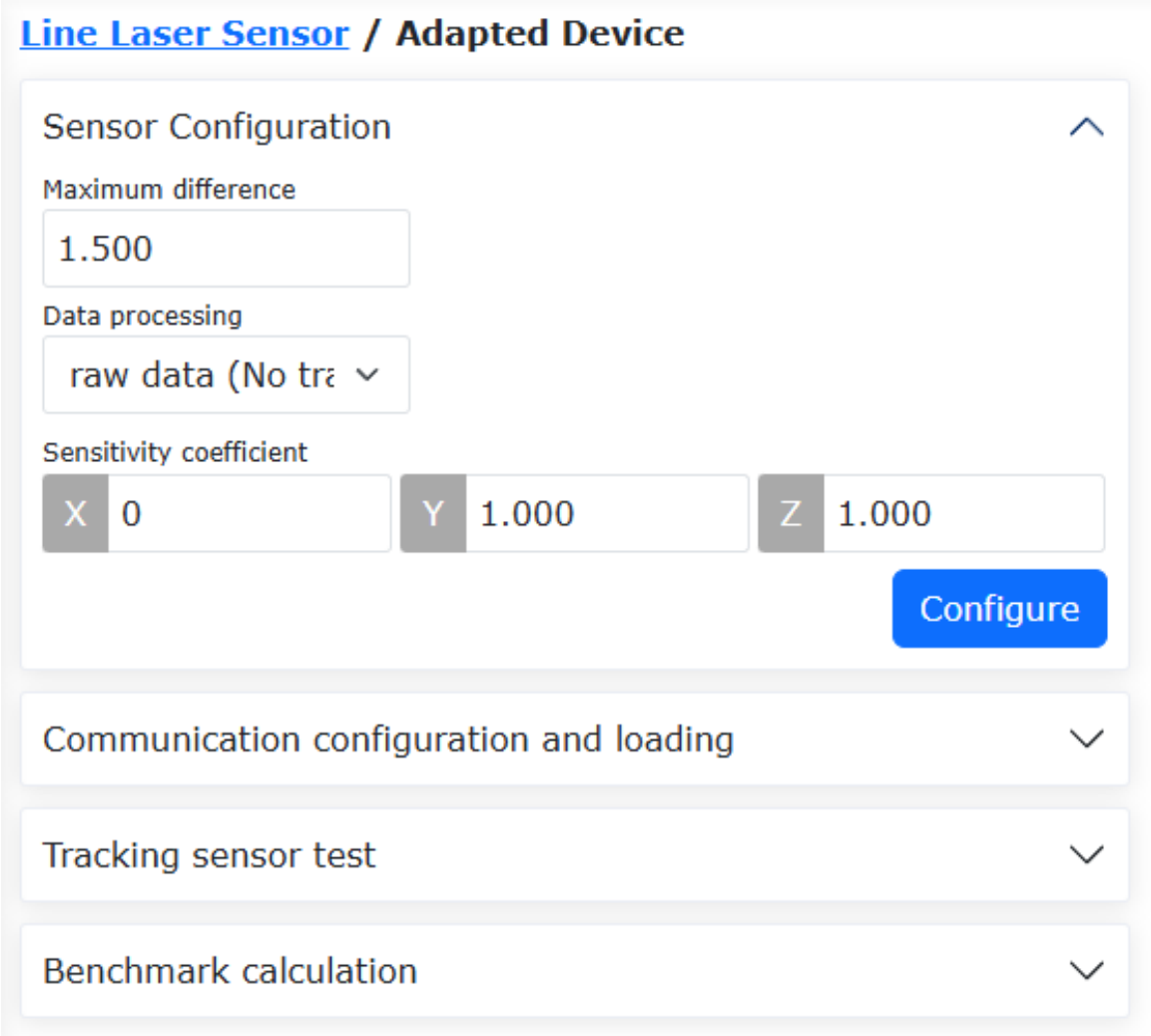


Figure 9.8-3-3 Line Laser Sensor Configuration

Step2: Calibrate the tool coordinate system and the laser sensor coordinate system. Calibrate the tool coordinate system using the “Six-Point Method”, and calibrate the laser sensor coordinate system using the “Five-Point Method”. Tool coordinate system and laser sensor coordinate system calibration are not the focus of this function introduction. For detailed calibration methods, refer to the corresponding section of the user manual.

Step3: Adjust the workpiece and laser beam position as shown in the figure below. The black rectangle is the workpiece, the red line segment is the laser beam. The laser beam should be perpendicular to the workpiece edge to be tracked, and the workpiece movement direction should be parallel to the laser beam. The workpiece moves at a constant speed, recommended speed is 15mm/s. Too fast will degrade tracking performance.

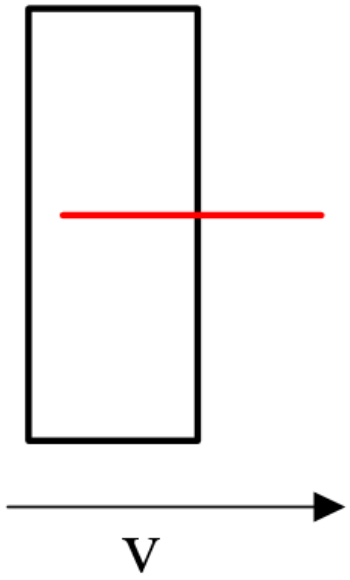


Figure 9.8-3-4 Workpiece and Laser Beam Relative Position Schematic

Step4: Click “Teach Program” - “Laser Tracking” - “Data Recording”, set function selection to “Record and Reproduce Simultaneously”, set fixed-point tracking motion type to “Robot Motion”, set fixed-point tracking trigger mode to “Time”, set tracking duration according to actual requirements. In this manual, 21s is used as an example. The settings of other parameters are the same as laser tracking using extended axis. Click the “Add” button below.

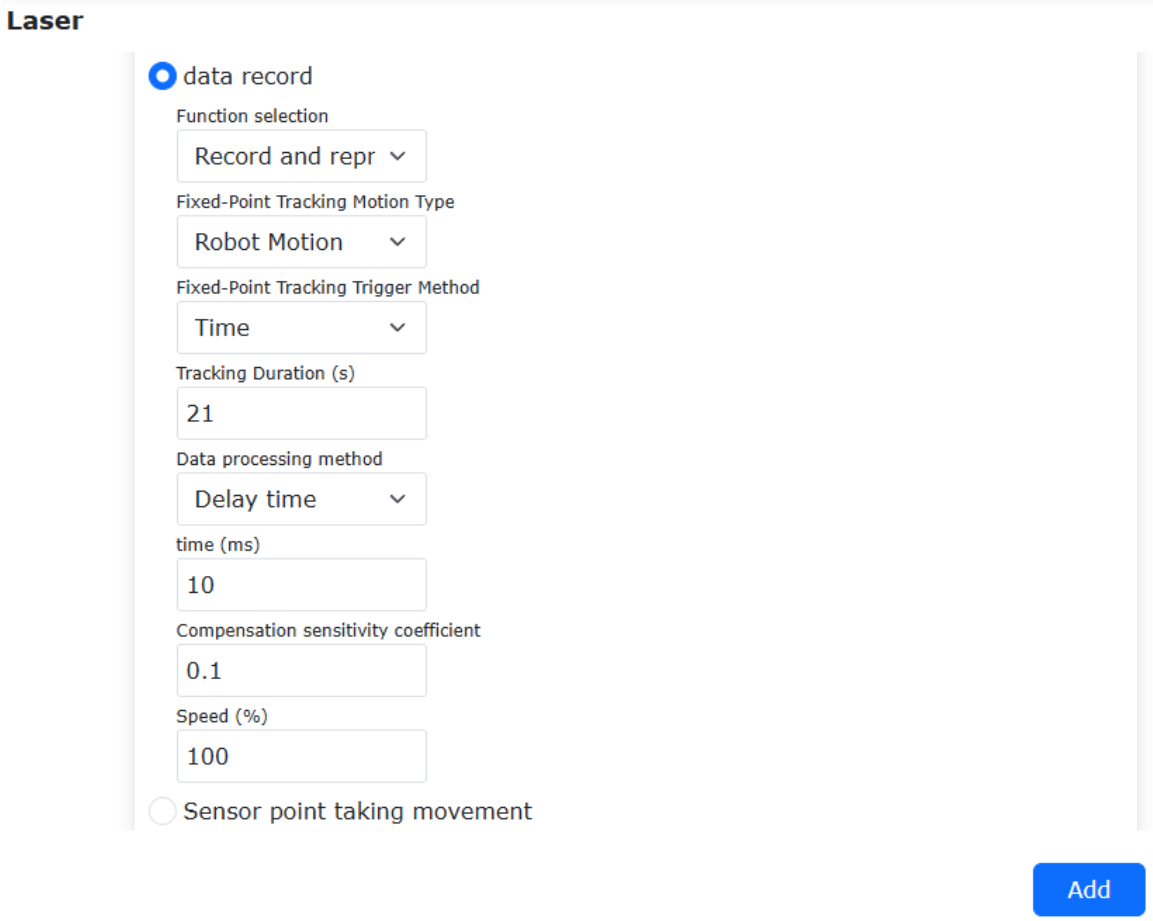


Figure 9.8-3-5 Custom Duration Tracking Parameter Settings

Step5: Click "Teach Program" - "Laser Tracking" - "Data Recording", set function selection to "Stop Recording", click the add button to generate the Lua program. When running this program, the robot will track for 21s and then exit tracking.

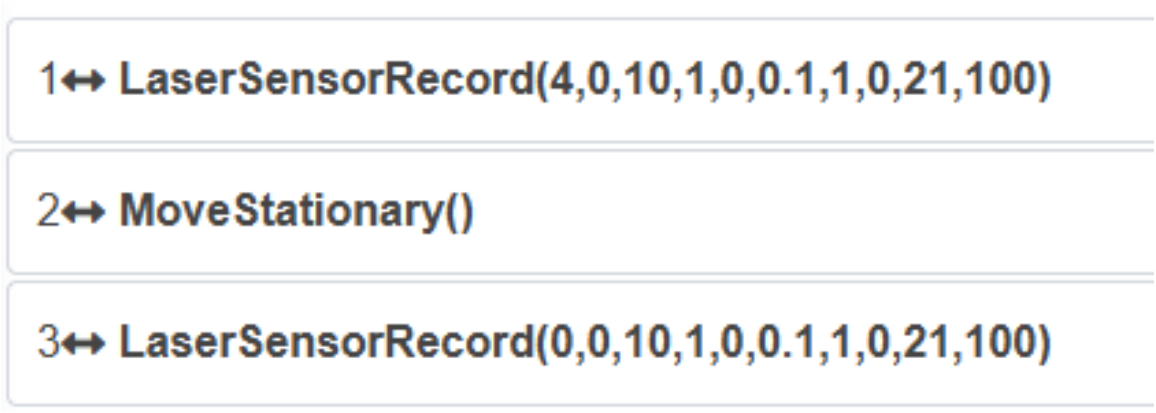


Figure 9.8-3-6 Custom Duration Tracking Typical Lua Program

1.3.9.8.3.4 IO Trigger Tracking Operation Process

Step1: Click “Initial Setup” - “Peripherals” - “Line Laser Sensor” - “Adapted Devices” to enter the configuration page. The configuration page includes “Sensor Configuration”, “Communication Configuration and Loading”, “Reference Calculation”, etc.

Click “Sensor Configuration” to set sensor input filter parameters, set the maximum difference according to actual conditions, select data processing as “Raw Data (No Transformation)”, set the sensitivity coefficient X direction to 0, set Y direction and Z direction according to actual conditions, recommended to set to 1; click “Communication Configuration and Loading” to input corresponding communication parameters to connect the laser sensor. For detailed configuration, refer to the corresponding section of the user manual.

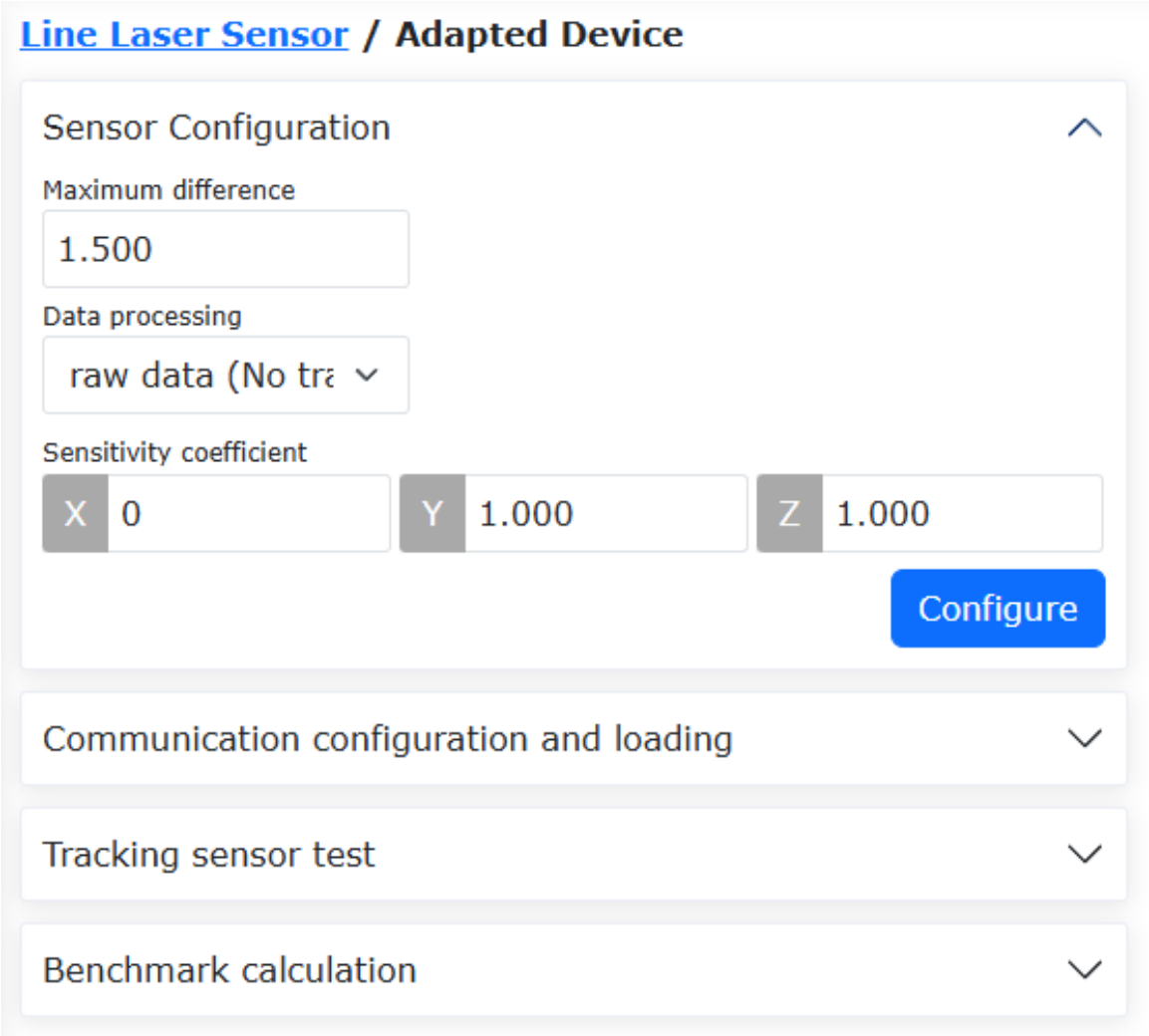


Figure 9.8-3-7 Line Laser Sensor Configuration

Step2: Calibrate the tool coordinate system and the laser sensor coordinate system. Calibrate the tool coordinate system using the “Six-Point Method”, and calibrate the laser sensor coordinate system using the “Five-Point Method”. Tool coordinate system and laser sensor coordinate system calibration are not the focus of this function introduction. For detailed calibration methods, refer to the corresponding section of the user manual.

Step3: Adjust the workpiece and laser beam position as shown in the figure below. The black rectangle is the workpiece, the red line segment is the laser beam. The laser beam should be perpendicular to the workpiece edge to be tracked, and the workpiece movement direction should be parallel to the laser beam. The workpiece moves at a constant speed,

recommended speed is 15mm/s. Too fast will degrade tracking performance.

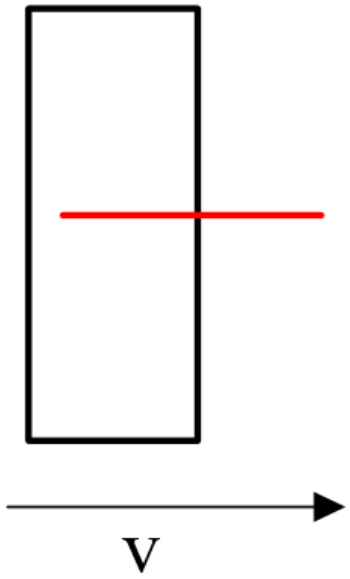


Figure 9.8-3-8 Workpiece and Laser Beam Relative Position Schematic

Step4: Click “Teach Program” - “Laser Tracking” - “Data Recording”, set function selection to “Record and Reproduce Simultaneously”, set fixed-point tracking motion type to “Robot Motion”, set fixed-point tracking trigger mode to “IO”. Tracking starts when IO is triggered and stops when the IO signal is removed. The settings of other parameters are the same as laser tracking using extended axis. Click the “Add” button below.

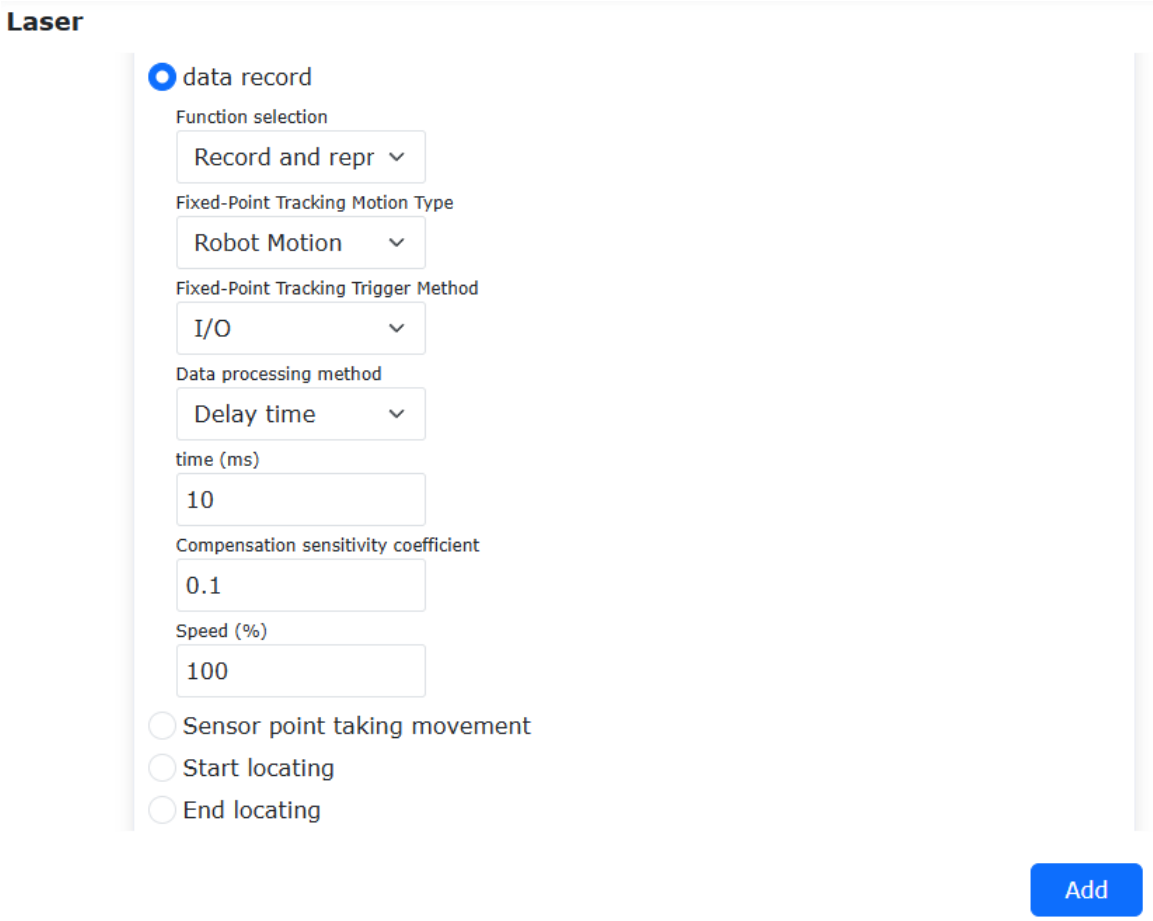


Figure 9.8-3-9 IO Tracking Parameter Settings

Step5: Click “Teach Program” - “Laser Tracking” - “Data Recording”, set function selection to “Stop Recording”, click the add button to generate the Lua program. When running this program, tracking starts when IO is triggered and stops when the IO signal is removed.

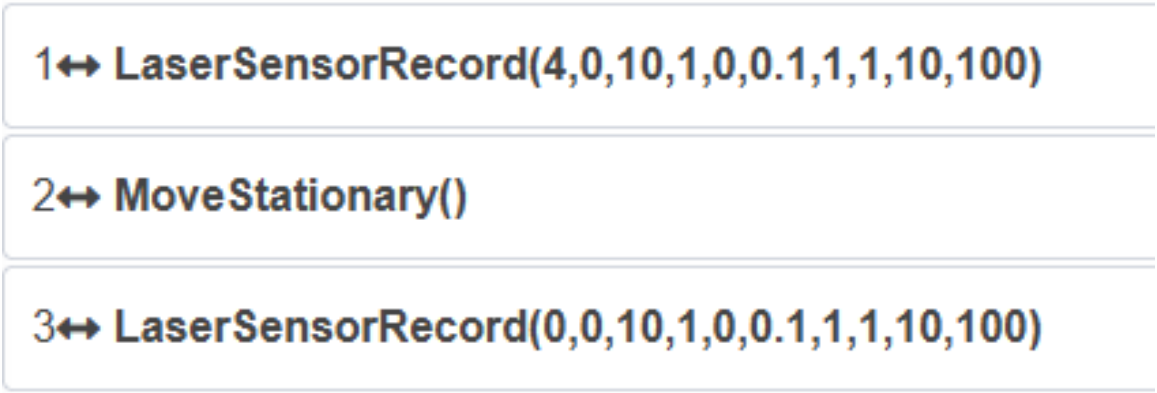


Figure 9.8-3-10 IO Tracking Typical Lua Program

1.3.9.8.4 LT-Rec command

Click the “LT-Rec” icon to enter the LT-Rec command editing interface.

This command realizes the function of taking out the starting point and end point of laser tracking recording, so that the robot can automatically move to the starting point position, which is suitable for the occasion where the movement starts from the outside of the workpiece and laser tracking recording is performed. At the same time, the host computer can obtain the information of the starting point and end point in the recorded data. for follow-up exercise.

Realize the adjustable function of laser tracking and reproduction speed, so that the robot can record at a very fast speed, and then reproduce according to the normal welding speed, which can improve the working efficiency.

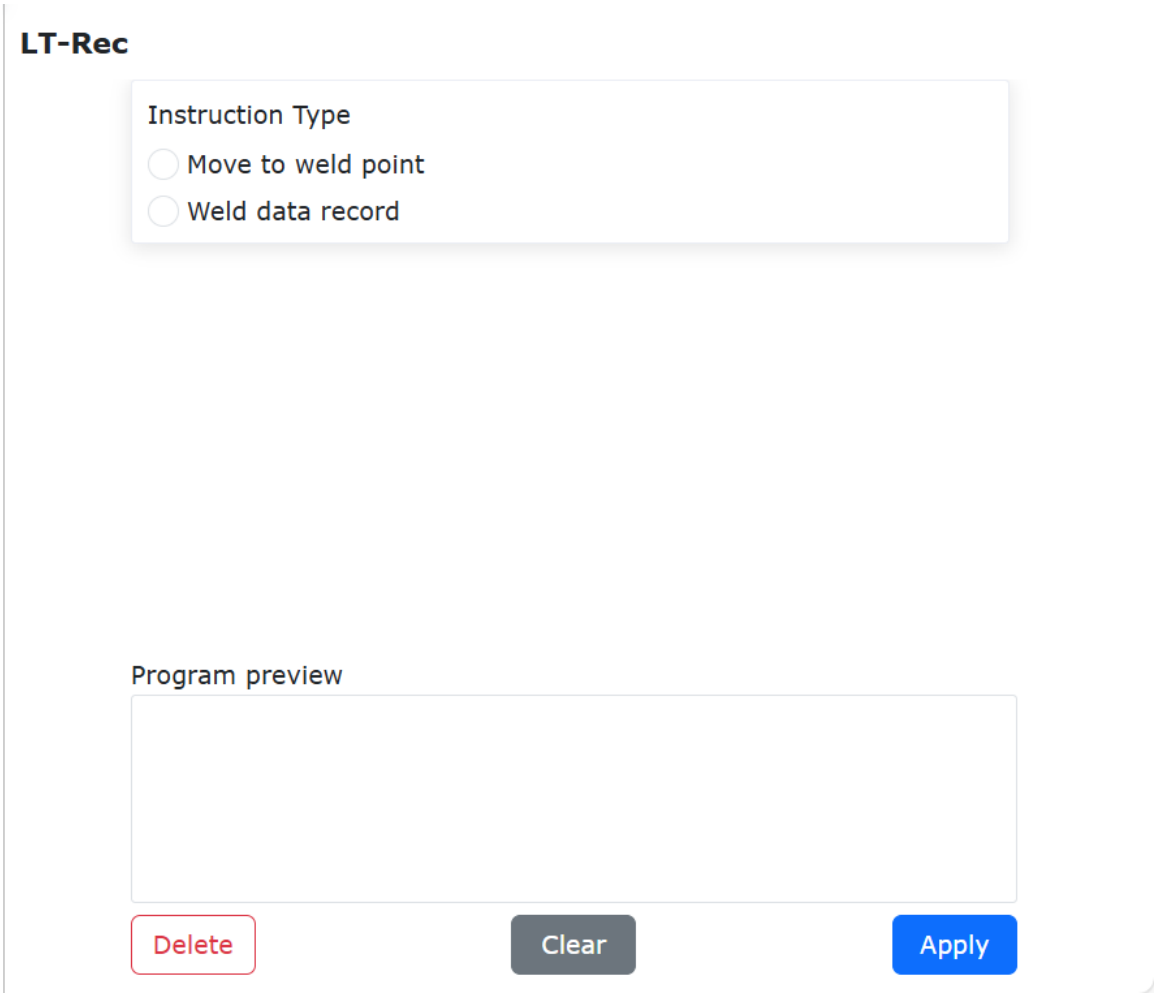


Figure 9.8-4 LT-Rec command interface

1.3.9.8.5 W-Search command

Click the "W-Search" icon to enter the W-Search command editing interface.

This command is a welding wire positioning command, including three commands of start of positioning, end of positioning and calculation of offset. This command is generally used in welding scenes and requires the combination of welding machine and robot IO and motion commands.

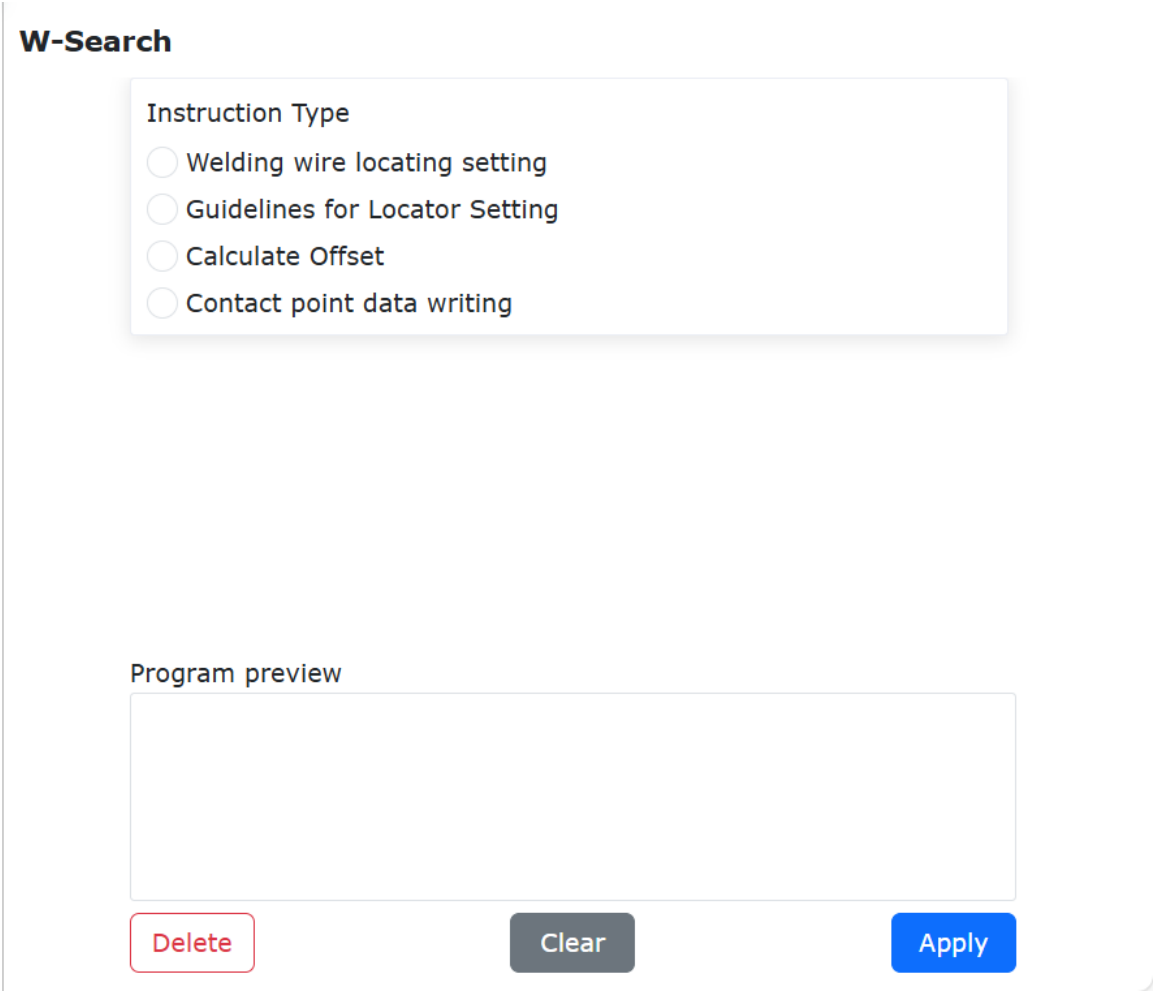


Figure 9.8-5 W-Search command interface

In writing a program, usually first set the start command of the search, and then add two LIN instructions to determine the direction of the search. After the search is successful, obtain the calculated offset, and pass the offset through the overall offset command. To take effect into the real welding motion command, the program example is as follows.

```
1 WireSearchStart(0,10,10,0,10,10,0)
2 Lin(1,100,-1,0,0)
3 Lin(1,100,-1,0,0)
4 WireSearchEnd(0,10,10,0,10,10,0)
```

Figure 9.8-5-1 W-Search example(1D)

1.3.9.8.6 Weld-Trc command

Click the "Weld-Trc" icon to enter the Weld-Trc command editing interface.

This command realizes the robot seam tracking and uses the deviation detection of the welding seam to compensate the trajectory, and the arc sensor can be used to detect the seam deviation.

Step1: Upper and lower compensation reference current setting method: feedback, set the upper and lower reference current start counting and the upper and lower reference current counting

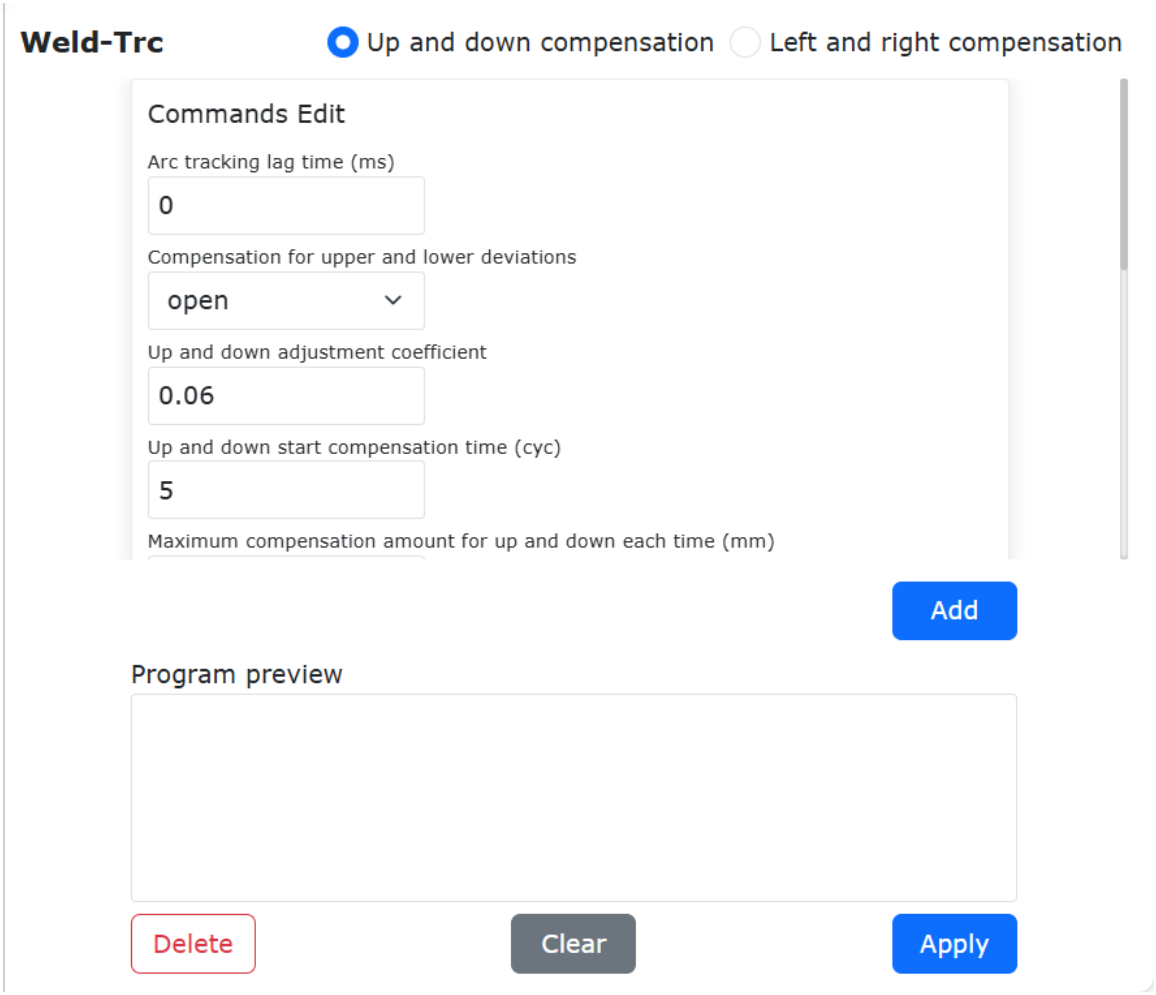


Figure 9.8-6-1 Weld-Trc command interface-Feedback

Step2:Upper and lower compensation reference current setting method: constant, set the upper and lower reference current

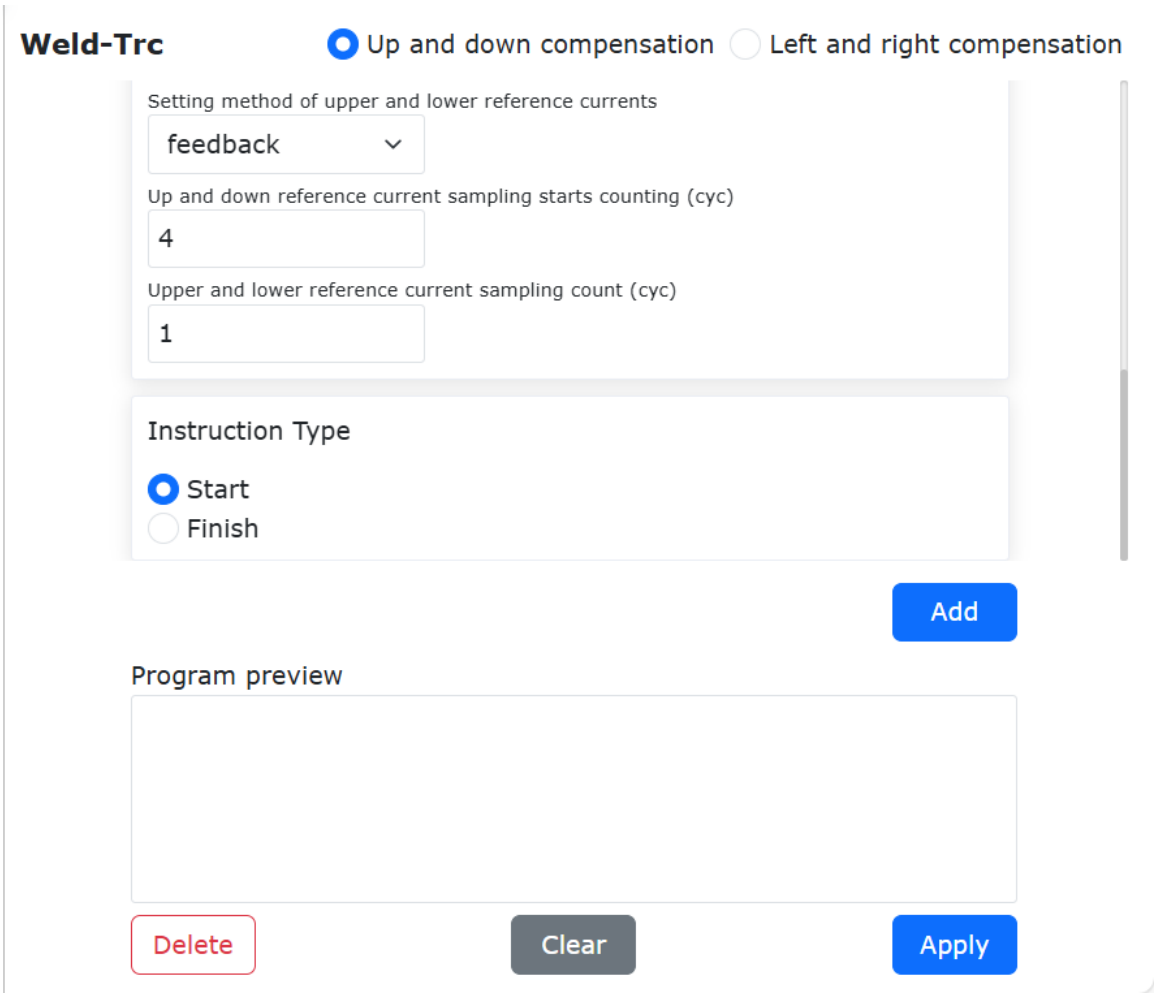


Figure 9.8-6-2 Weld-Trc command interface-constant

Step3:Left and right compensation parameter interactive page

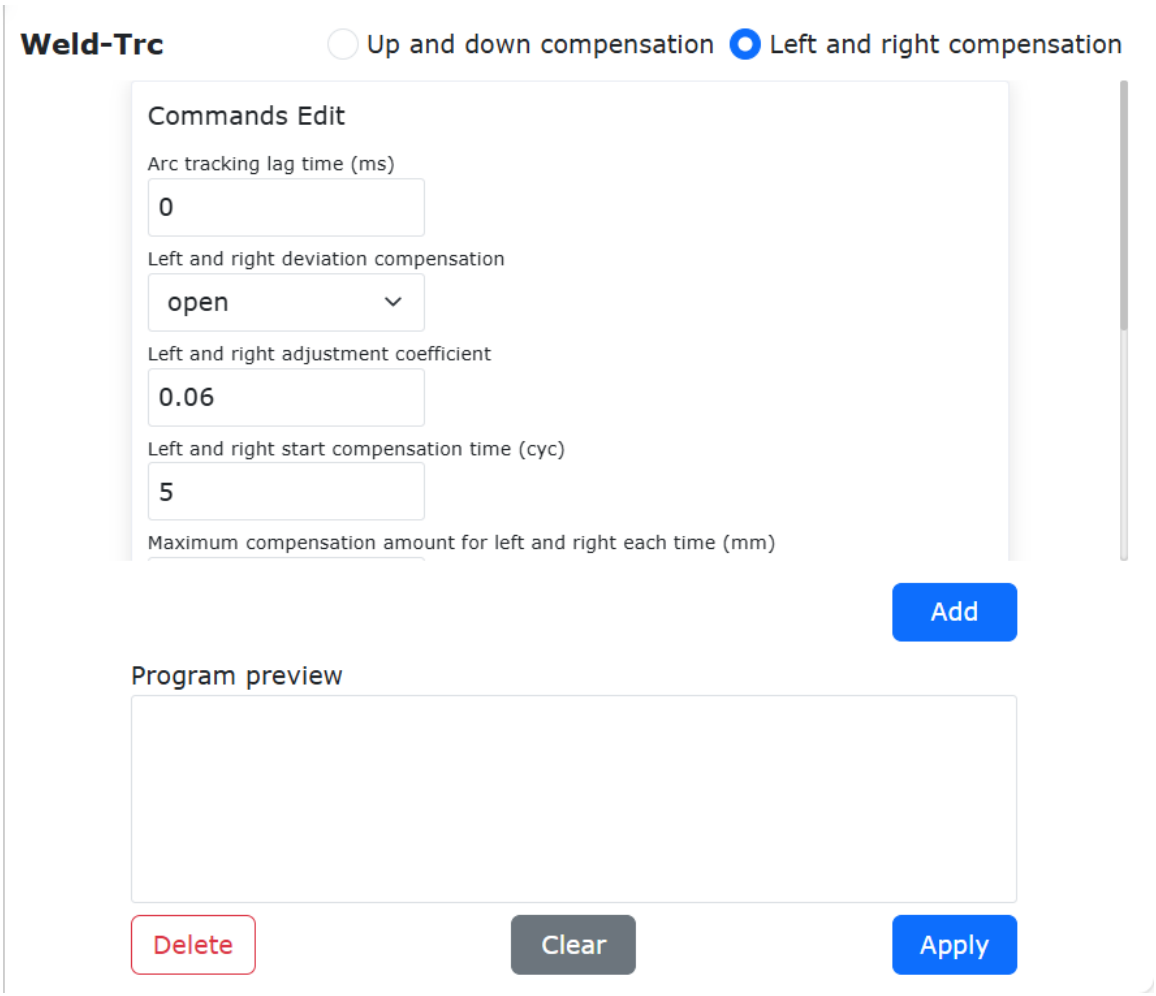


Figure 9.8-6-3 Weld-Trc command interface-left and right compensation parameters

1.3.9.8.7 Robot Arc Tracking System Composition

During the collaborative robot arc tracking welding process, the welding machine feeds back the real-time welding current and voltage signals to the robot. The robot performs welding trajectory position compensation based on the real-time feedback of welding current and voltage values to achieve the arc tracking effect. The feedback of current and voltage signals between the welding machine and the robot can be achieved in the following four ways. The first two require an additional PLC for data relay, while the latter two involve a direct connection between the welding machine and the robot control box:

CANopen or Other Bus Communication: If your welding machine supports bus communication protocols such as CANopen, EtherCAT, or ModbusTCP (e.g., OTC NBC-500RP, Megmeet A2 series, etc.), the PLC and the welding machine can communicate directly via the relevant communication protocol. The corresponding welding current signal can be transmitted directly to the PLC via communication and then fed back to the robot via UDP communication.

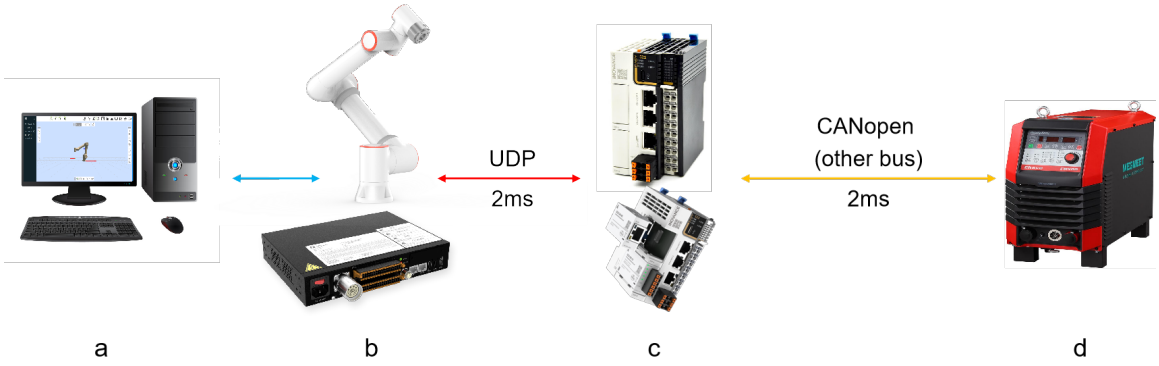


Figure 9.8-6-4 Robot Arc Tracking System Composition Topology Diagram (PLC and Welding Machine Bus Communication)

a-Computer; b-Robot and Control Box; c-PLC and Bus Communication Module; d-Welding Machine

PLC + IO Analog: The PLC can also directly collect analog signals and then convert the analog signals into current values with a certain conversion relationship to feed back to the robot. If your welding machine has a real-time welding current analog output channel, you can directly connect this channel to the analog input module of the PLC. If your welding machine does not have a real-time welding current analog output channel, you can connect an external Hall current sensor. The sensor collects the real-time welding current signal and converts the welding current signal into an analog signal output to the PLC analog input module.

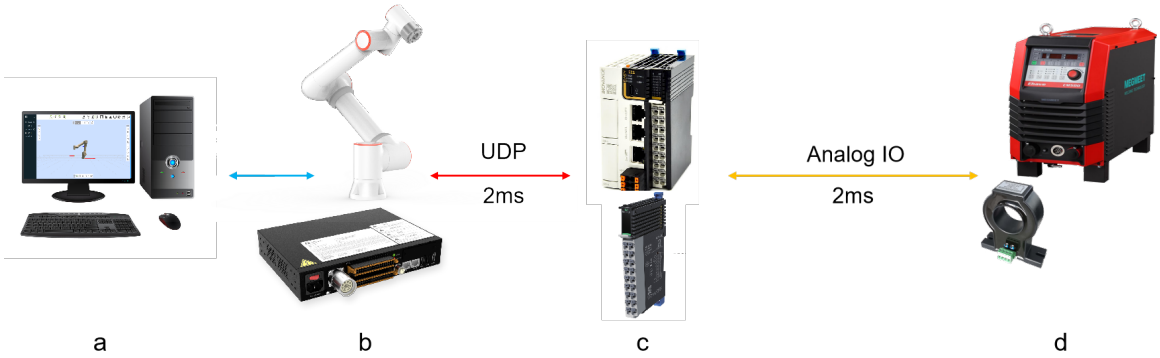


Figure 9.8-6-5 Robot Arc Tracking System Composition Topology Diagram (PLC Collecting Analog Signals)

a-Computer; b-Robot and Control Box; c-PLC and Analog Input Module; d-Welding Machine and Hall Current Sensor

Control Box AI: The robot control box IO ports have two analog input channels (0 ~ 10V). If your welding machine has a real-time welding current analog output channel, you can directly connect this channel to the analog input port of the control box. If your welding machine does not have a real-time welding current analog output channel, you can connect an external Hall current sensor. The sensor collects the real-time welding current signal and converts the welding current signal into an analog signal output to the analog input channel of the control box. There is usually a linear relationship between the analog input value and the actual welding current value. Detailed parameter settings are described later in “Arc Tracking Channel Configuration.”

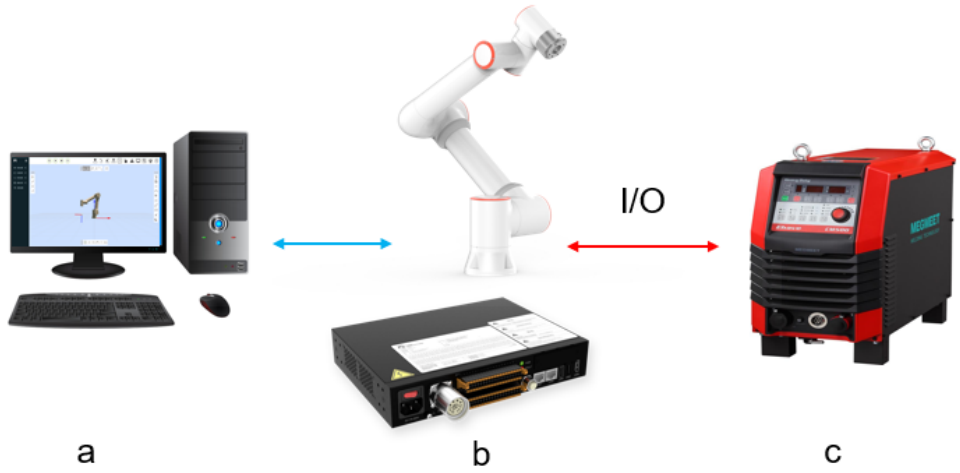


Figure 9.8-6-6 Robot Arc Tracking System Composition Topology Diagram (Control Box AI Collecting Analog Signals)

a-Computer; b-Robot and Control Box; c-Welding Machine and Hall Current Sensor

Ethernet Communication (ModbusTCP): If your welding machine supports ModbusTCP communication, the robot can directly control the welding and read real-time current and voltage feedback values via ModbusTCP. As shown in Figure 1-4. The ModbusTCP communication between the robot and the welding machine uses the control box peripheral open protocol. For details, see “8.6.6. Digital Communication Protocol (Modbus TCP).”

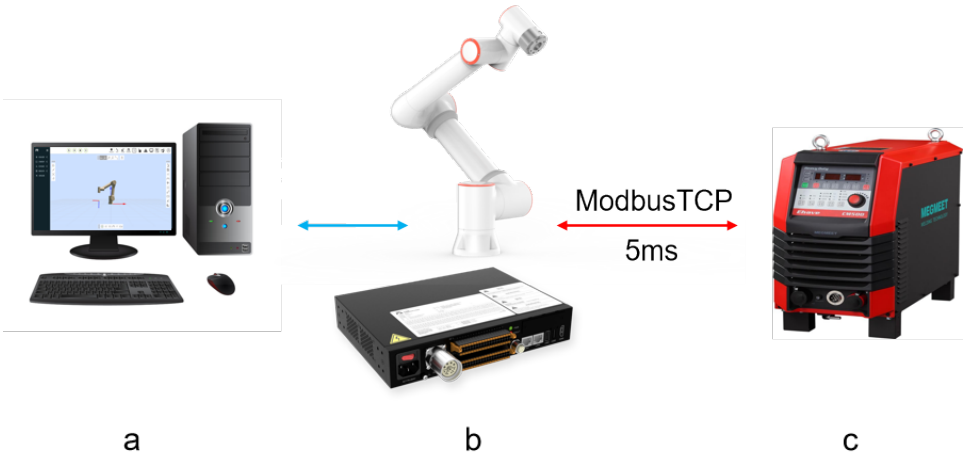


Figure 9.8-6-7 Robot Arc Tracking System Composition Topology Diagram (ModbusTCP Communication)

a-Computer; b-Robot and Control Box; c-Welding Machine

1.3.9.8.7.1 Welding machine model and setting

Table 9.8-1 Verified welding machine model

Verified welding machine model
MEGMEET ArtsenII CM350

Table 9.8-2 Welding machine function setting

Function number	Set the parameters
F18	20
F19	56

1.3.9.8.7.2 PLC model and settings

Table 9.8-3 Verified PLC models

Verified PLC models
INOVANCE Easy521

Table 9.8-4 PLC key settings

Settings	Set the contents
Communication protocols	CANOPEN
Feedback current sampling source	Feedback data from welding machine
Synchronization period	2ms

`Annex:PLC Program`

1.3.9.8.7.3 Arc tracking function

1Introduction to the arc tracing feature

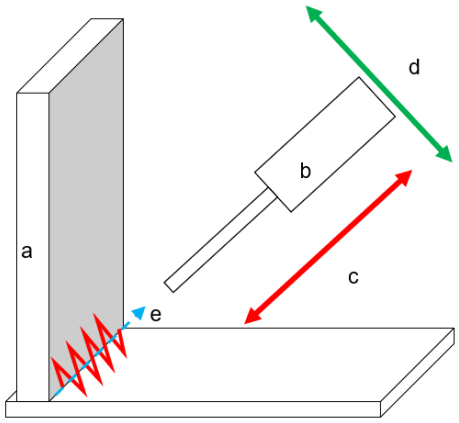


Figure 9.8-7-1 Typical arc tracing scenario

Typical scenarios for the arc tracing function include: a. welding the workpiece (the weld groove is at a right or acute angle), b. the welding gun, e. weld center line.

The arc tracking function can tracing the welding groove through the collected welding current information and the swing parameters: c. up and down (depth) direction tracking and d. left and right (center line) direction tracking.

2Communication Configuration

CANopen or Other Bus Communication:

Open WebApp and click “Initial”->”Peripheral” -> “Welder” in turn.

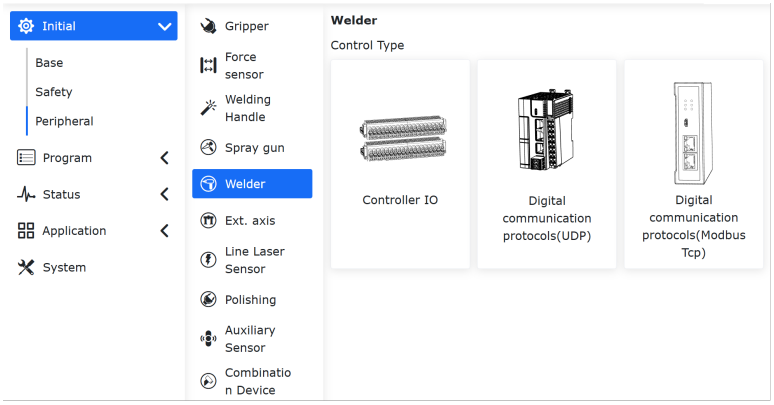


Figure 9.8-7-2 Open welding config

The control type is selected as “Digital communication protocol”. Since the robot communicates with PLC through UDP, UDP communication parameters need to be configured. The meanings of each parameter are as follows:

IP address:IP address of PLC for UDP communication;

Port number:PLC UDP communication port number;

Communication cycle:The cycle of UDP communication between the robot and the PLC, the default is 2ms;

Packet loss detection cycle and Packet loss times:When the number of lost packets in the packet loss detection period exceeds the specified value, the robot reports an error message indicating UDP Packet loss Exception and automatically cuts off the communication;

Communication interruption confirmation time:The robot does not receive a complete PLC feedback packet within this period of time, which will report “UDP communication interruption” error alarm, and cut off UDP communication;

Automatic reconnection after communication interruption:Whether the robot automatically reconnects after detecting UDP communication interruption;

Reconnection period and Num of reconnections:After automatic reconnection of UDP communication interruption is enabled and UDP communication interruption is detected, the robot reconnects at the set period. When the reconnection times reach the maximum set value and the connection is not successful, the robot reports an error alarm of “UDP communication interruption” and disconnects the UDP communication at the same time.

After configuring the above parameters, click the “Configure” and “Load” buttons successively.

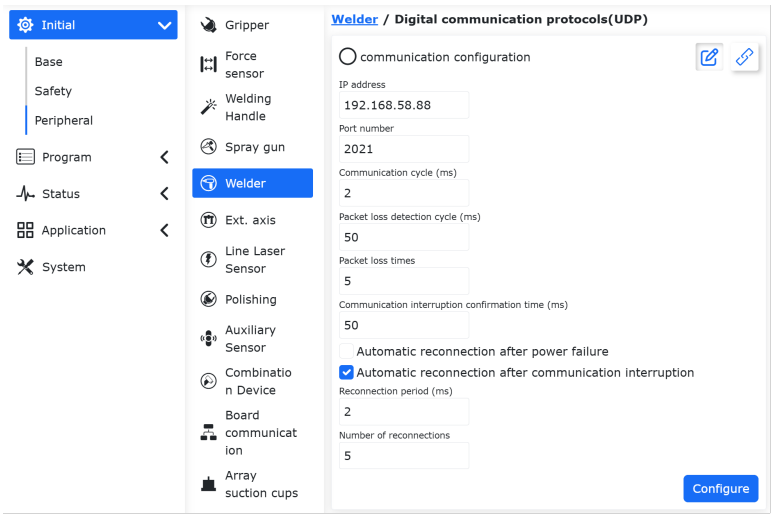


Figure 9.8-7-3 Selecting the control type

PLC + IO Analog:

Same as “CANopen or Other Bus Communication,” the PLC program converts the input analog data into current and voltage data in the UDP communication protocol and sends it to the robot.

Control Box AI:

No communication configuration is required. Simply connect the I/O cables between the control box and the welding machine correctly. The real-time welding current and voltage feedback analog lines from the welding machine are input to the robot control box AI0 and AI1 ports.

Ethernet Communication (ModbusTCP):

Correctly connect the network cable between the robot and the welding machine. In WebApp, click sequentially on “Initial Settings,” “Peripherals,” “Control Box,” “Peripheral Open Protocol.” Upload the welding machine communication protocol to the robot, then click “Configure” and “Load” buttons sequentially. The robot will then establish a ModbusTCP communication connection with the welding machine.

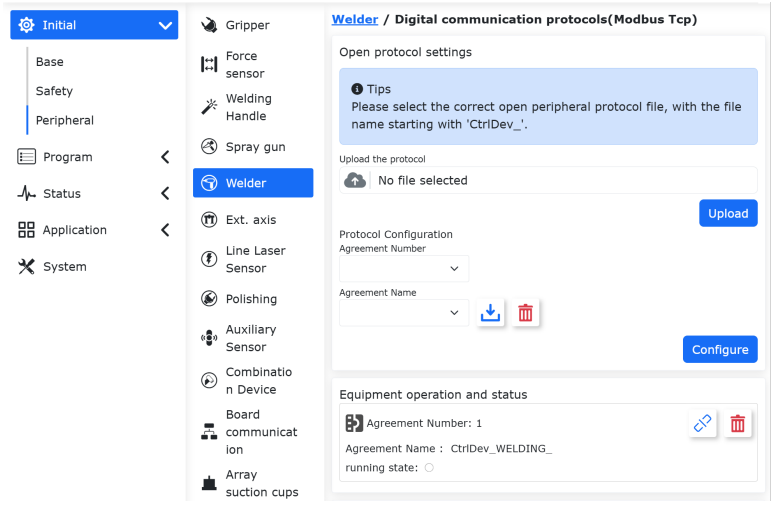


Figure 9.8-7-4 Establishing Arc Tracking Ethernet Communication

Note: The arc tracking effect depends on fast real-time welding current and voltage data feedback. If the feedback frequency is slow, it may lead to weld seam tracking failure. Therefore, when using ModbusTCP for welding ma-

chine communication, it is necessary to reasonably set the communication cycle in the protocol. The recommended communication cycle is less than 10ms.

3) Channel Configuration

CANopen or Other Bus Communication:

Click sequentially on “Initial Settings” -> “Peripherals” -> “Welding Machine” -> “Digital Communication Protocol (UDP).”

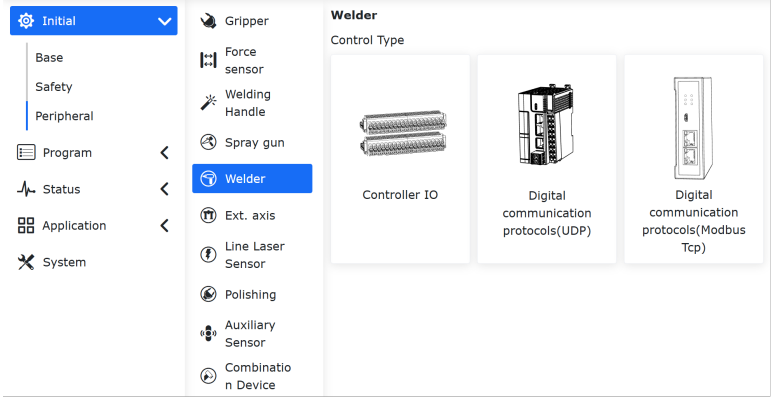


Figure 9.8-7-5 Selecting Welding Machine Control Type as “Digital Communication Protocol (UDP)”

Find the “Arc Tracking Channel” at the bottom of the page. Select the corresponding extended AI channel based on the actual configuration. The default welding current AI channel is “Aux-AI0” and the welding voltage AI channel is “Aux-AI1”. Click the “Configure” button.

Note: The UDP communication protocol between the robot and the PLC is detailed in “Appendix 1: Robot UDP Communication Protocol.” The feedback data from PLC to the robot in the protocol includes the actual welding current and voltage feedback input channels in bytes 74~77.

During welding, the PLC collects real-time welding current signals via CANOpen or other buses and provides feedback to the robot for arc tracking through the actual welding current and voltage values in bytes 74~77.

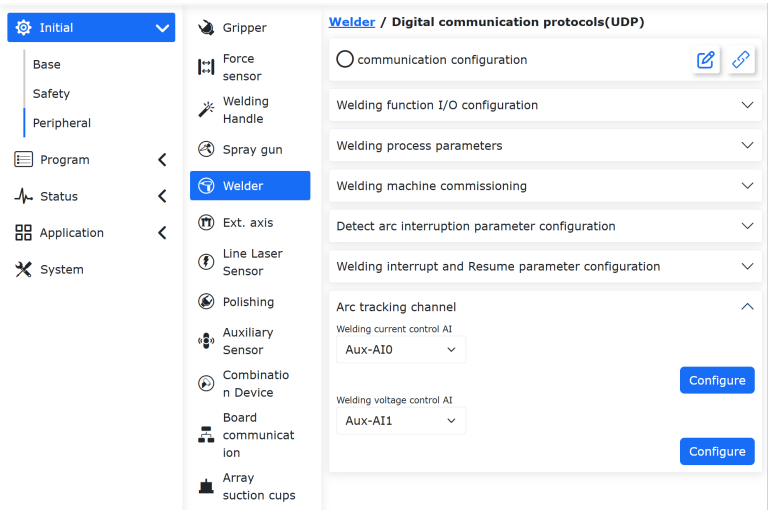


Figure 9.8-7-6 Bus Communication Arc Tracking Channel Configuration

PLC + IO Analog:

The configuration is the same as “CANopen or Other Bus Communication.” In the PLC program, the user converts the read analog input into actual welding current and voltage feedback values through numerical conversion, and assigns these values to the actual welding current and voltage feedback input channels (bytes 74~77) in the feedback data packet from PLC to the robot in the UDP communication protocol between the robot and the PLC.

Control Box AI:

Click sequentially on “Initial Settings” -> “Peripherals” -> “Welding Machine,” “Controller I/O.”

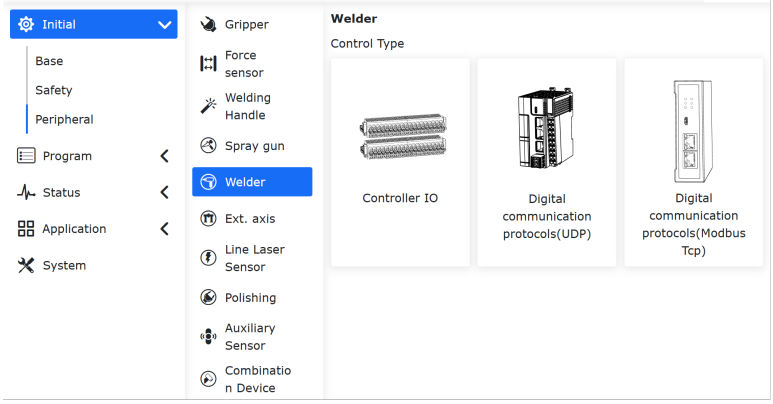


Figure 9.8-7-7 Selecting Welding Machine Control Type as “Controller I/O”

Find the “Arc Tracking Channel” at the bottom of the page. Select Welding Current AI as “Ctrl-AI0” and Welding Voltage AI as “Ctrl-AI1”. Click the “Configure” button. The control box analog input 0~10V often has a linear relationship with the actual feedback current and voltage values, so it is necessary to configure the actual welding current and voltage values corresponding to different analog inputs.

In the “Analog Current-Voltage Relationship Diagram” section of the AI channel configuration, the parameter settings for the “A-V” and “V-V” interfaces need to refer to the analog input/output table/chart of the welding machine being used.

For example, configure the lower and upper limits of the welding current for the control box current analog AI as 0A and 500A, respectively. Configure the lower and upper limits of the output voltage for the control box current analog AI as 0V and 5V, respectively, as the configuration parameters for the “A-V” interface in the “Analog Current-Voltage Relationship Diagram” section of the AI channel configuration. Click “Configure” to complete the configuration of the control box current analog AI channel.

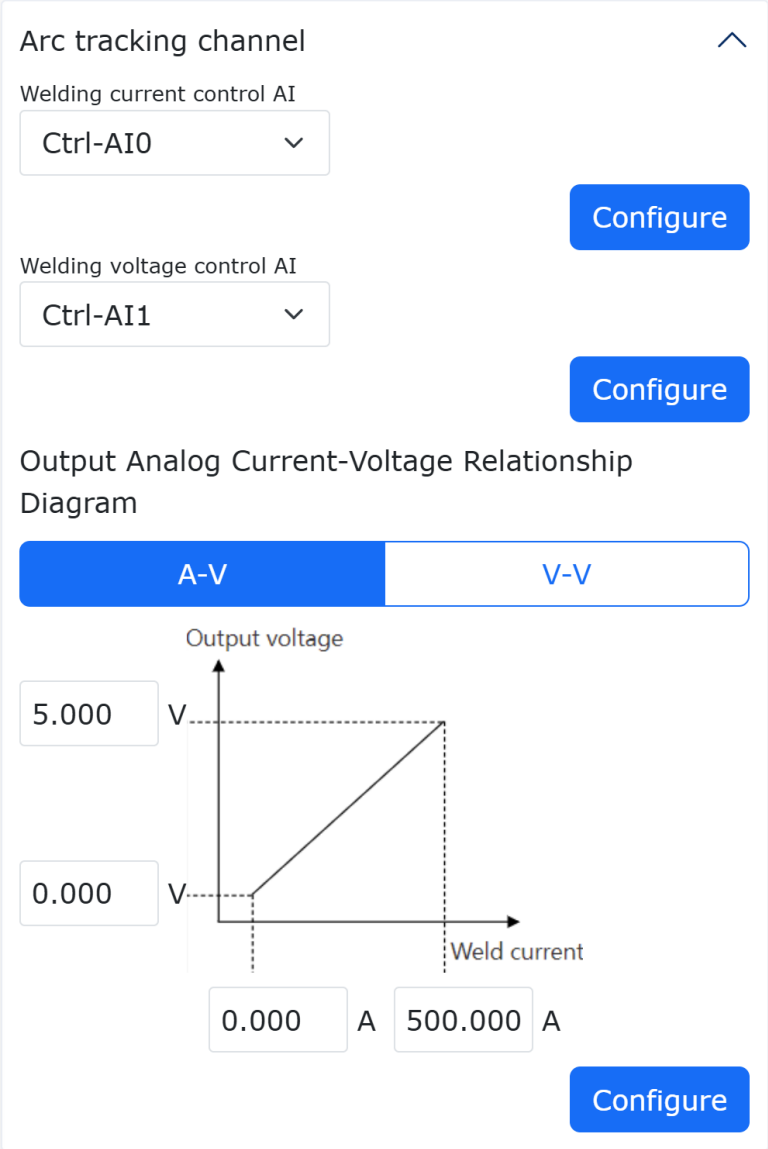


Figure 9.8-7-8 Control Box Current Analog AI Configuration

For example, configure the lower and upper limits of the welding voltage for the control box voltage analog AI as 0V and 50V, respectively. Configure the lower and upper limits of the output voltage for the control box voltage analog AI as 1.018V and 10V, respectively, as the configuration parameters for the “V-V” interface in the “Analog Current-Voltage Relationship Diagram” section of the AI channel configuration. Click “Configure” to complete the configuration of the control box voltage analog AI channel.

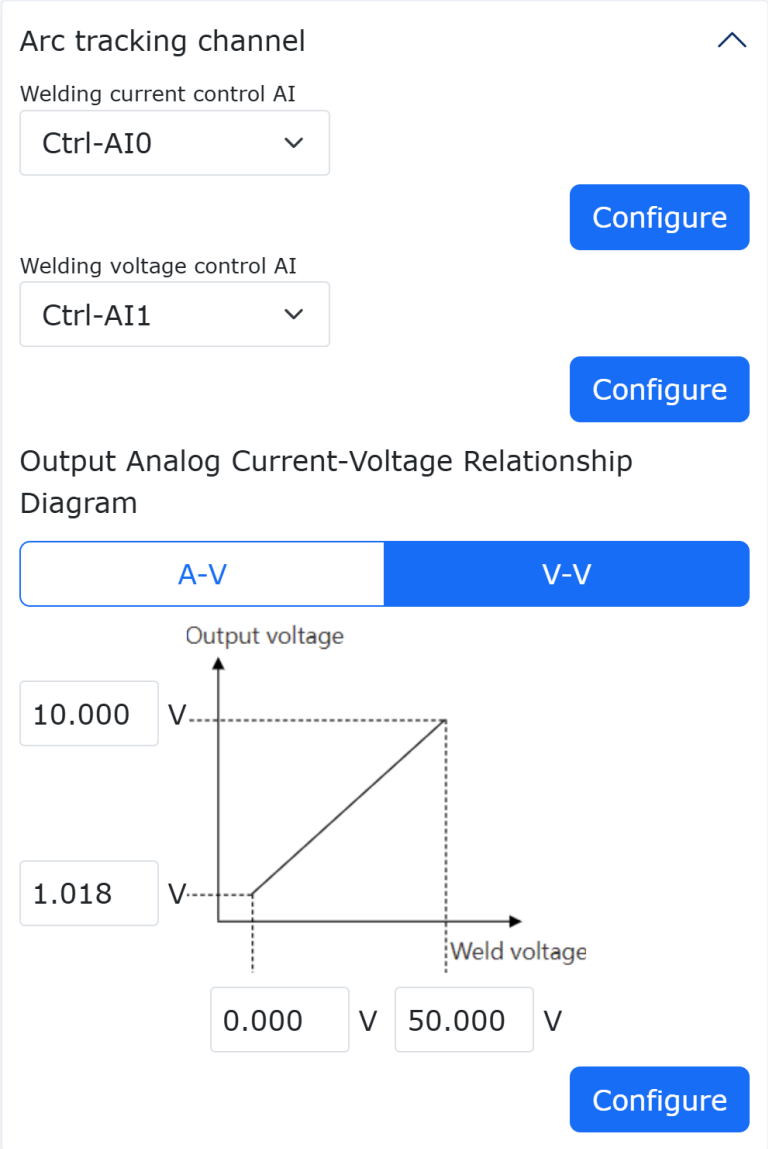


Figure 9.8-7-9 Control Box Voltage Analog AI Configuration

Ethernet Communication (ModbusTCP):

Click sequentially on “Initial Settings,” “Peripherals,” “Welding Machine,” “Digital Communication Protocol (Modbus TCP).”

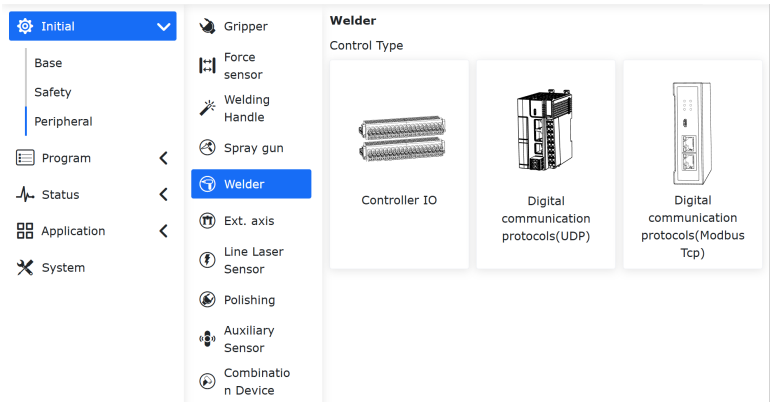


Figure 9.8-7-10 Selecting Welding Machine Control Type as “Digital Communication Protocol (Modbus TCP)”

Find the “Arc Tracking Channel” at the bottom of the page. Select Welding Current AI as “Ethernet” and Welding Voltage AI as “Ethernet”. Click the “Configure” button.

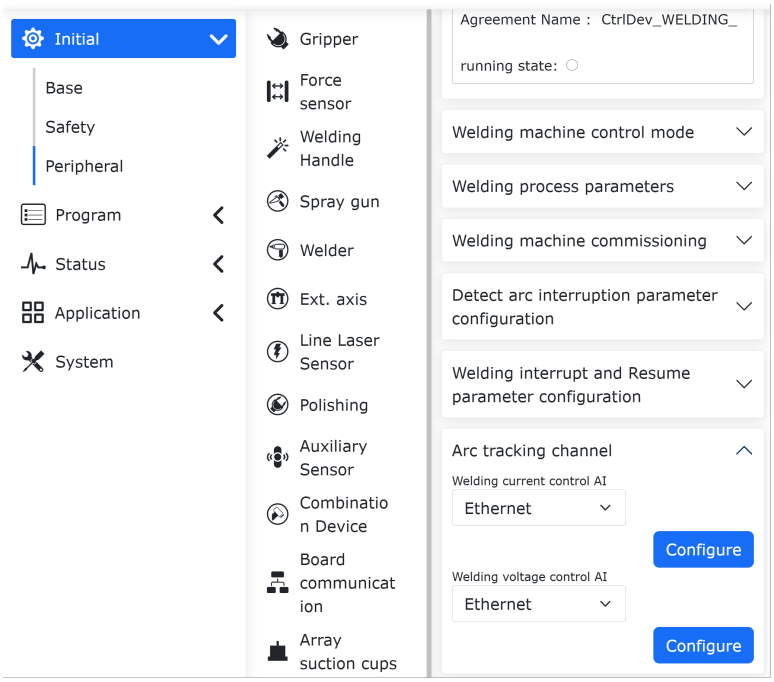


Figure 9.8-7-11 Ethernet Communication Arc Tracking Channel Configuration

4Introduction to the use of arc tracing function commands

The arc tracking function can be adapted to the swing welding movement, inserting the arc tracking start command after the swing welding arc starts, and inserting the arc tracking end command before the swing welding arc extinguishing.

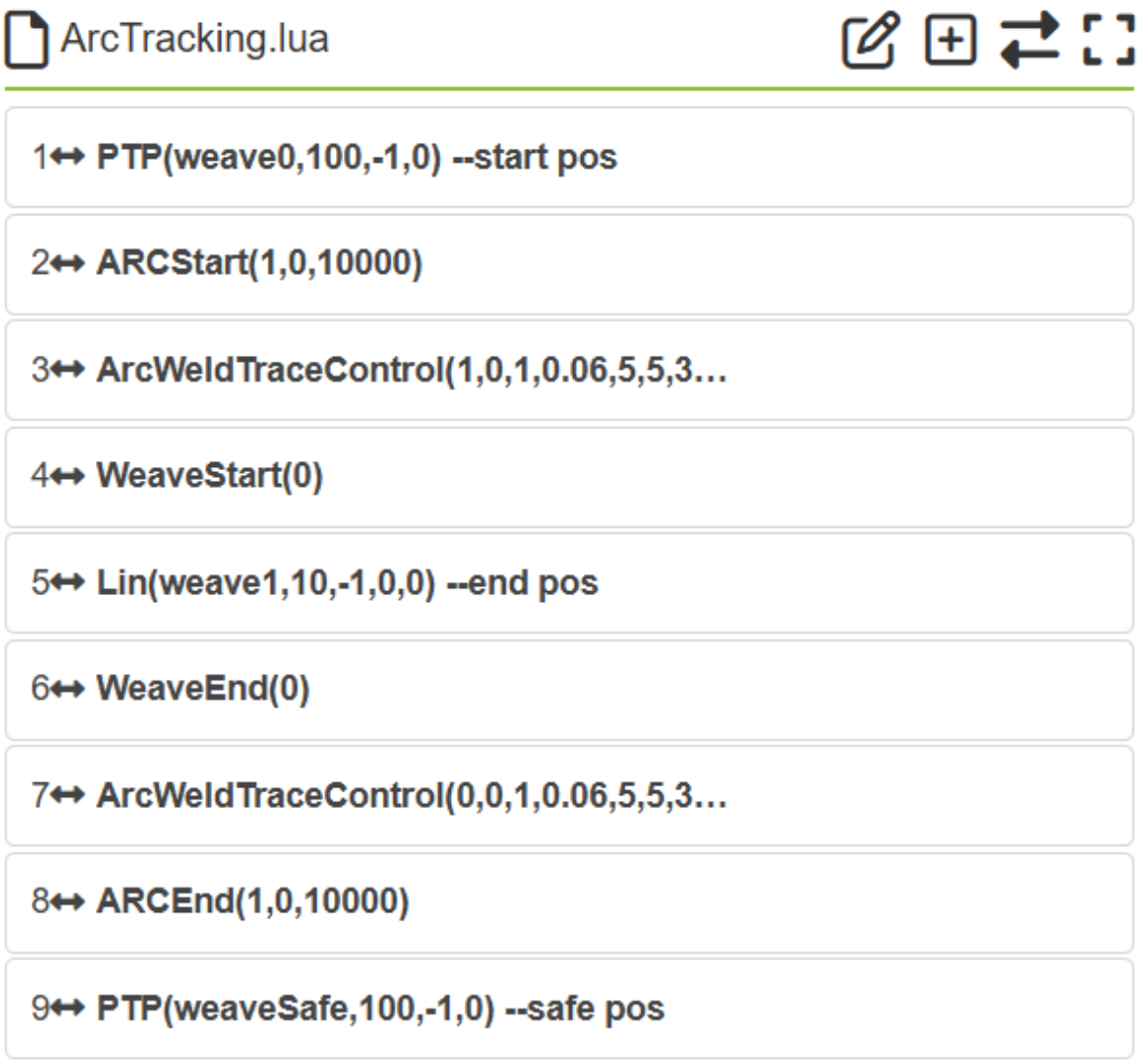


Figure 9.8-7-12 Typical example of arc tracing

5)Introduction to the parameters of the function interface

Table 9.8-5 Arc tracking up-down compensation module

The name of the parameter	Meaning	Description
Arc tracking lag time	The time for the feedback current to lag	The default is 0ms, do not adjust it
Compensation for up-down deviations	Up-down compensating switches	You can choose to “on” or “off”
Up-down adjustment coefficients	The coefficient of the relationship between the current and the compensation distance (adjustment sensitivity)	The welding tends to be in a short-circuit transition state, and the current signal-to-noise ratio gradually decreases, so it is recommended to reduce the sensitivity
Up-down start compensating period	The fastest periods to start the up-down compensation	Associated with the swing frequency, it is better to open when the current tends to be stable for 3~4s after arcing. If the oscillation frequency is 1 Hz, the parameter can be 4, if the frequency is 2 Hz, the parameter can be 8, and so on
The maximum distance of up-down compensation each period	The maximum distance of compensation per up-down compensation period	According to the welding scene setting, the faster the weave frequency, the smaller the compensation distance
The maximum distance of compensation for the up-down totals	The maximum cumulative distance of compensation for a single complete welding process	According to the welding scene setting, the larger the weld deviation, the larger the setting
Up-down coordinate system selection	The coordinate system in which the compensation value is compensated	If there is a welding weave, select “Weave”, otherwise select “Tool” or “Frame”
Up-down reference current setting method	Selection of the reference current acquisition method	You can choose “Feedback” by reading the feedback current, or “Constant” by filling in the current value directly
The up-down reference current samples start counting	The number of periods for which the reference current is harvested with a delay	Associated with the weave frequency, it is better to open when the current tends to be stable for 3~4s after arcing. If the weave frequency is 1 Hz, the parameter can be 4, if the frequency is 2 Hz, the parameter can be 8, and so on
Up-down reference current sampling counts	Reference current feedback mode, the statistical period of collecting the reference current	Default 1 CYC
Up-down reference currents	Reference current constant mode, reference current value	It can be filled in manually to achieve the desired compensation height

Table 9.8-6 Arc tracking left-right compensation module

The name of the parameter	Meaning	Parameter description
Arc tracking lag time	The time for the feedback current to lag	The default is 0ms, do not adjust it
Compensation for left-right deviations	Left-right compensation switches	You can choose to “On” or “Off”
Left-right compensation coefficients	The coefficient of the relationship between the current and the compensation distance (adjustment sensitivity)	The welding tends to be in a short-circuit transition state, and the current signal-to-noise ratio gradually decreases, so it is recommended to reduce the sensitivity
Left-right compensation start counting	The fastest count to start left-right compensation function	Associated with the weave frequency, it is better to start when the current tends to be stable for 3~4s after arcing. If the weave frequency is 1 Hz, the parameter can be 4, if the frequency is 2 Hz, the parameter can be 8, and so on
Left-right the maximum distance of compensation each time	The maximum distance of compensation per compensation cycle	According to the welding scene setting, the faster the swing frequency, the smaller the compensation distance
The maximum distance of compensation for the left -right totals	The maximum cumulative distance of compensation for a single complete welding process	According to the welding scene setting, the larger the weld deviation, the larger the setting

6)Scope of application

Table 9.8-7 Up-down compensation On, Left-right compensation Off

Key parameters	Parameter range
Weave frequency Hz	0 (without welding swing), 0.5 to 2 (with weld swing)
Weave amplitude mm	0 (without welding swing), 3 through 7 (with welding swing)
Set the voltage V	>17
Set the current A	>160

Table 9.8-8 Up-down compensation Off, Left-right compensation On

Key parameters	Parameter range
Weave frequency Hz	0.5 to 2
Weave amplitude mm	3 to 7
Set the voltage V	>17
Set the current A	>160

Table 9.8-9 Up-down compensation On, Left-right compensation On

Key parameters	Parameter range
Weave frequency Hz	0.5 to 2
Weave amplitude mm	3 to 7
Set the voltage V	>19
Set the current A	>210

7)Precautions

1) The left-right compensated arc tracking function can only be adapted to symmetrical triangle or sine weave based on line trajectory.
2) The starting position of the welding to be able using the compensation function must be accurately above the weld (the axis of the welding gun is in the center of the fillet weld), and the welding gun should not be too close to the seam, otherwise there is a risk of hitting the welding gun.
3) The material on both sides of the groove of the workpiece needs to be consistent.
4) The coordinate size and attitude of the workpiece need to be accurately calibrated using the 6-point method.
5) If the deviation between the set trajectory and the seam is larger, the maximum compensation distance each time and the total maximum compensation distance should be larger too.
6) The deviation between the set trajectory and the end point of the weld should not be larger than 100mm/m, and too large the deviation may cause the welding wire or even the welding gun to hit the workpiece, so that the welding position deviates from the preset trajectory (the weave is not in place), resulting in the arc tracking function can not work normally.
7) If a small set current and voltage is selected for welding, the compensate coefficient of arc tracking should be reduced accordingly to reduce the instability compensation caused by the burr of the feedback current.
8) When different coordinate systems are selected for arc tracking, the positive and negative signs of the up-down and left-right compensation coefficients may need to be adjusted, which can be judged by test welding in addition to the direction of the corresponding coordinate system. If the welding trajectory (left) is taught on an inclined plate, the welding trajectory (right) moves in the direction of descent based on the tilt gradient of the swing plane after arc tracking is enabled, and the height of the welding torch at the end is close to the starting point, then the sign of the adjustment coefficient is correct.

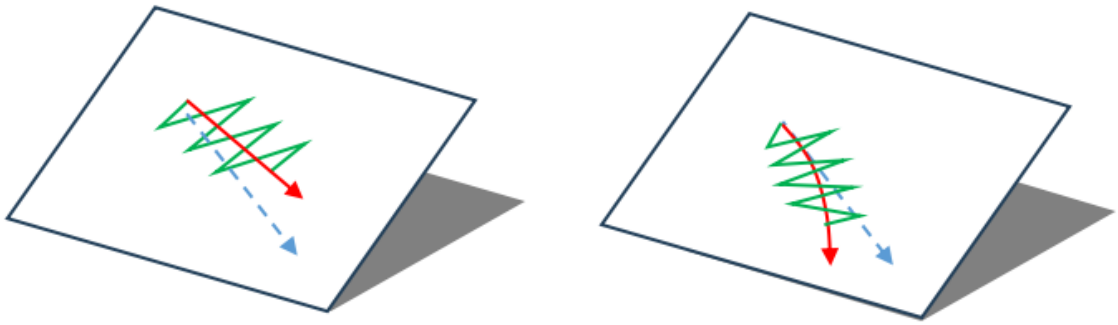


Figure 9.8-7-13 Set the tilt swing trajectory (left), weld trajectory when the symbol is correct (right)

1.3.9.8.8 Adjust command

Click the “Adjust” icon to enter the Adjust command editing interface.

This command adaptively adjusts the posture of the welding torch for the scene of welding tracking. After recording the three corresponding posture points, add the posture adaptive adjustment command according to the actual direction of the robot’s movement. See Robot Peripherals chapter for details.

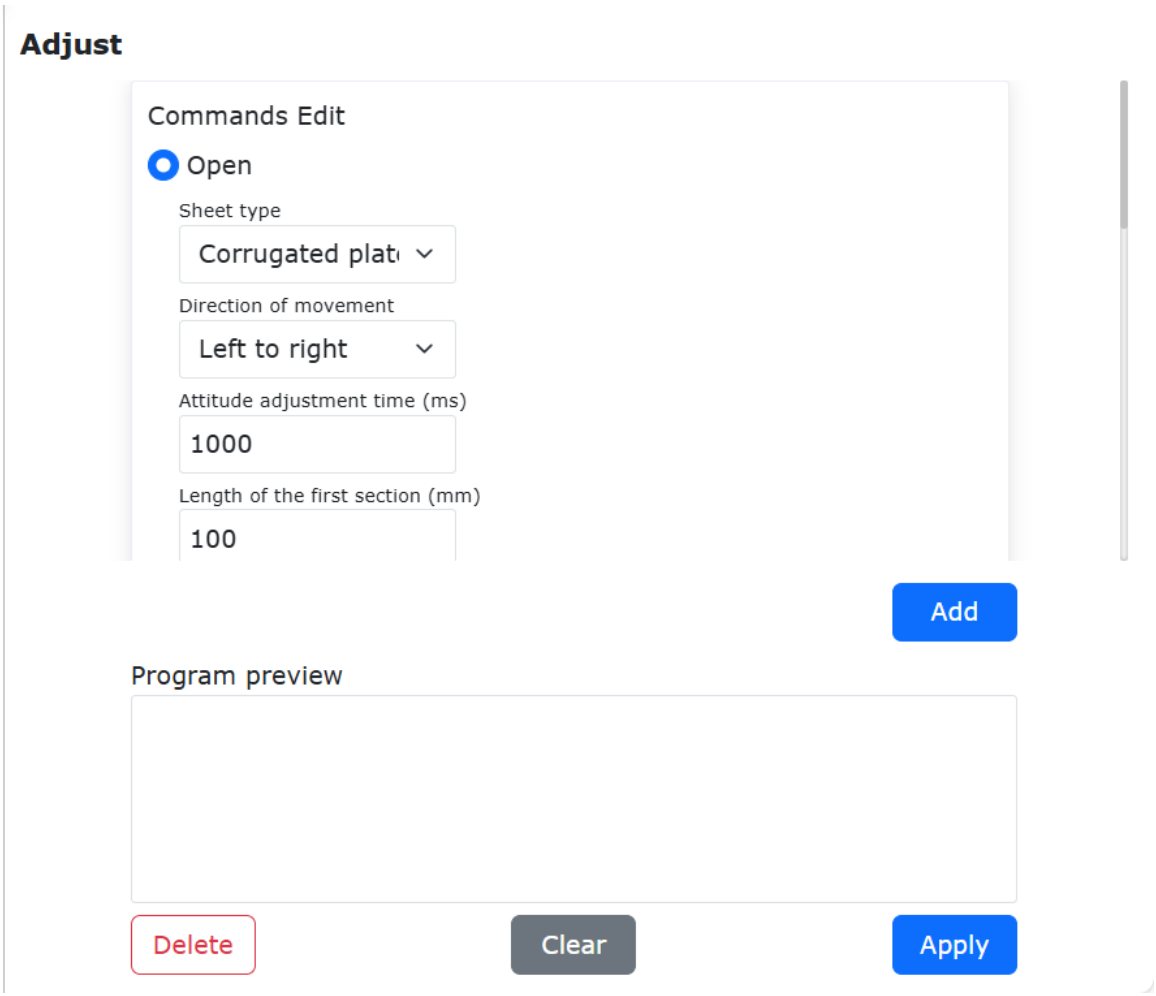


Figure 9.8-8 Adjust command interface

1.3.9.9 Force control command interface



Figure 9.9 Force control command interface

1.3.9.9.1 F/T command

Click the "F/T" icon to enter the F/T command editing interface.

The command includes nine commands: FT_Guard (collision detection), FT_Control (constant force control), FT_Compliance (compliance control), FT_Spiral (spiral insertion), FT_Rot (rotation insertion), FT_Lin (linear insertion), FT_FindSurface (surface positioning), FT_CalCenter (center positioning), FT_Click (click force detection), see the robot peripherals chapter for details.

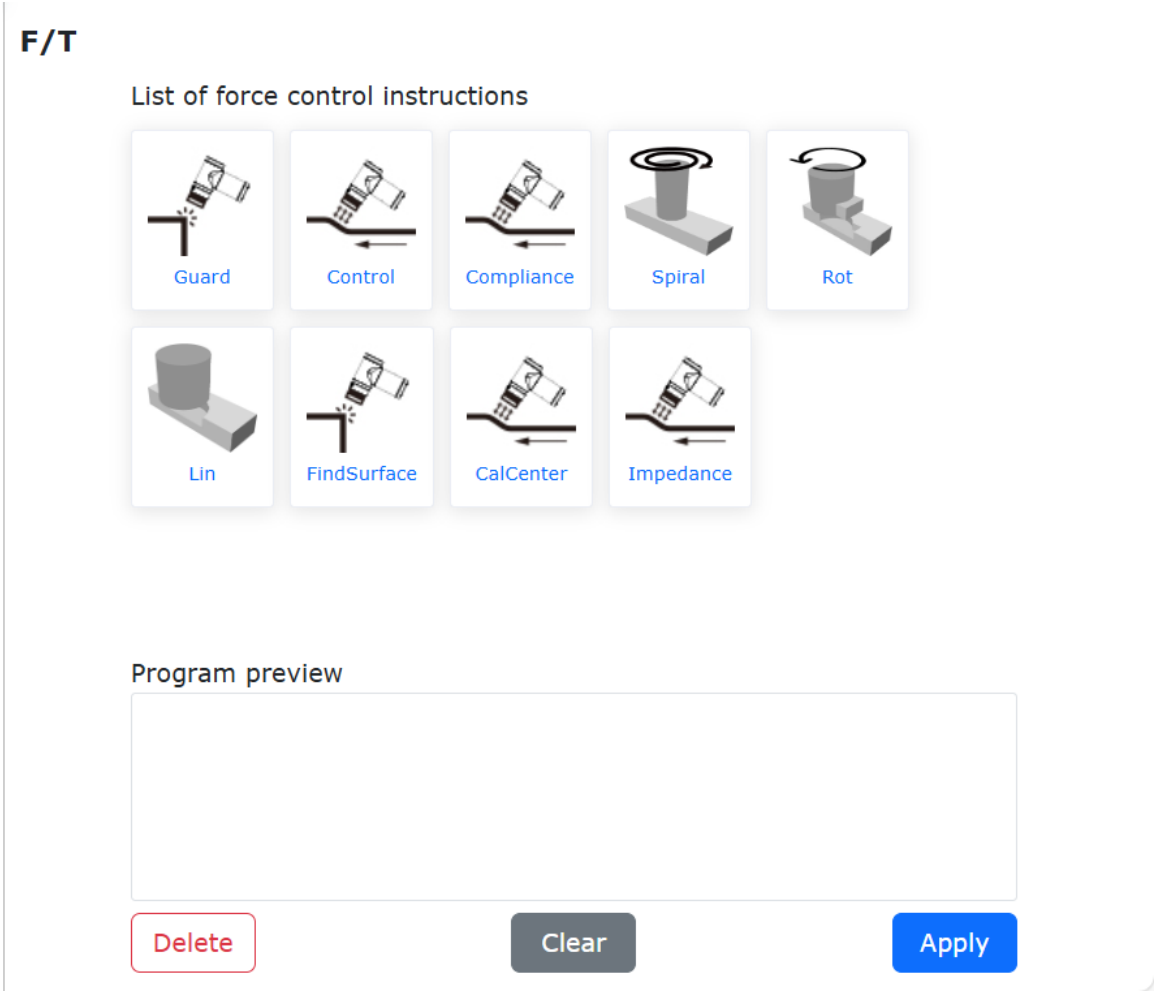


Figure 9.9-1 F/T command interface

1.3.9.9.1.1 Force Control Rotation Insertion Optimization Function

1.3.9.9.1.2 Overview

The force control rotation insertion function is typically used for undertaking rotation insertion actions. Before running the action, the robot end-effector needs to be moved to a fully aligned taught hole position. According to the application scenario, set corresponding motion parameters and an external force detection failure handling strategy. After completion, if the detected external force does not reach the set threshold, the user can independently choose to stop the entire program (function configured as "Error", interface displays red error) or choose to continue motion (function configured as "Warning", interface displays yellow warning).

1.3.9.9.1.3 Operation Process

Step1: Click sequentially on "Teach Program" -> "Program Programming" -> "Force Control Set" -> "Rot" instruction. Set corresponding motion parameters according to the actual application scenario. The external force detection failure handling strategy can be set to "Error" or "Warning". When configured as "Error", if the robot detects that the external force is always less than the set threshold and the set rotation angle has been reached, an error will be reported on the interface and subsequent program execution will stop. When configured as "Warning", if the robot detects that the external force is always less than the set threshold and the set rotation angle has been reached, a warning will be displayed on the interface and subsequent program execution will continue.

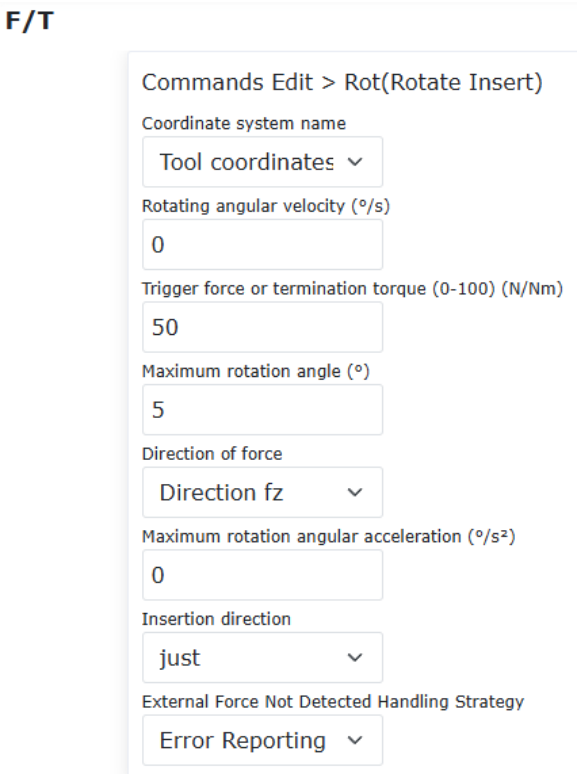


Figure 9.9-2 Force Control Rotation Insertion Parameter Configuration

Step2: The force control rotation insertion function needs to be combined with the "FT_Control" function for motion, with the same motion parameters set. Typical Lua programs with the external force detection failure handling strategy set to "Error" and "Warning" are shown in the figures below respectively.

```
1↔ Lin(P1,100,-1,0,0,0,100,200)
2↔ FT_Control(1,2,0,0,1,0,0,0,0,0,10,0,0,0,0.0001,0,0,0,0,0,0,1000,10,0,0,0,0,0,0,0,0,0,0,0)
3↔ FT_RotInsertion(0,2,2,12,2,5,1,0)
4↔ FT_Control(0,2,0,0,1,0,0,0,0,0,10,0,0,0,0.0001,0,0,0,0,0,0,1000,10,0,0,0,0,0,0,0,0,0,0,0)
5↔ Lin(P2,100,-1,0,0,0,100,200)
```

Figure 9.9-3 Typical Lua Program Configured as "Error"

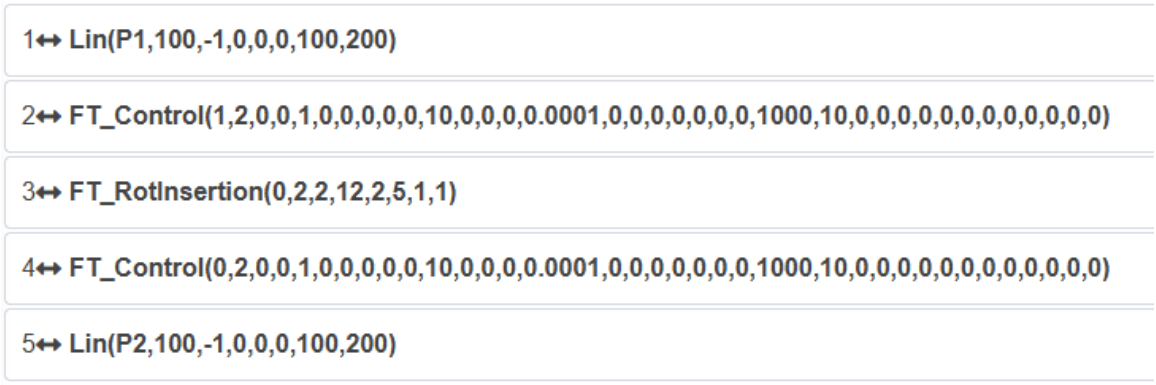


Figure 9.9-4 Typical Lua Program Configured as “Warning”

1.3.9.9.2 Torque command

Click the “Torque” icon to enter the Torque command editing interface.

This command is a torque recording command, which realizes the real-time torque recording collision detection function. Click the “Torque Record Start” button to continuously record the collision situation during the operation of the motion command, and the recorded real-time torque is used as the theoretical value of the collision detection judgment to reduce the probability of false alarms. When the set threshold range is exceeded, the collision detection duration is recorded. Click the “Torque Recording Stop” button to stop recording. Click “Torque Record Reset” to return the status to the default state.

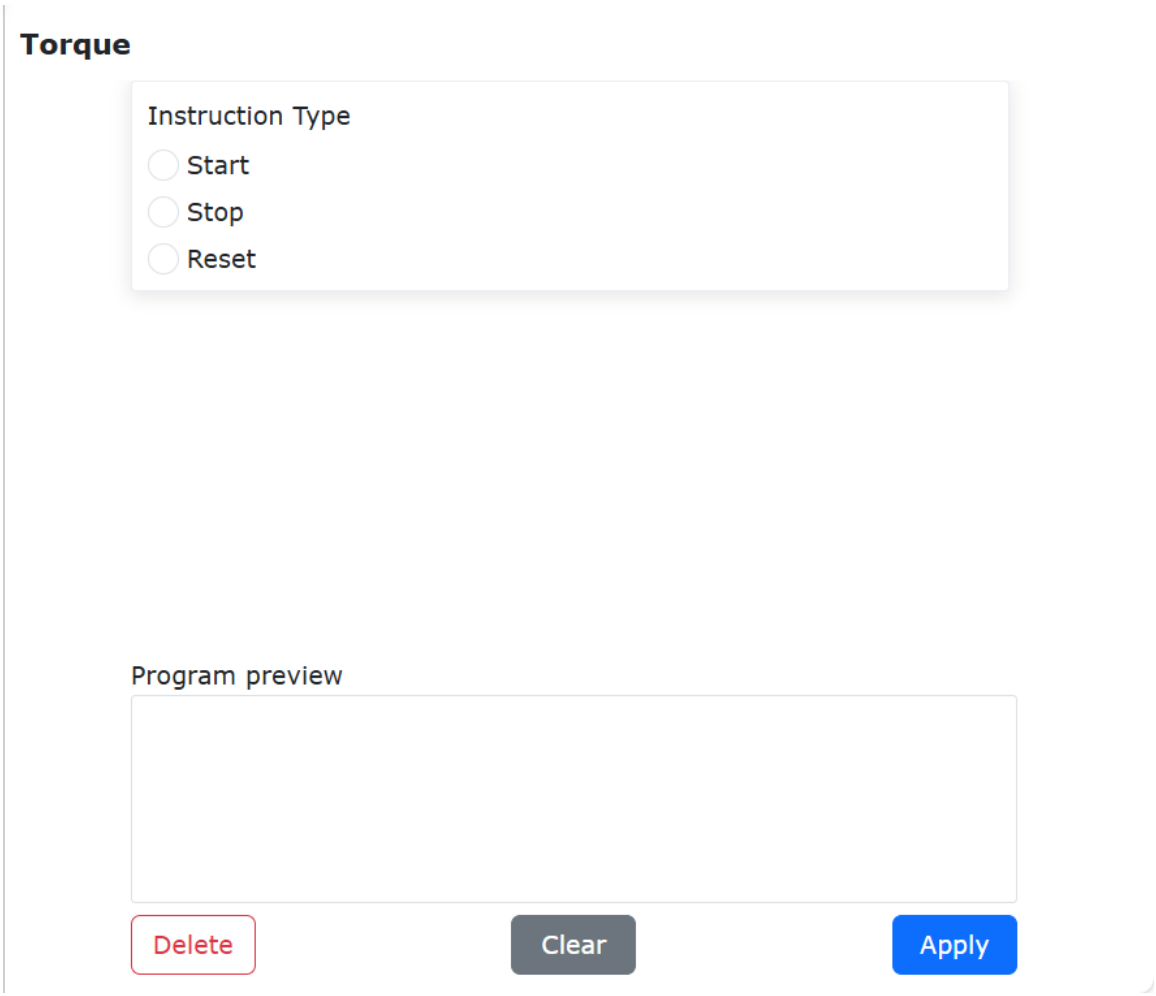


Figure 9.9-2 Torque command interface

1.3.9.10 Visual command interface

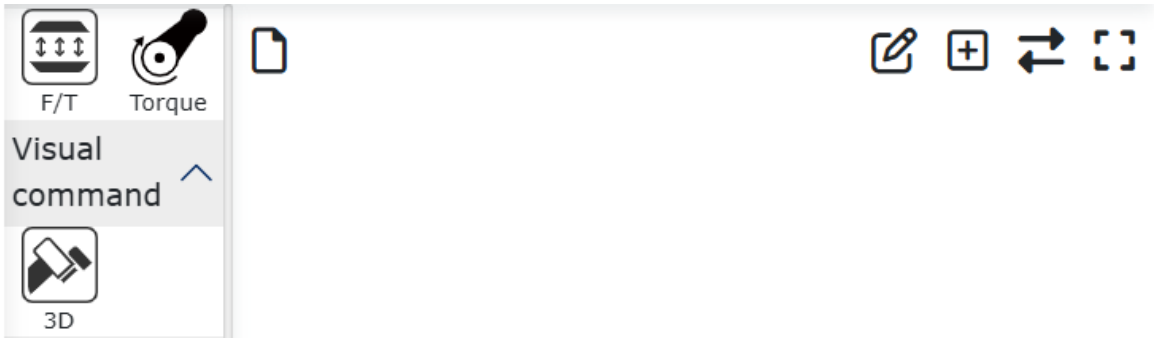


Figure 9.10 Visual command interface

1.3.9.10.1 3D command

Click the “3D” icon to enter the 3D command editing interface.

This command generates commands for 3D vision program examples. Users can refer to the generated programs and communicate with other vision devices, including two program case references of camera calibration and camera capture.

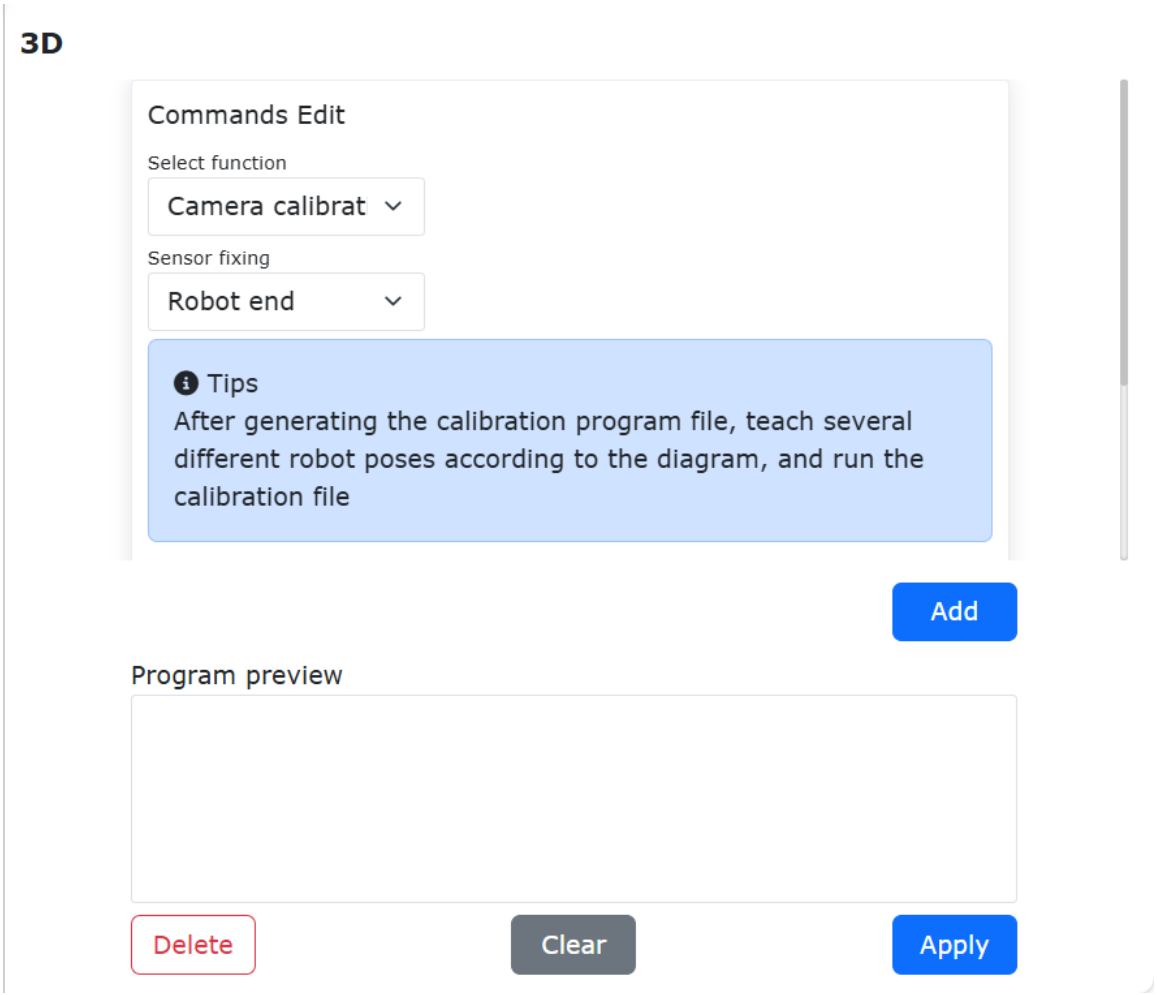


Figure 9.10-1 3D command interface

1.3.9.11 Palletizing command interface



Figure 9.11 Palletizing command interface

1.3.9.11.1 Pallet command

Click the “Pallet” icon to enter the Pallet command editing interface.

This instruction generates instructions for the palletizing program, which is consistent with the matrix movement function in Section 3.9.6. For details, refer to that chapter.

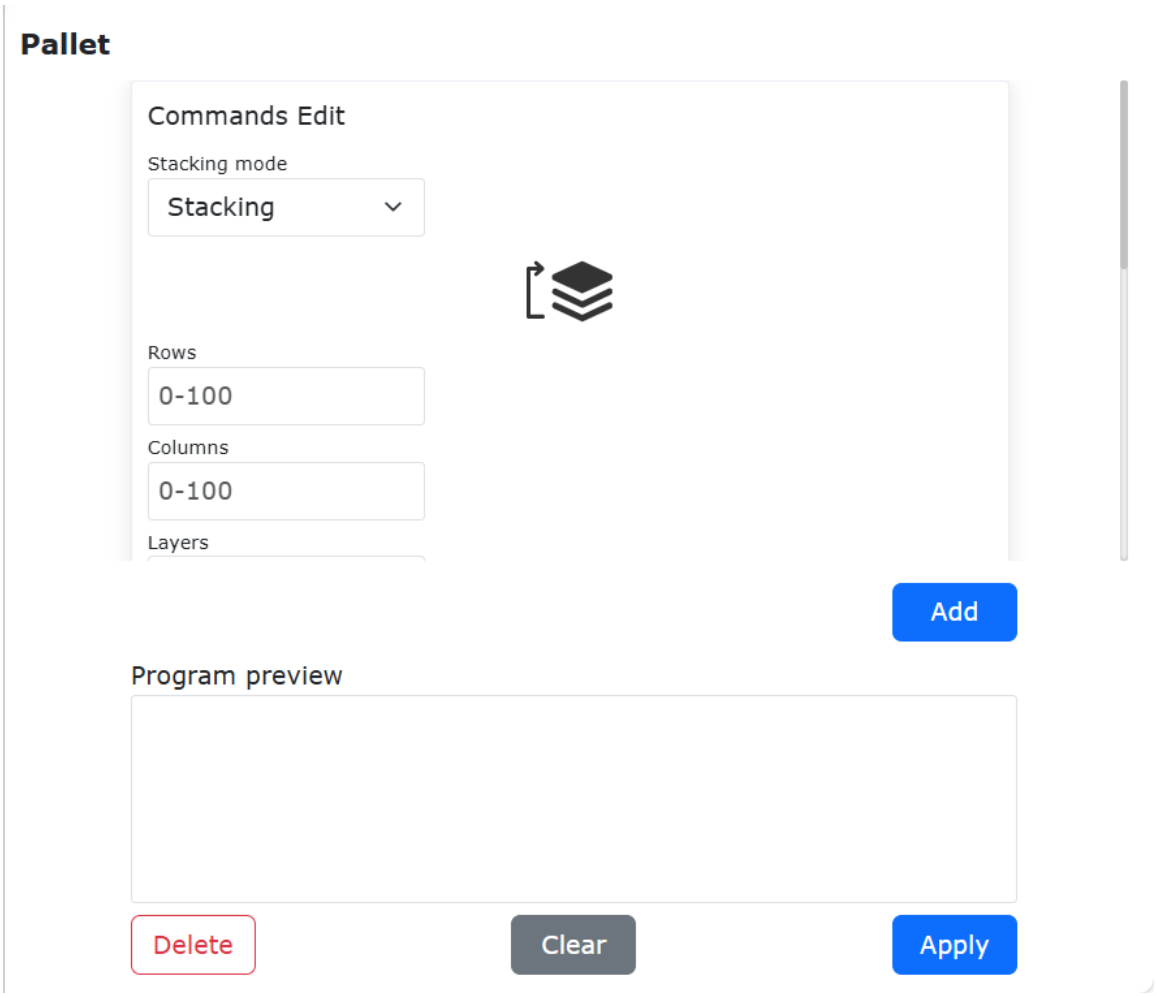


Figure 9.11-1 Pallet command interface

This function controls the regular movement of the manipulator by setting the three-point coordinates and the values of the row and column layer and layer height, which is suitable for common palletizing applications. The first step is to select the robot movement mode, “PTP” or “Line”, the second step is to set the robot movement path, “head-to-tail walking method” or “bow walking method”, the third step is to set the stacking method, “stack stacking” or “unstacking”.

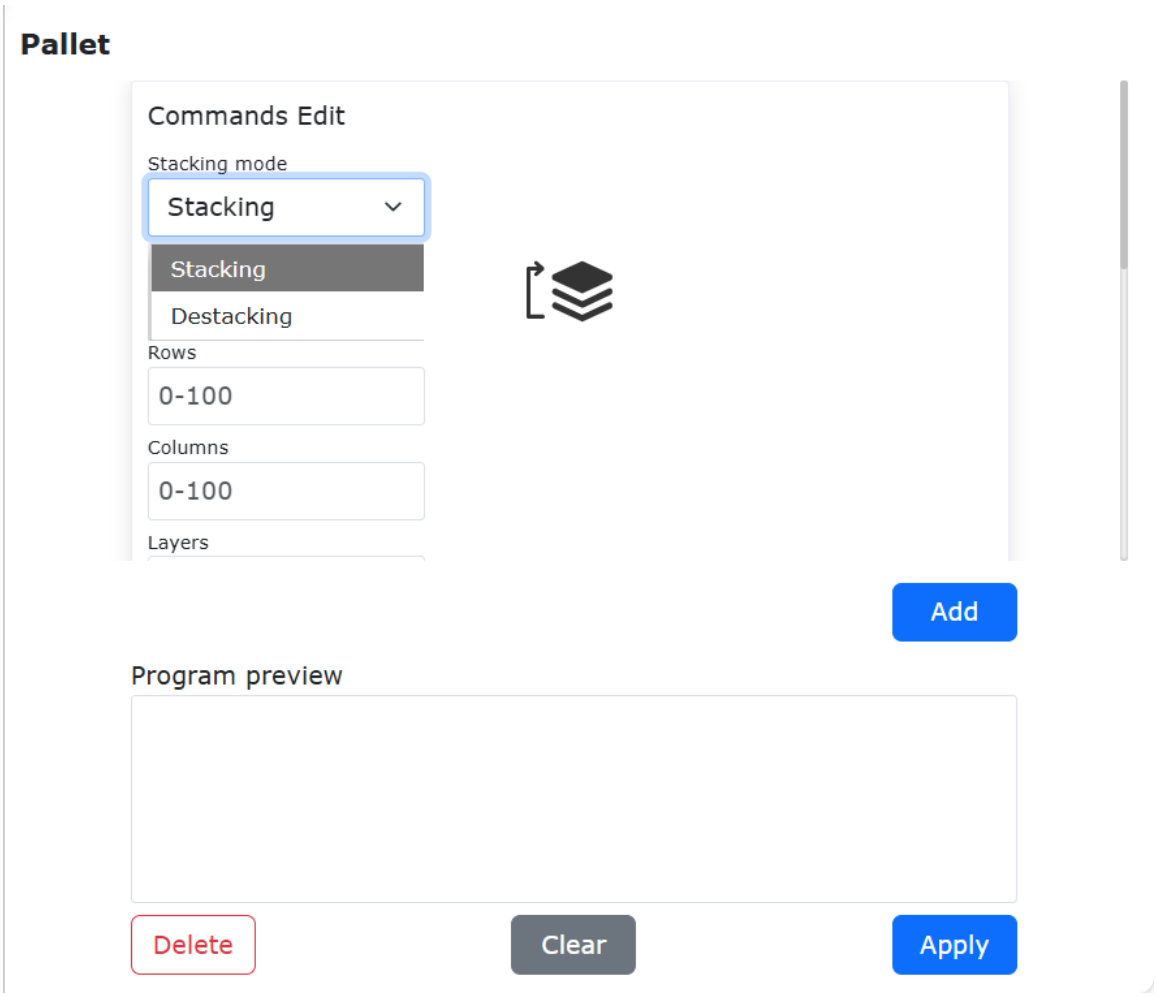


Figure 4.11-2 Matrix move

The fourth step is to teach three points according to the path. The first point is the starting point of the first row, and the arm posture is determined by this point during the whole movement process. The second point is the end point of the first row, and the third point is the end point of the last row. The fifth step is to set the number of rows and columns. The sixth step is to set the number of layers and the height of each layer. The last step is to name the matrix motion program file, and a matrix movement program is generated successfully.

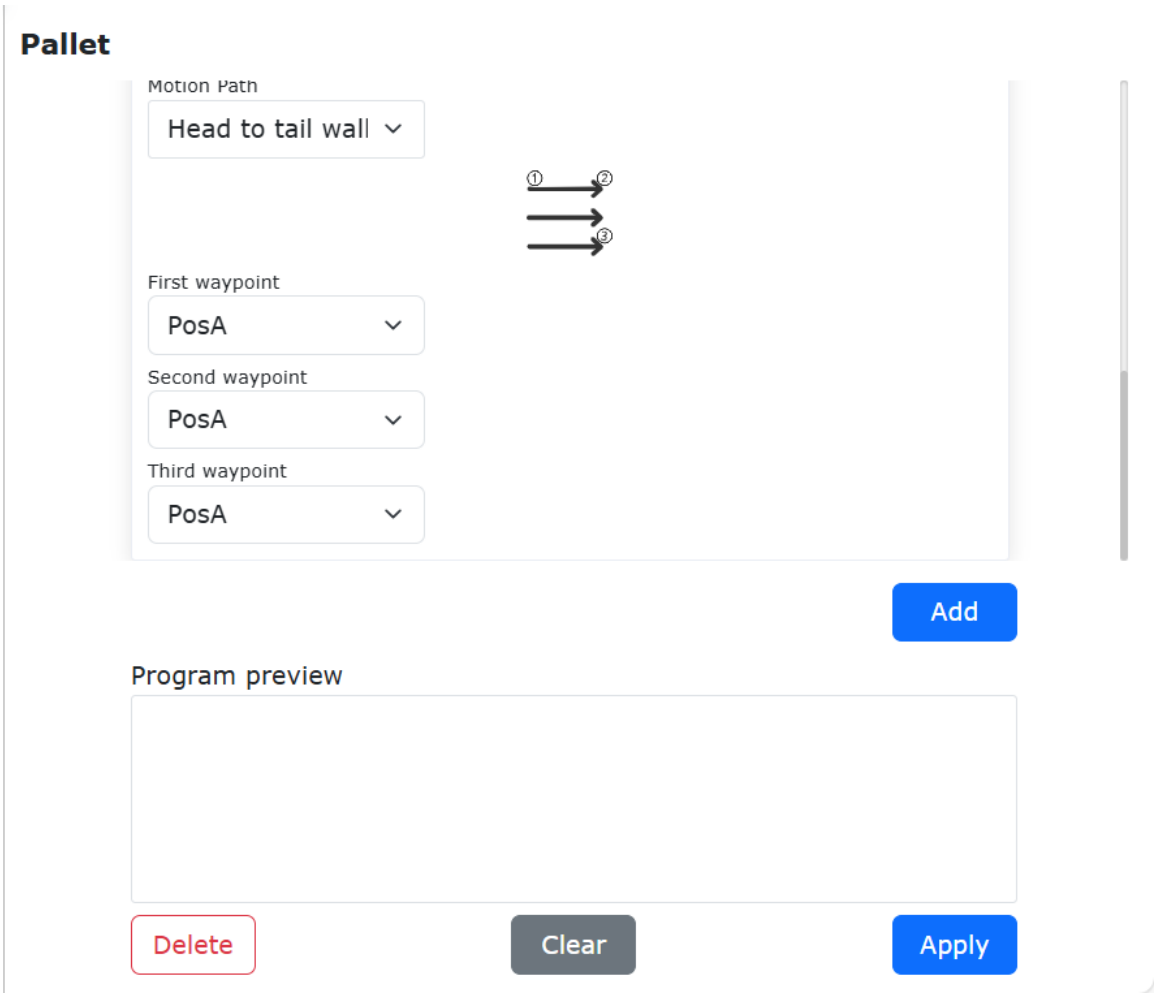


Figure 4.11-3 Matrix move

1.3.9.12 Communication command interface

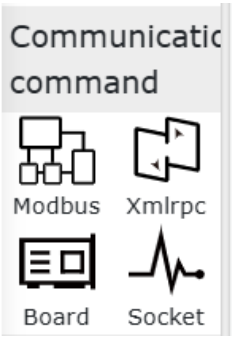


Figure 9.12 Communication command interface

1.3.9.12.1 Modbus command

Click the “Mobus” icon to enter the Modbus command editing interface.

The command function is a bus function based on the ModbusTCP protocol. The user can control the robot to communicate with the ModbusTCP client or server (the master station communicates with the slave station) through relevant instructions, and perform read and write operations on coils, discrete quantities, and registers.

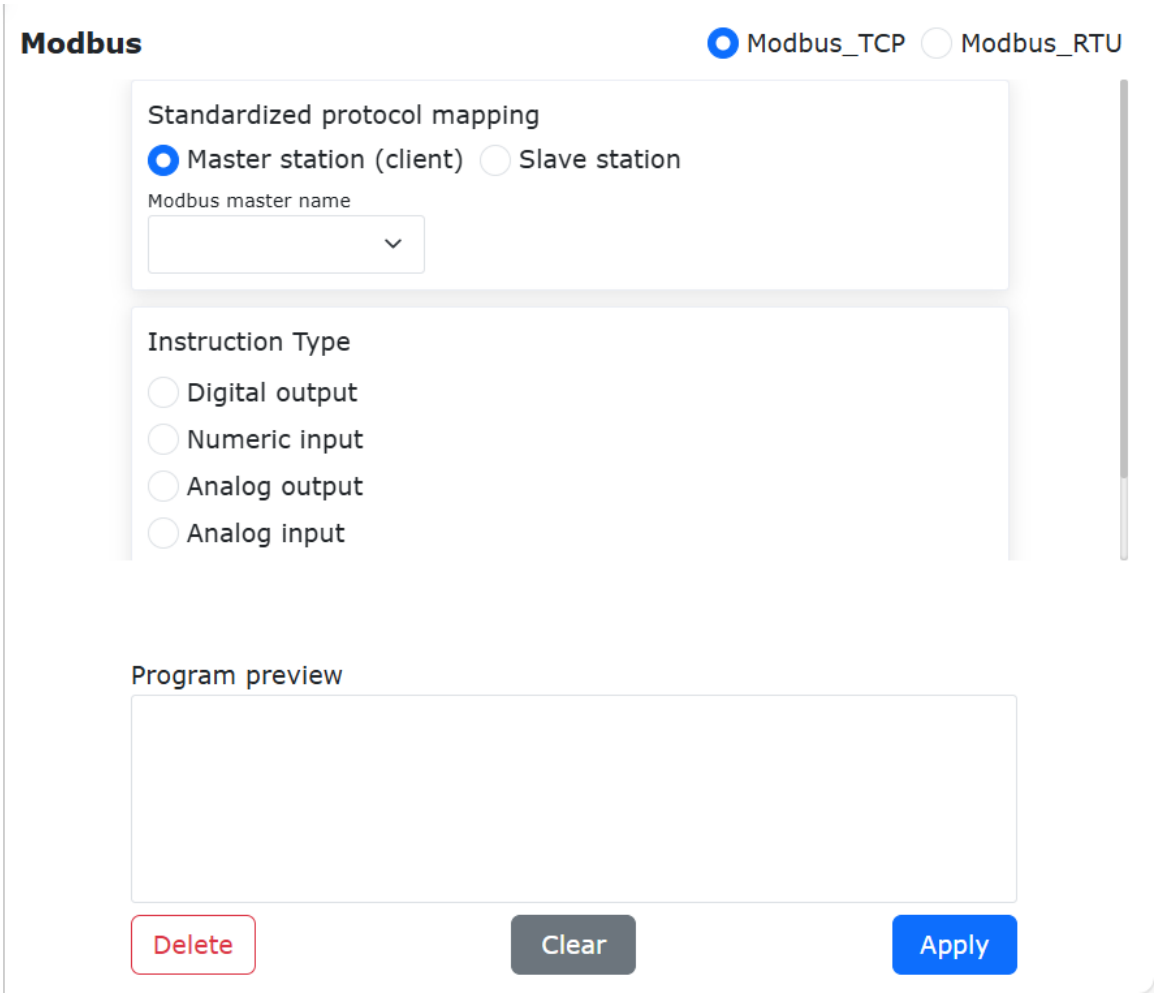


Figure 9.12-1 modbus command master interface

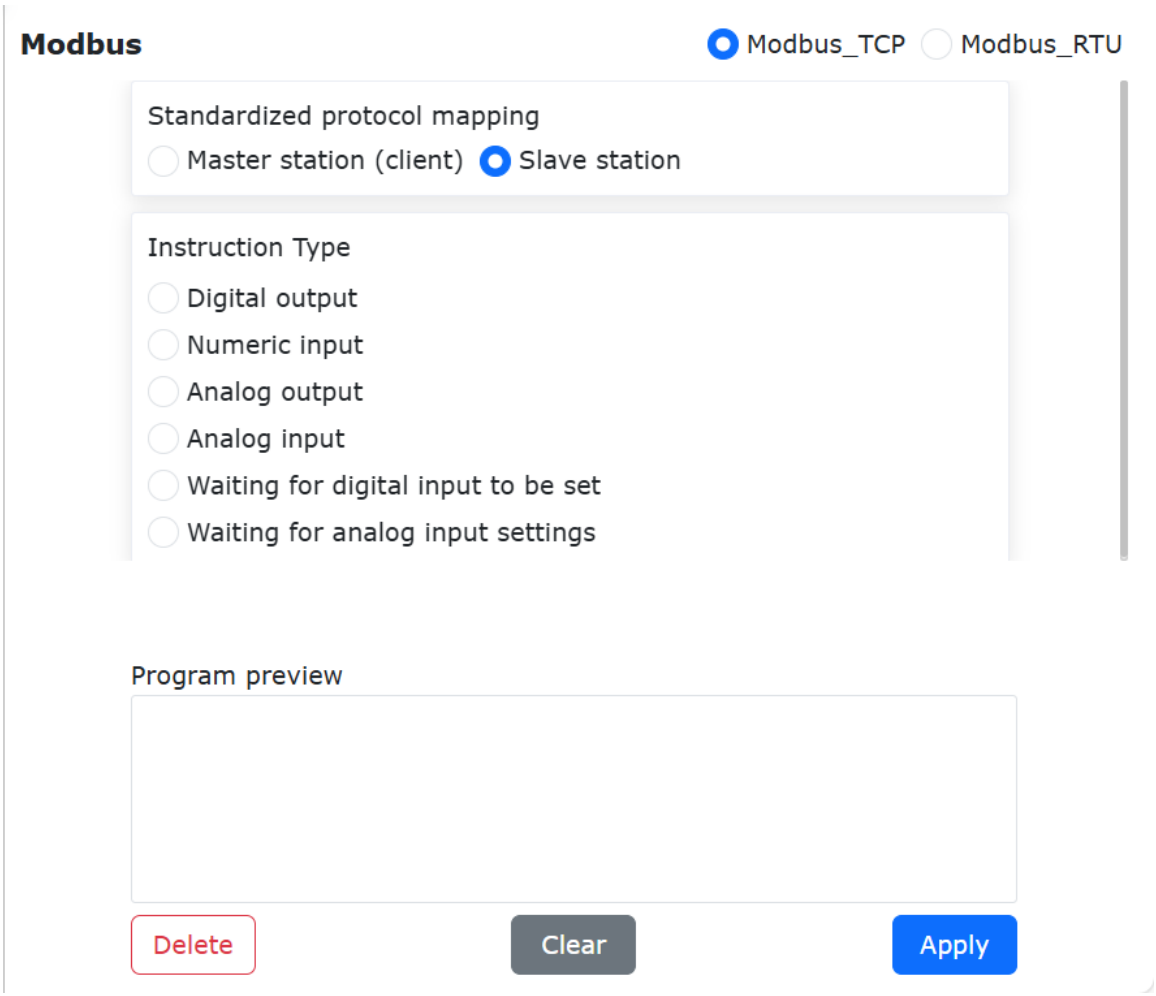


Figure 9.12-2 modbus command slave interface

For more operating functions of ModbusTCP, please contact us for consultation.

1.3.9.12.2 Xmlrpc command

Click the “Xmlroc” icon to enter the Xmlrpc command editing interface.

XML-RPC is a remote procedure call method for transferring data between programs using xml over sockets. In this way, the robot controller can call the function (with parameters) in the remote program/service and obtain the returned structured data. The Robot Controller handles all the details of writing XML-RPC client messages and handling conversions between data types and XML.

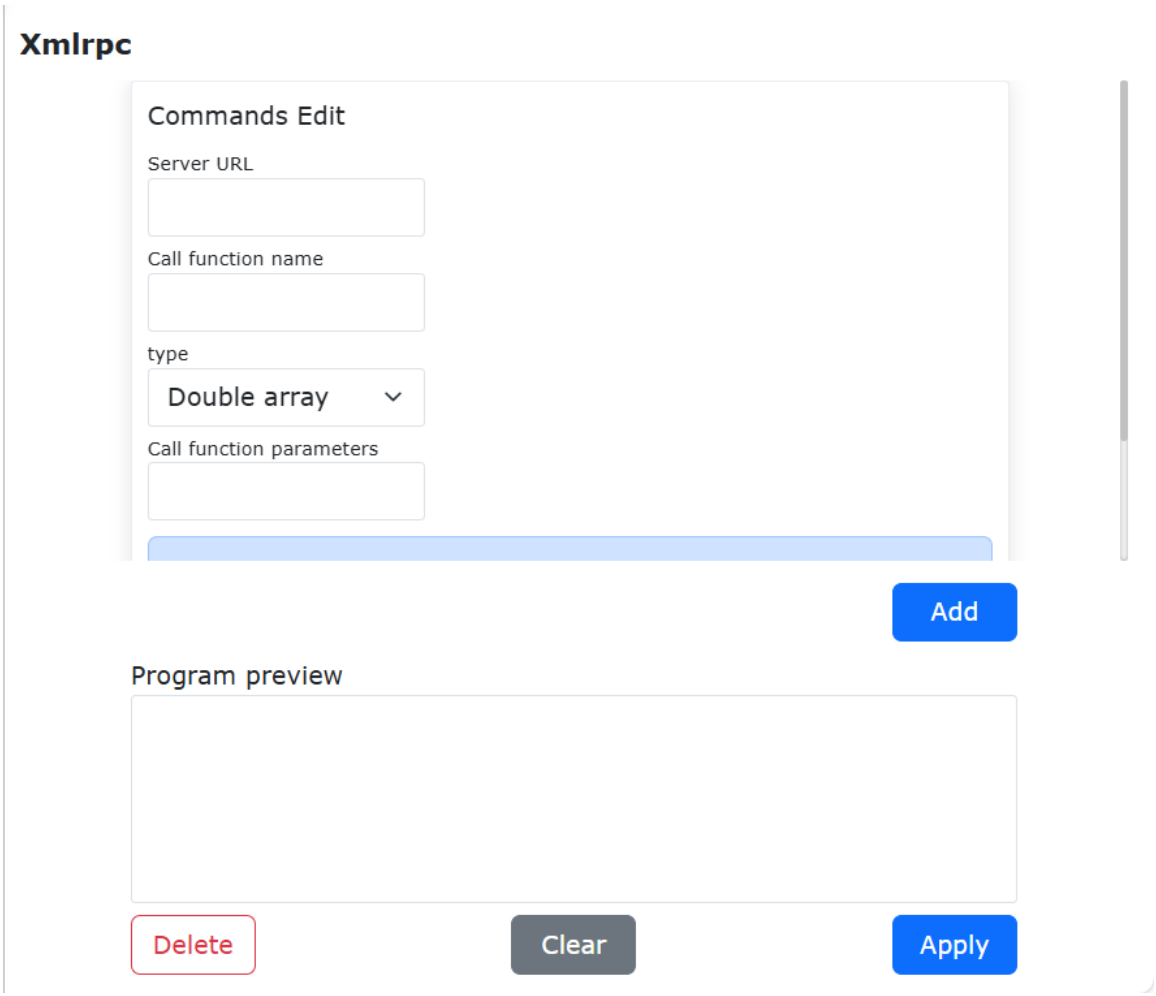


Figure 9.12-3 Xmlrpc command interface

Important:

1) The controller acts as a client to connect to the remote custom port;
2) The controller acts as a client to call the remote function;
3) Support calling different remote functions;
4) Support string array parameter input and character array result return, the number of array elements can be customized;

Support double-type array parameter input and double-type array result return, the number of array elements can be customized;

```
XmlrpcClientCall(serverUrl,methodName,tableType,param)

serverUrl server url, for example:"http://192.168.58.29:50000/RPC2"

methodName Call function name, "example.add"

tableType 1-double array, 2-string array
```

(continues on next page)

(continued from previous page)

```
param call function parameters
```

```
XmlrpcClientCall(error, result)

error 0 - no error, 1 - error

result If the parameter is passed in as a double array, the result is a double array,

If the parameter is passed in as an array of string type, the result will be an array of␣
↪string type
```

1.3.9.13 Auxiliary command interface

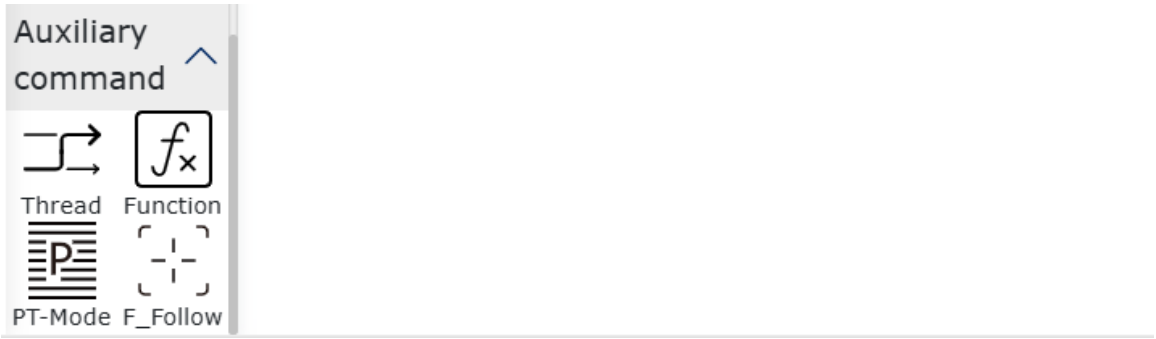


Figure 9.13 Auxiliary command interface

1.3.9.13.1 Thread command

Click the "Thread" icon to enter the Thread command editing interface.

The Thread command is an auxiliary thread function. Users can define an auxiliary thread to run simultaneously with the main thread. The auxiliary thread mainly performs data interaction with external devices, supports socket communication, obtains robot DI status, robot DO status settings, and obtains robot status information. Thread data interaction, the data obtained by the main thread through the auxiliary thread is used to control the judgment of the robot's motion logic, the screenshot of the user program example:

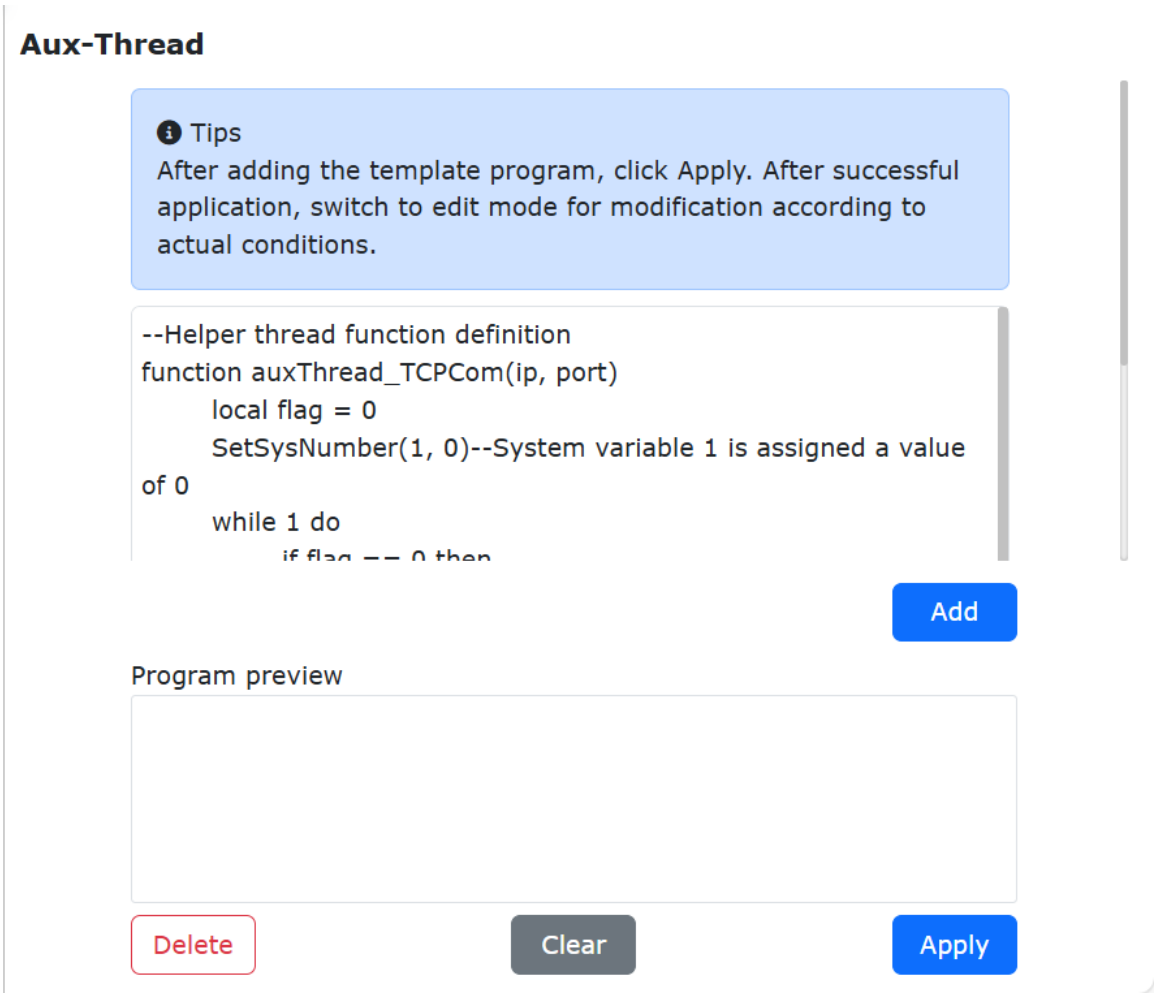


Figure 9.13-1 Thread program example

1.3.9.13.2 Function command

Click the “Function” icon to enter the Function command editing interface.

This command is to call the function interface function, provide the robot interface function to the customer to choose, and remind the customer of the parameters required by the function, which is convenient for the customer to write script commands, and more functions are being added one after another.

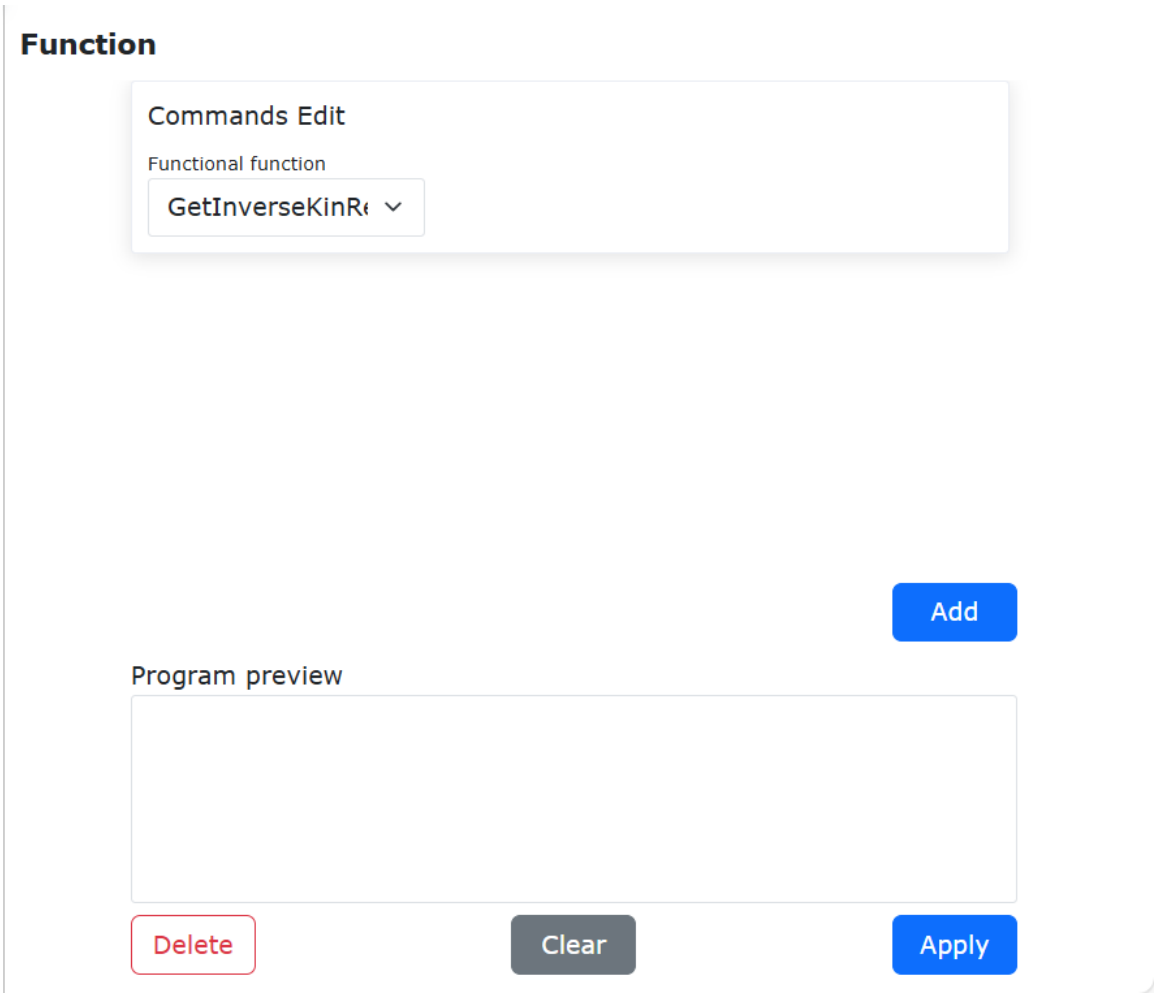


Figure 9.13-2 Function command interface

1.3.9.13.3 PT-Mode command

Click the “PT-Mode” icon to enter the PT-Mode command editing interface.

This command is mainly used for mode switching between system mode and point table mode. Teaching points in different point tables can be applied by switching point tables.For details, see Chapter 11 - Points.

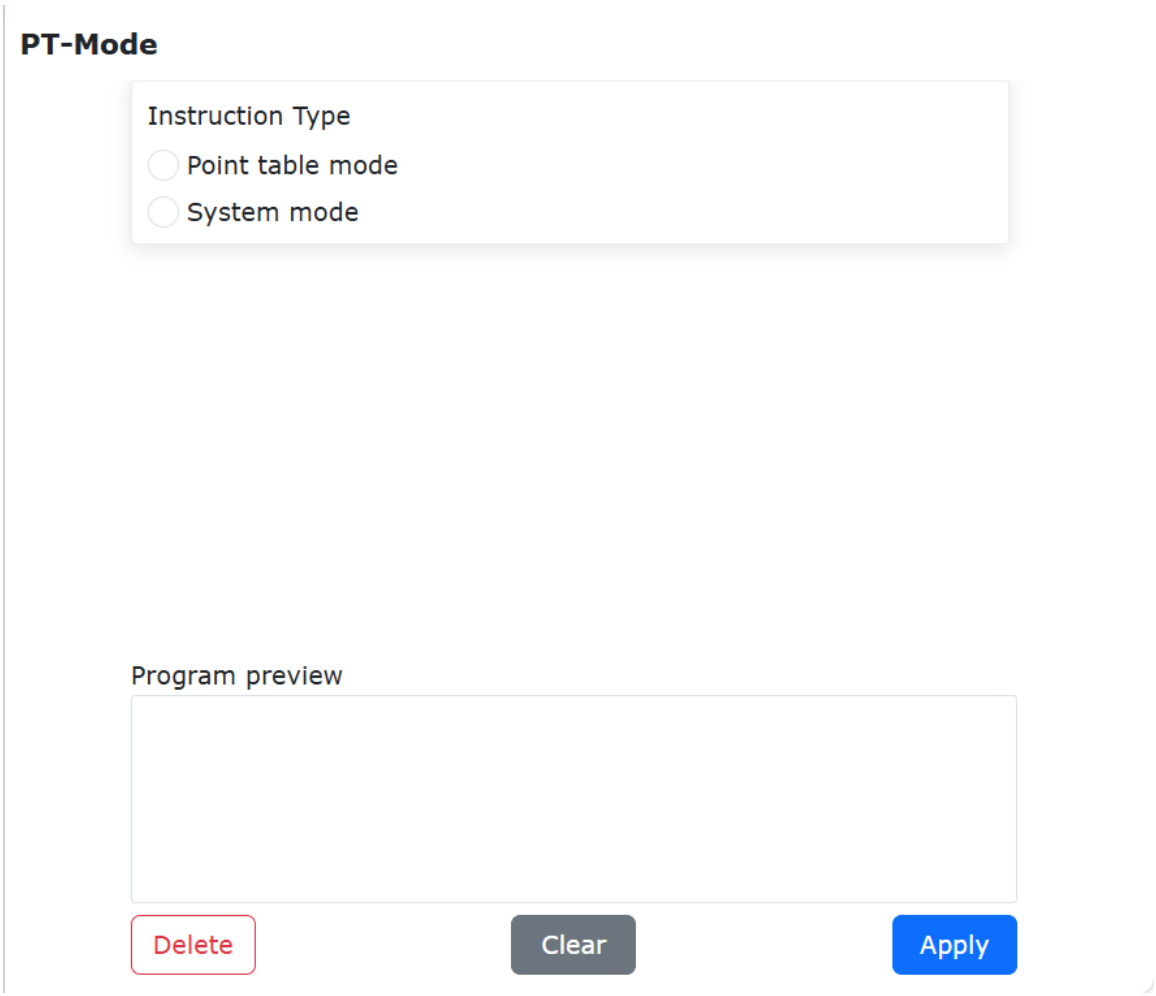


Figure 9.13-3 PT-Mode command interface

1.3.9.14 Teaching program is not saved for verification

On the program teaching page, after opening/creating a new program, if the teaching program is modified, the program is not saved.

If you click “Open”, “New”, “Export”, “Rename” and other related file operations, the “Do you want to save this program” pop-up box will be triggered, prompting “The current program has changed. Do you want to save the changes to this program?” ,As shown below.

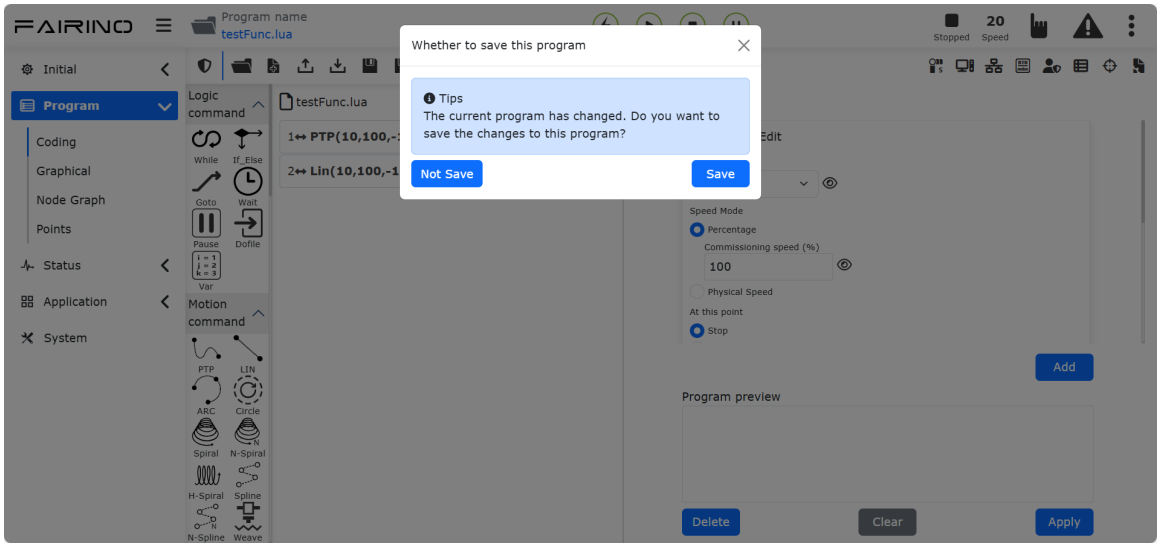


Figure 9.14-1 The current page program does not save verification

Step1:Click the “Not Save” button, and the program will restore the unmodified data and continue to perform previous related file operations.

Step2:Click the “Save” button, the unsaved Lua program is saved successfully, and the previous related file operations continue to be performed.

If you leave the program teaching page and switch to other pages, the prompt “Do you want to save this program” will also be triggered, and you will still stay on the current teaching program page, as shown below.

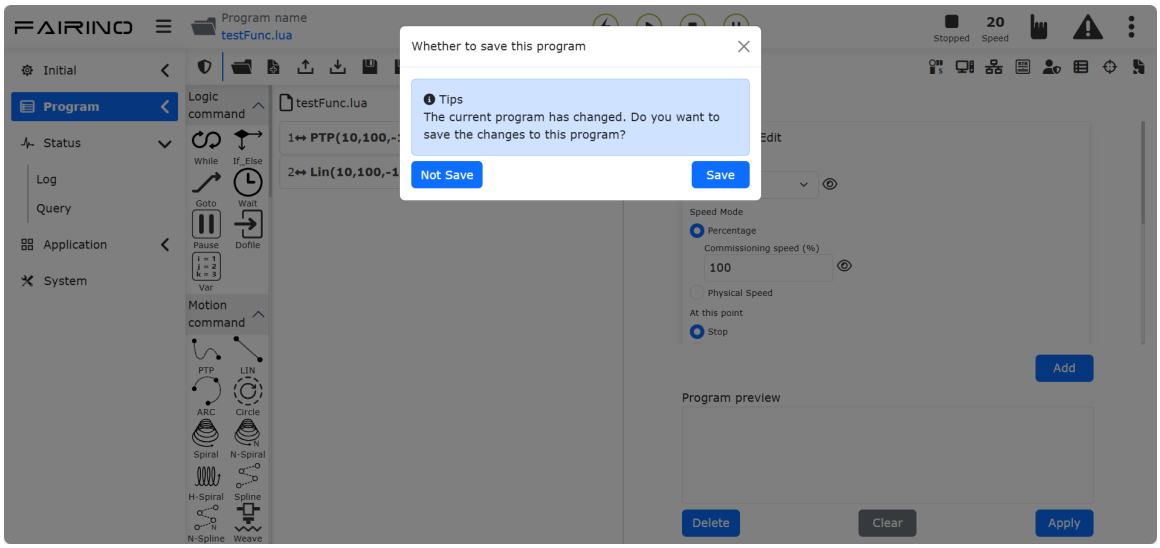


Figure 9.14-2 Switch page program does not save verification

Step1:Click the “Not Save” button to jump to the previously selected page.

Step2:Click the “Save” button, and the unsaved Lua program is saved successfully and jumps to the previously selected page.

1.3.9.15 Teaching program encryption

The teaching procedure is divided into a state of encryption and non-encryption. The encryption level is divided into first-level encryption and secondary encryption. Among them, the level of first-level encryption is the highest, and the secondary is second. All teaching programs are displayed and set in the form of program encryption information in “System Settings - Custom Information”. Encryption level descriptions are provided to the right of the table.

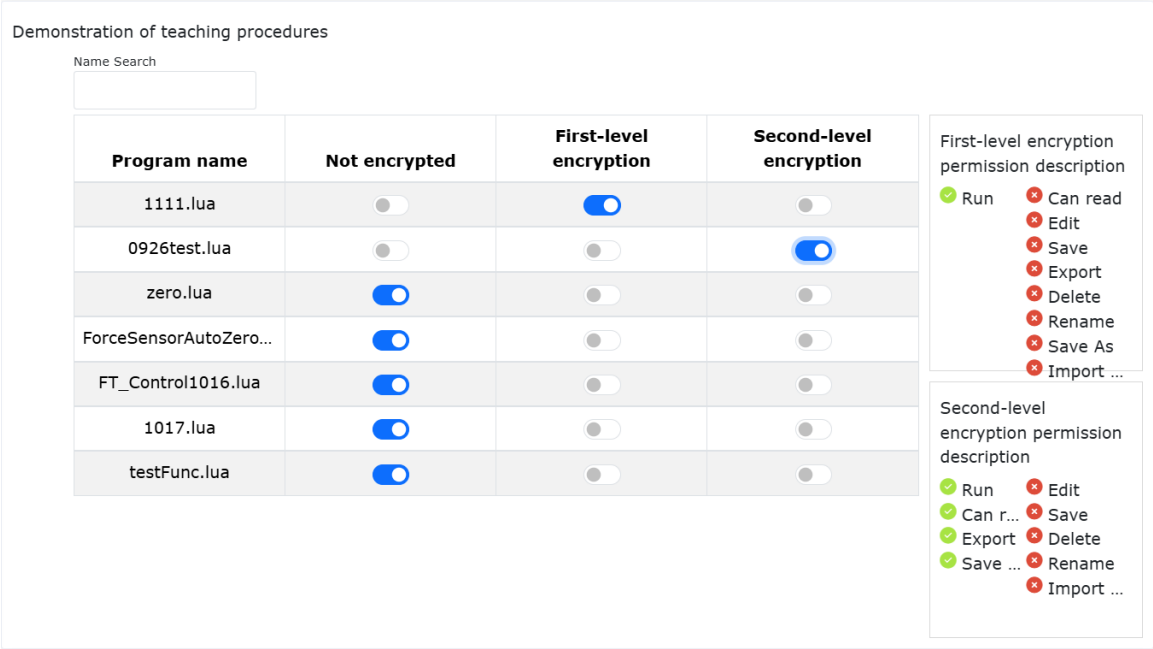


Figure 9.15-1 Demonstration of teaching procedures

When the program is a first-level encryption state, after opening the program: the corresponding “export”, “preservation”, “existing as”, “copy”, “cut”, “delete”, “delete”, “delete”, “delete”, “delete” The buttons such as “upward”, “downward” and “editing mode switching” will be grayed. Click the icon to be invalid and it will prompt that the current program is in an encrypted state. The program “renamed” icon will hide. Add instruction bars and program editing areas are invisible and prompts to be locked in first-level encryption.

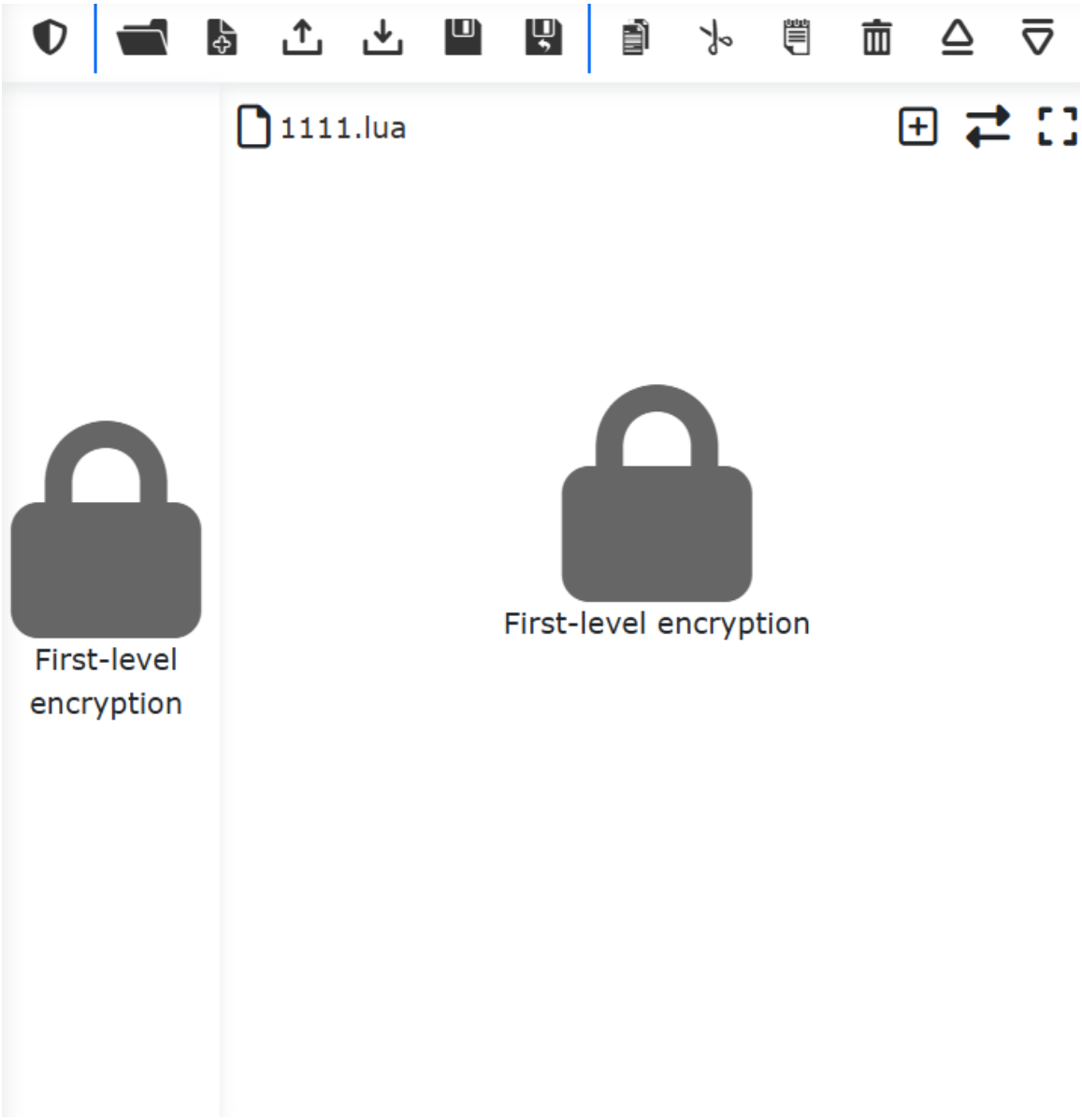


Figure 9.15-2 Program first-level encryption interface

When the program is second-level encryption, after opening the program on the “Program Demonstration” page: the corresponding “savings”, “copy”, “shear”, “paste”, “delete”, “upper”, “upper” in the operating bar The buttons such as the “Move” will turn ashes. Click the icon to be invalid and it will prompt that the current program is encrypted. The program “renamed” icon will hide. The adding instruction bar is not visible and prompts to be locked in a secondary encryption. The program editing area can browse the reading program normally.

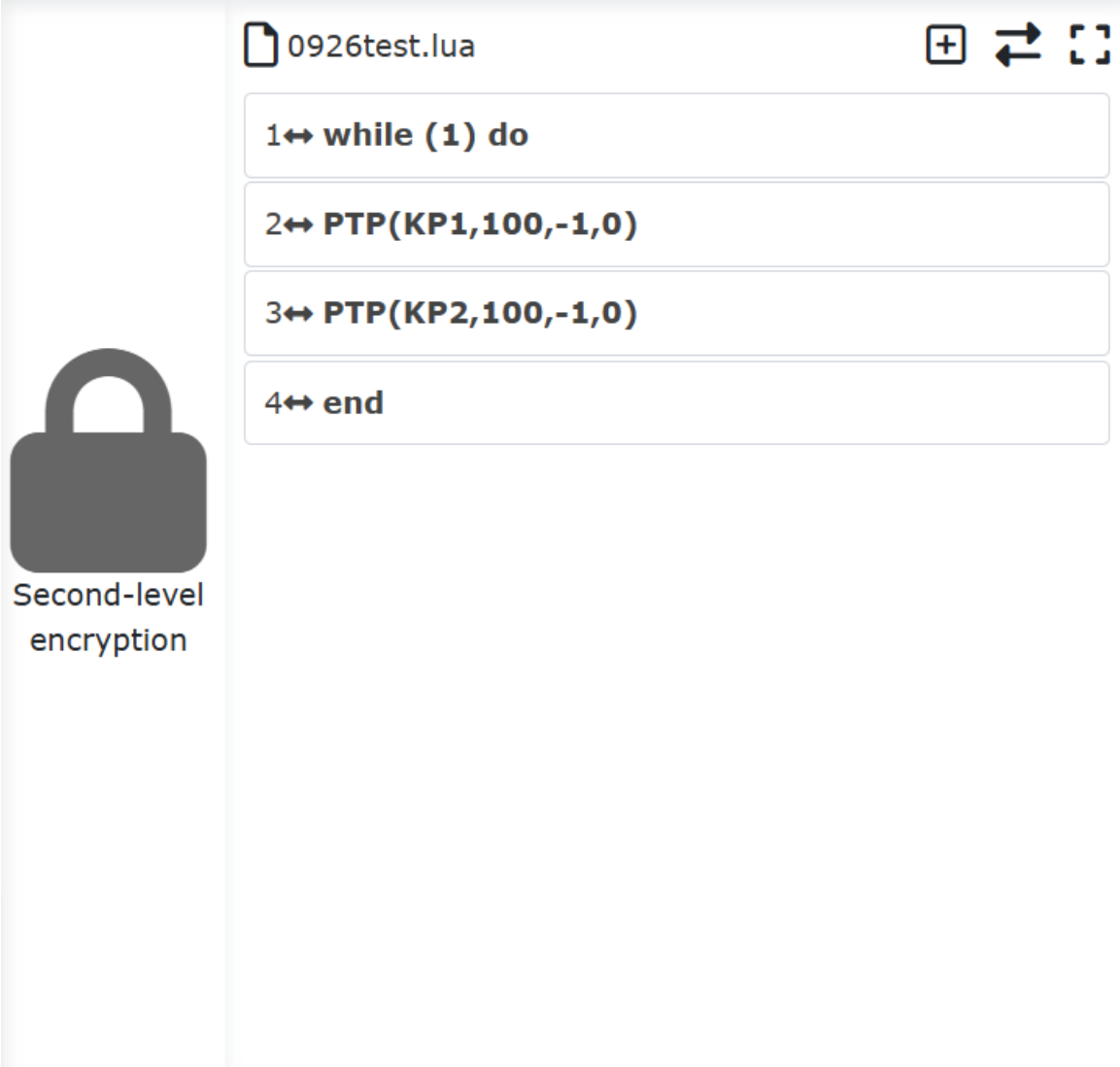


Figure 9.15-3 Program second-level encryption interface

Both first -level encryption and second -level encryption can use the “export” function. Verification operations will be performed when importing. If the program of the same name is an encrypted file, the import operation will be interrupted and indicated that the coverage of the encryption program cannot be introduced.

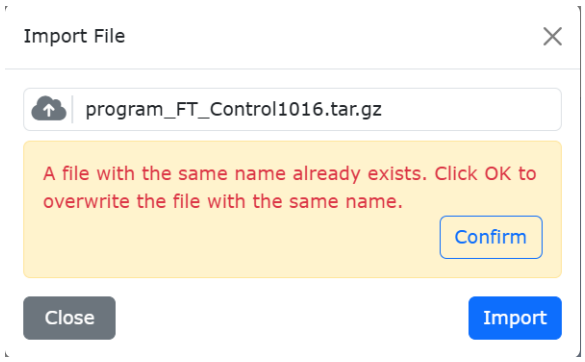


Figure 9.15-4 Program import

1.3.9.16 Local teaching point

The local teaching point is bound to the current teaching program. When adding a program command, it can only be applied to the current teaching program and cannot be used in other teaching programs.

Add: Click the "Add Local Teaching Point" icon on the far right of the program file name to add local teaching points. (For detailed records of local teaching points, please refer to the teaching point records in robot operation)

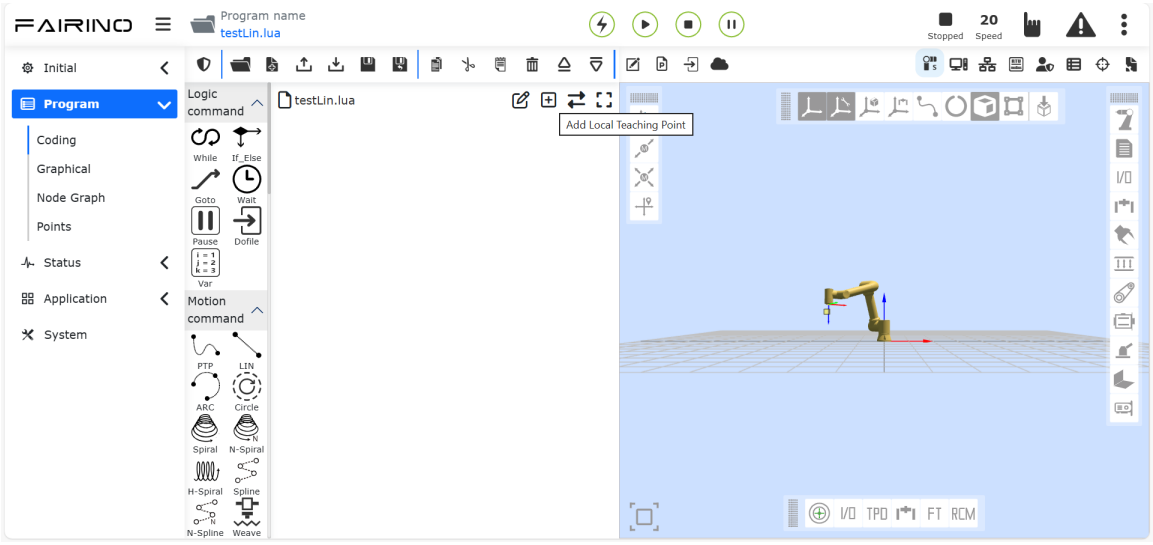


Figure 9.16-1 Add local teaching points

Delete: Click the serial number column of the table to select the local teaching point to be deleted, and then click the "Delete" icon in the upper right corner of the title of the local teaching point to delete the local teaching point.

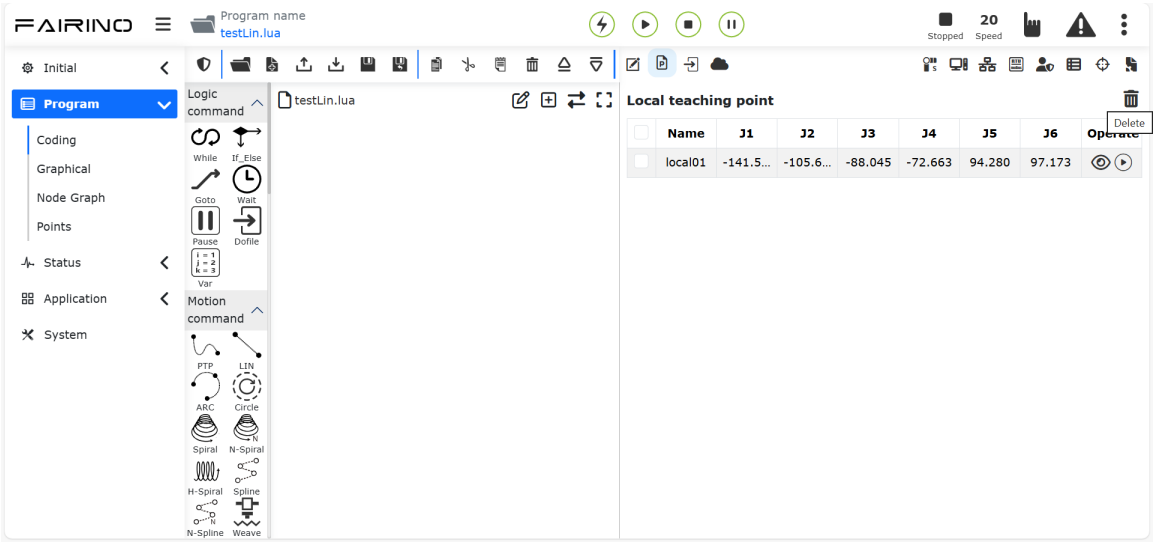


Figure 9.16-2 Delete local teaching point

Run: Click the "Start Running" icon in the data operation bar of the local teaching point table to perform a single-point operation of the local teaching point and move the robot to the position of this point.

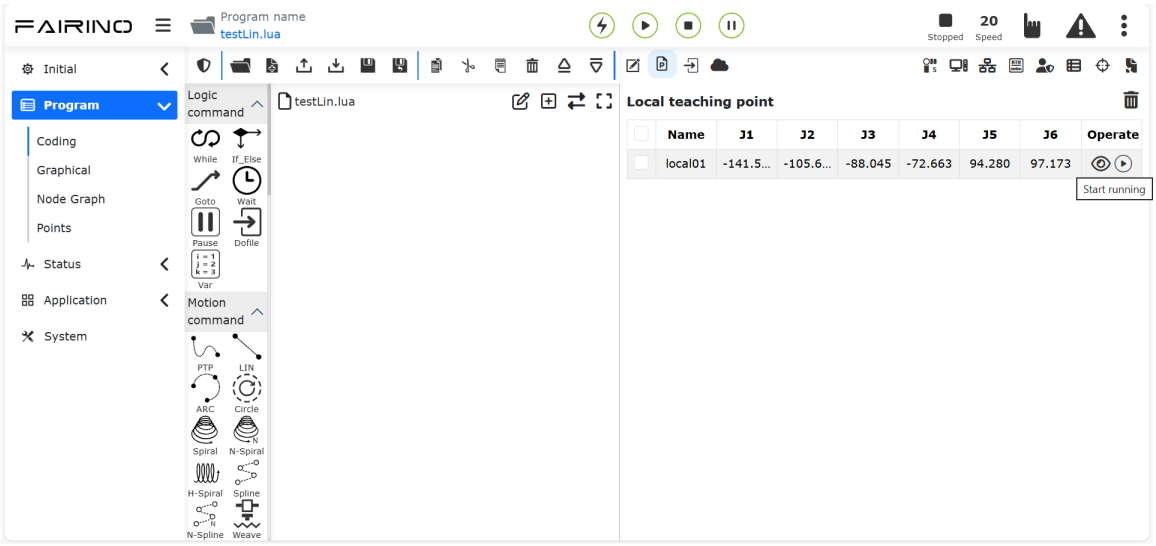


Figure 9.16-3 Run local teaching point

Details: Click the “Details” icon in the data operation bar of the local teaching point table to view the details of the local teaching point.

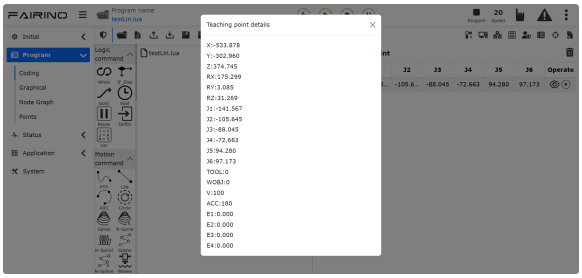


Figure 9.16-4 Local teaching point details

1.3.9.17 Current program backup

After the user modifies the teaching program and clicks save, the “backup” function of the current program is triggered (the backup time is 1 year), and the initial content of the current program is saved and displayed on the right side, which is convenient for the user to compare the modified content. Users can view the corresponding program backup content by selecting a date, and click the “Delete” icon in the upper right corner to delete the current program backup content. The content of the current program backup can only be viewed, not modified.

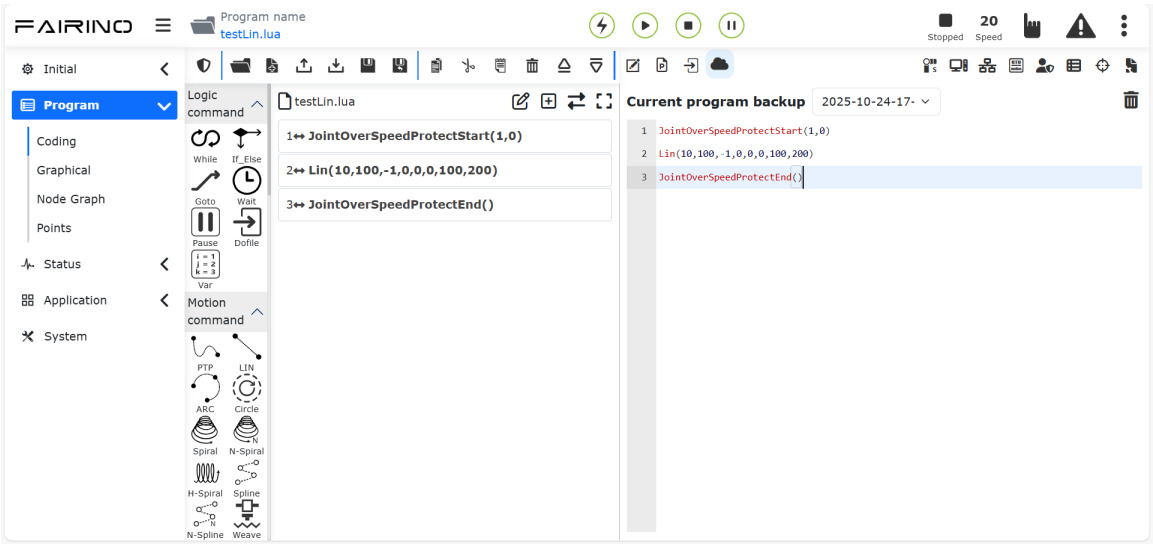


Figure 9.17 Current program backup

1.3.9.18 Modbus TCP Communication

ModbusTCP is a commonly used communication protocol in industrial production. Faao collaborative robots provide two ways to communicate with your device: ModbusTCP master and ModbusTCP slave.

The collaborative robot supports up to 8 ModbusTCP masters to communicate with external devices at the same time, and each master supports up to 128 registers; the collaborative robot ModbusTCP slave has 128 coils, 128 discrete inputs, 64 holding registers and 64 input registers (holding registers and input register data types include unsigned, signed and floating point types). At the same time, some ModbusTCP slave input register addresses of collaborative robots are dedicated to feedback information such as the current robot's joint position and movement speed, and some coil register addresses are dedicated to controlling the robot to start the program, stop the program, set the control box DO and other functions.

The robot ModbusTCP slave only supports establishing a connection with one master station. The robot can communicate with different devices as a master and a slave at the same time. The following is a detailed usage method.

1.3.9.18.1 ModbusTCP master

Before using the cooperative robot as the ModbusTCP master station to communicate with your equipment, please check the network connection between your equipment and the robot, and confirm that the network interfaces are in the same network segment.

There are several steps to use the robot ModbusTCP master station:

- Add the master station;
- Add registers;
- Communication test;
- Writing user programs;
- Run user programs.

1.3.9.18.1.1 Add ModbusTCP master

Open the WebApp, click “Teaching” and “Program Teaching” in turn to create a new user program “testModbusMaster.lua”.

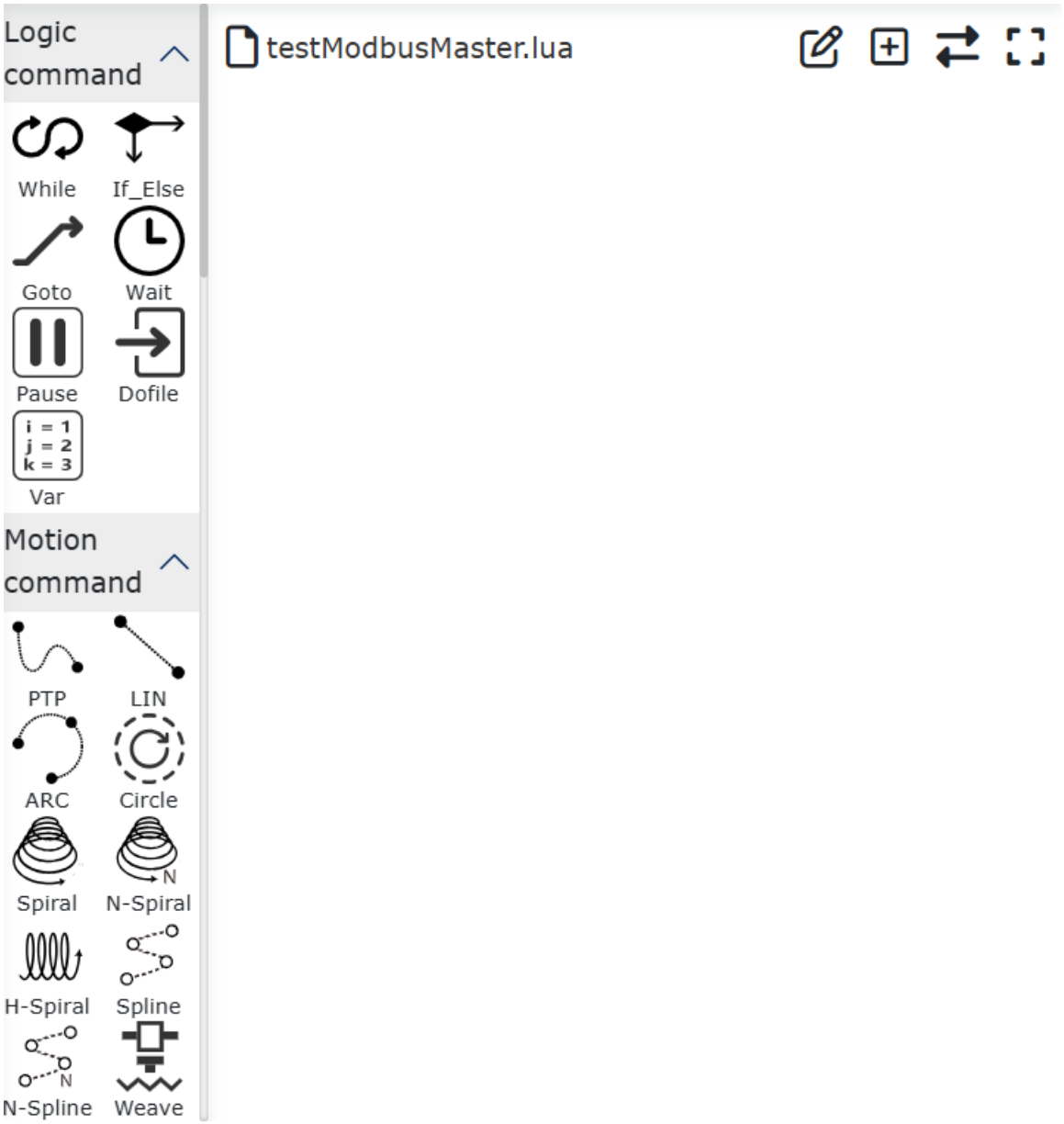


Figure 9.18-1 Create ModbusTCP master station user program

Click “ModbusTCP Settings” button to open the ModbusTCP function configuration page.

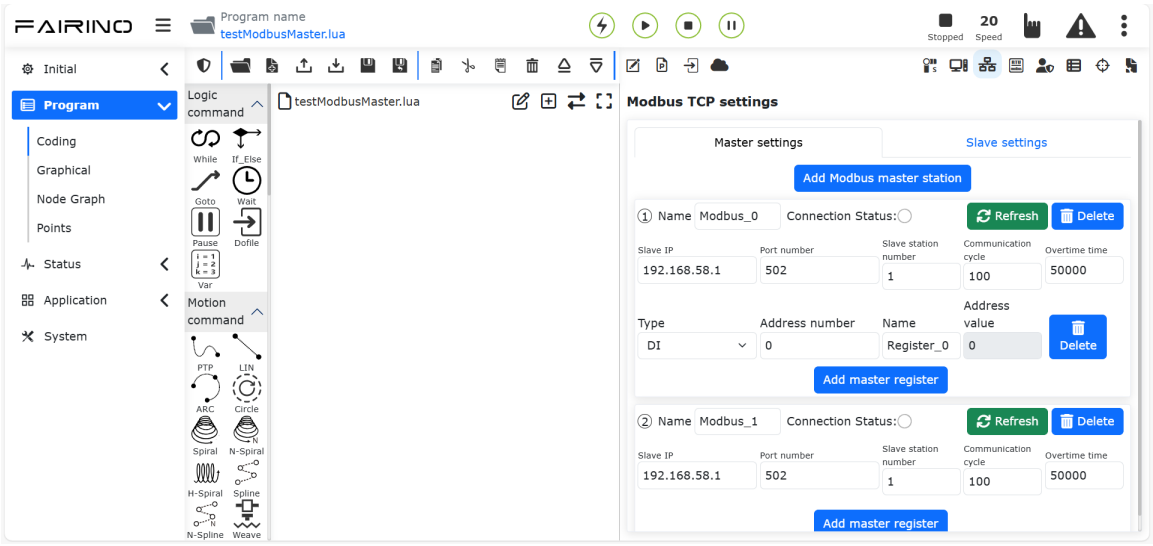


Figure 9.18-2 Open ModbusTCP Settings

Click “Master settings” and “Add Modbus master station” in turn to finish adding a ModbusTCP master station.

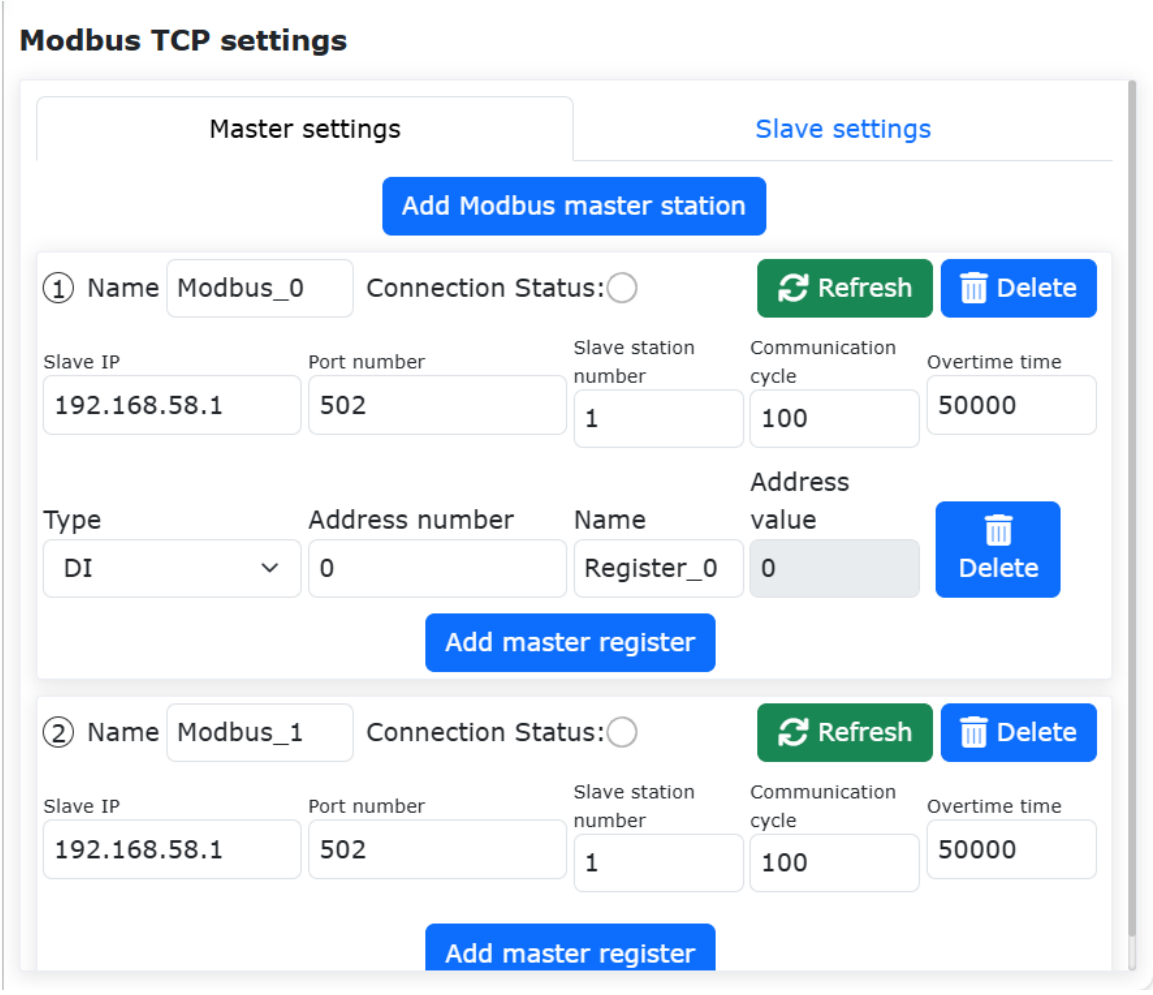


Figure 9.18-3 Add “ModbusTCP Master Station

According to your equipment, enter the name, slave ip, port number, slave number, communication cycle, and timeout in turn. The specific meanings of the above parameters are as follows:

Name:ModbusTCP master station name. Robots can create up to 8 master stations to connect with corresponding slave stations. Different master stations can be distinguished by unique names, such as PLC, camera, data acquisition card and FRRobot1;

Slave IP:the slave IP address to which the ModbusTCP master station of the robot is connected;

Note: The robot and slave equipment must be connected through the network cable first, and the IP addresses of the robot and slave equipment must be in the same network segment.

Port number:The port number of ModbusTCP slave station to be connected;

Slave station number:ModbusTCP slave station number to be connected;

Communication period: The period (ms) when the robot ModbusTCP master station inquires about the slave station status, which only affects the update speed of the slave station register data on the ModbusTCP Settings page, but does not affect the speed of reading or writing the ModbusTCP slave station register value in the lua program of the user.

Timeout Period: When using the ModbusTCP read/write interface for operations, if the system fails to establish a connection after exceeding the timeout period, it will report a Modbus not connected error. Unit: ms, valid range: 100-60000.

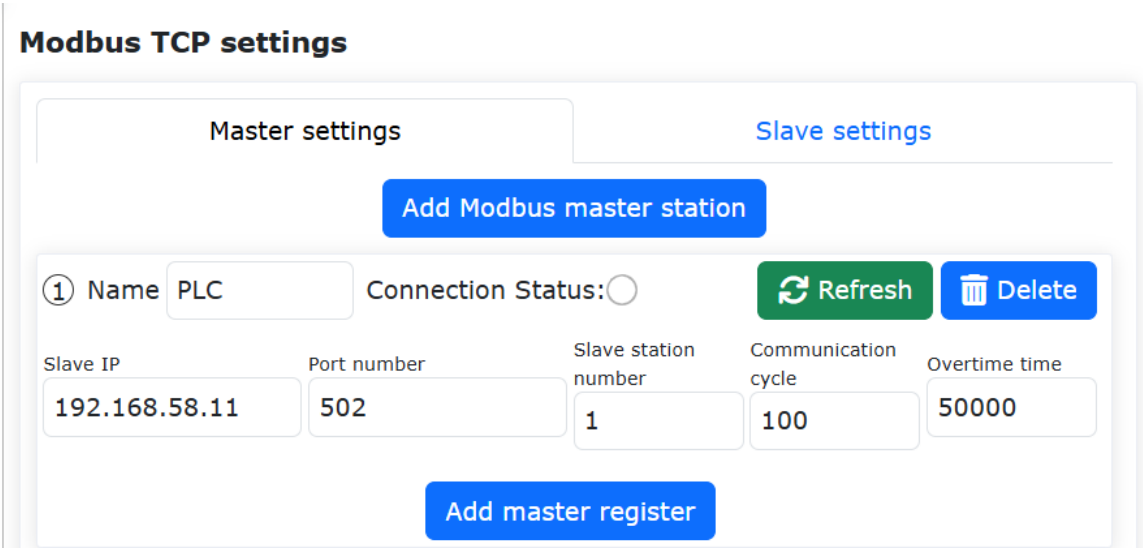


Figure 9.18-4 Setting ModbusTCP master station parameters

After the above parameters are correctly input, the ModbusTCP master station of the robot automatically establishes a connection with the configured slave station. After the connection is successful, the “Connection status” indicator on the page lights up.

Note: If you have confirmed that the relevant parameters of ModbusTCP master station are correctly configured, but the robot is not successfully connected with your equipment, please check the following configurations:

The physical network connection between the robot and the slave equipment;

The IP addresses of the two network physical ports of the robot teaching device and the control box are different, please confirm whether they are connected to the correct network port;

Please confirm whether the network port of the robot and the network port of the slave station equipment are in the same network segment. If the IP address of the robot is 192.168.58.2, the IP address of the slave station equipment

must be 192.168.58.0~192.168.58.255, and it cannot be the same as the IP address of the robot;

Check whether the port number of the slave equipment is the same as the set port number. (If the connection status indicator is flashing, it means that the register address in the master station is wrong. Please check whether the register type and address are correct)

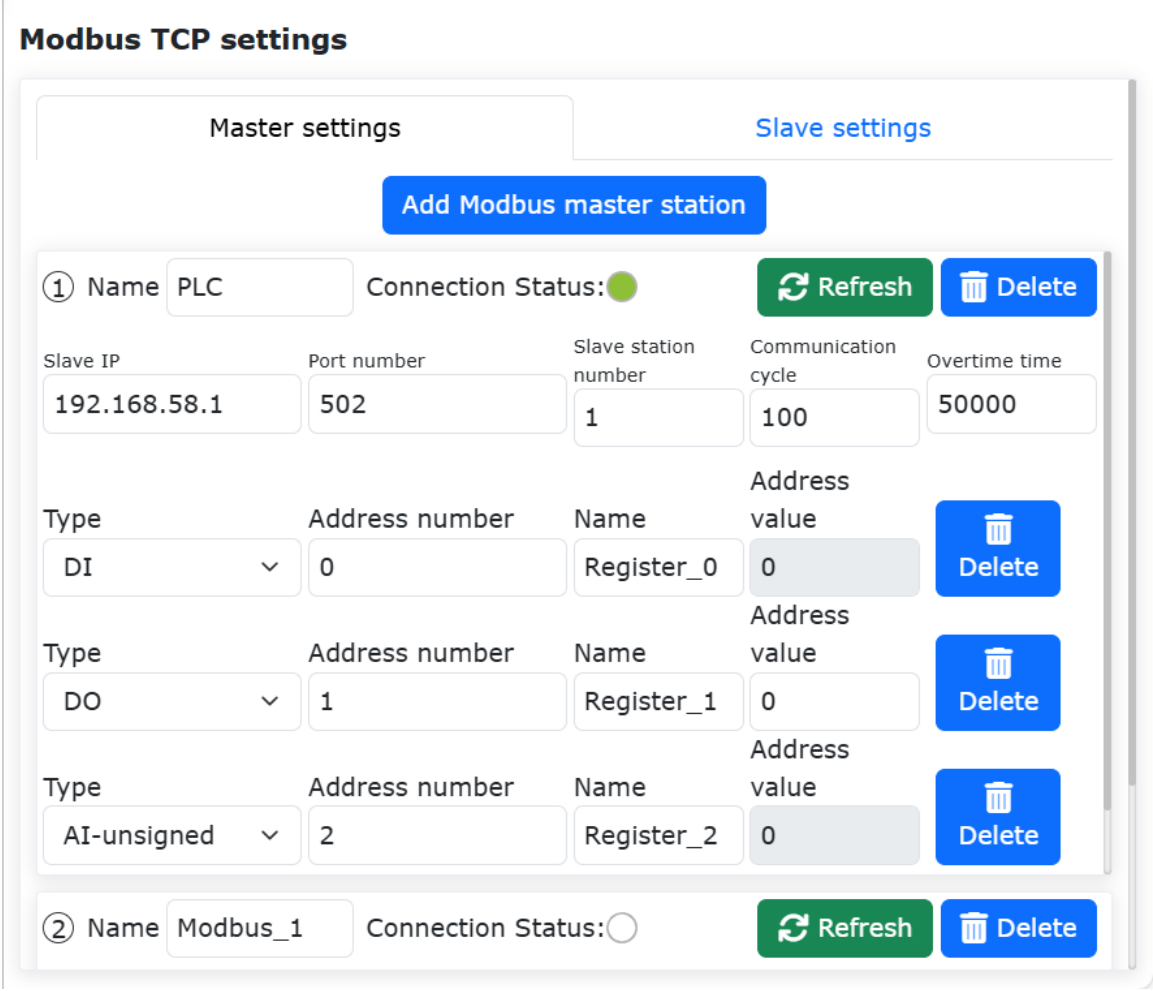


Figure 9.18-5 Connection status of Modbus TCP master station

At this point, we have completed the creation of a ModbusTCP master station for the robot. If you click “Add Modbus Master Station” again, you can create a new ModbusTCP master station (Figure 7). The robot can support up to 8 master stations to communicate with external devices at the same time. Double-click the “Delete” button in the upper right corner of the Modbus master station to delete the Modbus master station.

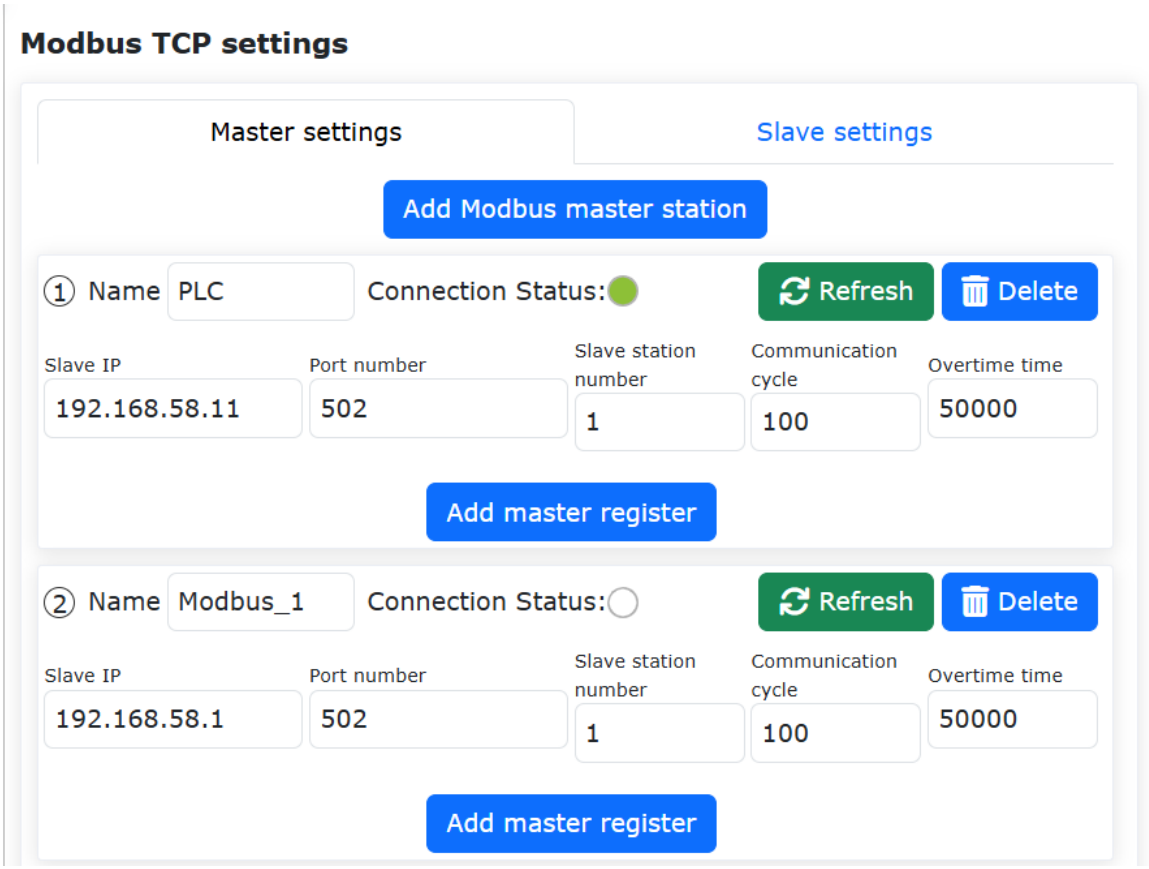


Figure 9.18-6 Add ModbusTCP master station again

1.3.9.18.1.2 ModbusTCP master add register

Click the “Add master register” button to add a register for this master station.

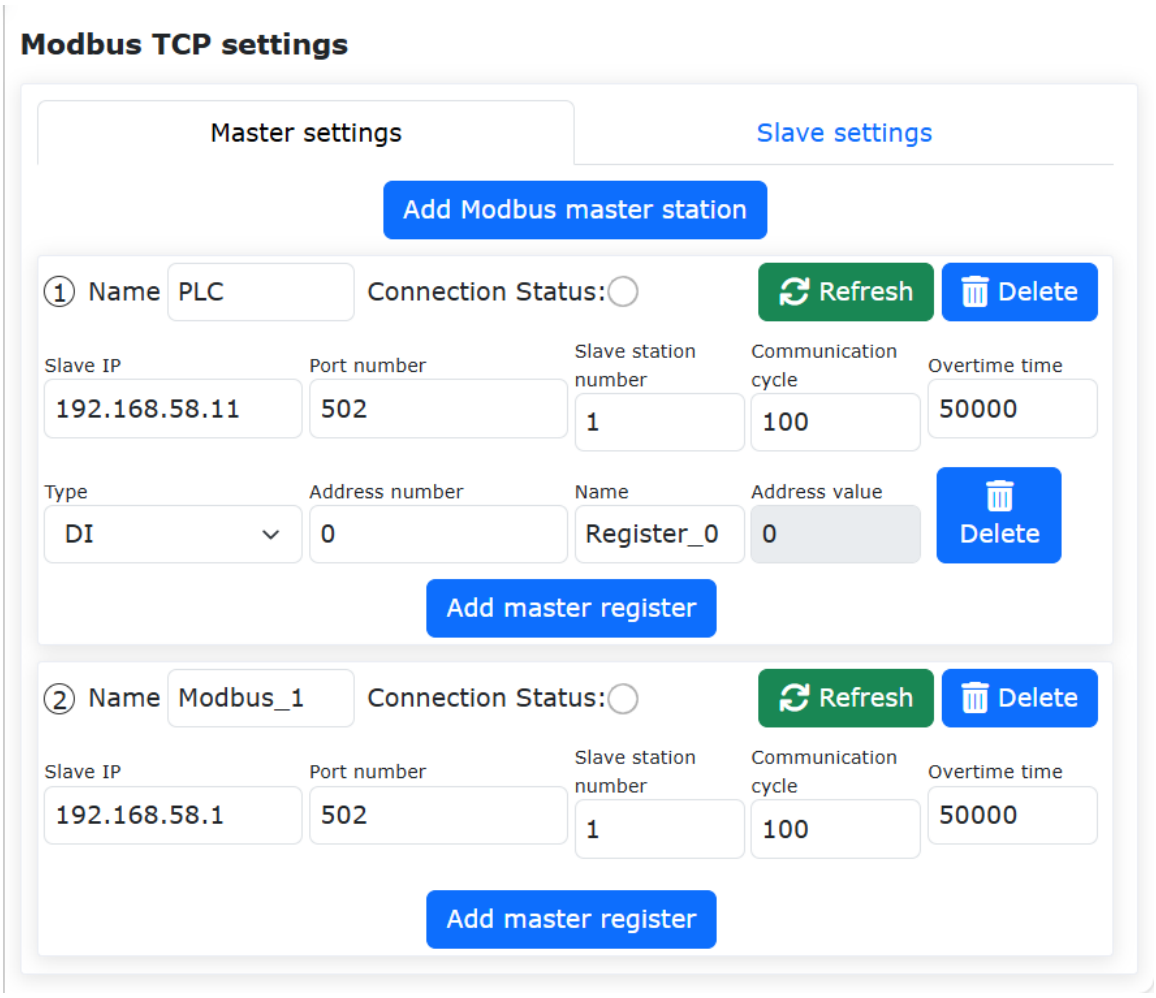


Figure 9.18-7 Add ModbusTCP master station register

Select the modbus register type, input address number and name of the master station in turn, and the meanings of each parameter are as follows:

Type:Register type, DI:discrete input; DO:coil; AI-unsigned:unsigned input register (0-65535); AI-signed:signed input register (-32768-32767); AI-float:float type input register (the data length of float type register is 32 bits, occupying two signed or unsigned registers); AO-unsigned:unsigned holding register (0-65535); AI-signed:signed holding register (-32768-32767); AI-float:float type holding register (the data length of float type register is 32 bits, occupying two signed or unsigned registers), among which the floating-point type registers in AI and AO are big-end display.

Address number:The address of ModbusTCP slave register to be read or written;

Name: The alias of the register. The ModbusTCP master station of the robot can set up at most 128 different registers, and each register can be distinguished by different names according to the actual meaning, such as “Start”, “Servo in Place” and “Liquid Level”.

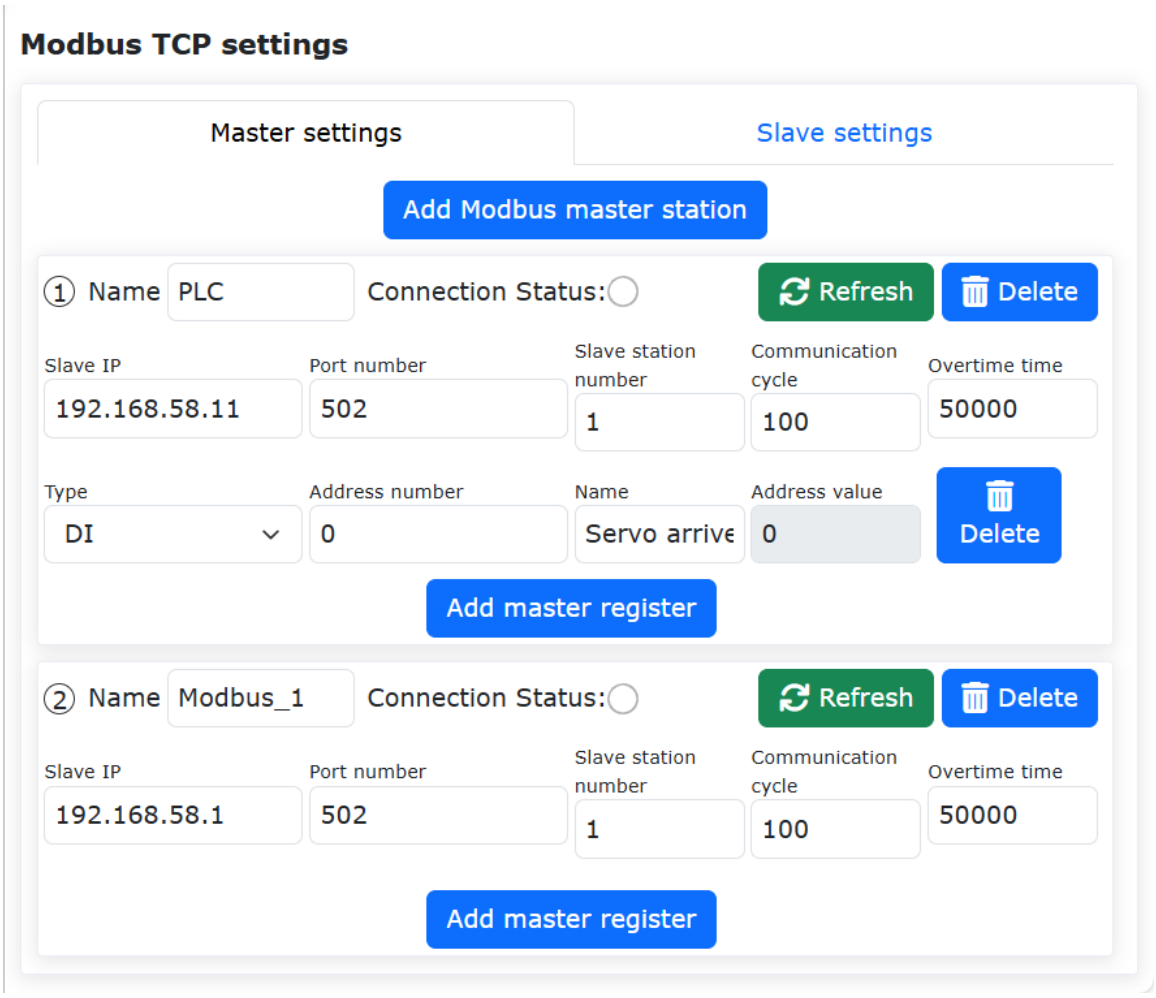


Figure 9.18-8 Setting ModbusTCP master station register parameters

Click the “Add Master Register” button again to add another master register, and double-click the “Delete” button on the right side of the register to delete it, a register is added for each type.

Note: If the master station connection status indicator flashes after adding the master station register, it means that the master station register address cannot be read. Please check whether the register type and address are correct.

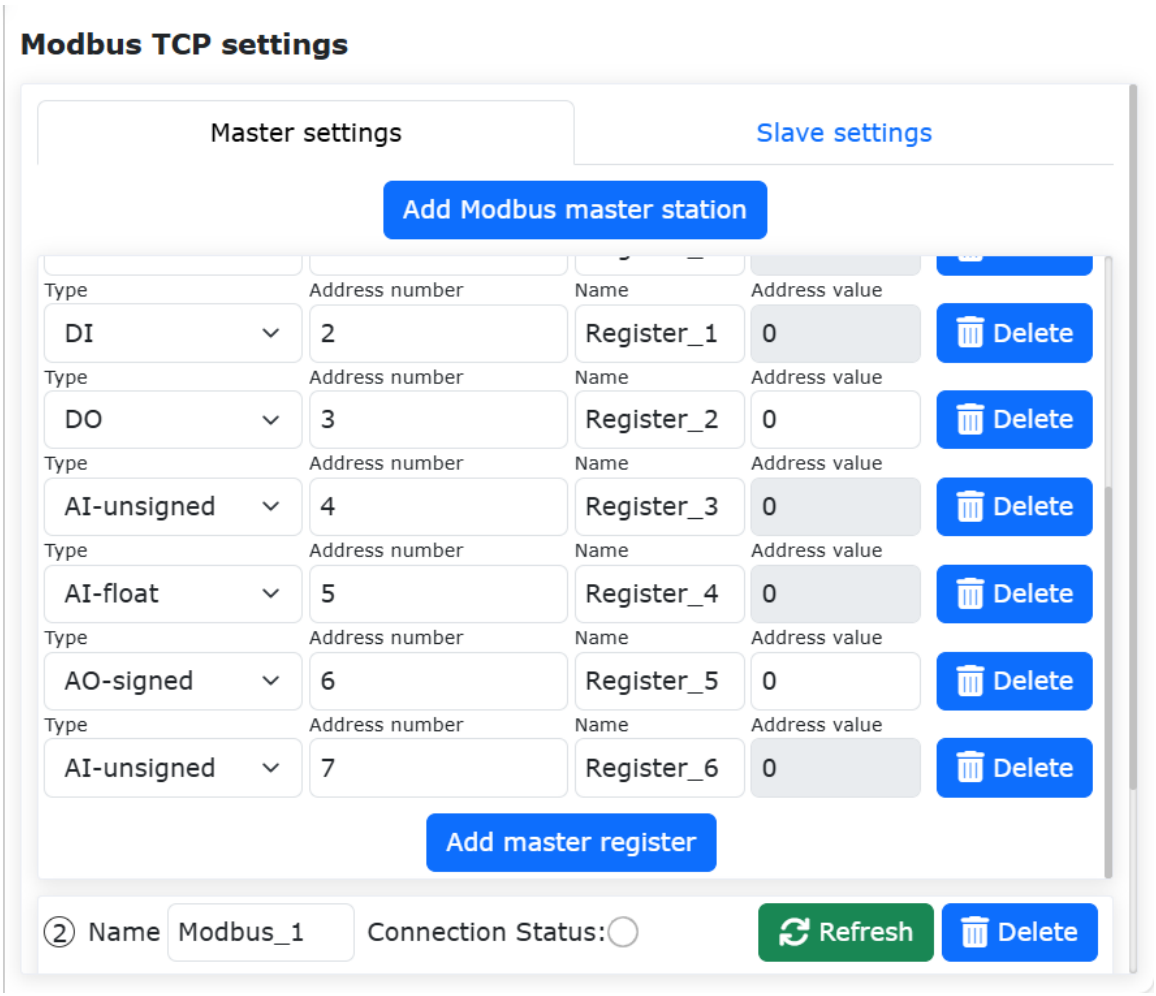


Figure 9.18-9 Adding Multiple Master Station Registers

1.3.9.18.1.3 ModbusTCP master communication test

Before the communication test, please check whether the “Cnnection status” indicator of ModbusTCP master station is always on. If the indicator is always on, it means that the current connection has been successful.

The robot Modbus master station register has an “address value” value box for displaying the current register value, in which the registers of DI (discrete input) and AI (input register) are read-only and the corresponding address values are gray non-editable value boxes.

When the value of the corresponding address of the slave station changes, the robot master station displays the current value synchronously corresponding to the register address value. DO (coil) and AO (holding register) are readable and writable registers, so their addresses are white editable value boxes, which can be used to read the values of the corresponding registers of ModbusTCP slave stations or modify the values of the registers on the Modbus master station setting page of the robot.

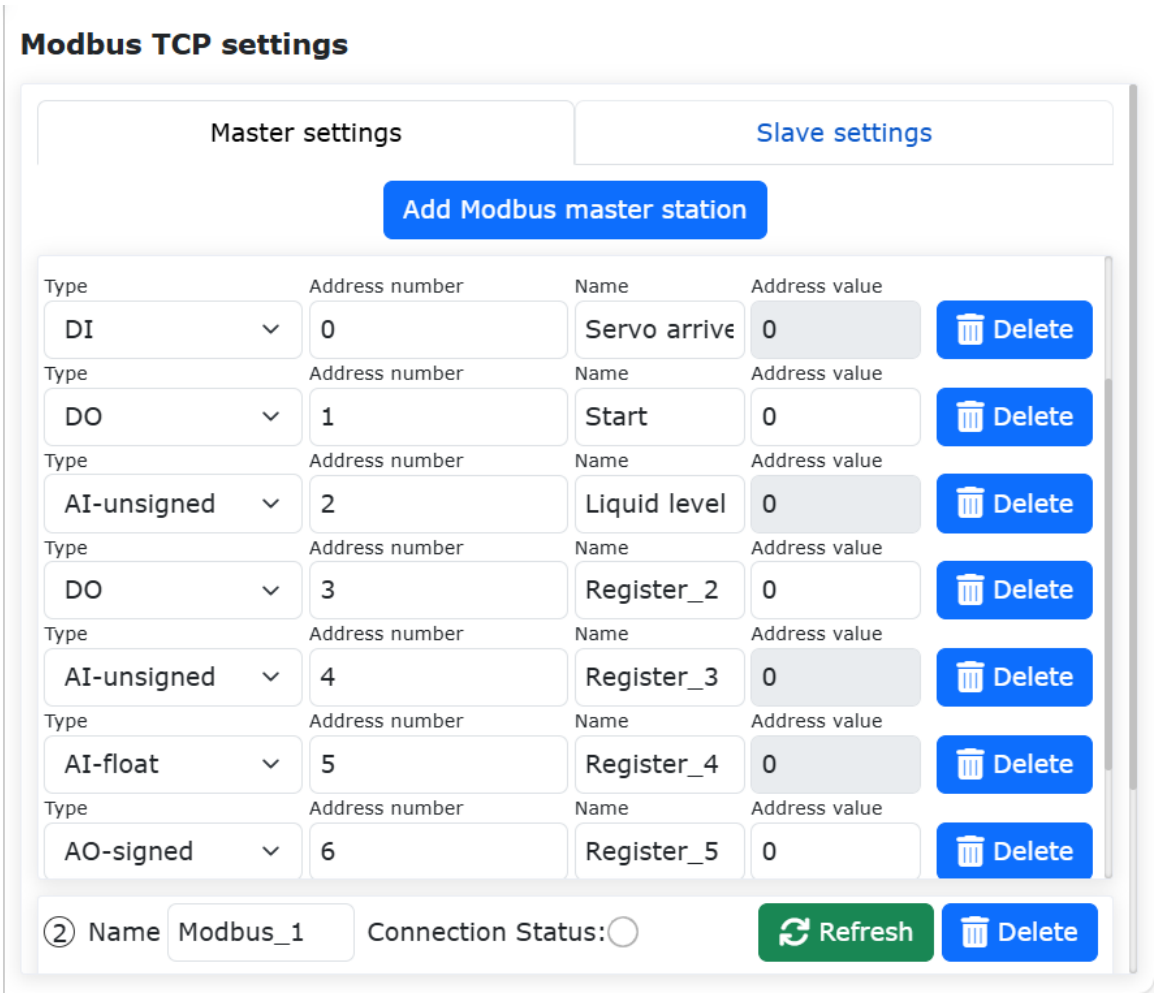


Figure 9.18-10 Modbus master station address value

1. Numerical monitoring of DI and AI type registers of master

On the external ModbusTCP slave device, set the address value of No.255 of DI (discrete input) register to 1, change the address value of No.257 of AI (input register) to No.123, change the address value of No.258 to -123, and change the address value of No.259 to 123.3. At this time, the address value of the corresponding register on the robot Modbus master station setting page will be displayed accordingly.

Note: Because the register with address 259 is a floating-point register, it actually occupies two 16-bit registers 259 and 260 to store a floating-point number, so you can't set a separate register to operate the No.260 register of AI, otherwise a numerical error will occur.

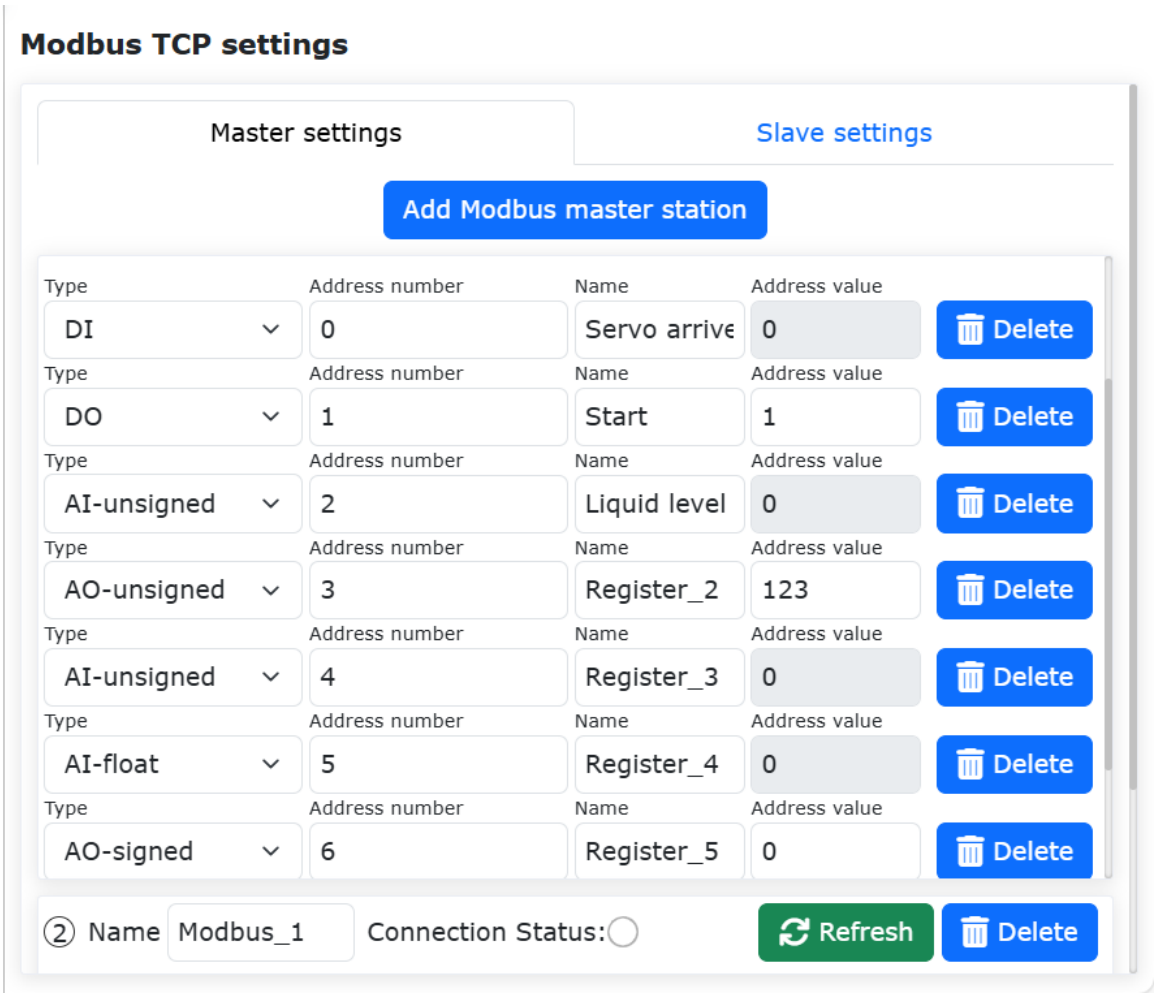


Figure 9.18-11 Modbus master station displays the values of DI and AI registers

2. Numerical writing of DO and AO type registers in the master

In the Modbus master station setting page of the robot, enter 1 in the No.255 address value input box of the DO (coil) type register named 'Start', and enter 65535,-32767 and 128.78 input boxes of the AO (holding register) named 'Target position A', 'Target position B' and 'Target position C' respectively.

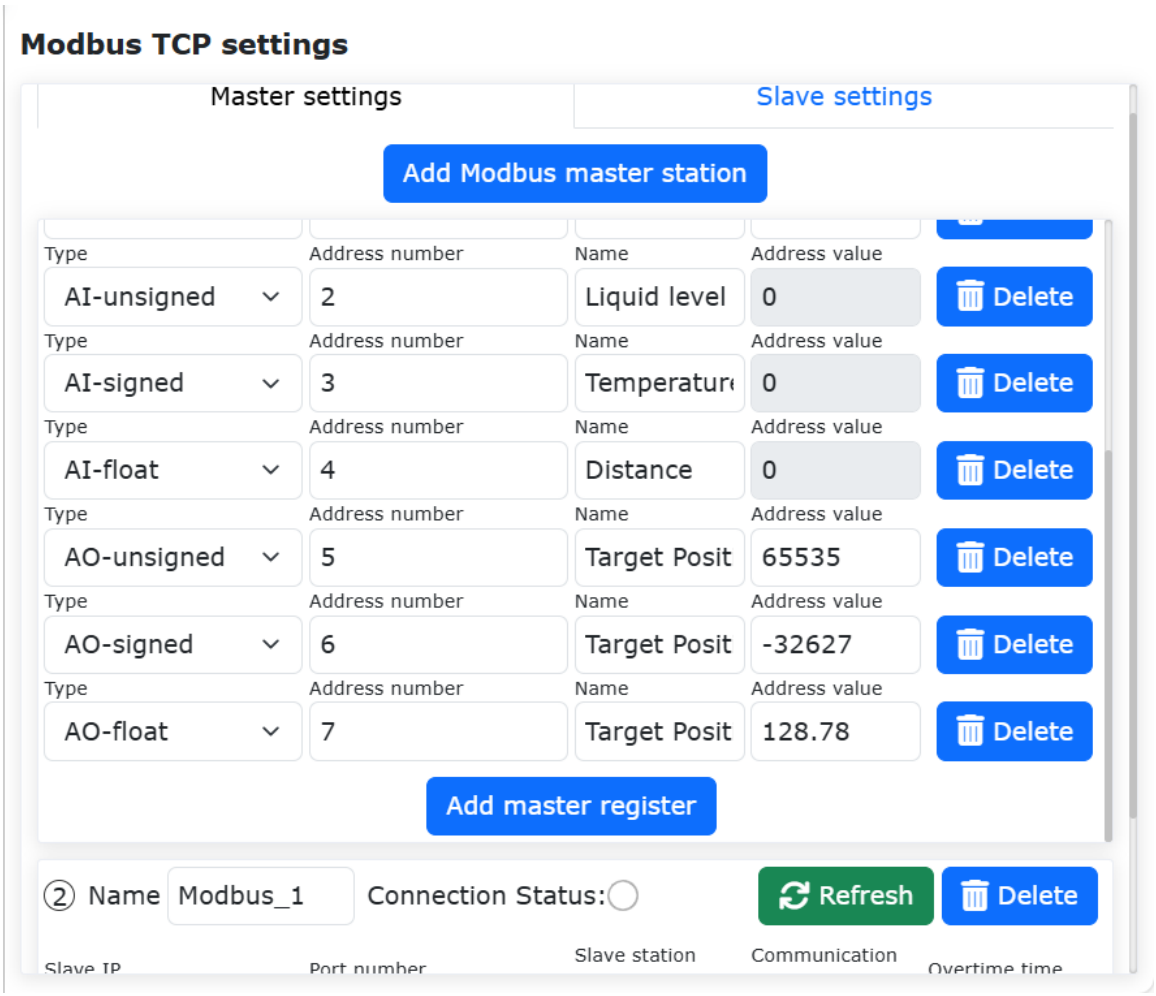


Figure 9.18-12 Modbus master station writes DO and AO registers

3. Numerical monitoring of DO and AO type registers of master If you change the values of DO (coil) and AO (holding register) in ModbusTCP slave station, the register address value of ModbusTCP master station setting page will not be updated immediately. You need to click the “Refresh” button in the upper right corner of master station configuration, and then the register address values of DO and AO on the page will be updated.

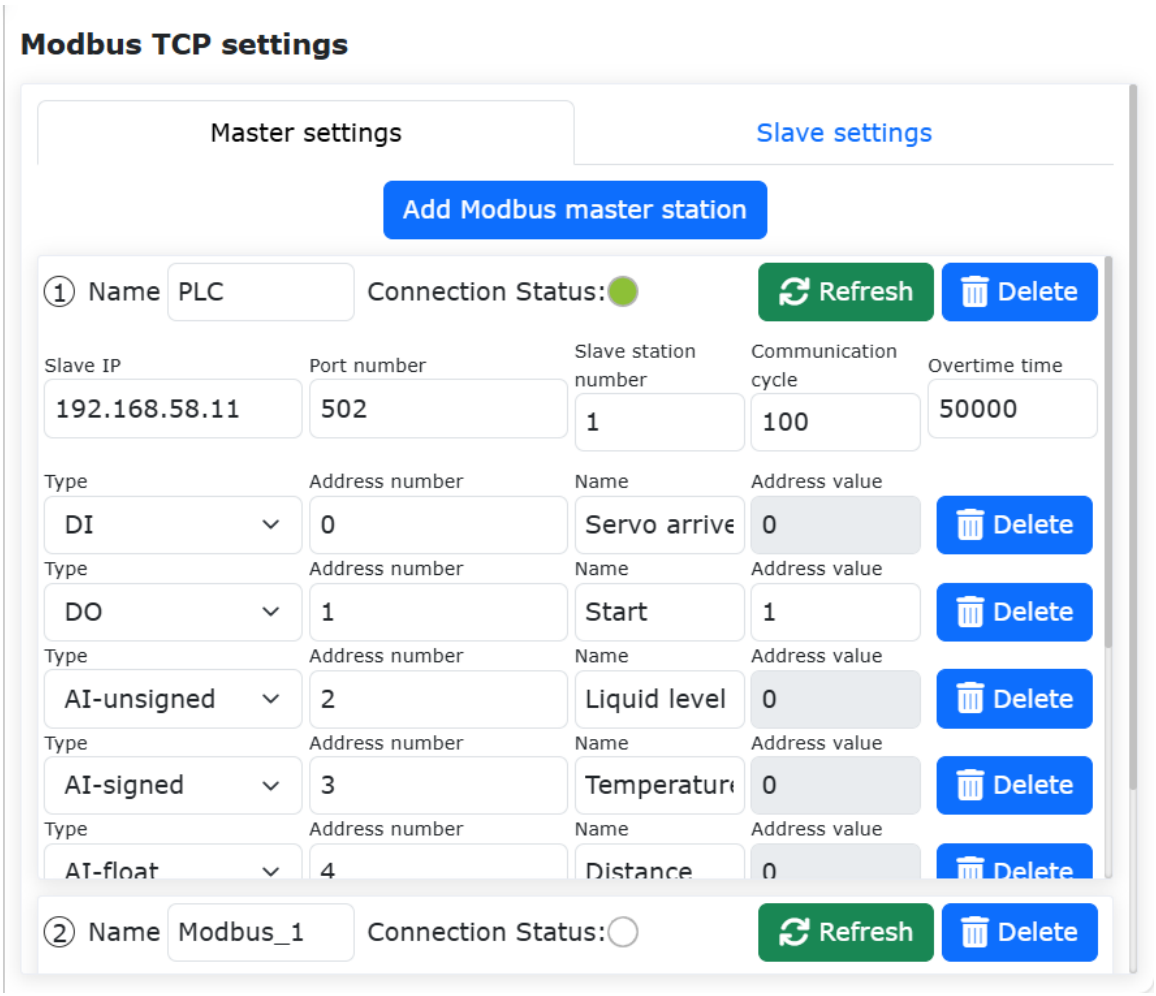


Figure 9.18-13 Refresh the DO and AO address values of ModbusTCP master station.

1.3.9.18.2 Write ModbusTCP master program

Click “Program” and “Communication command” in turn to open the communication command adding page.

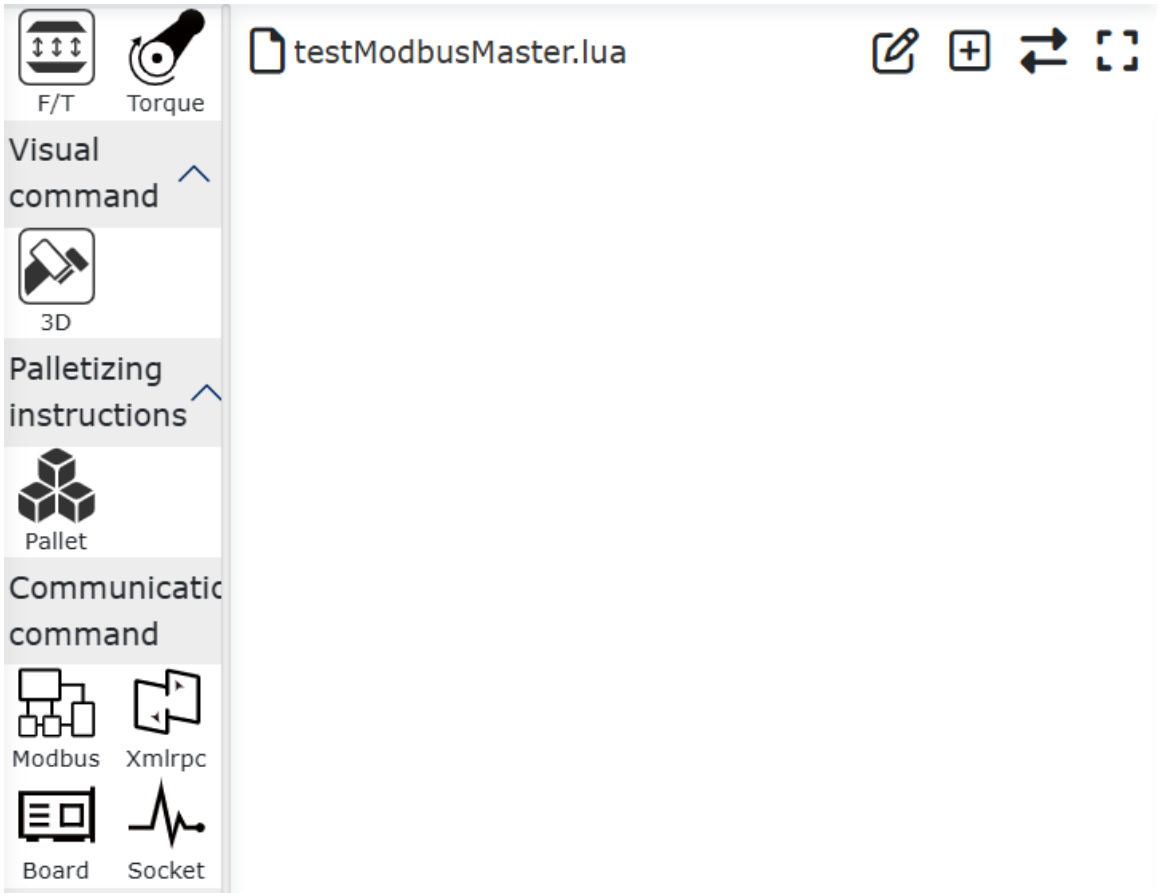


Figure 9.18-14 Open the communication command add page

Click “Modbus”.

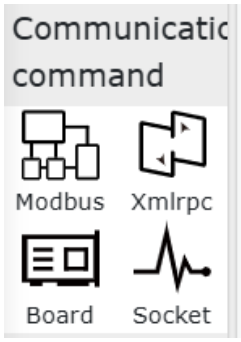


Figure 9.18-15 Select Modbus

Click “Modbus_TCP”.

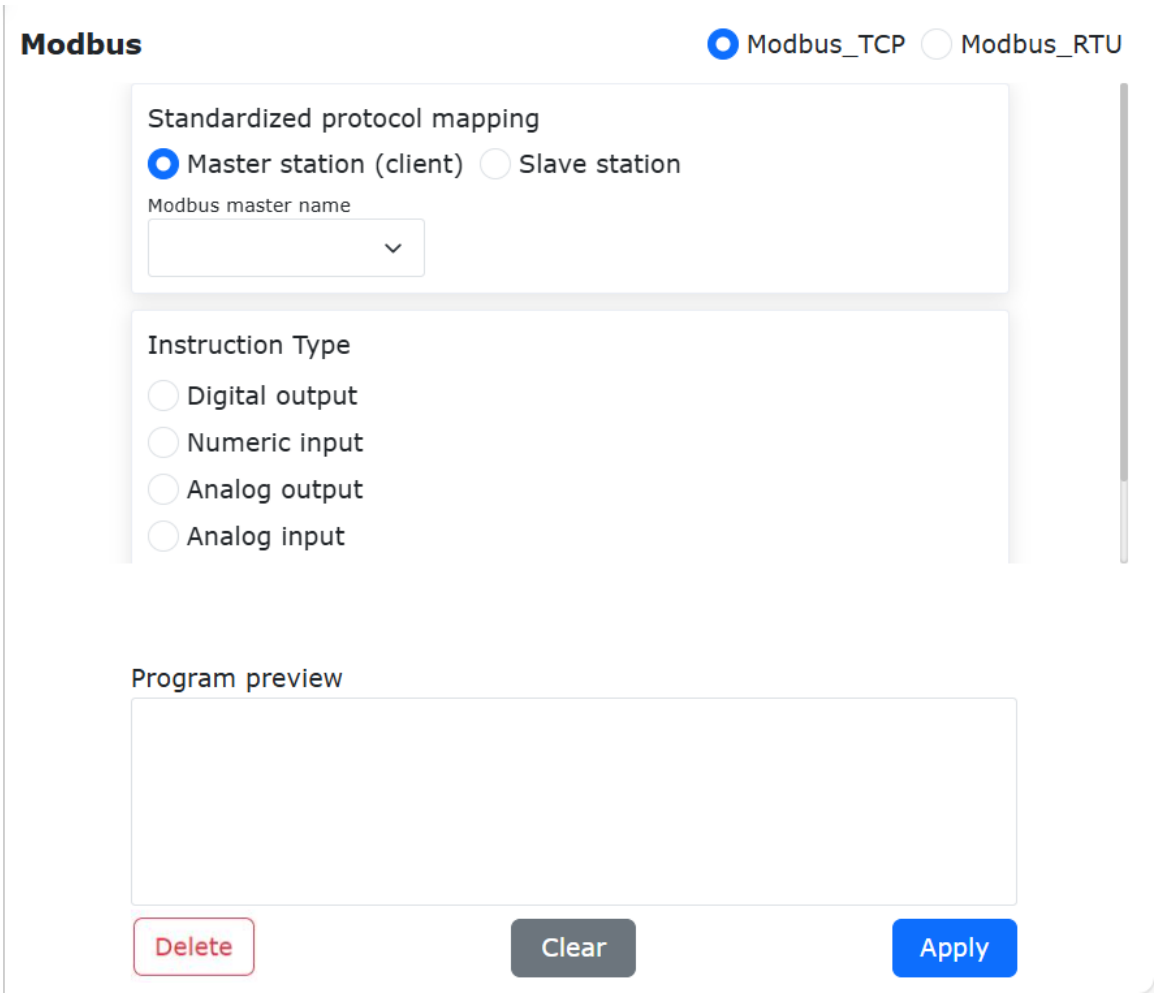


Figure 9.18-16 Select Modbus_TCP

Select “Master station (client)” to open the ModbusTCP master station command addition page.

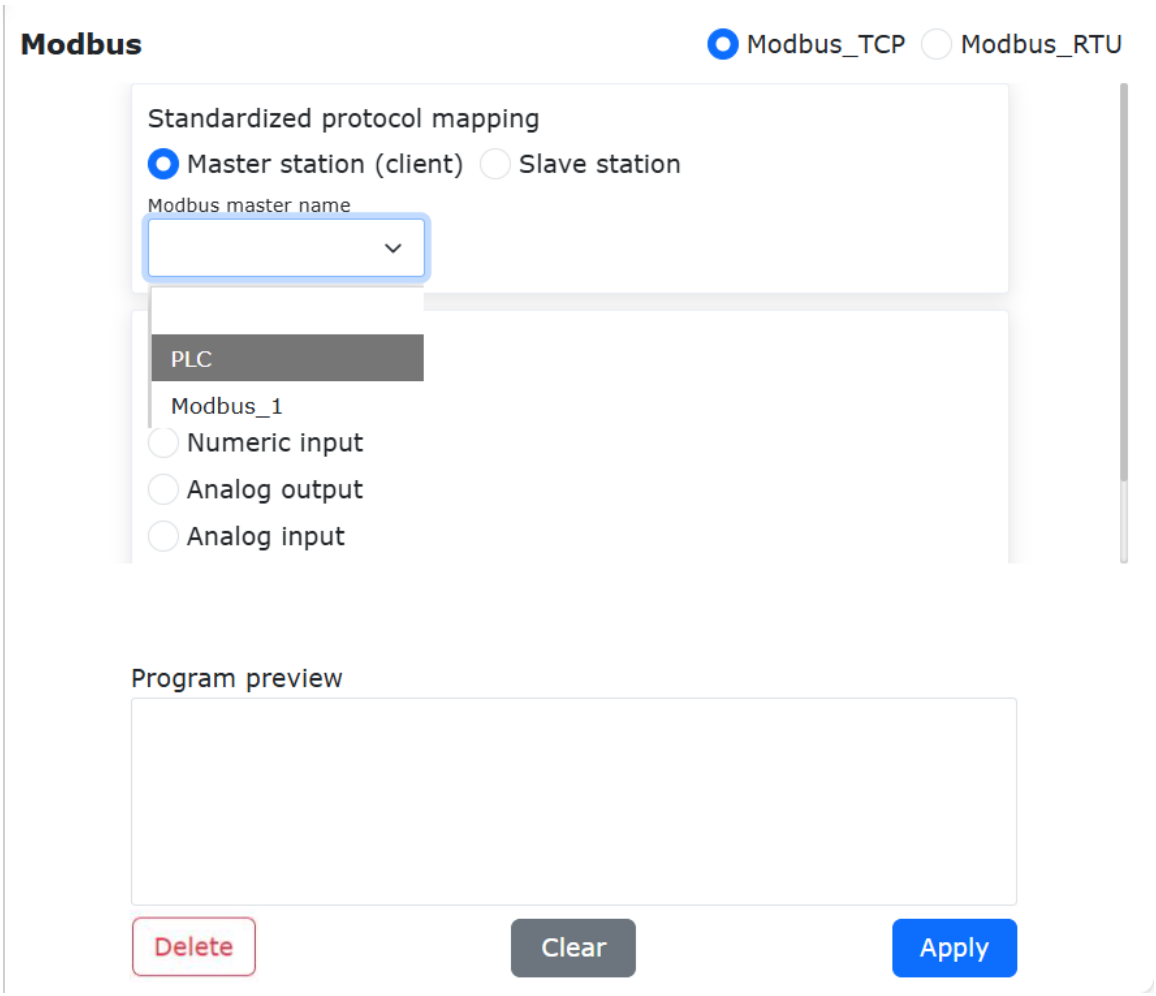


Figure 9.18-17 Selecting Master Station (Client)

1. Write a single digital output DO (coil)

Select “Modbus master name” as the master station ‘PLC’ added on the Modbus master station setting page before, with DO name as ‘Start’, register number as 1 and register value as 1, and click the “Write digital output” button. Finally, scroll to the bottom of the page and click the “Apply” button (Figure 21).

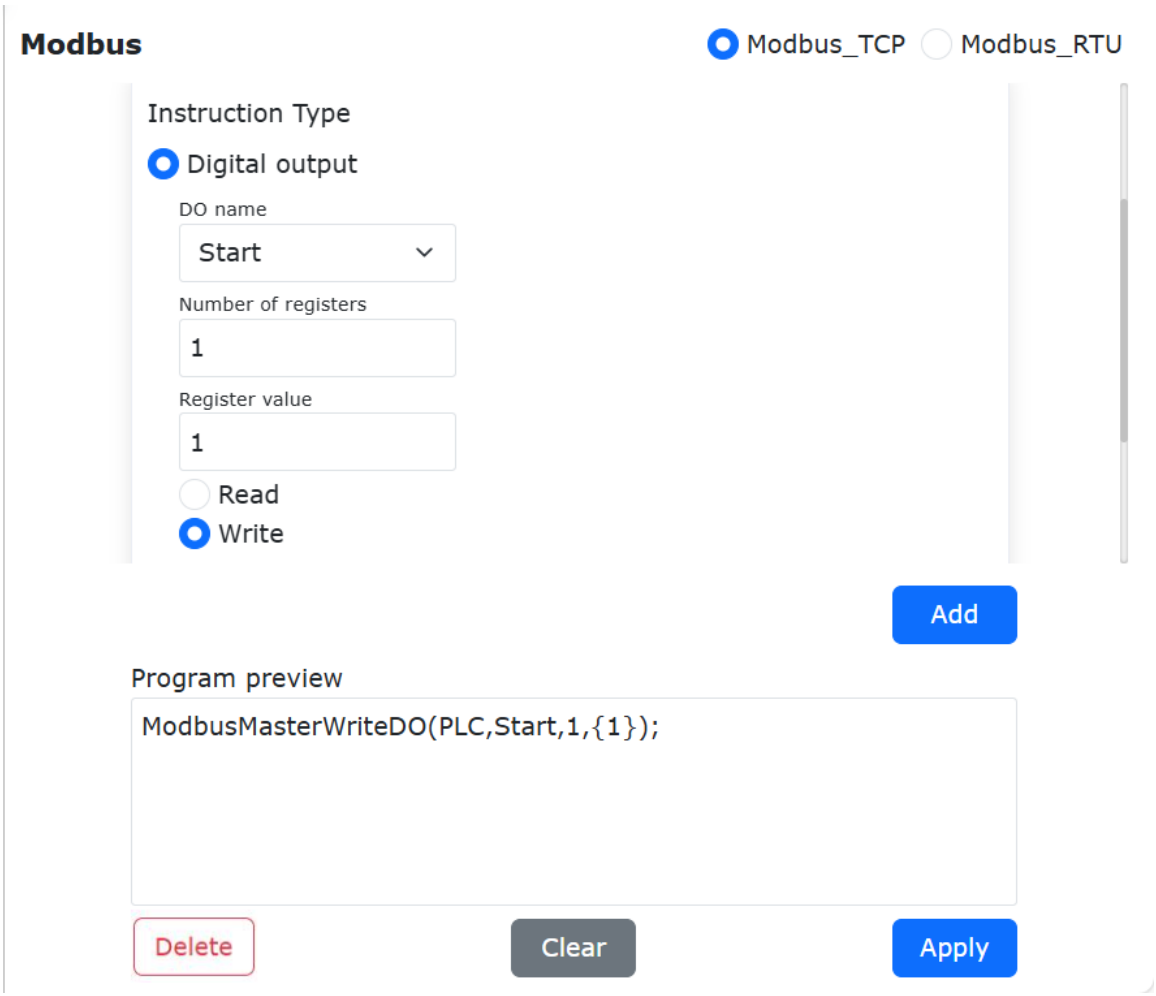


Figure 9.18-18 Adding Write Single Digital Output

At this time, the robot program “testModbusMaster.lua” has added an command for the Modbus master station of the robot to write a single digital output. Switch the robot to the automatic mode, click the start button, and the robot will write the address value of the coil register ‘Start’ corresponding to the master station “PLC” as 1.

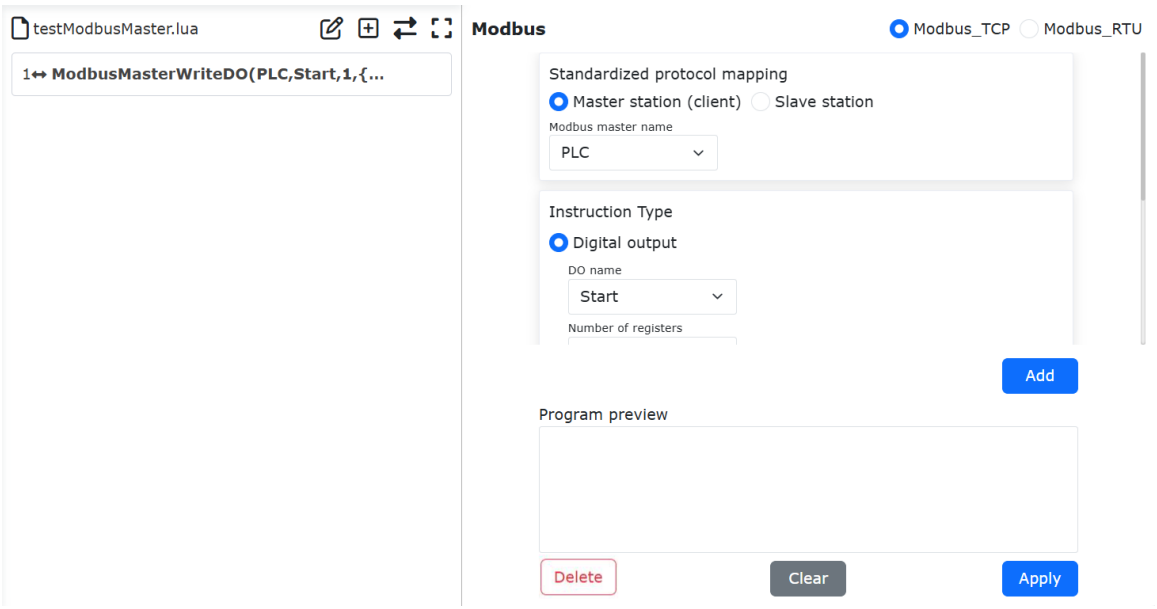


Figure 9.18-19 Writing a single coil lua program

2. Write multiple digital outputs DO (coils)

Open the ModbusTCP master station command adding page, select Modbus master name as the master station 'PLC' added in the Modbus master station setting page before, the name of DO is 'Start', the number of registers is 5, and the register values are 1,0,1,0,1. the number of register values should correspond to the set number of registers, and multiple register values should be separated by English commas, and click the 'Write digital output' button. Finally, scroll to the bottom of the page and click the "Apply" button.

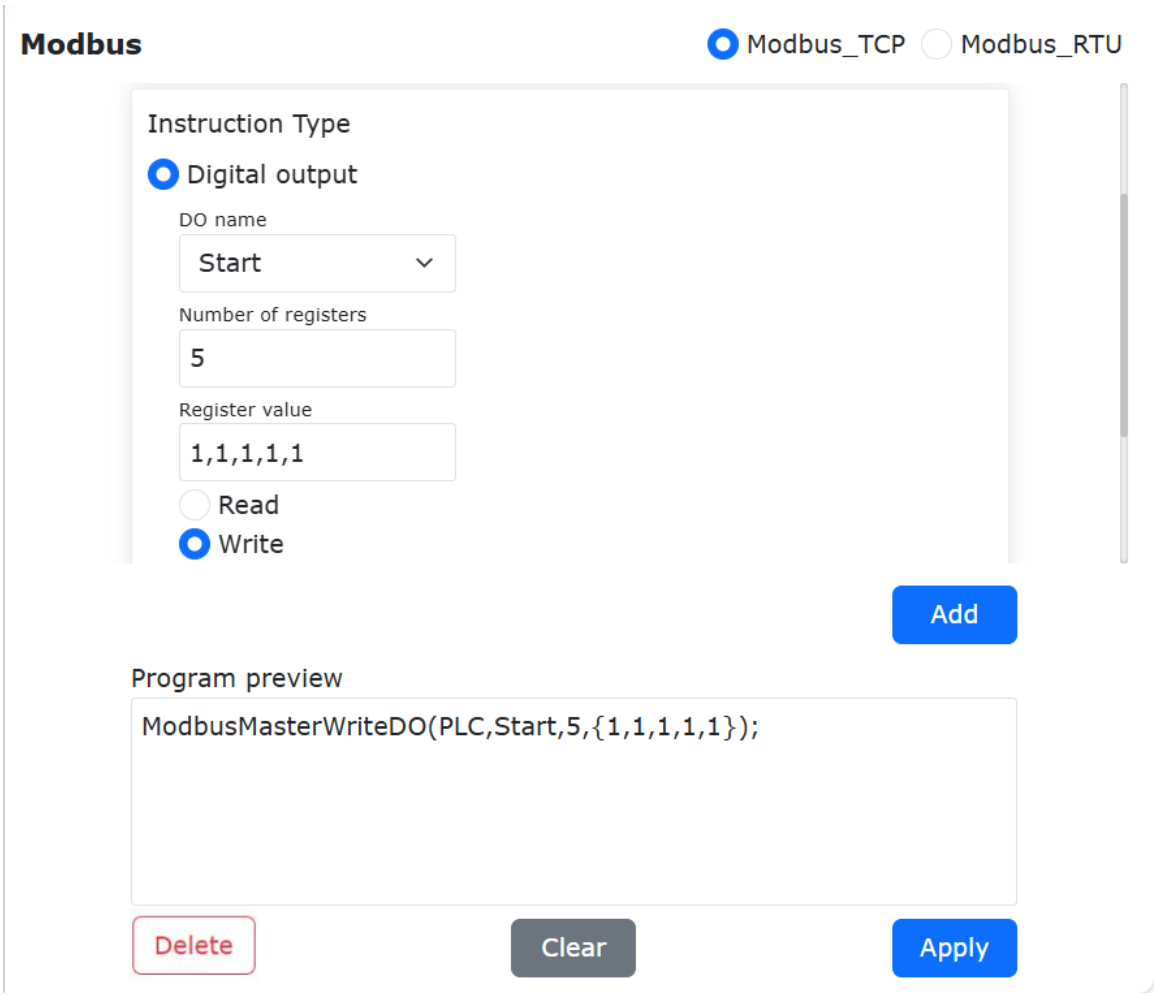


Figure 9.18-20 Configures writing multiple digital outputs

At this time, the robot program "testModbusMaster.lua" has added an command that the robot Modbus master station writes multiple digital outputs, so as to switch the robot to the automatic mode, click the start button, and the robot will write the values of the coil register 'Start' corresponding to the master station 'PLC' and the following four coils as 1, 0, 1, 0 and 1 respectively.

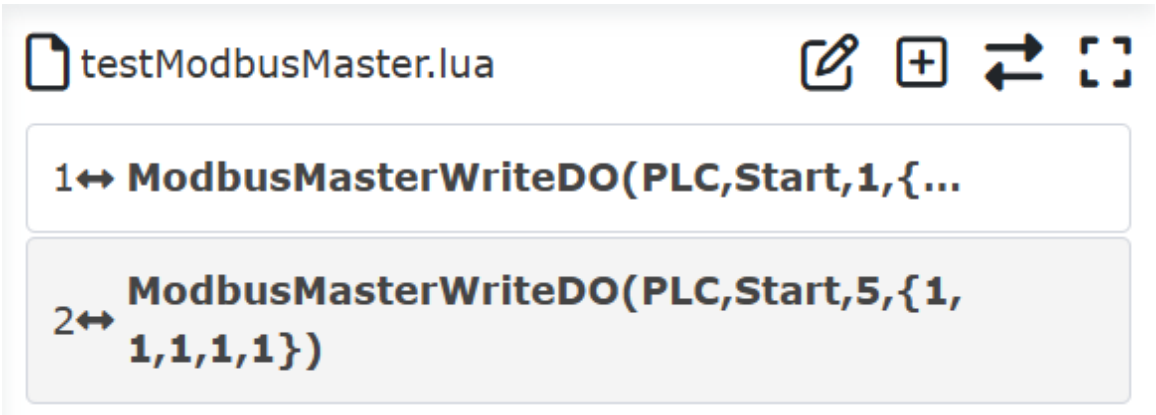


Figure 9.18-21 Programming multiple coils lua

3. Read a single digital output DO (coil)

Open the ModbusTCP master station command addition page, select “Modbus master name” as the master station “PLC” added in the Modbus master station setting page before, the DO name is “Start”, the number of registers is 1, and the register value does not need to be filled in, and click “Read digital output”. Finally, scroll to the bottom of the page and click the “Apply” button.

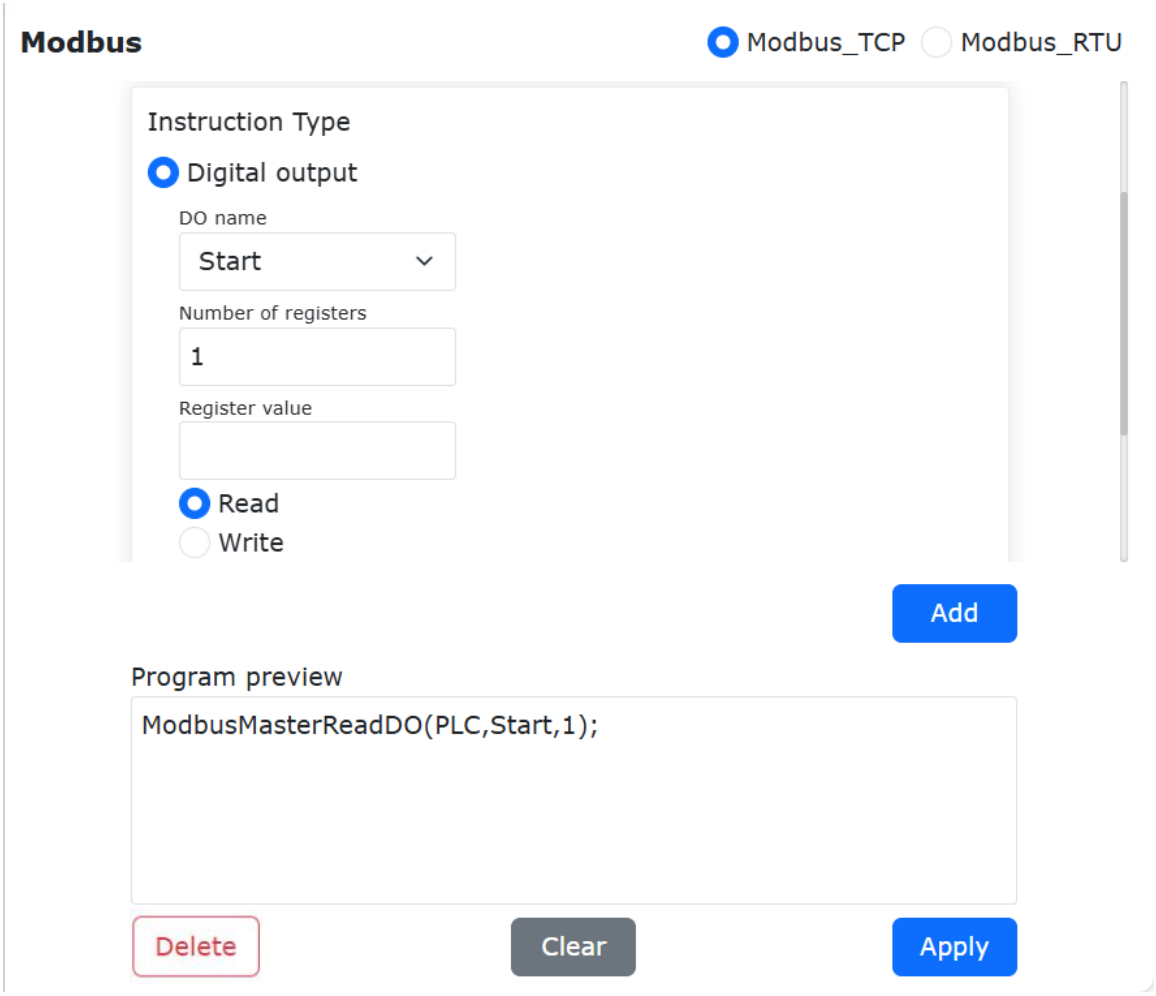


Figure 9.18-22 Configuring Reading a Single Digital Output

At this time, the robot program “testModbusMaster.lua” has added an command for the robot Modbus master station to read a single digital output.

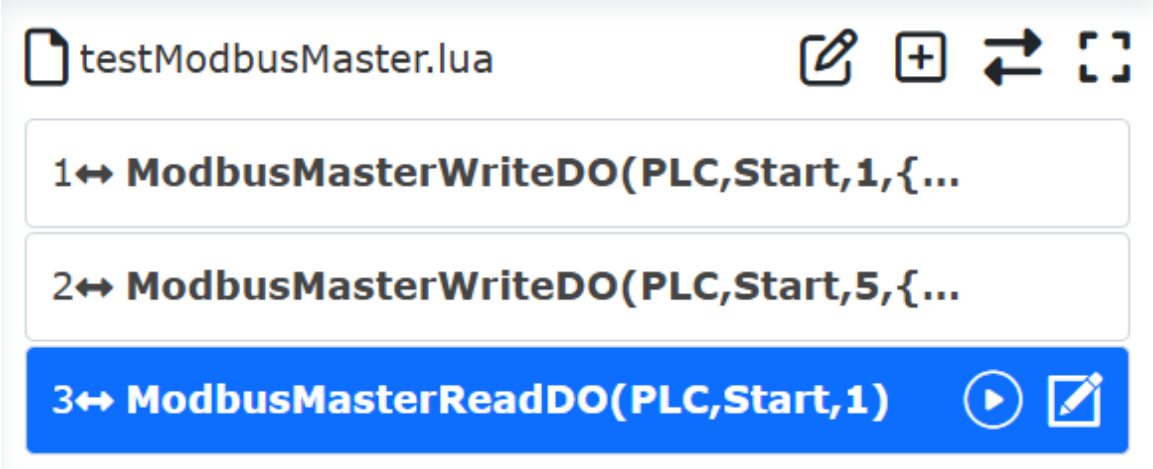


Figure 9.18-23 Program for reading a single coil

Usually, after reading the Modbus register, the read value is stored in a variable, so it is necessary to define a variable to store the read value. As shown in figure 29, click the "switch mode" Button to switch the robot lua program to an editable state, and write the variable "startValue" with added return value before the "ModbusMasterReadDO" command, and the value read after executing the program will be stored in "startValue".

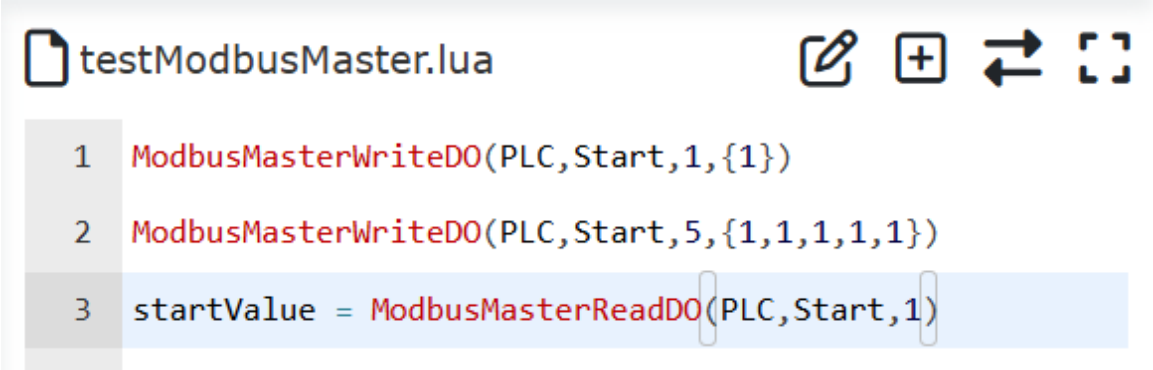


Figure 9.18-24 Reading a single digital output into a variable

The register value of coil type is only 0 and 1, so different operations can be carried out by judging the different register values in robot programs. As shown in Figure 30, click the "Switch Mode" button to switch the robot teaching program to the non-editable mode, and add two joint motion commands to move to two different points "P1" and "P2" respectively.

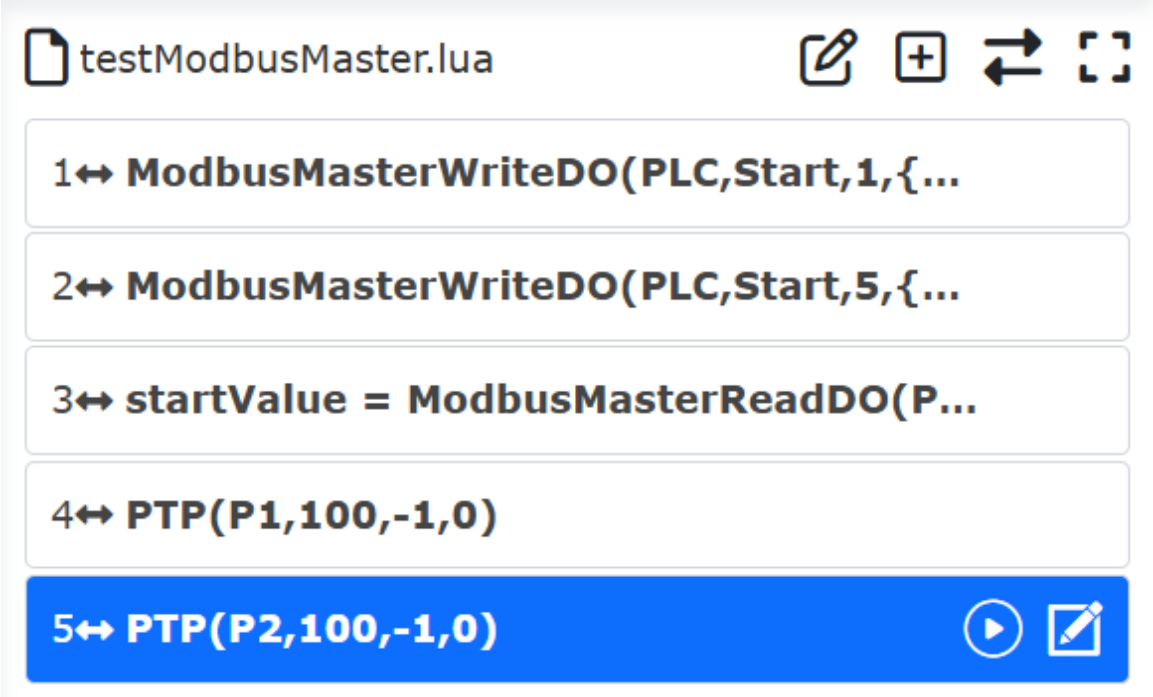


Figure 9.18-25 Add motion commands of different points

The program is switched to the editable mode again, and the judgment condition of the coil value "startValue" is written. When the value of "startValue" is 1, the robot moves to the point "P1", otherwise the robot moves to the point "P2".

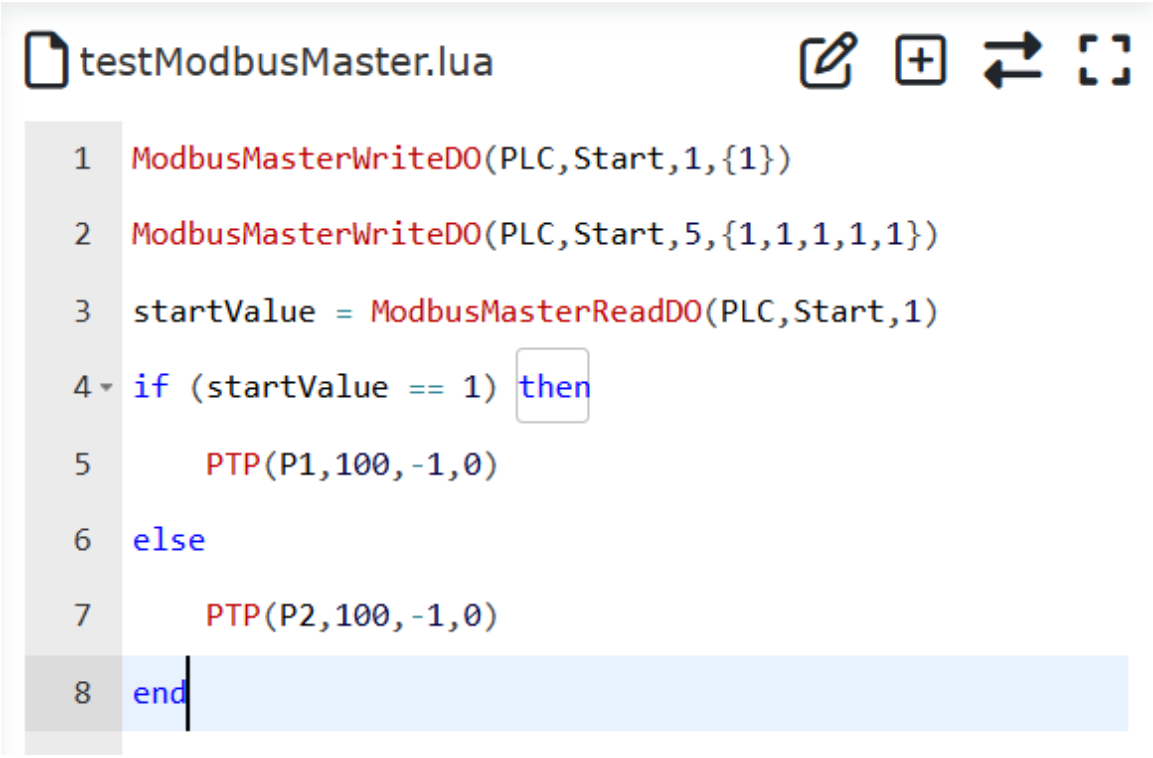


Figure 9.18-26 Move to different points according to different coil values

Finally, switch the robot program to non-editable mode, switch the robot to automatic mode, and start the running

program on the premise of confirming safety. The first two lines of the program set the DO value of the coil named "Start" to 1, so the robot will move to the "P1" point after executing the program.

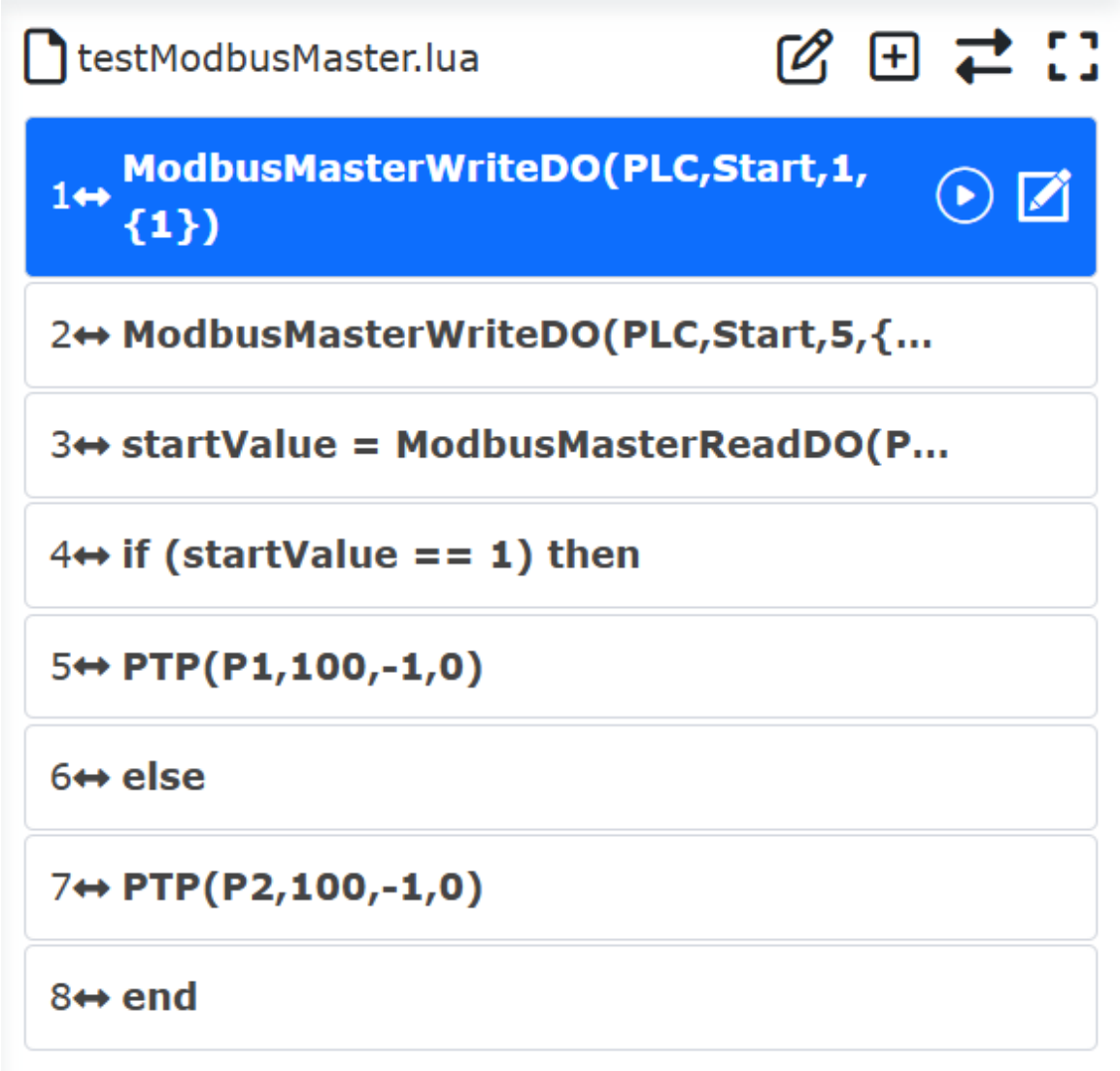


Figure 9.18-27 Read the register value of a single coil and move it

4. Read multiple digital DO (coils)

Open the ModbusTCP master station command adding page, select "Modbus master name" as the master station "PLC" added in the Modbus master station setting page, the name of DO is "Start", the number of registers is 6, and the register value does not need to be filled in, and click "Read digital output". Finally, scroll to the bottom of the page and click the "Apply" button (Figure 34).

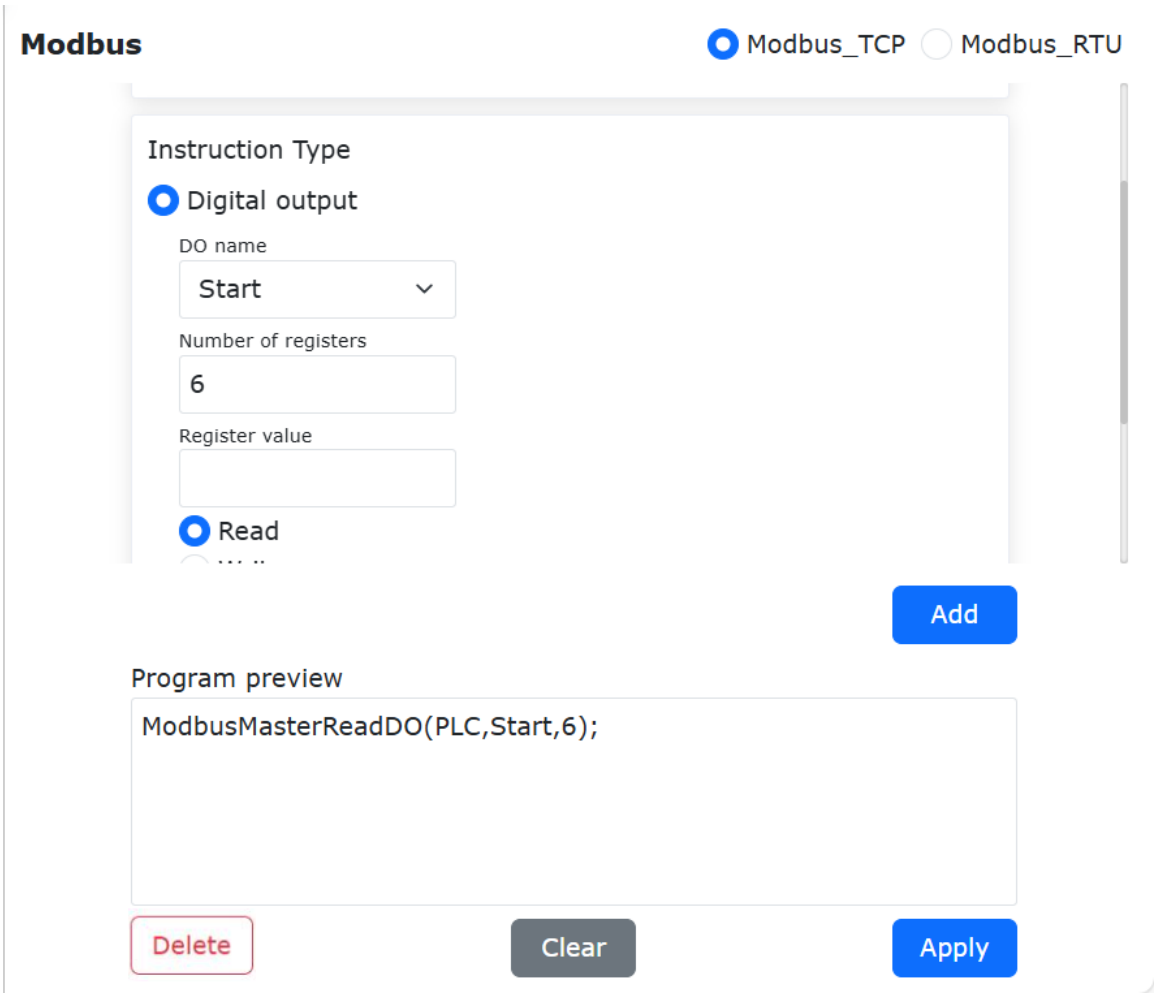


Figure 9.18-28 Configuring Reading Multiple Digital Outputs

At this time, the robot program “testModbusMaster.lua” has added an command that the robot Modbus master station reads multiple digital outputs.

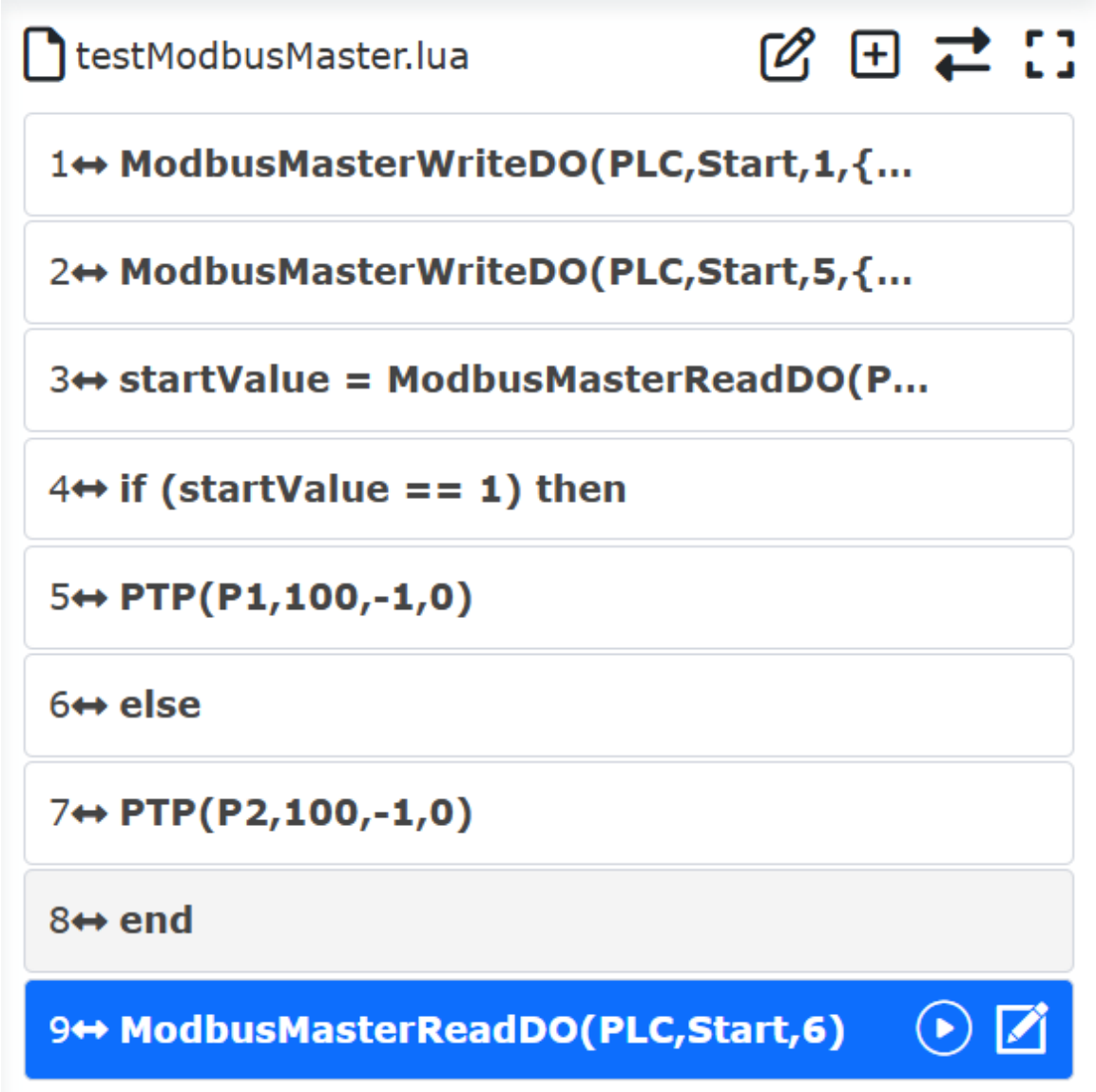


Figure 9.18-29 Reading multiple digital output programs

Click the “switch mode” button “” to switch the robot lua program to the editable state. Since the number of readings is six, it is necessary to write and add six return value variables “value1,value2,Value3,Value4, Value5,value6” before the “ModbusMasterReadDO” command, and the six register values read after the program execution will have the above six variables respectively.

testModbusMaster.lua

```
1  ModbusMasterWriteDO(PLC,Start,1,{1})
2  ModbusMasterWriteDO(PLC,Start,5,{1,1,1,1,1})
3  startValue = ModbusMasterReadDO(PLC,Start,1)
4  if (startValue == 1) then
5      PTP(P1,100,-1,0)
6  else
7      PTP(P2,100,-1,0)
8  end
9  value1,value2,value3,value4,value5,value6 = ModbusMaste
```

Figure 9.18-30 Reading Multiple Digital Outputs and Storing Variables

5. Read DIgital input DI (discrete input)

Open the ModbusTCP master station command addition page, select “Modbus master name” as the master station “PLC” previously added in the Modbus master station setting page, the DI name is “Servo arrive”, the number of registers is 2, and click “Read digital input”. Finally, scroll to the bottom of the page and click the “Apply” button (Figure 38).

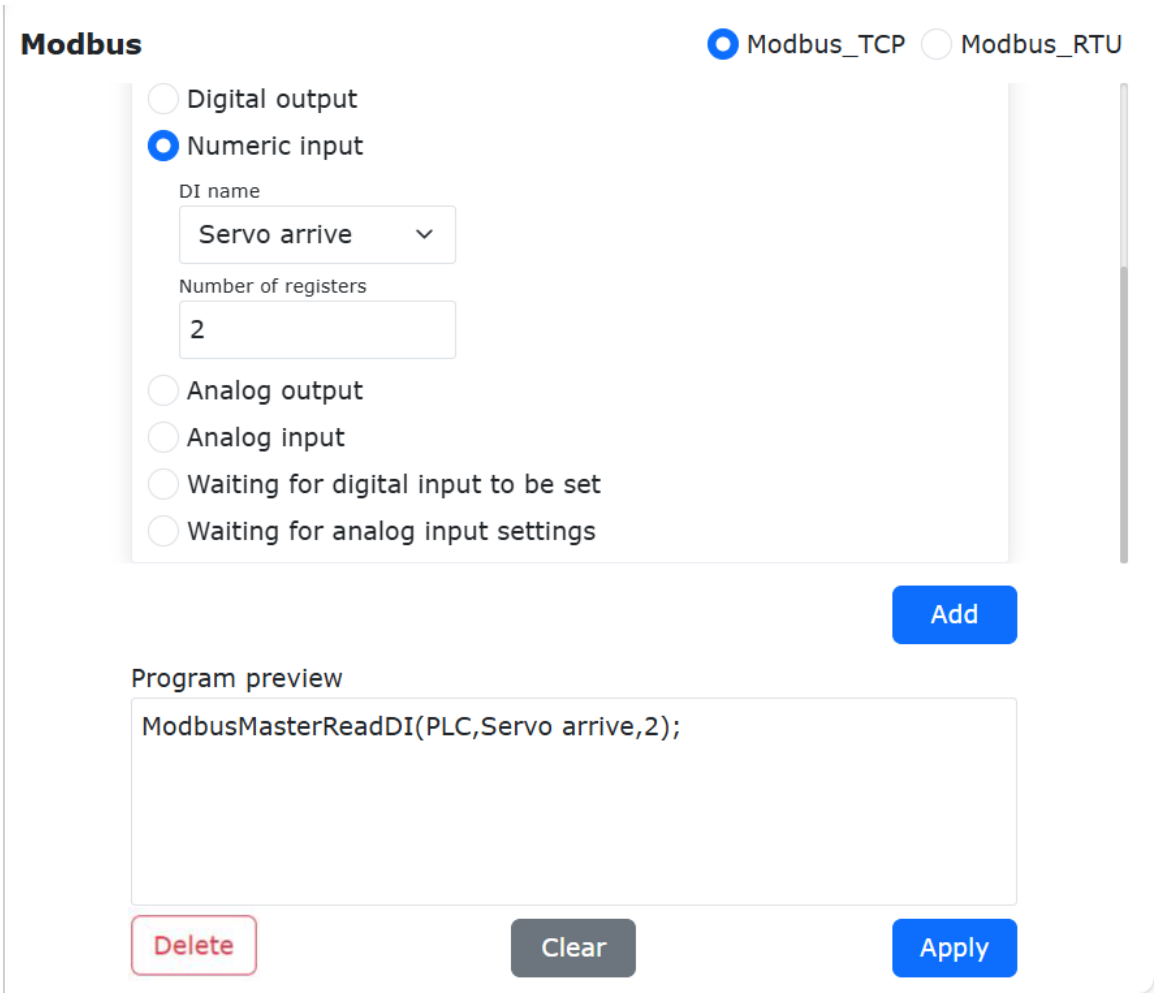


Figure 9.18-31 Configuring Read Digital Input

At this time, the robot program "testModbusMaster.lua" has added an command for the Modbus master station to read digital input.

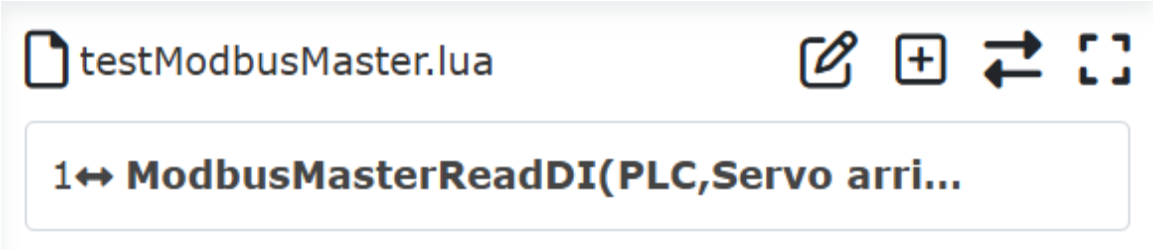


Figure 9.18-32 Reading digital input program commands

Click the "Switch Mode" button "" to switch the robot lua program to an editable state, and write the return value variables "state1,state2" before the "ModbusMasterReadDO" command. The two digital input values read after executing the program will be stored in the variables "state1" and "state2" respectively, and you can control the robot to do different operations by judging the variable values.

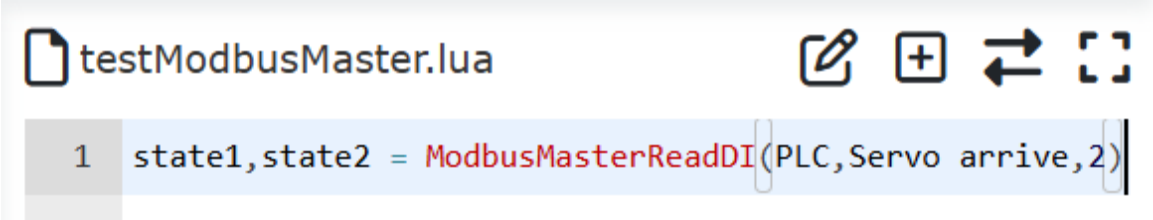


Figure 9.18-33 Reading digital input and storing variables

6. Read and write analog input AI (input register) and analog output AO (holding register) The reading and writing operations of analog input AI (input register) and analog output AO (holding register) are basically the same as those of digital input DI (discrete input) and digital output DO (coil), but the difference is that the data range of the latter is only 0 or 1, while the data range of the former is larger, so the specific operations can refer to the programming of digital input and digital output, and only the reading input AI (Figure 41) and writing analog output AO are shown here.

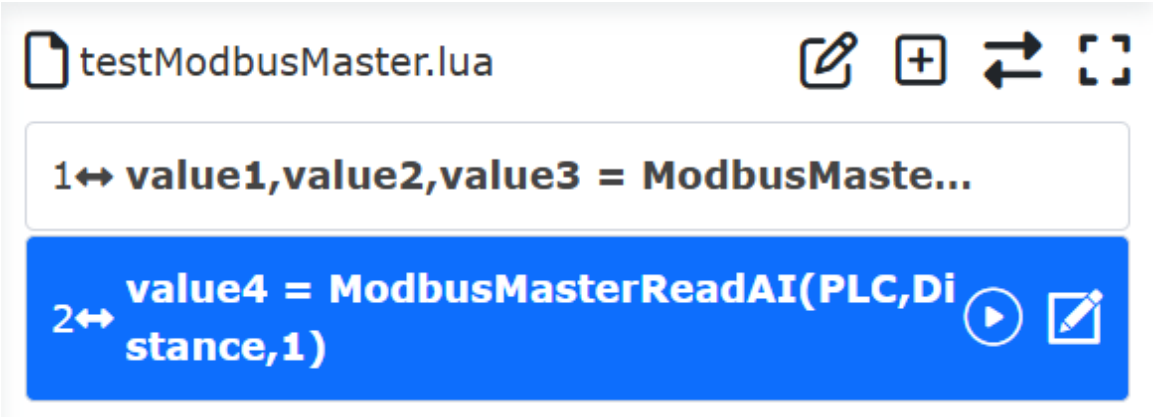


Figure 9.18-34 Reading analog input AI

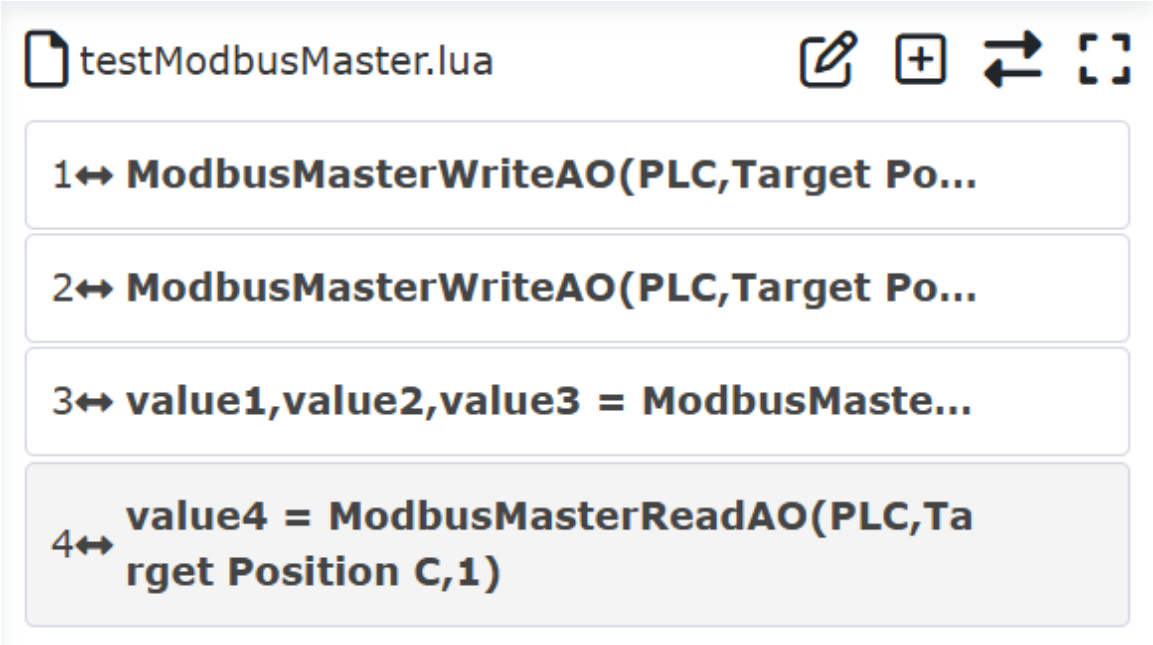


Figure 9.18-35 Read and write analog output AO

7. Waiting for digital input

Open the ModbusTCP master station command addition page, find the “Wait for digital input to be set”, that is, wait for DI discrete input setting, select the “servo in place” register with DI name as the configuration, the waiting status is “True” and the timeout time is 5000 ms. Click “Add” button, and finally click “Apply” button.

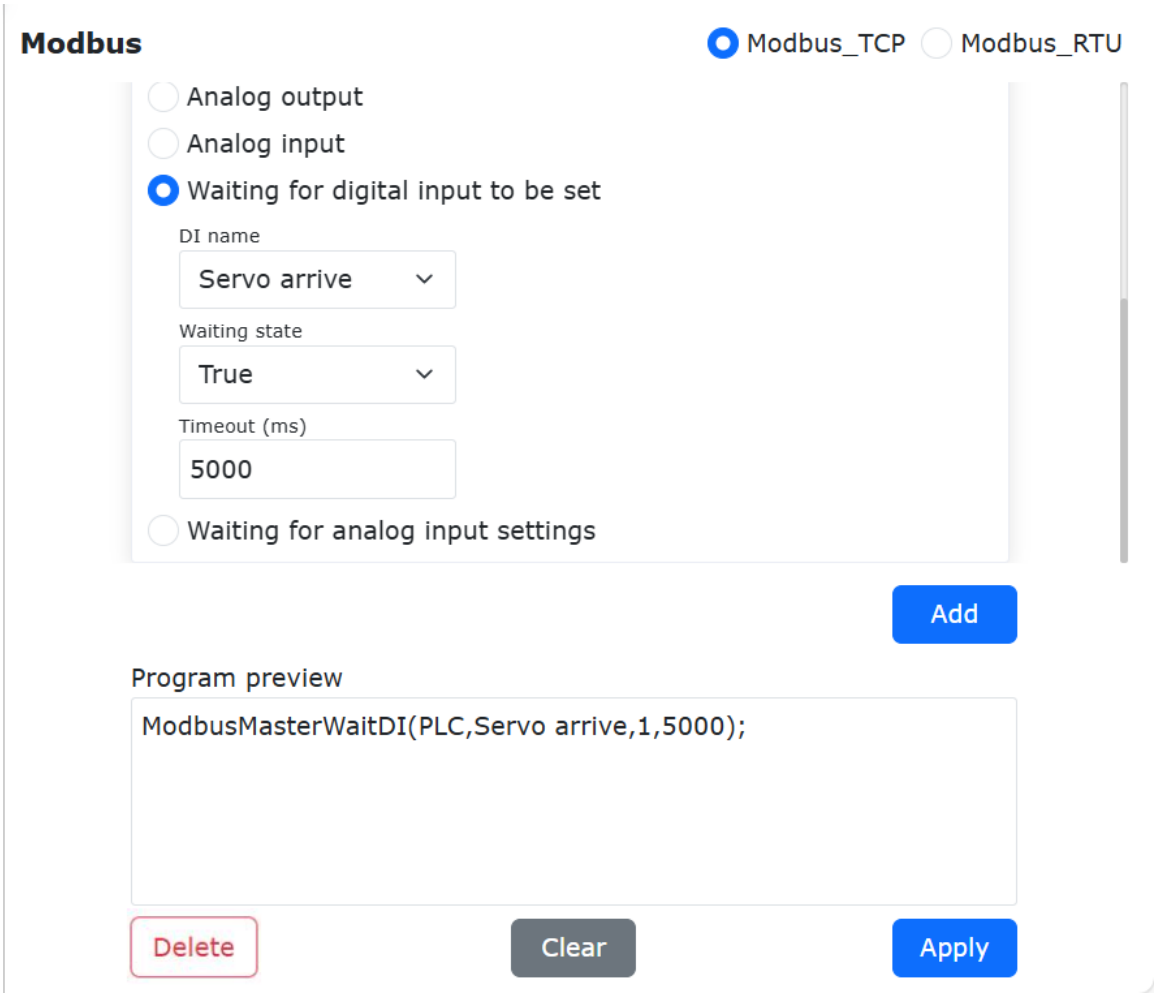


Figure 9.18-36 Add the command of waiting for DI input

At this time, the robot program “testModbusMaster.lua” has added an command that the robot Modbus master station waits for DIgital input DI. After starting the program, the robot will wait for the value of the “Servo arrive” register of the “PLC” master station to become true, that is, the value 1. Since the set timeout is 5s, when the “Servo arrive” signal is still 0 after the robot waits 5s, the robot program will report a timeout error and the program will stop.



Figure 9.18-37 Waiting for Digital Input DI Program

8. Waiting for analog Input

Open the ModbusTCP master station command addition page, find “Waiting for analog input setting”, that is, waiting for the setting of AI input register, select the configured “liquid level” register with AI name, waiting status of “>”, register value of 255 and timeout time of 5000 ms.. Click “Add” button, and finally click “Apply” button.

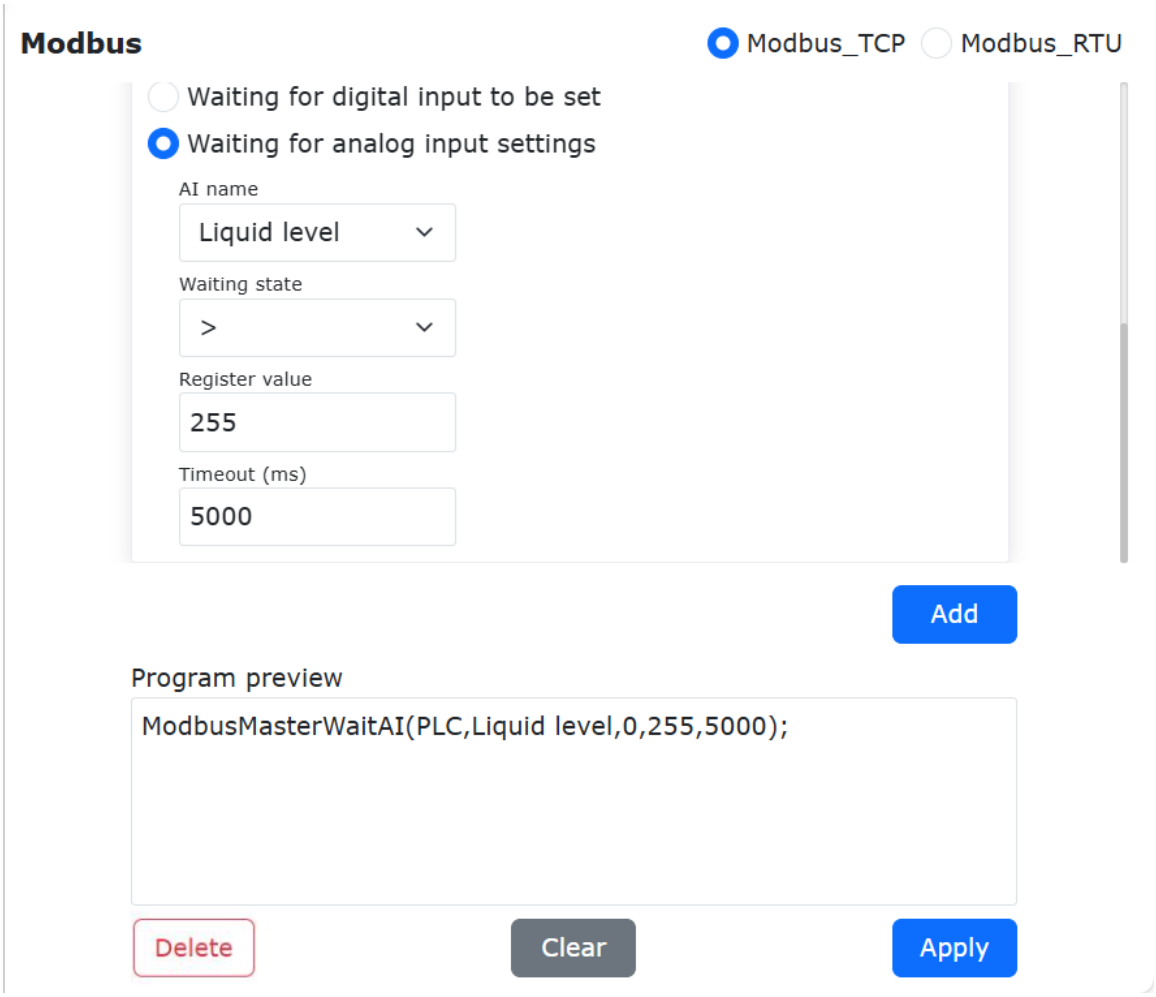


Figure 9.18-38 Add Waiting for Analog Input

At this time, the robot program “testModbusMaster.lua” has added an command that the robot Modbus master station waits for the AI input register value. After starting the program, the robot will wait for the “Liquid level” register value of the “PLC” master station to be greater than 255. Since the set timeout is 5s, when the “Liquid level” signal is still less than 255 after the robot waits for 5s, the robot program will report a timeout error and the program will automatically stop running.



Figure 9.18-39 Waiting for AI Input Register Program

Open the ModbusTCP Master command addition page, find the “Wait for Analog Input Setting” (i.e., wait for AI input register setting). Select the AI name as the configured “Liquid Level” register, set the wait condition to “=”, the register value to 255, and the timeout to 5000ms. Click the “Add” button, and finally click the “Apply” button.

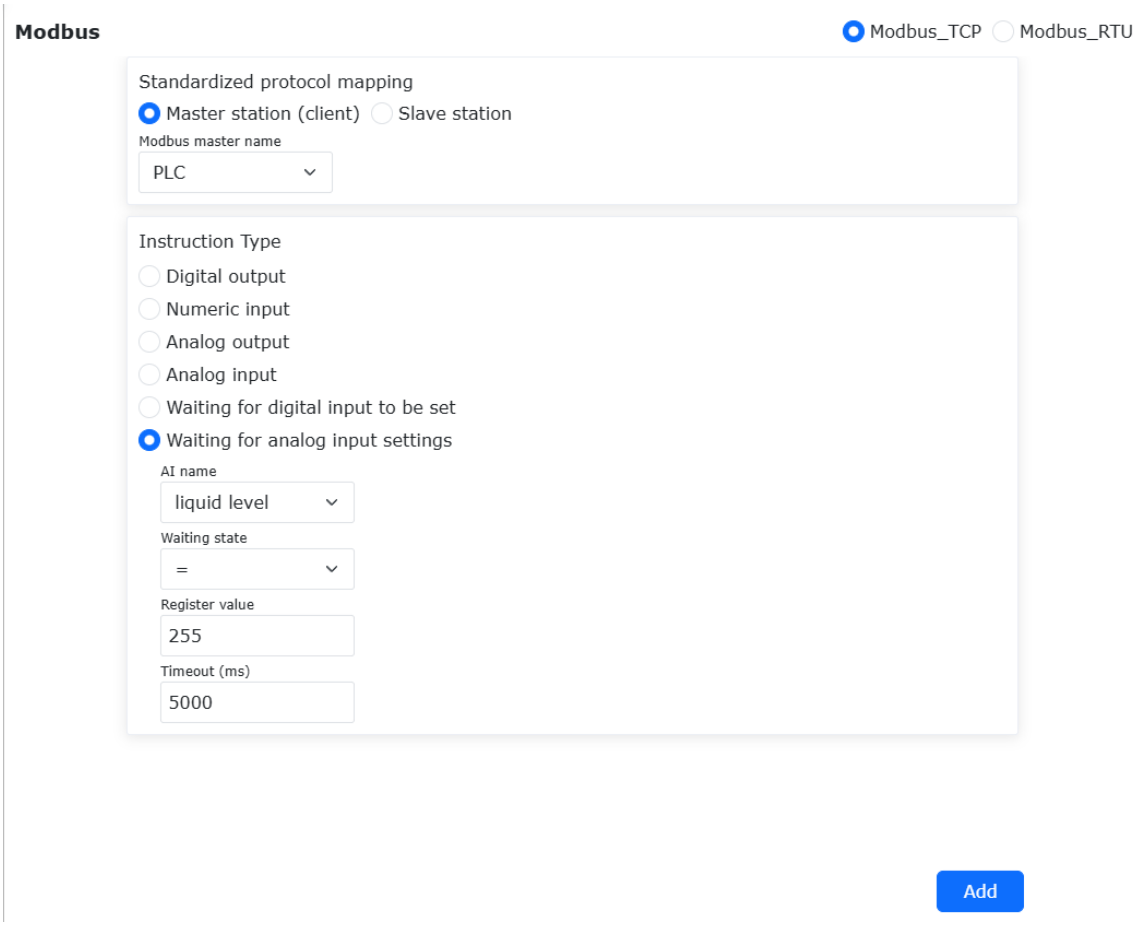


Figure 9.18-39-2 Add Wait for Analog Input

At this point, a Modbus Master command to wait for the AI input register value has been added to the robot program “test.lua”. After starting the program, the robot will continuously wait for the “Liquid Level” register value from the “PLC” master to equal 255. Since the timeout is set to 5s, if the “Liquid Level” signal still does not equal 255 after the robot waits for 5 seconds, the robot program will report a timeout error and automatically stop running.

1.3.9.18.3 ModbusTCP slave

Robot ModbusTCP slave station provides four types of registers: Digital input (coil), Digital output (discrete input), Analog input (holding register) and Analog output (input register). The digital input and analog input are mainly used for the robot to read the data of the external ModbusTCP master station to control the robot operation, while the digital output and analog output are mainly used for the robot to send data signals to the external ModbusTCP master station equipment, and the external master station equipment reads the relevant register values to control the equipment operation. In addition to the above-mentioned general input and output, the robot also provides some “Functional digital input (coils)” for the external master station equipment to control the robot’s start-up program and stop-up program, and provides some input registers to display the current robot’s state information, including the robot’s current Cartesian position, the robot’s current running state, etc. (please refer to Annex 1: ModbusTCP slave address mapping table). The use process of robot ModbusTCP slave station mainly includes: parameter configuration; Communication test; Programming.

1.3.9.18.3.1 ModbusTCP slave communication parameter configuration

Open the WebApp, click “Teaching” and “Program Teaching” in turn to create a new user program “testModbus-Slave.lua”.

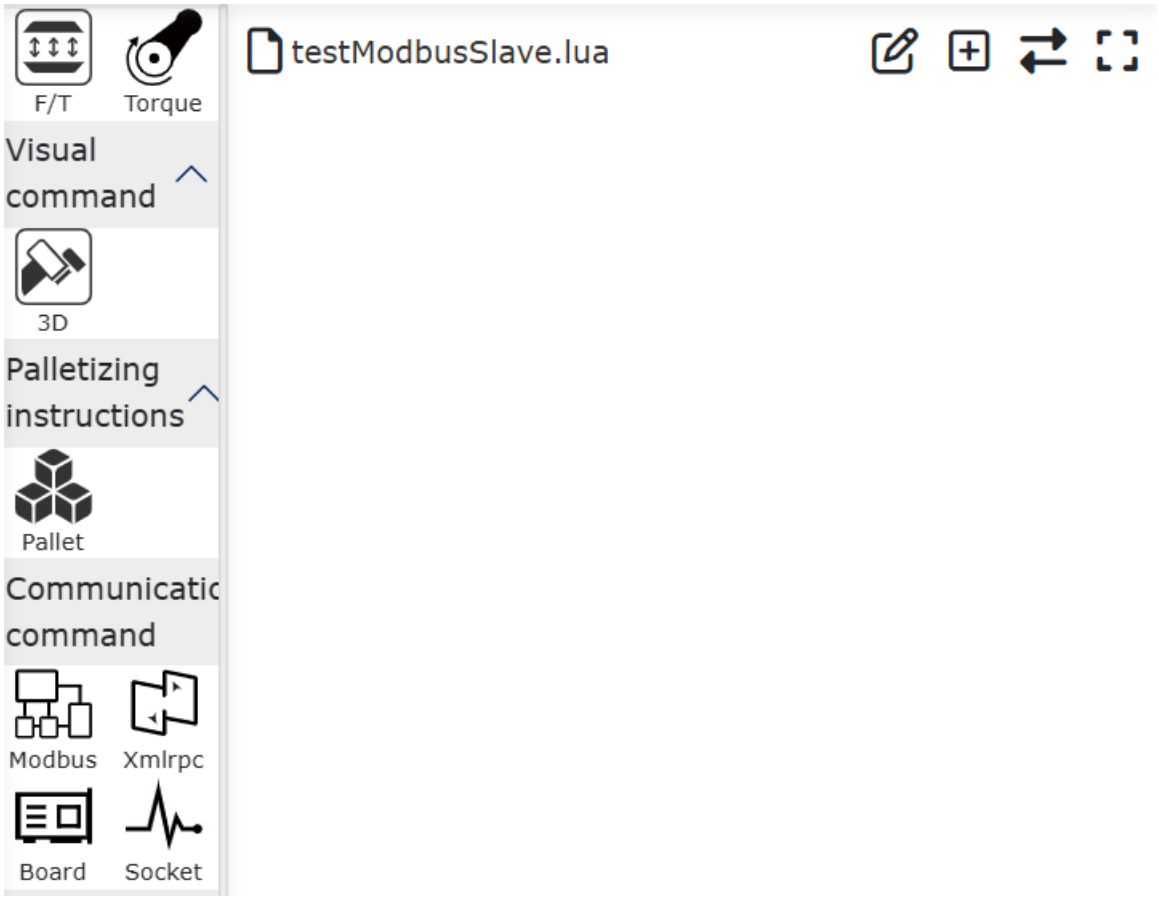


Figure 9.18-40 Create ModbusTCP slave station user program

Click “ModbusTCP Settings” button to open the ModbusTCP function configuration page.

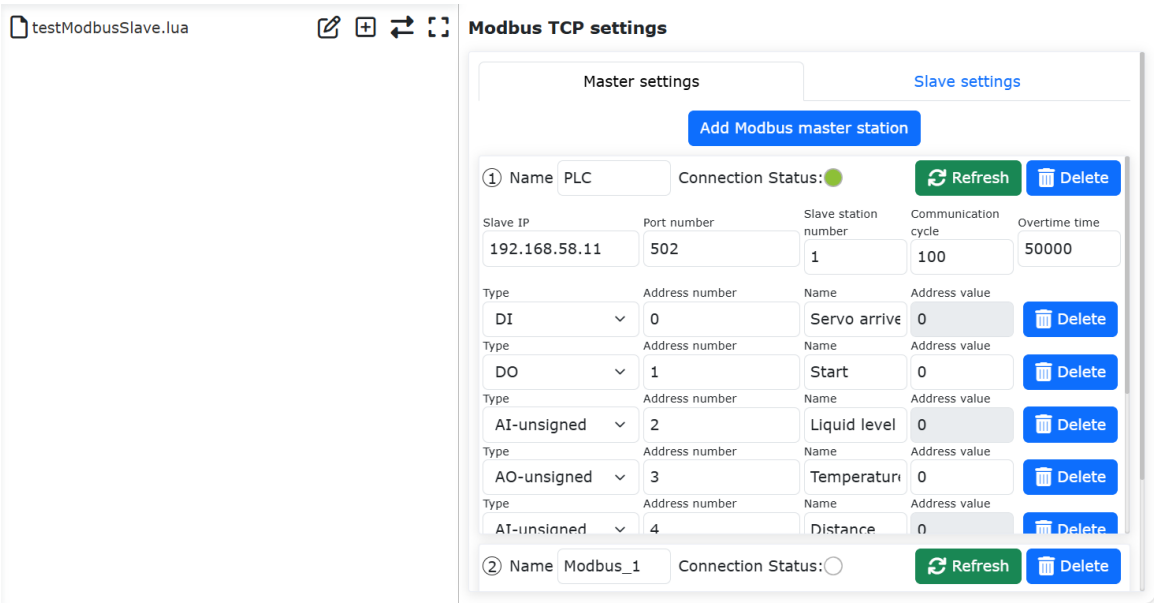


Figure 9.18-41 Open ModbusTCP Settings

Click “Slave Settings” in turn, and enter the IP, Port number and Slave station number of the robot slave station, where “IP” is the IP address of the robot slave station, FAIRINO cooperative robot has two network ports, the teaching device and the control box and the IP addresses of the two ports are different. Enter the correct IP address according to the network port where the external device is connected to the robot slave station (it is recommended that you use the network port on the control box).You have to restart robot after change the IP address, port number or slave station number of the robot ModbusTCP slave station,otherwise it won’t take effect.

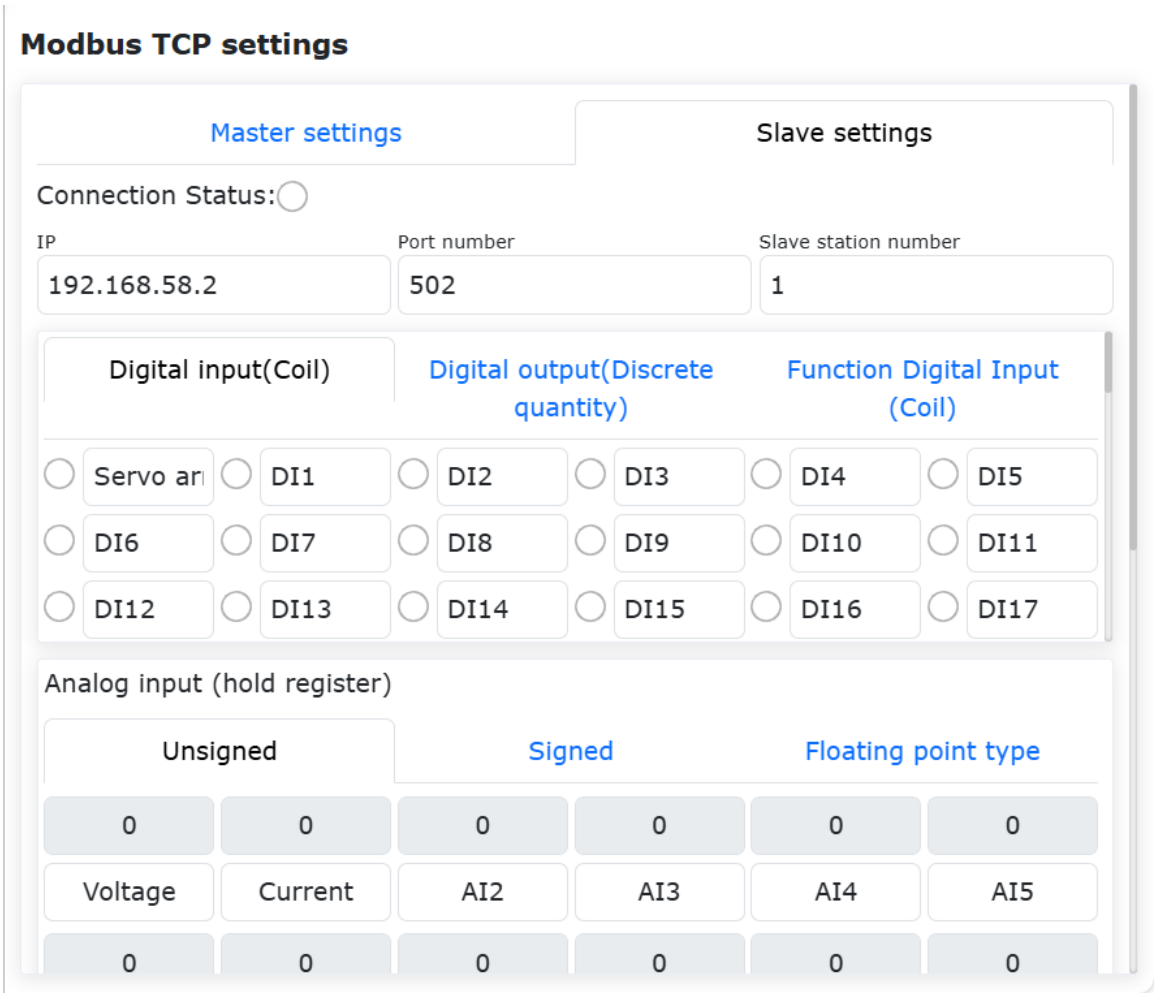


Figure 9.18-42 ModbusTCP slave settings

After the parameters of ModbusTCP slave station are set and the robot is restarted, the external master station equipment can establish a connection with the robot slave station through the set parameters. After the connection is successful, the “Connection status” indicator light on the robot slave station setting page will light up.



Figure 9.18-43 slave station connection status indicator light

1.3.9.18.3.2 ModbusTCP slave communication test

1. Digital input (coil)

The robot ModbusTCP slave station provides 128 coil registers, and their register addresses are 100~127.

Note: Please refer to Annex I: ModbusTCP slave station address mapping table.

Aliases can be set for the general registers of robot ModbusTCP slave stations as shown in Figure 52. Modify the name of the robot slave station coil register DI0 to be “A in place” and the name of DI1 to be “B in place”. According to the address mapping table, the Modbus coil addresses of “A in place” and “B in place” are 100 and 101, respectively. Set the robot slave station coil register addresses 100 and 101 to 1 on the external ModbusTCP master station equipment, At this time, two register indicators on the monitoring page of the robot ModbusTCP slave station light up .

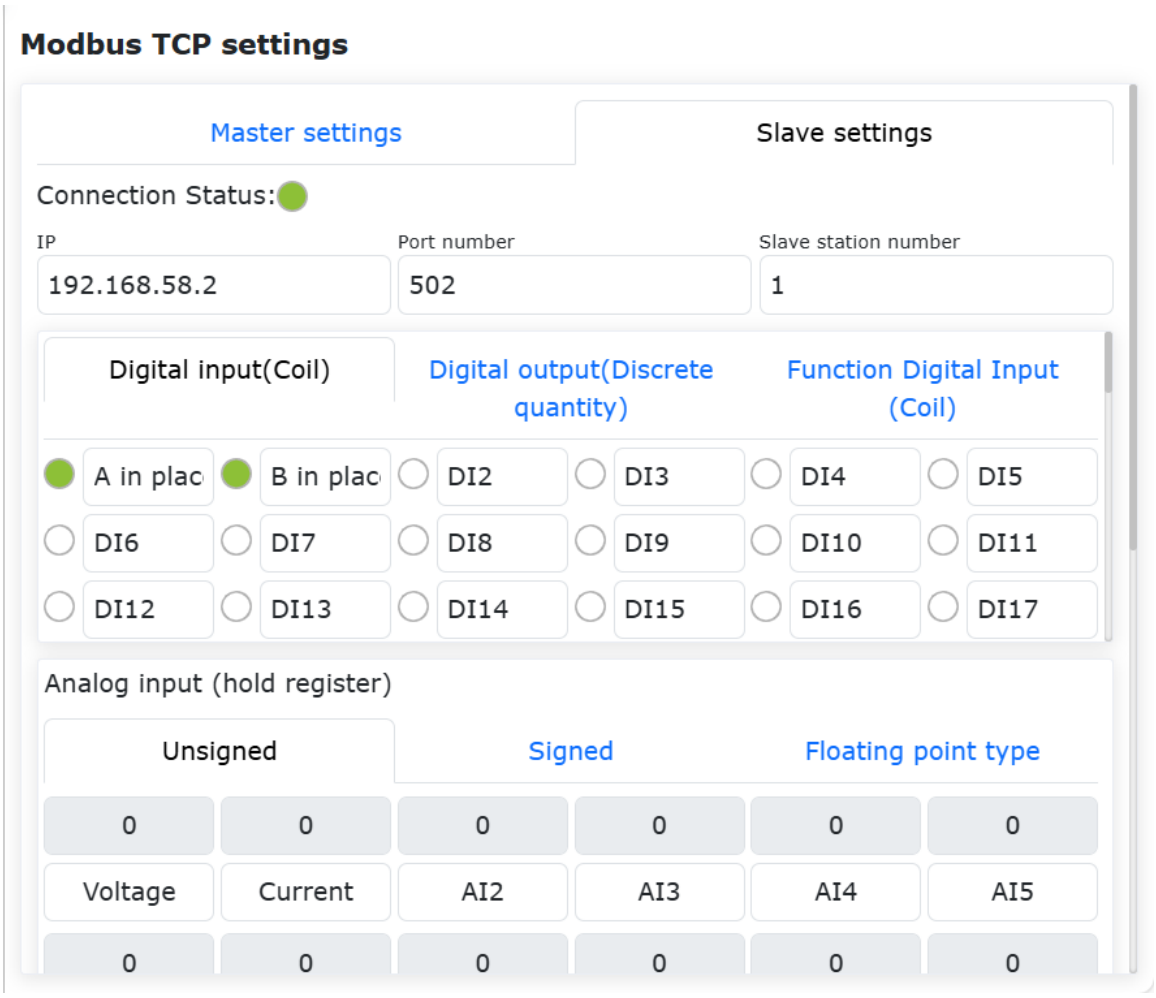


Figure 9.18-44 ModbusTCP slave station coil state monitoring

2. Digital output (discrete input)

The robot ModbusTCP slave station provides 128 discrete input registers, and their register addresses are 100~127.

Note: Please refer to Annex 1: ModbusTCP slave station address mapping table for specific definitions.

Similarly, the robot ModbusTCP slave's discrete input register can also be set with aliases, as shown in Figure 53. Click "Digital Output (Discrete Input)" to modify the name of the robot slave's discrete input register DO0 as "A Start" and DO1 as "B Start". According to the address mapping table, the Modbus discrete input addresses of "A Start" and "B Start" are 100 and 101 respectively. Click "A Start" to correspond to the discrete input indicator light, which lights up, and the value of the corresponding register address 100 becomes 1, which can be read from the external ModbusTCP master station equipment.

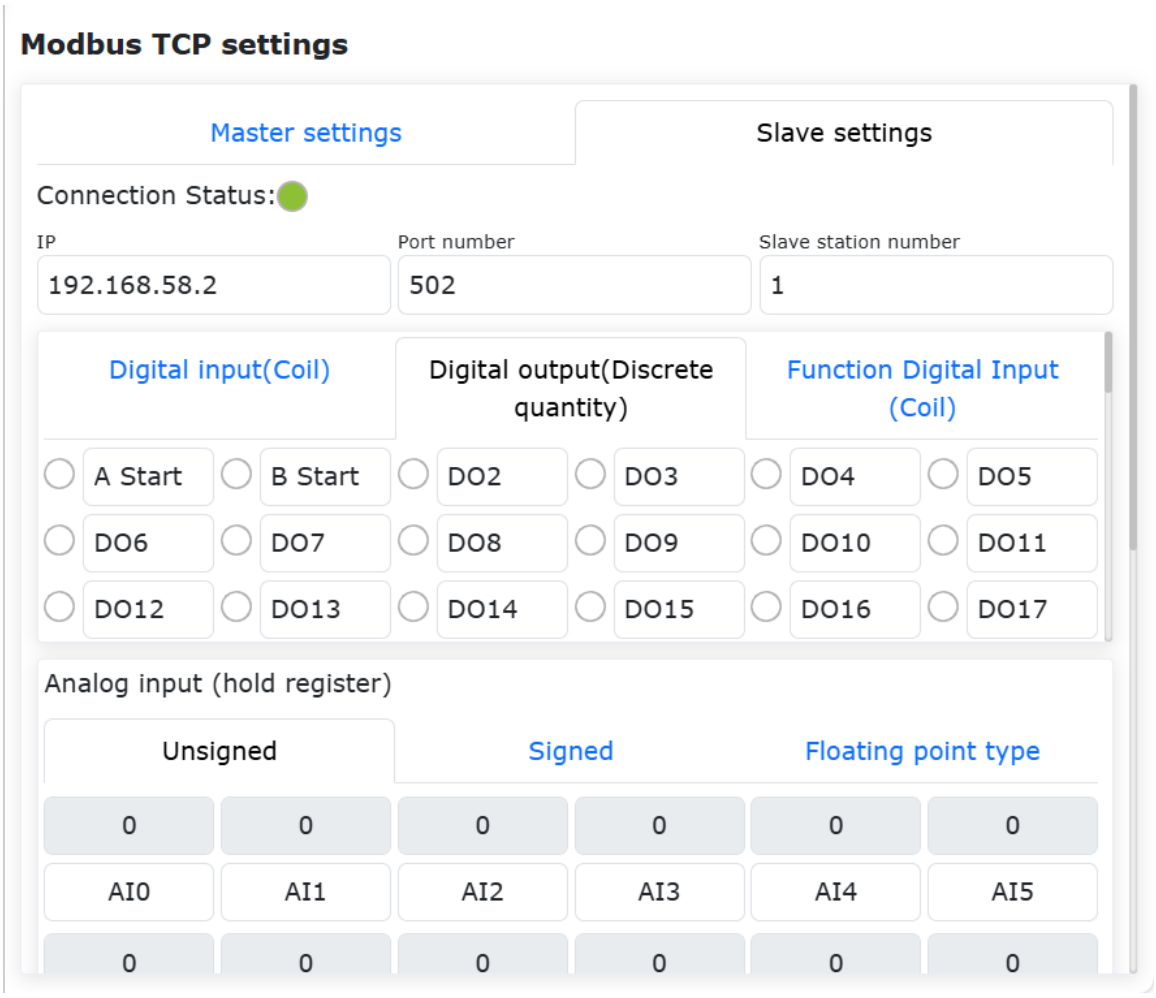


Figure 9.18-45 ModbusTCP slave discrete input control

3. Analog input (holding register)

The robot provides 64 holding registers of unsigned, signed and float types, with the addresses of AI0~AI63 ranging from 100 to 195.

Note: text Please refer to Annex 1: ModbusTCP slave address mapping table for specific definitions, in which the data range of unsigned register is 0~65535, the data range of signed register is -32768~32767, and the float register is displayed at the big end.

As shown in Figure 53, the names of AI0 and AI1 are changed to “voltage” and “current” respectively, and the addresses of the two registers are found to be 100 and 101 respectively from the ModbusTCP slave address mapping table. Therefore, when the connected master station equipment modifies the register address values of the holding registers 100 and 101, the register address values of “voltage” and “current” on the ModbusTCP slave station monitoring page of the robot are updated and displayed synchronously, and the robot’s analog input is mainly used for reading the external master station equipment values.

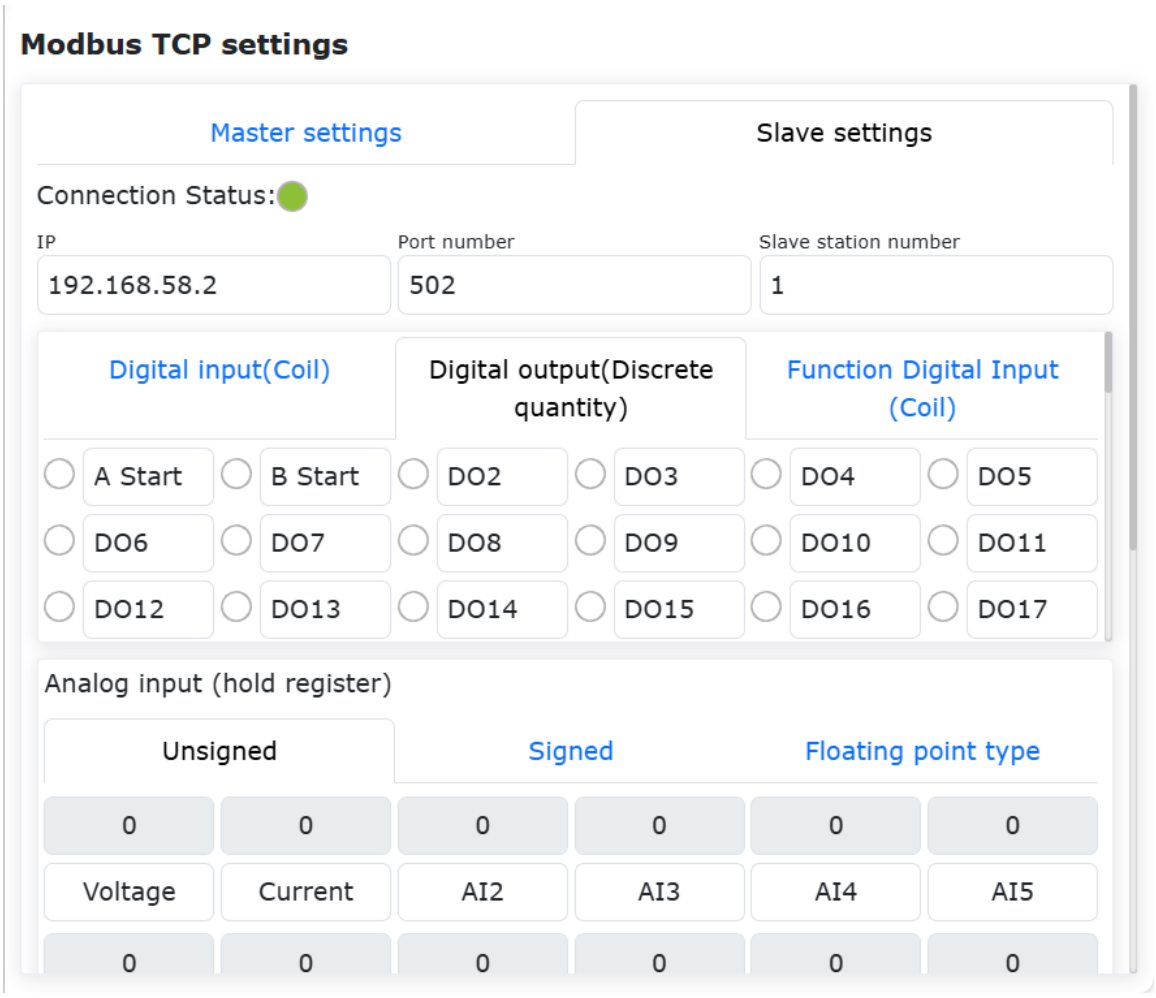


Figure 9.18-46 ModbusTCP slave analog input monitoring

4. Analog output (input register)

The robot provides 64 input registers of unsigned, signed and floating-point types, with the addresses of AO0~AO63 ranging from 100 to 195.

Note: text please refer to Annex 1: ModbusTCP slave address mapping table, in which the data range of unsigned register is 0~65535, the data range of signed register is -32768~32767, and the floating-point register is displayed at the big end.

As shown in figure 55, the names of AO0 and AO1 are changed to “position A” and “position B” respectively, and the values of the input registers are 2000 and 1500 respectively. The addresses of the two registers are 100 and 101 respectively from the station address mapping table of ModbusTCP. Therefore, when the connected master station equipment reads the address values of the input registers 100 and 101, the set values can be obtained, and the simulated output of the robot slave station is mainly used for the robot to transfer to the external master station equipment.

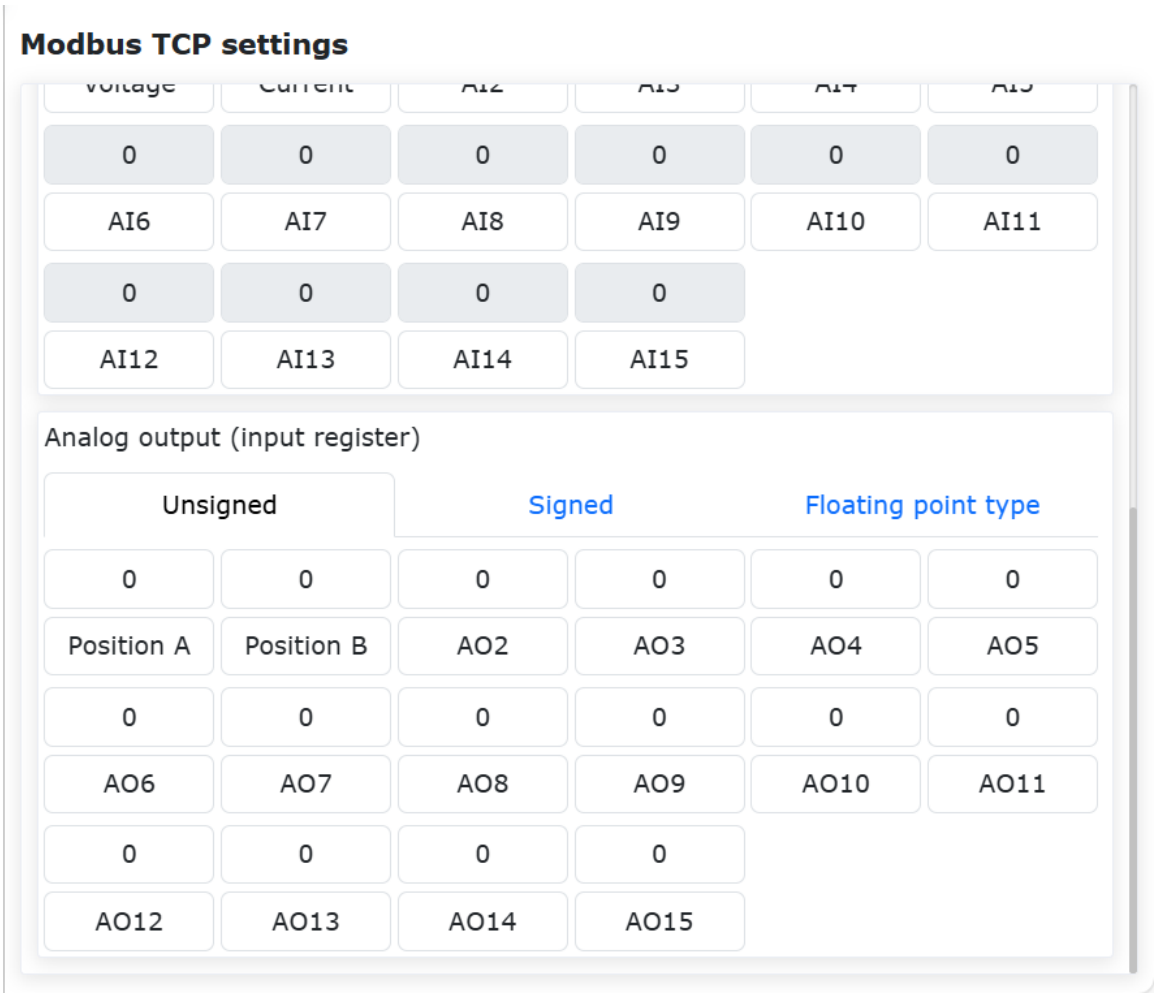


Figure 9.18-47 Modbus slave station modifies analog input

1.3.9.18.3.3 ModbusTCP slave programming

Click “All” and “Communication command” in turn to open the communication command adding page.

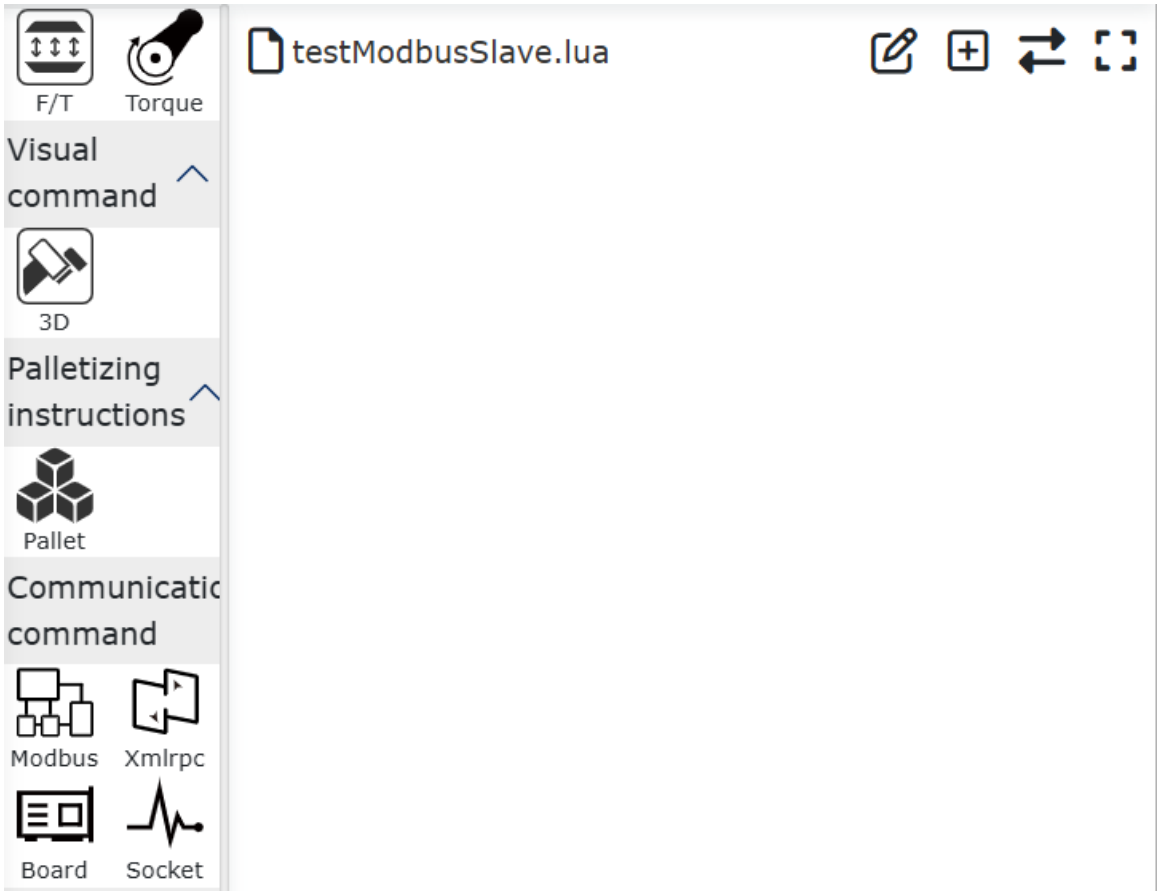


Figure 9.18-48 Open the communication command add page

Click “Modbus”.

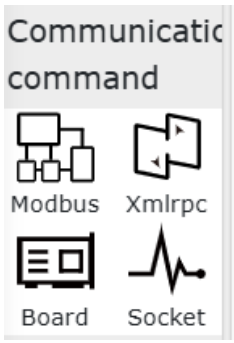


Figure 9.18-49 Select Modbus

Click “Modbus_TCP”.

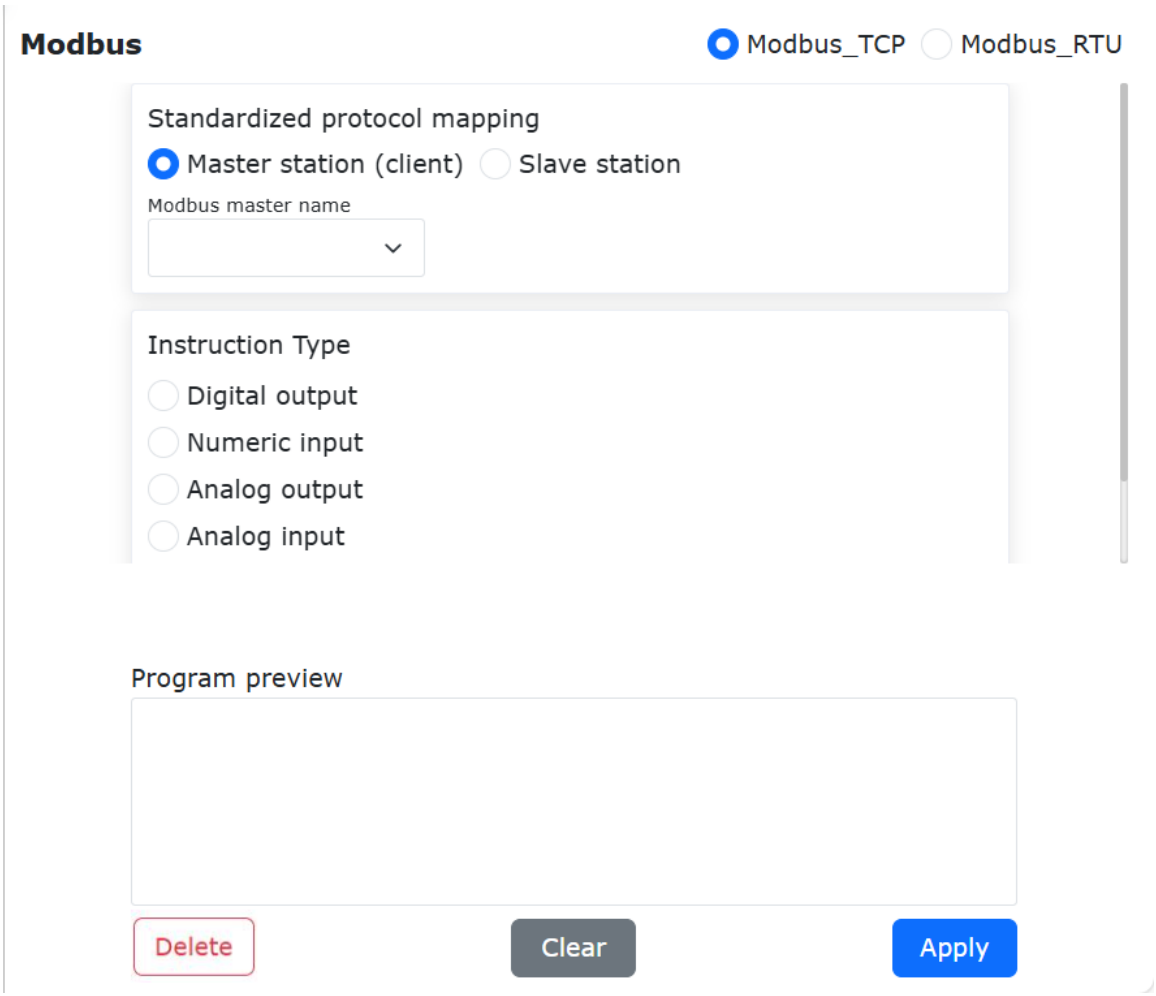


Figure 9.18-50 Select Modbus_TCP

Select “slave station” to open the ModbusTCP slave station command adding page (figure 60).

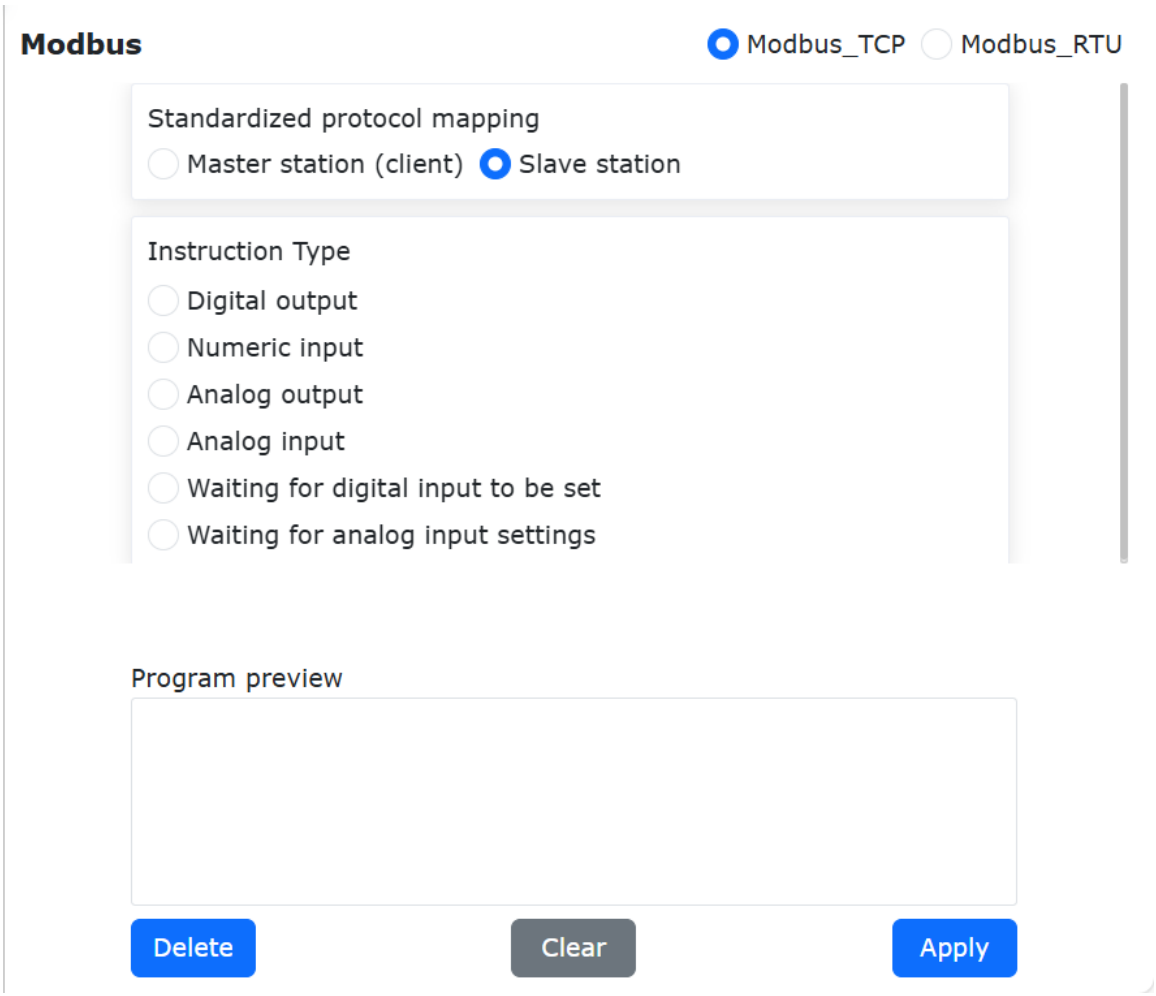


Figure 9.18-51 slave command addition

1. Write a single digital output DO (discrete input)

Select the DO name as “A Start”, the number of registers is 1 and the register value is 0, and click “Write Single Digital Output”. Finally, scroll to the bottom of the page and click the “Apply” button (Figure 62).

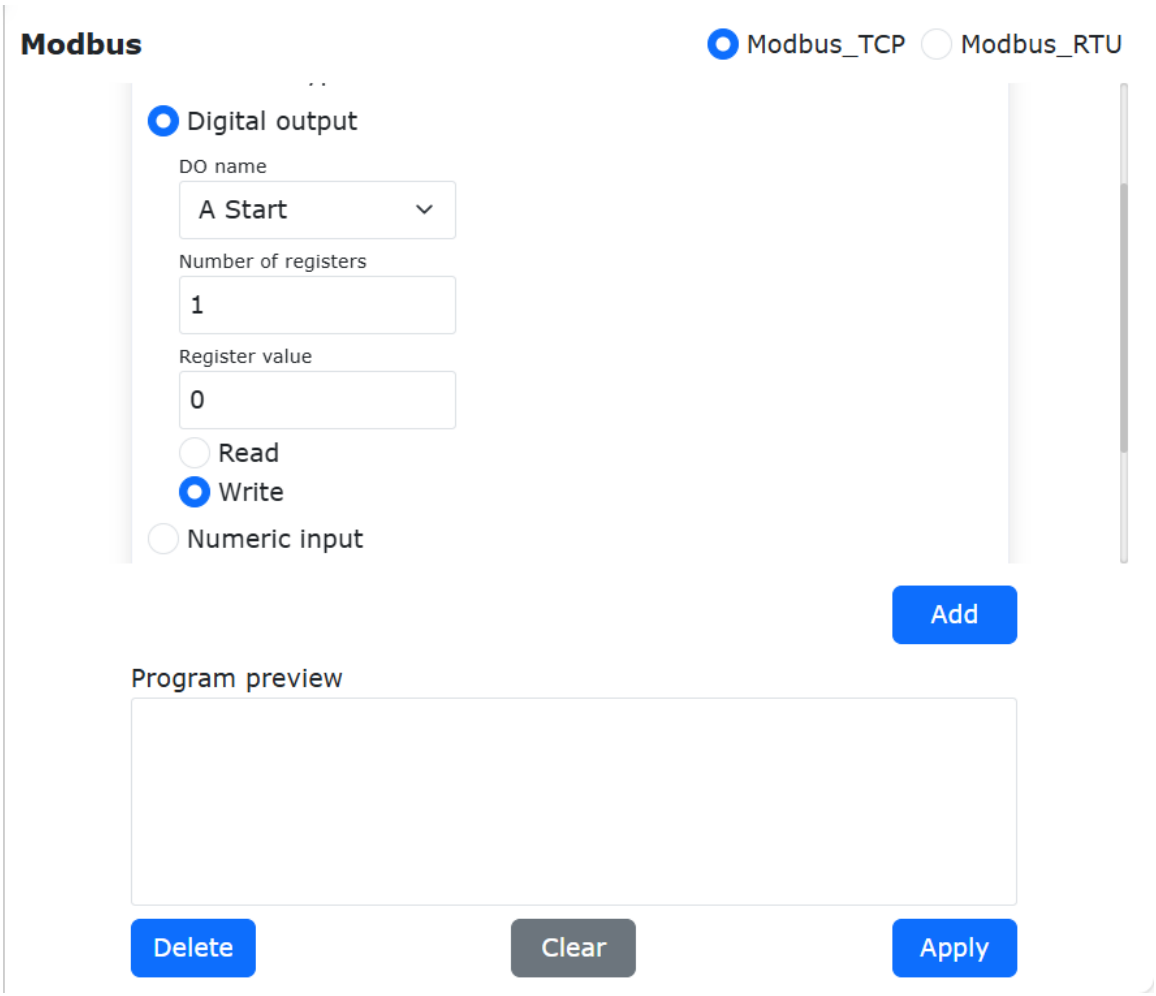


Figure 9.18-52 Adding Write Single Digital Output command

At this time, the robot program “testModbusSlave.lua” has added an command for the robot Modbus slave station to write a single digital output, switch the robot to the automatic mode, click the start button, and the robot will write the address value of the corresponding digital output named “A Start” as 0.

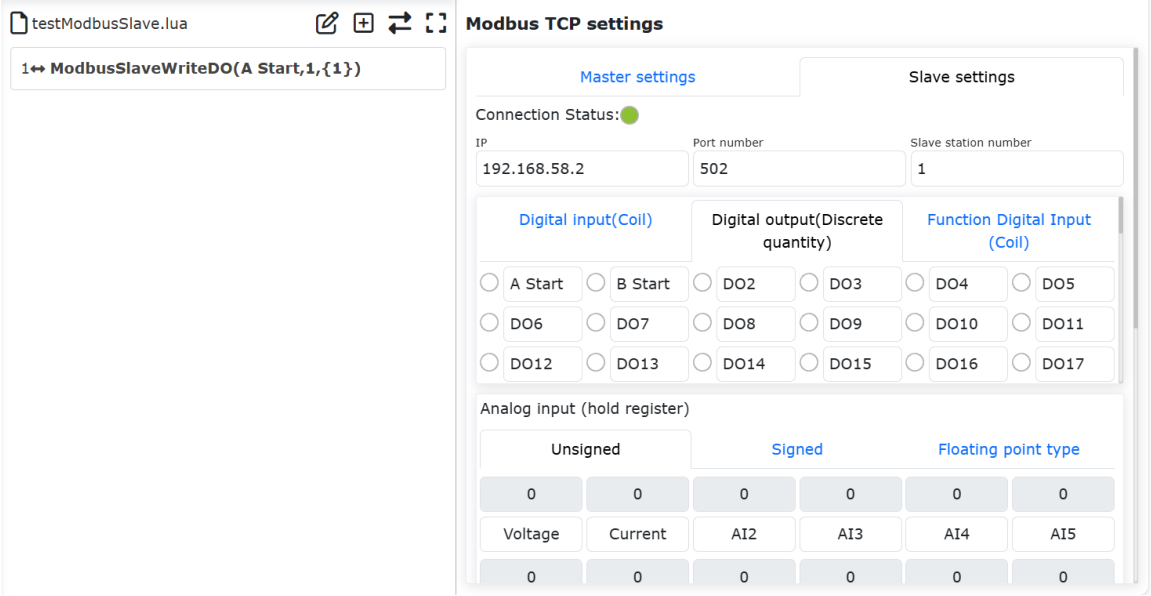


Figure 9.18-53 Write a single digital output LUA program

2. Write multiple digital outputs DO (discrete input)

Open the ModbusTCP slave command addition page, find the “Digital Output Settings”, select the DO name as “A Start”, the number of registers is 5, and the register values are 1,0,1,0,1, where the number of register values corresponds to the set number of registers, and multiple register values are separated by English commas, and click “Write Digital Output”. Finally, scroll to the bottom of the page and click the “Apply” button (Figure 65).

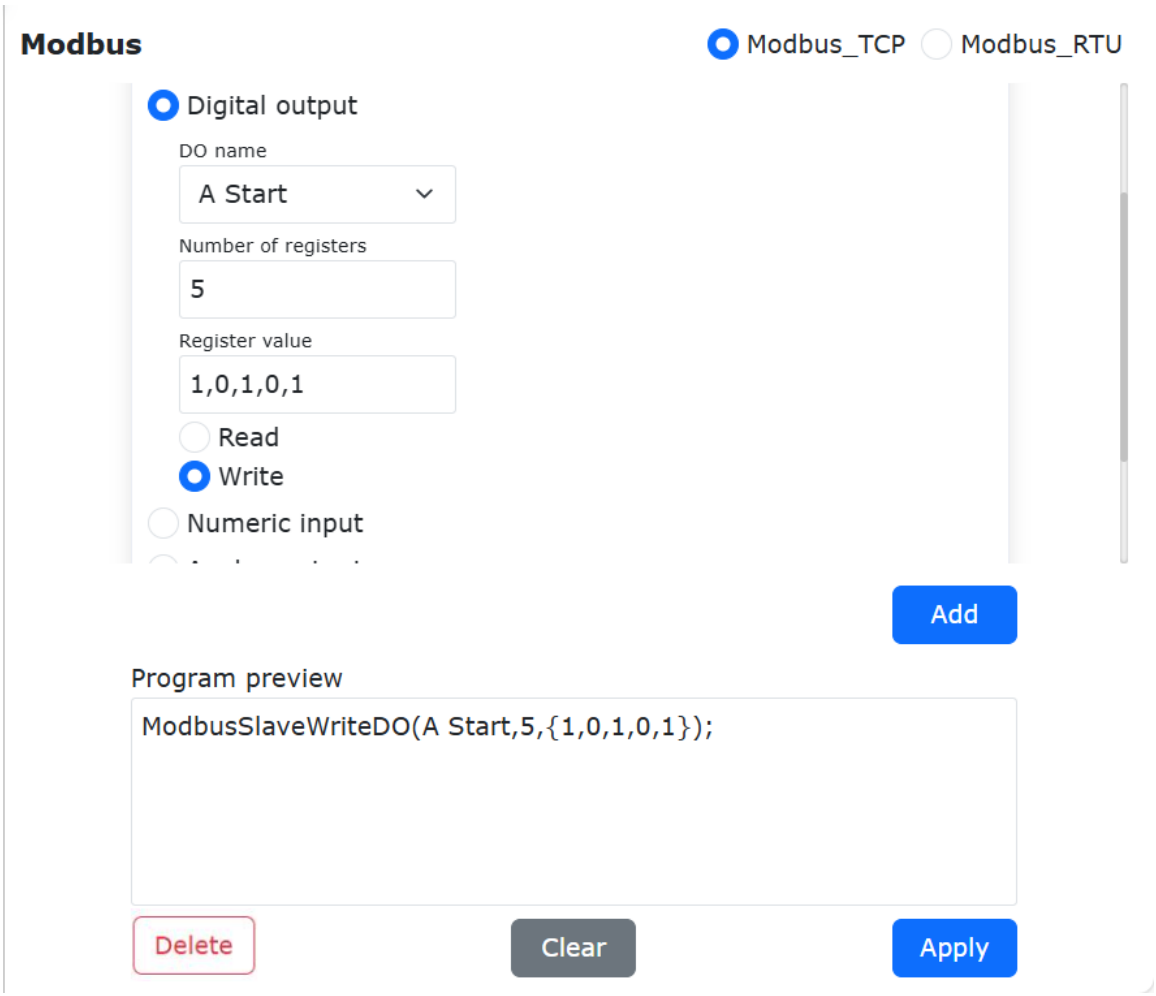


Figure 9.18-54 Configures Writing Multiple Digital Outputs

At this time, the robot program "testModbusSlave.lua" has added an command for the robot Modbus slave station to write multiple digital outputs. Switch the robot to the automatic mode, click the start button, and the robot will start the slave station "A" and write the values of its four discrete input registers as 1, 0, 1, 0 and 1 respectively.



Figure 9.18-55 Write multiple digital output LUA programs

3. Read a single digital output DO (discrete input)

Open the ModbusTCP master station command addition page, find “Digital Output Settings”, the name of DO is “A Start”, the number of registers is 1, and the register value does not need to be filled in. Click “Read Digital Output”. Finally, scroll to the bottom of the page and click the “Apply” button (Figure 68).

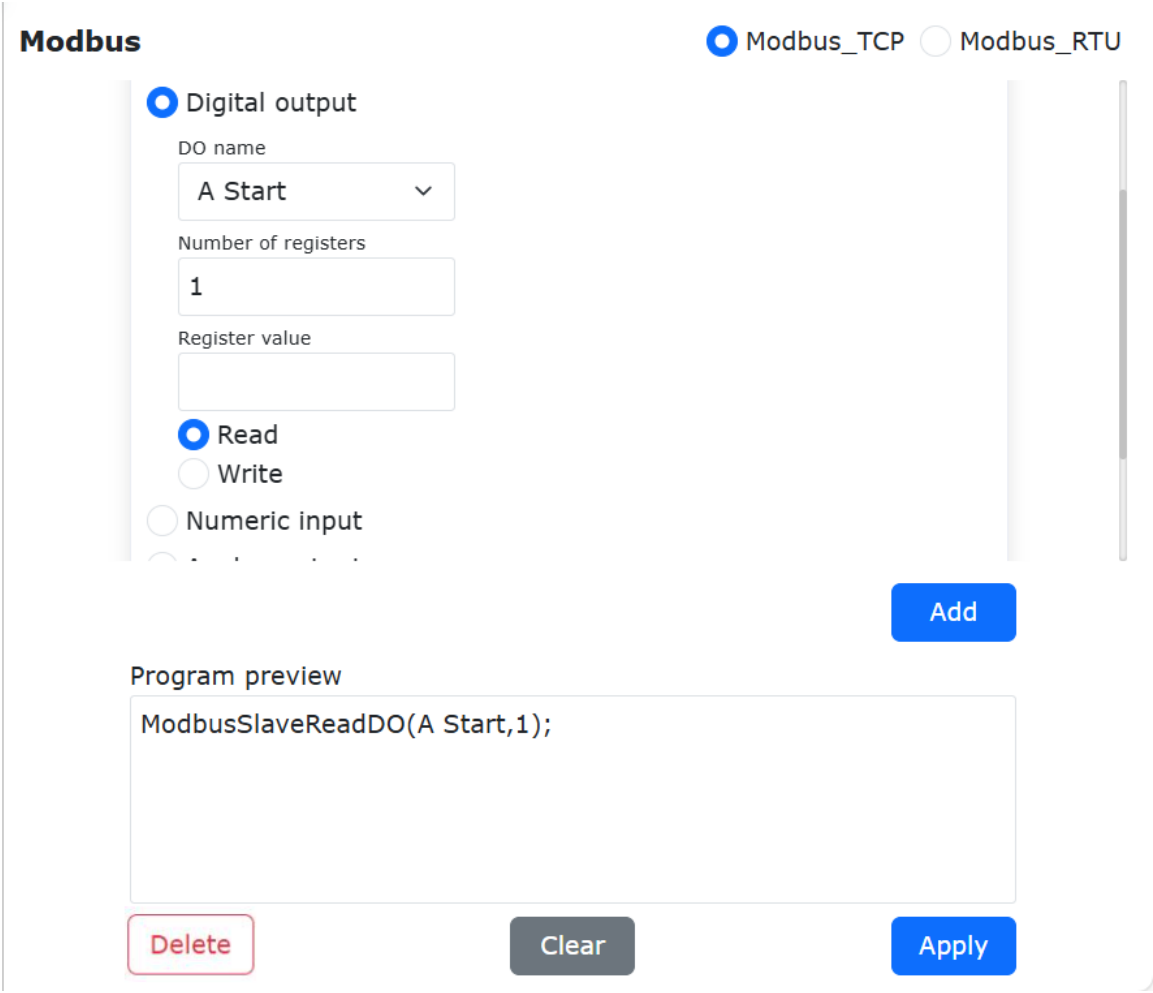


Figure 9.18-56 Configuring Reading a Single Digital Output

At this time, the robot program “testModbusSlave.lua” has added an command for the robot Modbus to read a single digital output from the station.

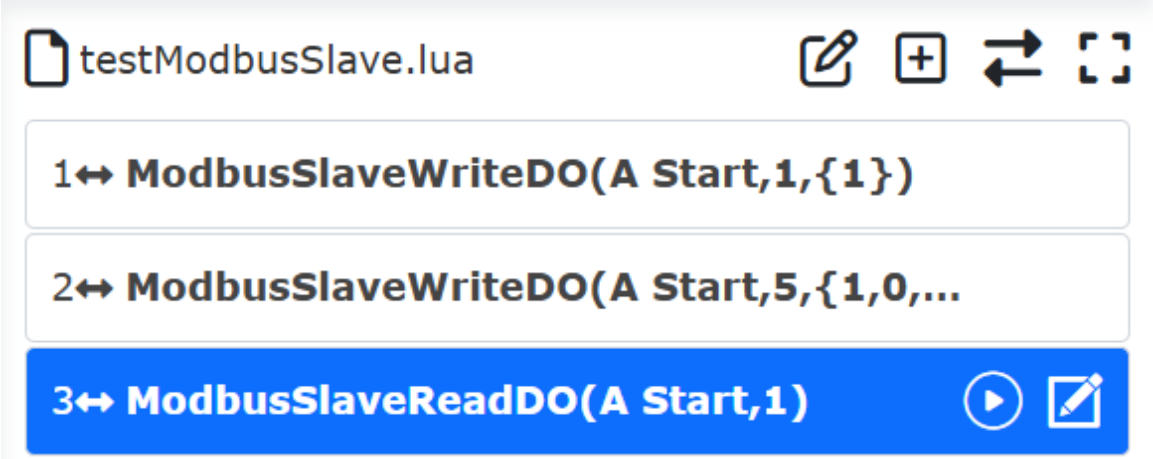


Figure 9.18-57 Read a single digital output program

Usually, after reading the Modbus register, the read value is stored in a variable, so it is necessary to define a variable to store the read value. As shown in figure 70, click the “switch mode” button “” to switch the robot lua program to the editable state, and write the variable “AStartValue” with added return value before the “ModbusSlaveReadDO” command, and the value read after executing the program will be stored in “AStartValue”.

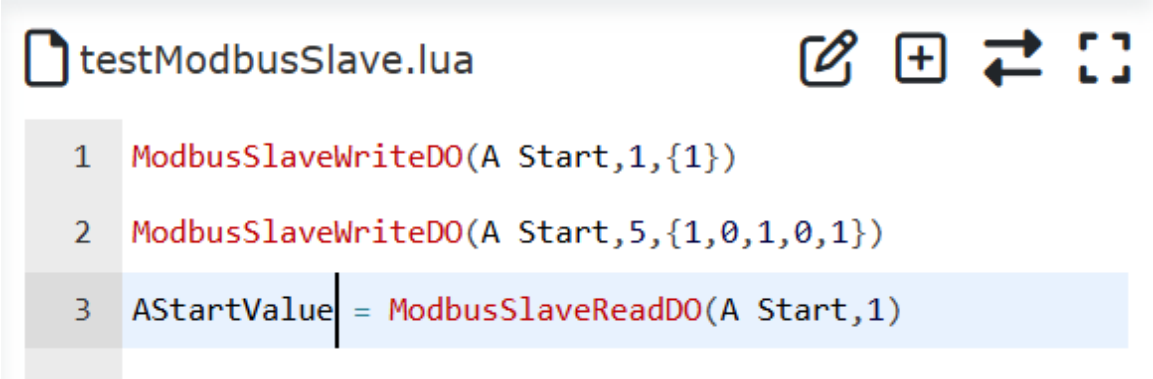


Figure 9.18-58 Reading a single digital output into a variable

The register value of coil type is only 0 and 1, so different operations can be carried out by judging the different register values in robot programs. As shown in Figure 71, click the “Switch Mode” button “” to switch the robot teaching program to the non-editable mode, and add two joint motion commands to move to two different points “P1” and “P2” respectively.

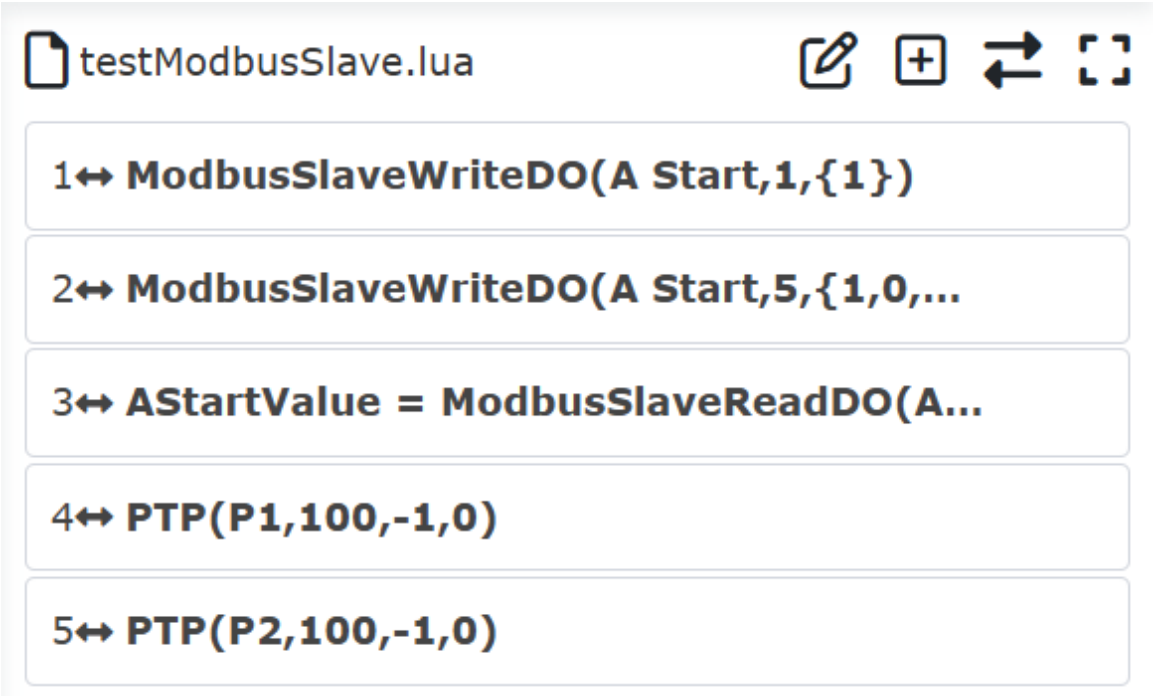


Figure 9.18-59 Add motion commands of different points.

As shown in Figure 72, the program is switched to the editable mode again, and the judgment condition of the digital output value “AStartValue” is written. When the value of “AStartValue” is 1, the robot moves to the point “P1”, otherwise the robot moves to the point “P2”.

testModbusSlave.lua

```
1  ModbusSlaveWriteDO(A Start,1,{1})
2  ModbusSlaveWriteDO(A Start,5,{1,0,1,0,1})
3  AStartValue = ModbusSlaveReadDO(A Start,1)
4  if (AStartValue == 1) then
5      PTP(P1,100,-1,0)
6  else
7      PTP(P2,100,-1,0)
8  end
```

Figure 9.18-60 Move to different points according to different digital output values

Finally, switch the robot program to non-editable mode, switch the robot to automatic mode, and start the running program on the premise of confirming safety. As shown in figure 73, the second line of the program sets the DO value of the digital output named “A Start” to 1, so the robot will move to the “P1” point after executing the program.

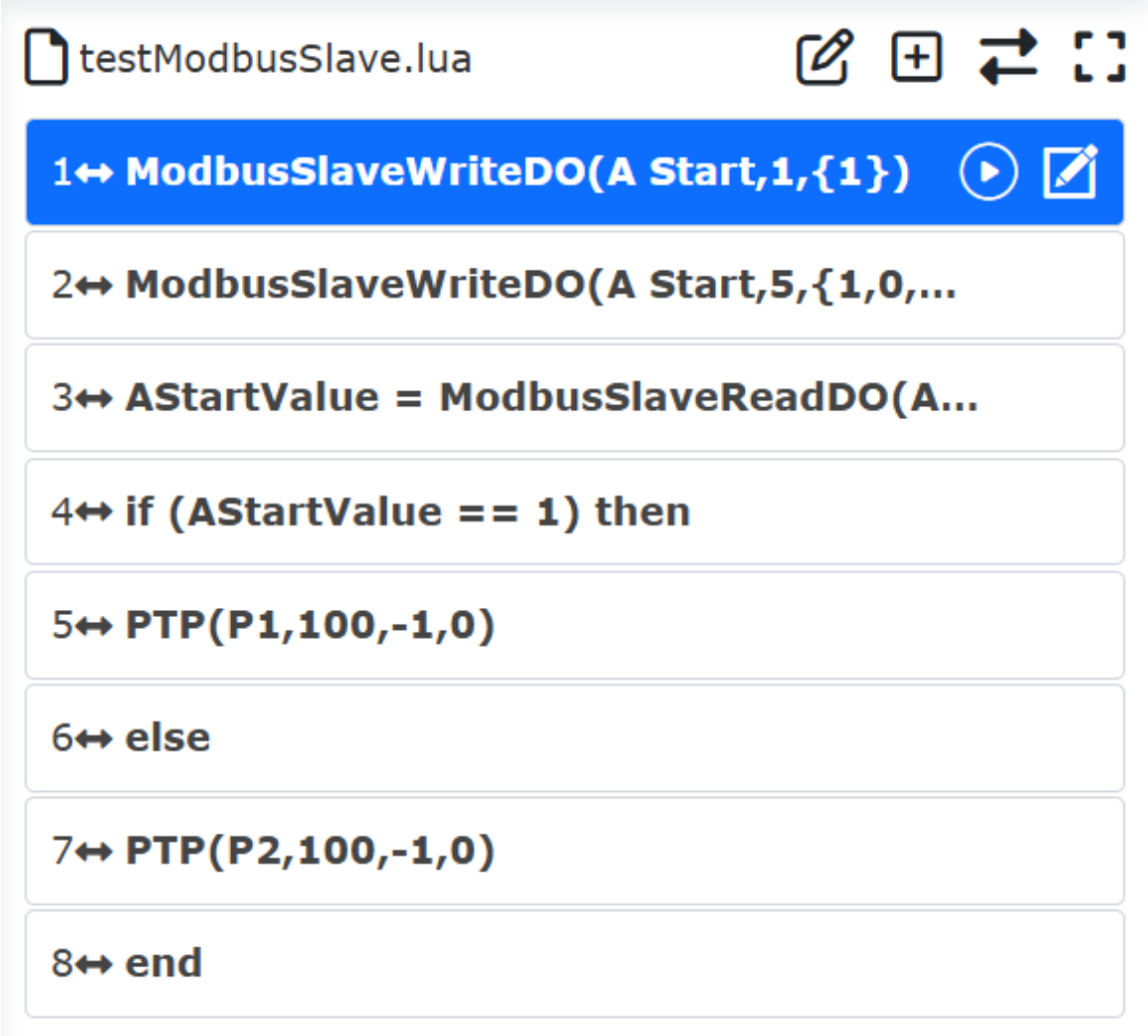


Figure 9.18-61 Read the register value of a single coil and move it

4. Read multiple digital outputs DO (discrete input)

As shown in Figure 74, open the ModbusTCP master station command addition page, find the “Digital Output Settings”, select the DO name as “A Start”, the number of registers is 2, and the register value does not need to be filled in, and click “Read Digital Output”. Finally, scroll to the bottom of the page and click the “Apply” button (Figure 75).

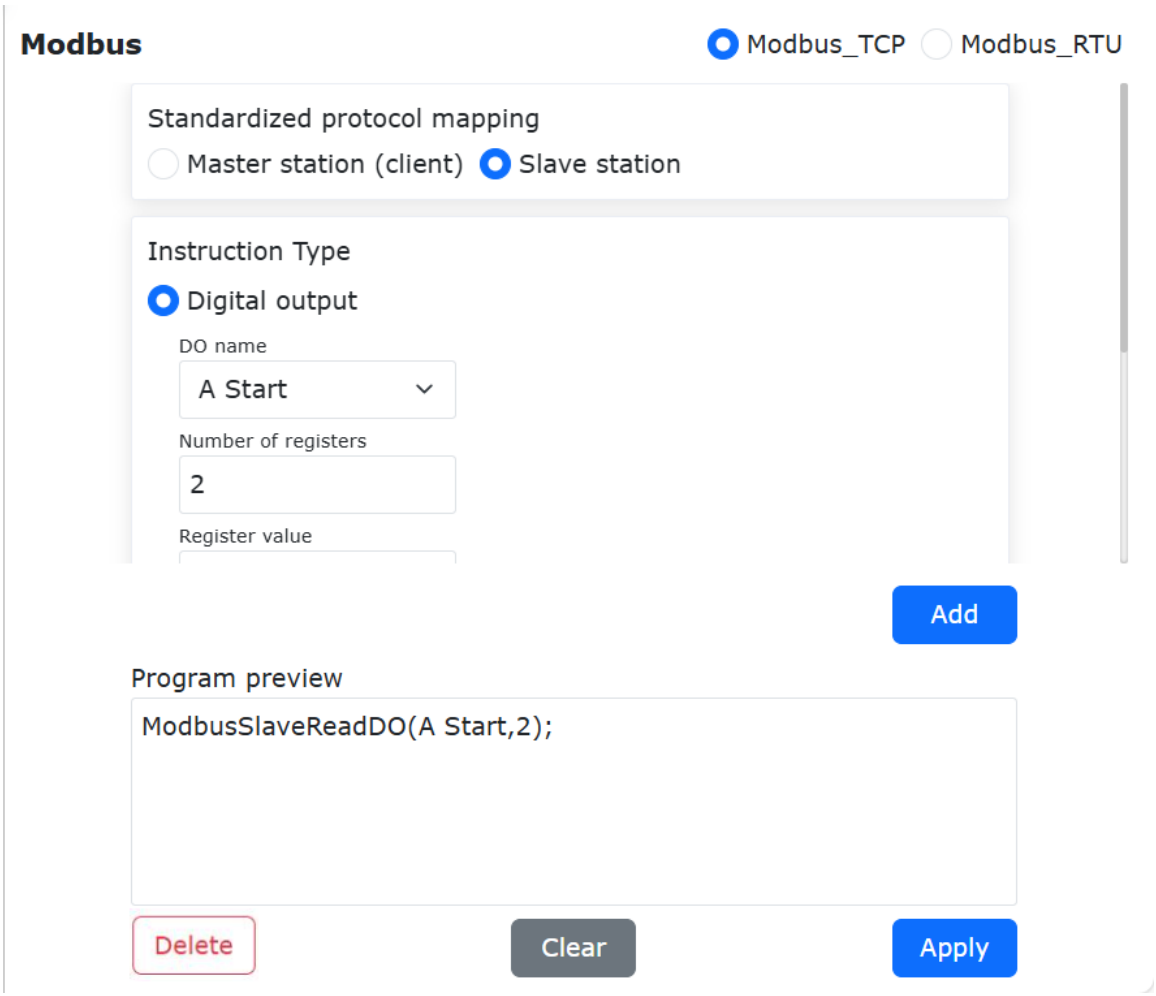


Figure 9.18-62 Configures Reading Multiple Digital Outputs

As shown in Figure 76, at this time, the robot program “testModbusSlave.lua” has added an command for the robot Modbus to read multiple digital outputs from the station.



Figure 9.18-63 Reading multiple digital output programs

As shown in figure 77, click the “switch mode” button “” to switch the robot lua program to the editable state. Since the number of readings is two, it is necessary to write and add two return values “value1,value2” before the “Modbus-SlaveReadDO” command, and the values of the two digital output registers read after executing the program will be stored in the above two variables respectively. Similarly, you can judge “value1” and “value2”.

testModbusSlave.lua

```
1  ModbusSlaveWriteDO(A Start,1,{1})
2  ModbusSlaveWriteDO(A Start,5,{1,0,1,0,1})
3  AStartValue = ModbusSlaveReadDO(A Start,1)
4  if (AStartValue == 1) then
5      PTP(P1,100,-1,0)
6  else
7      PTP(P2,100,-1,0)
8  end
9  value1,value2 = ModbusSlaveReadDO(A Start,2)
```

Figure 9.18-64 Reading Multiple Digital Outputs and Storing Variables

5. Read DIgital input DI (coil)

As shown in Figure 78, open the ModbusTCP slave command addition page, find the “Digital Input Settings”, select the DI name as “A in place” and the number of registers as 2, and click “Read Digital Input”. Finally, scroll to the bottom of the page and click the “Apply” button (Figure 79).

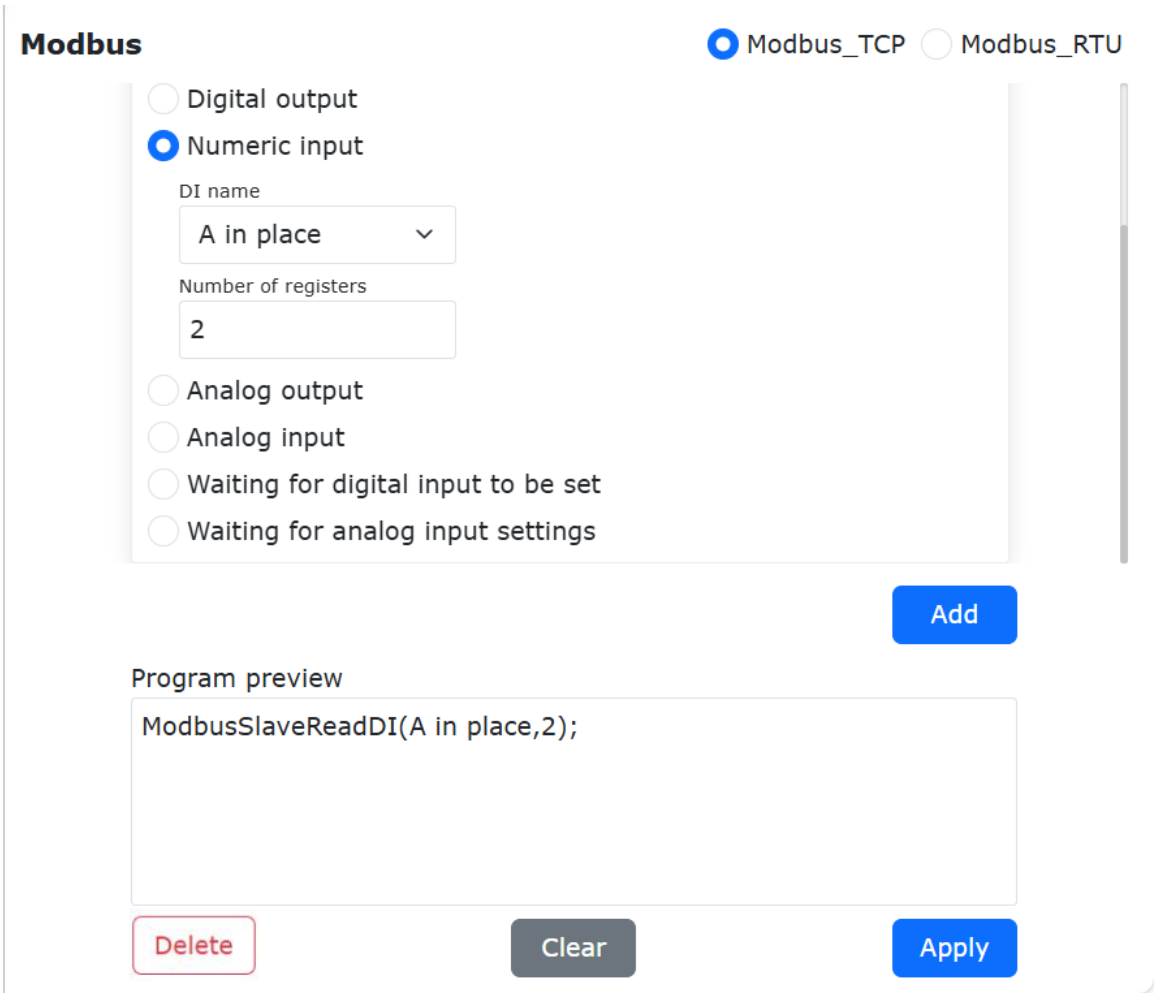


Figure 9.18-65 Configuring Read Digital Input

As shown in Figure 80, at this time, the robot program “testModbusSlave.lua” has added an command for the robot Modbus to read digital input from the station.



Figure 9.18-66 Reading digital input program commands

As shown in Figure 81, click the “switch mode” button “” to switch the robot lua program to an editable state, and write the return value variable “AState,BState” before the “ModbusSlaveReadDI” command. The two digital input values read after executing the program will be stored in the variables “AState” and “BState” respectively, and you can control the robot to do different operations by judging the variable values.

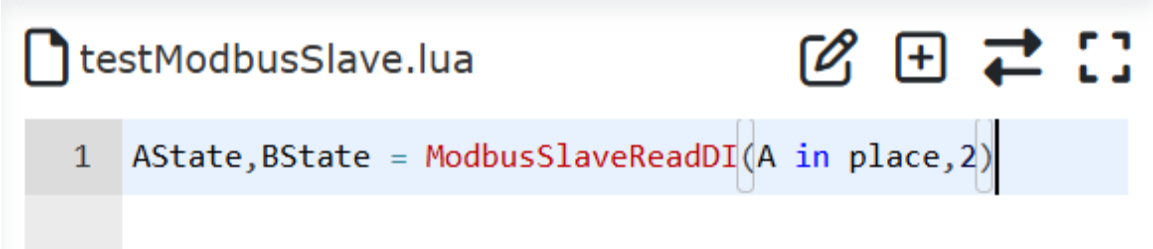


Figure 9.18-67 Reading digital input program

6. Read and write operations of analog output AO (input register) and analog input AI (hold register)

The reading and writing operations of analog output (input register) and analog input (holding register) are basically the same as those of digital output (discrete input) and digital input (coil), but the difference is that the data range of the latter is only 0 or 1, while the data range of the former is larger, so the specific operations can refer to the programming of digital output and digital input, and only the reading and writing operations of analog input (Figure 82) and analog output (Figure 83) are shown here.

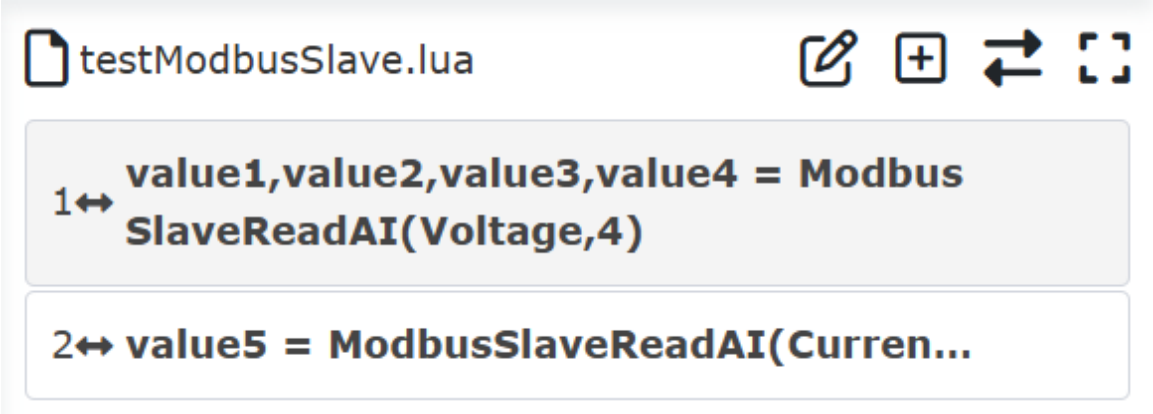


Figure 9.18-68 Reading Analog Input

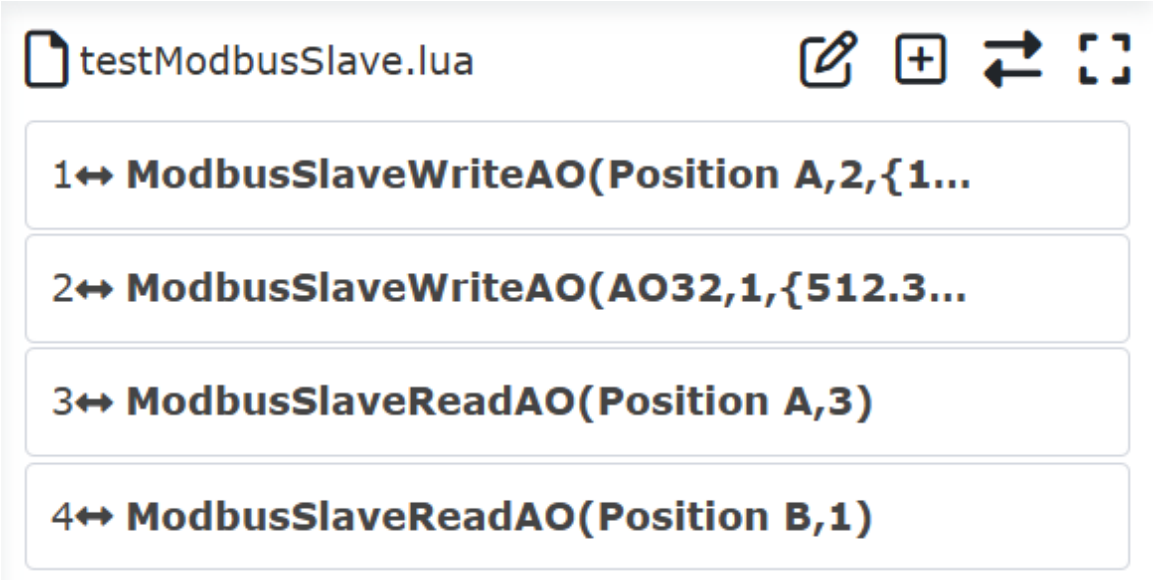


Figure 9.18-69 Reading and writing analog output

7. Waiting for digital input

As shown in Figure 84, open the ModbusTCP slave command addition page, find “Waiting for digital input settings”, select the “A in place” register with DI name as configuration, the waiting status is “True”, and the timeout time is 5000 ms. Click “Add” button, and finally click “Apply” button.

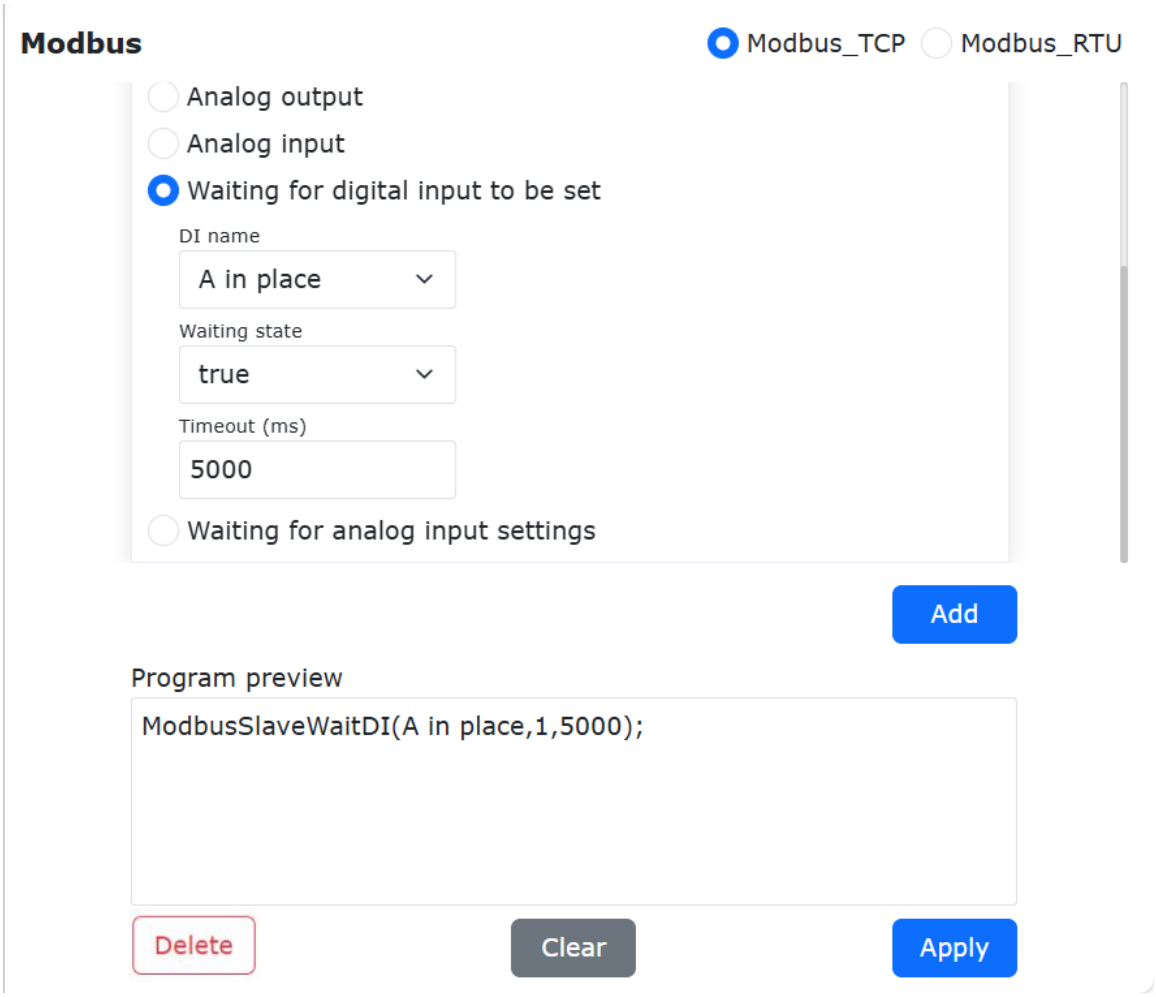


Figure 9.18-70 Add the command of waiting for digital input

As shown in Figure 85, at this time, the robot program “testModbusSlave.lua” has added an command that the robot Modbus slave station waits for digital input. After starting the program, the robot will wait for the value of the “A in place” coil register of the slave station to become true, that is, the value 1. Since the set timeout is 5s, when the “A in place” signal is still 0 after the robot waits for 5s, the robot program will report a timeout error and the program will automatically stop running.



Figure 9.18-71 Waiting for Digital Input Program

8. Waiting for analog input

As shown in figure 86, open the ModbusTCP slave command addition page, and find the voltage register with the AI name selected for “Waiting for Analog Input Settings”. The waiting state is “>”, the register value is 255, and the timeout time is 5000 ms.. Click “Add” button, and finally click “Apply” button.

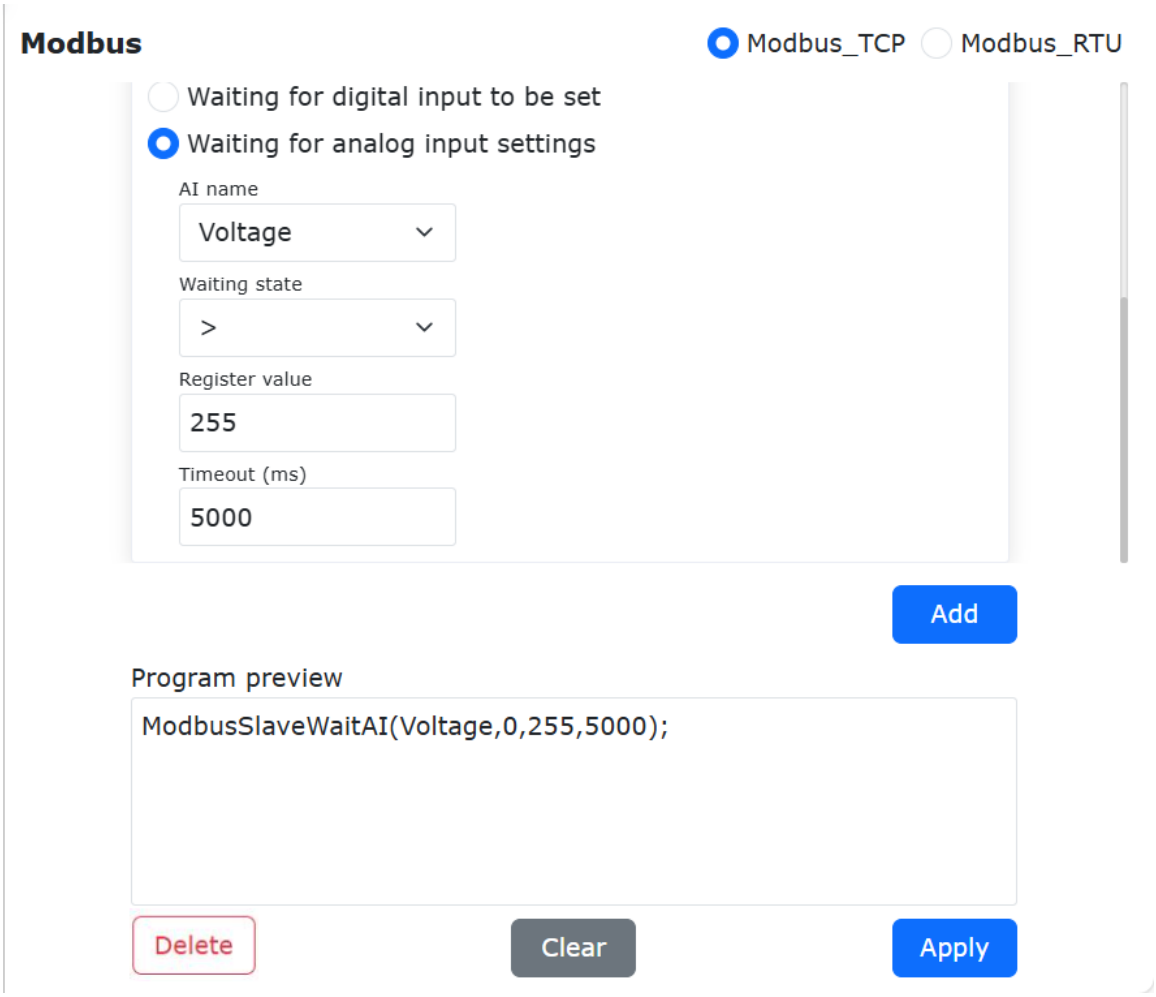


Figure 9.18-72 Add an command to wait for analog input

As shown in Figure 87, at this time, the robot program “testModbusSlave.lua” has added an command that the robot Modbus slave station waits for the analog input value. After starting the program, the robot will wait for the slave station’s “voltage” register value to be greater than 255. Since the set timeout time is 5s, when the robot waits for 5s and the “voltage” signal is still less than 255, the robot program will report a timeout error and the program will automatically stop running.

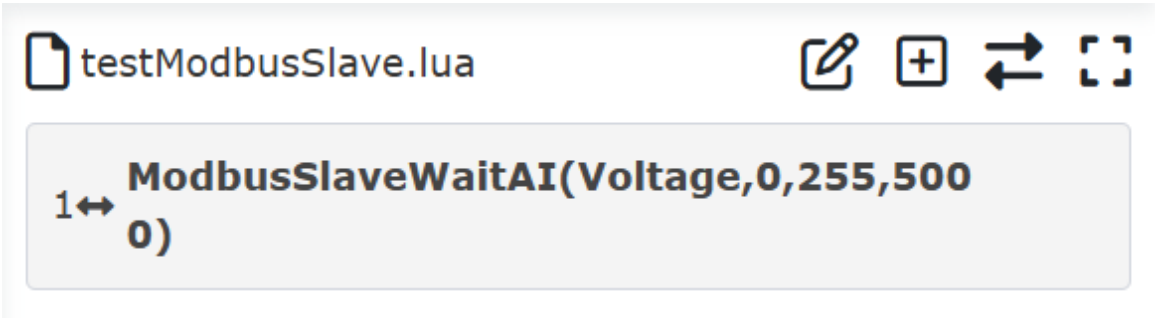


Figure 9.18-73 Waiting for Analog Input Register Program

Open the ModbusTCP Slave command addition page, find the “Wait for Analog Input Setting” (i.e., wait for AI input register setting). Select the AI name as the configured “Liquid Level” register, set the wait condition to “=”, the register value to 255, and the timeout to 5000ms. Click the “Add” button, and finally click the “Apply” button.

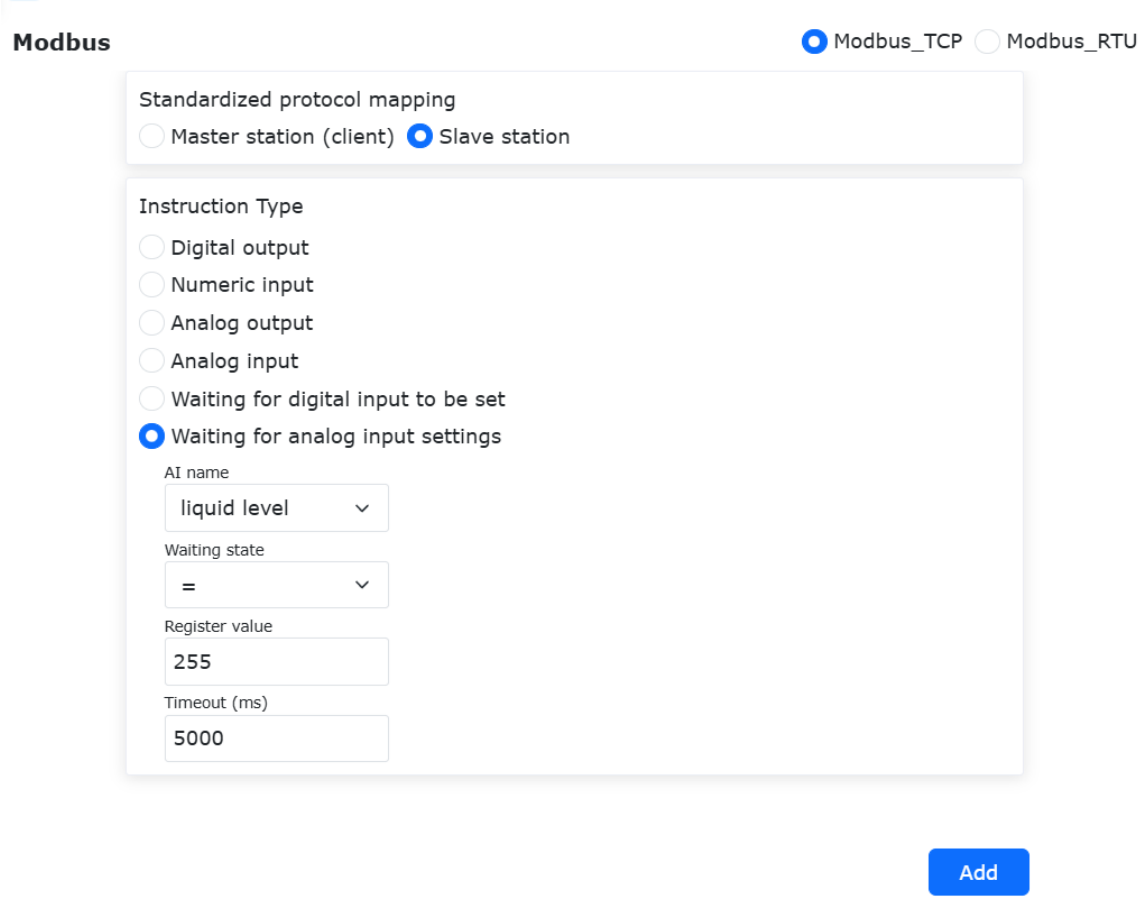


Figure 9.18-73-2 Add Wait for Analog Input

At this point, a Modbus Slave command to wait for the AI input register value has been added to the robot program “test.lua”. After starting the program, the robot will continuously wait for the “Liquid Level” register value to equal 255. Since the timeout is set to 5s, if the “Liquid Level” signal still does not equal 255 after the robot waits for 5 seconds, the robot program will report a timeout error and automatically stop running.

1.3.9.18.3.4 ModbusTCP slave robot state feedback and control

The ModbusTCP slave input register addresses 310~473 of the cooperative robot are used to feed back the real-time status of the robot (see Annex 1: ModbusTCP slave address mapping table). You only need to read the value of the corresponding register with the master station equipment to obtain the corresponding real-time status data of the robot.

The coil register addresses 300~599 of ModbusTCP slave station of the cooperative robot are used for the master station equipment to control the robot (see Annex 1: ModbusTCP slave address mapping table). Taking coil address 502 as an example, this address function indicates “startup program”.

When the robot is in automatic mode, the master station equipment sets the value of address 502 from 0 to 1, the robot automatically starts to run the currently configured program; Take the coil address 300 as an example. It is used to control the output of the robot control box DO0. When the external master station sets the coil address 300 from 0 to 1, the automatic output of the control box DO0 is valid. Similarly, when the external master station sets the coil address

300 from 1 to 0, the output of the control box DO0 is invalid. As shown in Figure 88, click "Function Digital Input (Coil)" on the ModbusTCP slave station setting page to monitor all current function digital inputs.

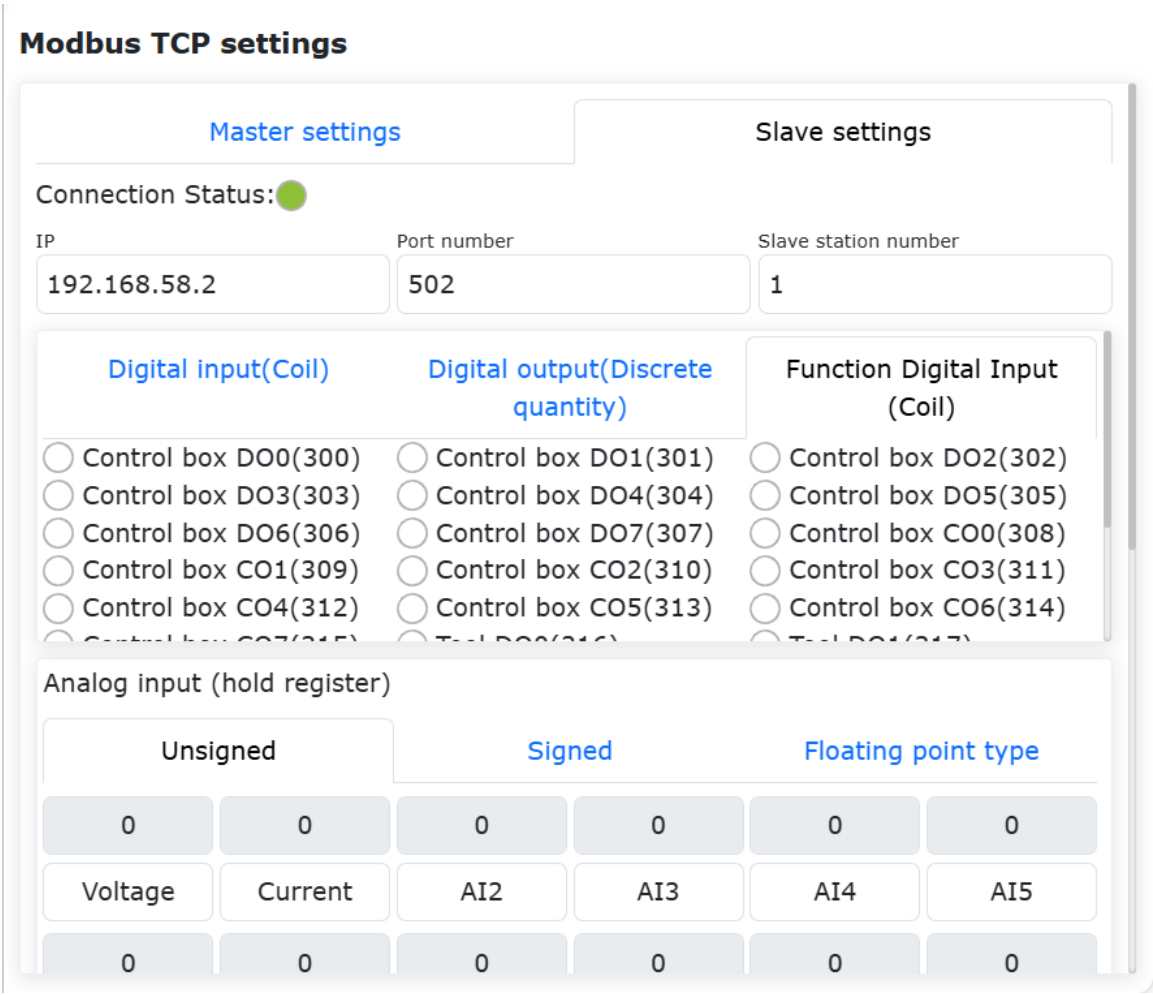


Figure 9.18-74 Digital Input of Robot Slave Station Function

FAIRINO MODBUSTCP Slave station address mapping table

address	type	name	data type	function	R/W	remarks
100	General digital input(coils)	DI0	BOOL	0x02	R/W	
101		DI1				
102		DI2				
103		DI3				
...		...				
227		DI127				
100	General digital output (Discrete input)	DO0	BOOL	0x01 0x05 0x15	R	
101		DO1				
102		DO2				
103		DO3				
...		...				
227		DO127				
100	Analog input (Holding register)	AI0	UINT16	0x04	R	
101		AI1				
102		AI2				
...		...				
115		AI15				
116		AI16	INT16			
117		AI17				
118		AI18				
...		...				
131		AI31				
132		AI32	FLOAT32 (Big Endian)			
133						
134		AI33				
135						
...		...				
194		AI63				
196						
100	Analog output (Input register)	AO0	UINT16	0x03 0x06 0x16	R/W	
101		AO1				
102		AO2				
...		...				
115		AO15				
116		AO16	INT16			
117		AO17				
118		AO18				
...		...				
131		AO31				
132		AO32	FLOAT32 (Big Endian)			
133						
134		AO33				
135						
...		...				
194		AO63				
196						
Robot state feedback						
310	Robot State (Input register)	Enabled state: 0-not enabled, 1-enabled	UINT16	0x04	R	
311		Robot modes 1-manual mode, 0-automatic mode				
312		Robot running state: 1-Stop, 2-Running, 3-Pause, 4-Drag				
313		Tool number				
314		Workpiece number				
315		Emergency stop state: 0-Normal, 1-Emergency stop				
316		Soft limit exceeded				
317		Main Error code				
318		Sub Error code				
319		Collision detection: 1-collision, 0-no collision				
320		Motion arrive				
321		Safety stop S10				
322		Safety stop S11				
330		TCP speed	FLOAT32 (Big Endian)			
340		Joint 1 position				
342		Joint 2 position				
344		Joint 3 position				
346		Joint 4 position				
348		Joint 5 position				
350		Joint 6 position				
352		Joint 1 velocity				
354		Joint 2 velocity				
356		Joint 3 velocity				
358		Joint 4 velocity				
360		Joint 5 velocity				
362		Joint 6 velocity				
364		Joint 1 current				
366		Joint 2 current				
368		Joint 3 current				
370		Joint 4 current				
372		Joint 5 current				
374		Joint 6 current				
376		Joint 1 torque				
378		Joint 2 torque				
380		Joint 3 torque				
382		Joint 4 torque				
384		Joint 5 torque				
386		Joint 6 torque				
388		TCP pos X				
390		TCP pos Y				
392		TCP pos Z				
394		TCP pos RX				
396		TCP pos RY				
398		TCP pos RZ				
400		TCP velocity X				
402		TCP velocity Y				
404		TCP velocity Z				
406		TCP velocity RX				
408		TCP velocity RY				
410		TCP velocity RZ				
430		Control box analog input AI0	UINT16			
432		Control box analog input AI1				
438		Tool analog input AI0				
450		Control box analog output AO0				
452		Control box analog output AO1				
458		Tool analog output AO0				
470		Digital inputs Bit0-Bit7 of the control box correspond to DI0-DI7. Bit8-Bit15 corresponds to CI0-CI7				
471		Tool end digital input Bit0-Bit15 correspond to DI0-DI15				
472		Digital outputs Bit0-Bit7 of the control box correspond to DO0-DO7. Bit8-Bit15 correspond to CO0-CO7				
473		Tool end digital output Bit0-Bit15 corresponds to DO0-DO15				
Robot control						
address	type	name	data type	function	R/W	remarks
300	Digital input(coils)	Control box DI00	BOOL	0x02	R/W	
301		Control box DI01				
302		Control box DI02				
303		Control box DI03				
304		Control box DI04				
305		Control box DI05				
306		Control box DI06				
307		Control box DI07				
308		Control box CI00				
309		Control box CI01				
310		Control box CI02				
311		Control box CI03				
312		Control box CI04				
313		Control box CI05				
314		Control box CI06				
315		Control box CI07				
316		Tool DI00				
317		Tool DI01				
318 ~ 499		reserve				
500		Pause				
501		Resume				
502		Start				
503		Stop				
504		Move to Job origin				
505		Manual automatic switching				
506		Start the main program				
507		level 1 reduction mode				
508		level 2 reduction mode				
509		level 3 reduction mode (stop)				
510		Reset all error				
511 ~ 599		reserve				

Figure 9.18-74 Modbus TCP Slave station address mapping table

Appendix 1: `Modbus TCP Slave station address mapping table`

1.3.9.19 Robot Backgrounder Function

1.3.9.19.1 Robot Backgrounder Function

The robot background program is a control program used to process the logical relationship of signals in the process of running the robot in the foreground motion program, and the background is also running, and the two are independent of each other in the operation relationship.

The background program can monitor the operating status of the foreground and at the same time send control signals to the foreground. The background program can also be connected with external devices through I/O communication to monitor and control the operation of robot peripheral equipment. Unlike the foreground teach-in, the background logic program can run commands, which cannot control any axis of motion. Therefore, it cannot be programmed with any robot axis motion commands. Only the logic control function and I/O communication function are retained.

When using a background program, the program scans in a loop from start to finish. The running cycle is 1 millisecond, and a delay function can be added to the background program to control the running cycle. During the execution of the background program, it is not affected by emergency stop, pause, or alarm.

Note: Run up to 8 background programs at the same time.

A maximum of 8 programs can be executed at the same time as background logic, and an error alarm will be sent when the number of programs exceeds the maximum value.

When the power is cut off, the background logic program will be automatically loaded and run according to the set state the next time the power is turned on.

1.3.9.19.1.1 Save the robot background program

Backgrounder creation, editing, and saving can only be used in the Backgrounder interface.

Step1:Open the robot background program interface. Open the teaching page, click “Program”, and then click “Coding”. Select the command background program in the upper left corner to enter the background program interface.

Note: The background program only contains logic judgment, assignment instructions, front-end control instructions, I/O interface instructions and Modbus communication instructions.

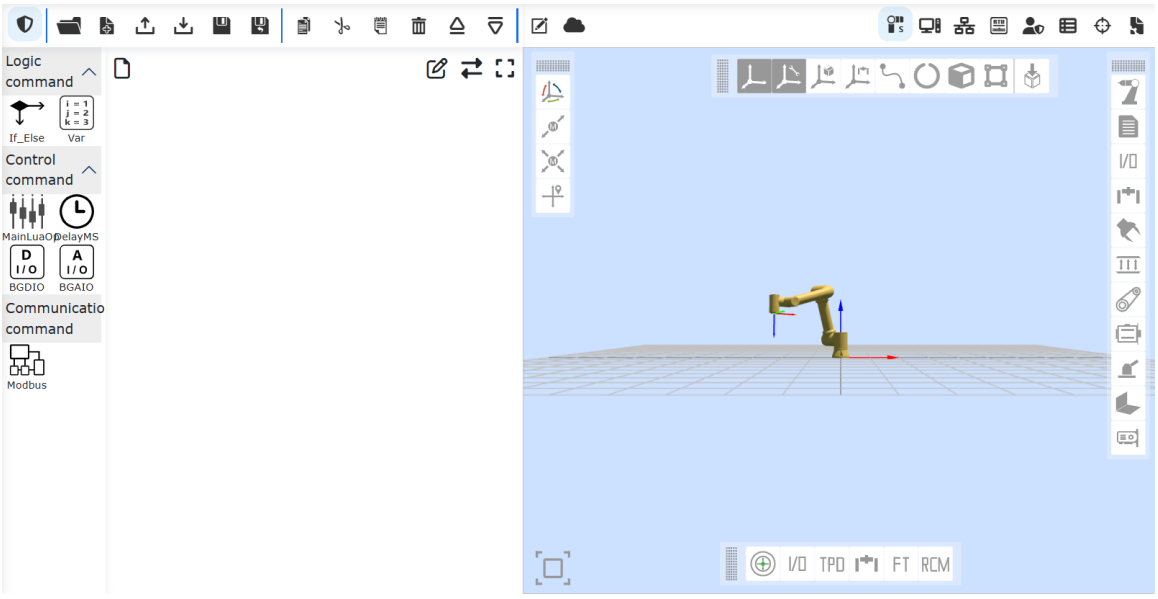


Figure 9.19-1 The robot background program interface

Step2:In manual mode, open the background teach program file. Click “New” to create a new teaching program file, edit the program, and click “Save” to save the file.

Note: The running cycle of the background program is 1 millisecond, and the provided delay function can be used in the program, as shown in the fourth line of the program in the following figure, to increase the delay of 1 second to control the running cycle.

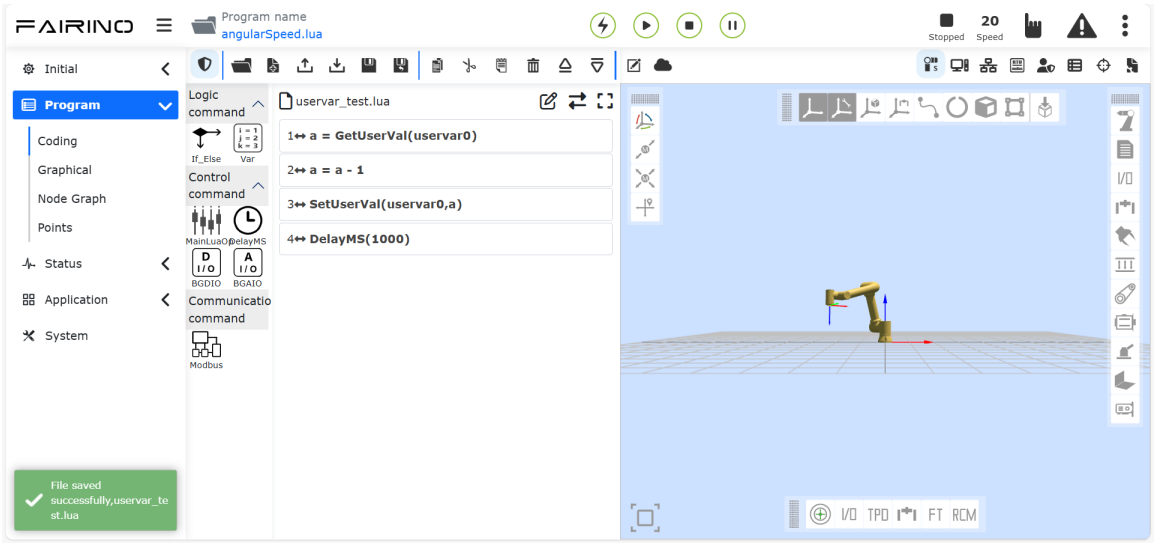


Figure 9.19-2 Save the robot background program

1.3.9.19.1.2 Robot background program management

Successfully saved backgrounders can be created, paused, resumed, and deleted in the backgrounder management interface. The backgrounder management interface allows you to intuitively see the running status of all created backgrounders. Green is running, and red is paused.

Step1:Create a background program. Click the Background Program Management button, select the saved background program from the drop-down box, and click Start Run to run the corresponding background program.

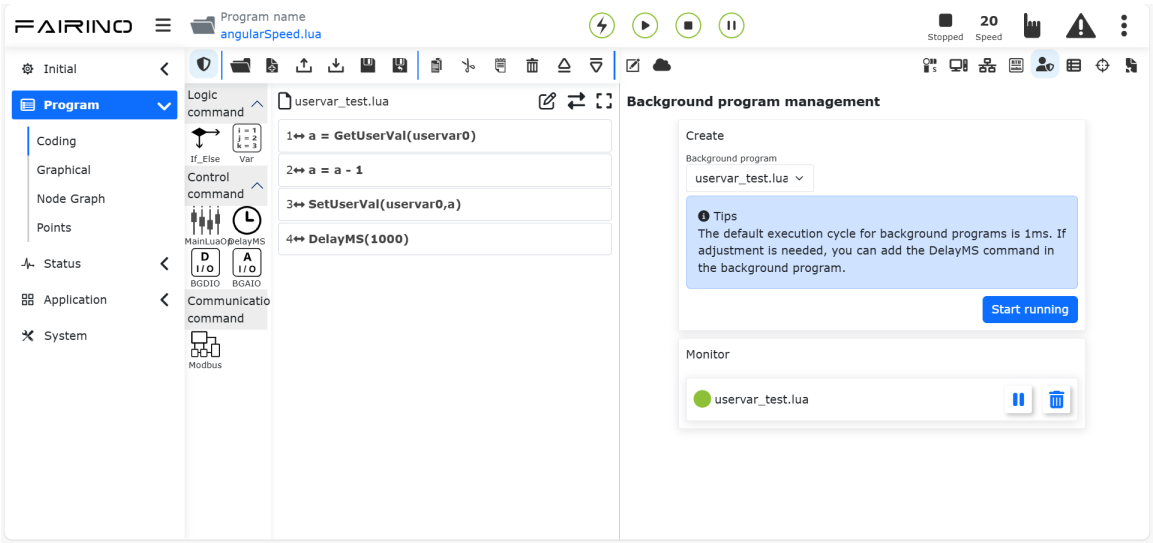


Figure 9.19-3 Start running the robot background program

Step2:Resume and pause the background program. In the background program management page, click Resume and Pause on the monitor to resume and pause the corresponding background program.Click Delete on the monitor program to delete the corresponding background program.

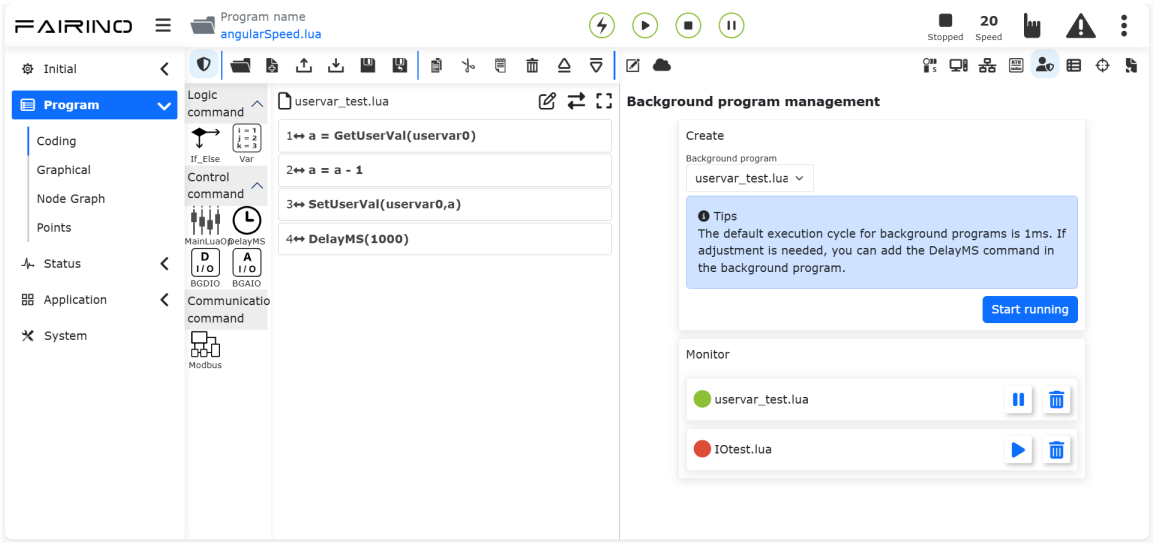


Figure 9.19-4 Pauseresumedelete the background program

1.3.9.19.2 Use of the robot user variables

Note: The new user variable function is applicable to the data interaction between the robot background program and the foreground program, or between different background programs.

1.3.9.19.2.1 Robot user variable management

Before using user variables, you can rename them to your liking. Open the teach-in page, click “Program”, “Coding”, and “user variable management”, which can be used in both the foreground program and the background program. Click on the variable name to change the variable name directly.

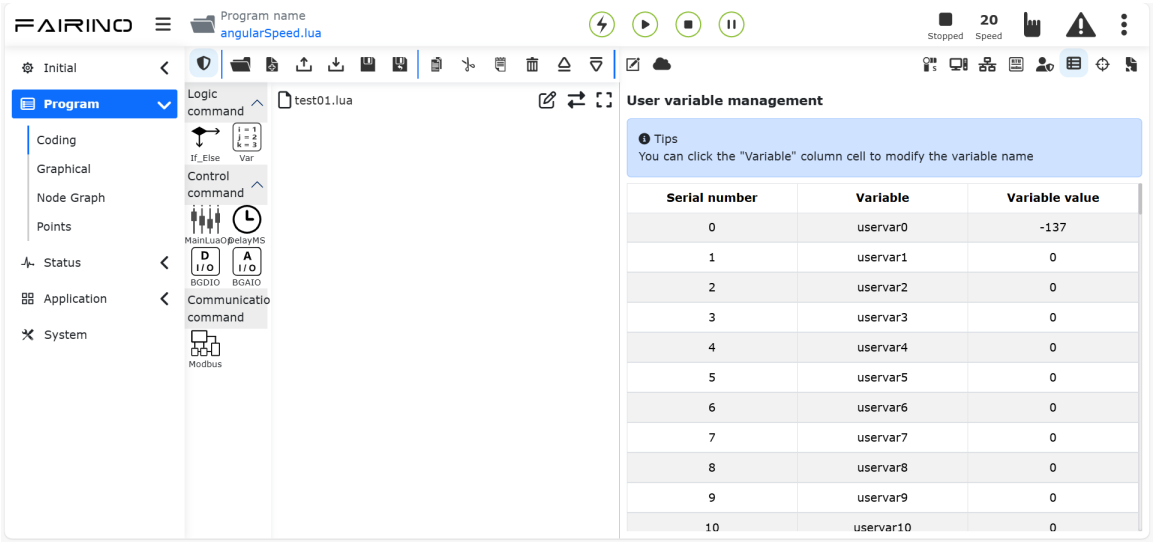


Figure 9.19-5 Robot user variable management

1.3.9.19.2.2 Robot user variable use

When user variables are used in foreground and background programs, only the user variable read/write interface can be used.

Step1:In manual mode, open the teach-in program file. Open the teaching page, click “Program”, click “Coding”, and click “New” to create a new teaching program file.

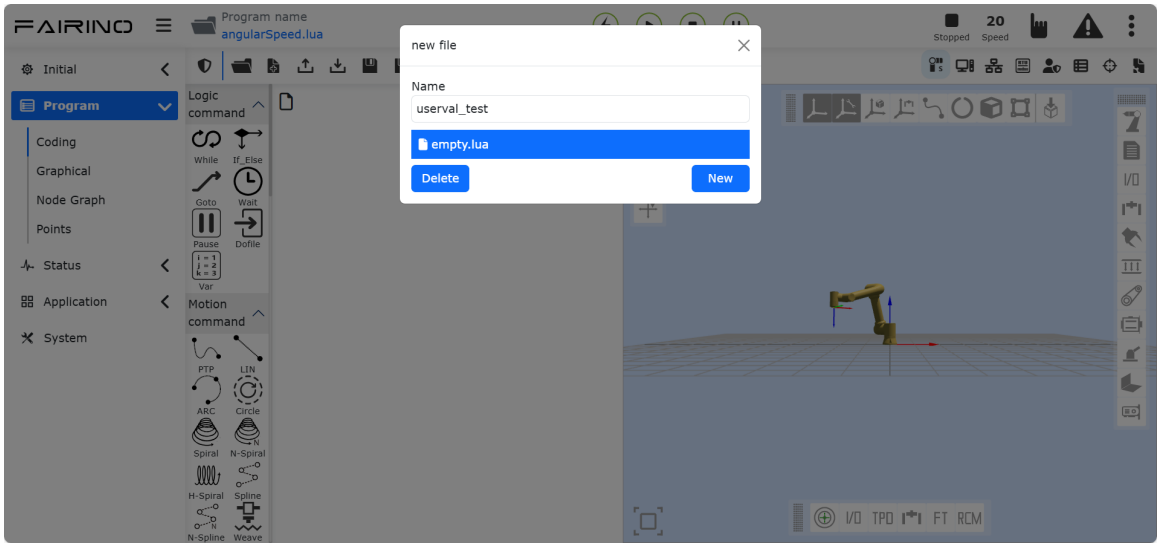


Figure 9.19-6 Create a new teach-in program file

Step2:Use the user variable read interface. Click the "Variable" command, select "User Variable", click the Get Variable Value drop-down box, select the user variable to be read, and click the "Add" and "Apply" buttons to write the user variable reading interface program.

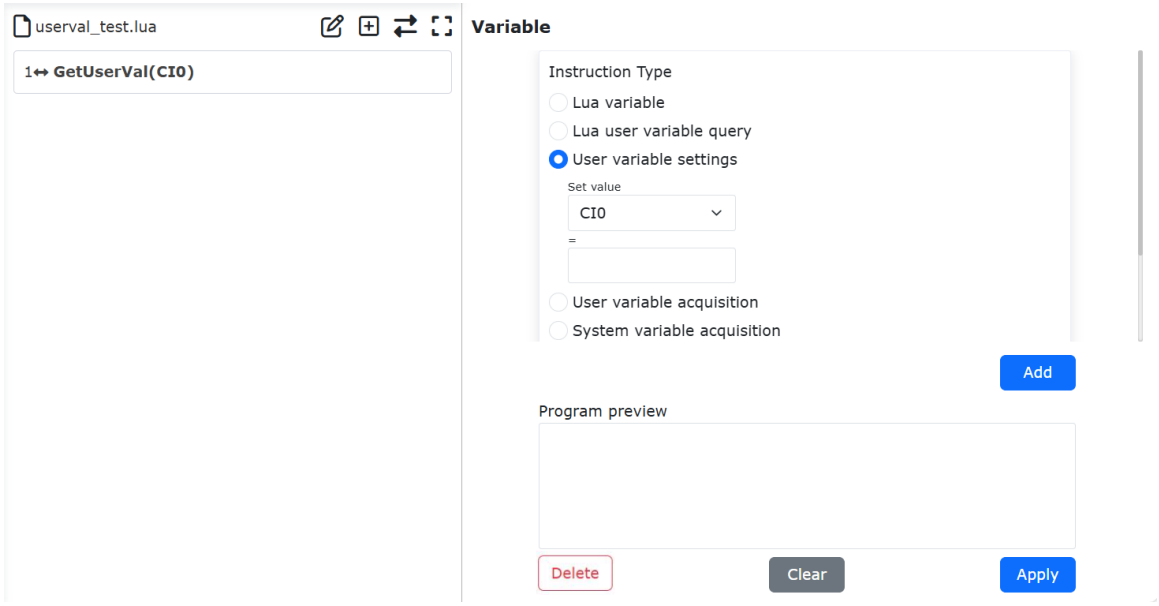


Figure 9.19-7 Use the user variable read interface

Step3:Use user variables to write interfaces. Click the "Variable" command, select "User Variable", click the Set Variable Value drop-down box, select the user variable to be set, and fill in the corresponding set value, which supports both constant and variable value. Click the Add and Apply buttons to write user variables and write interface programs.

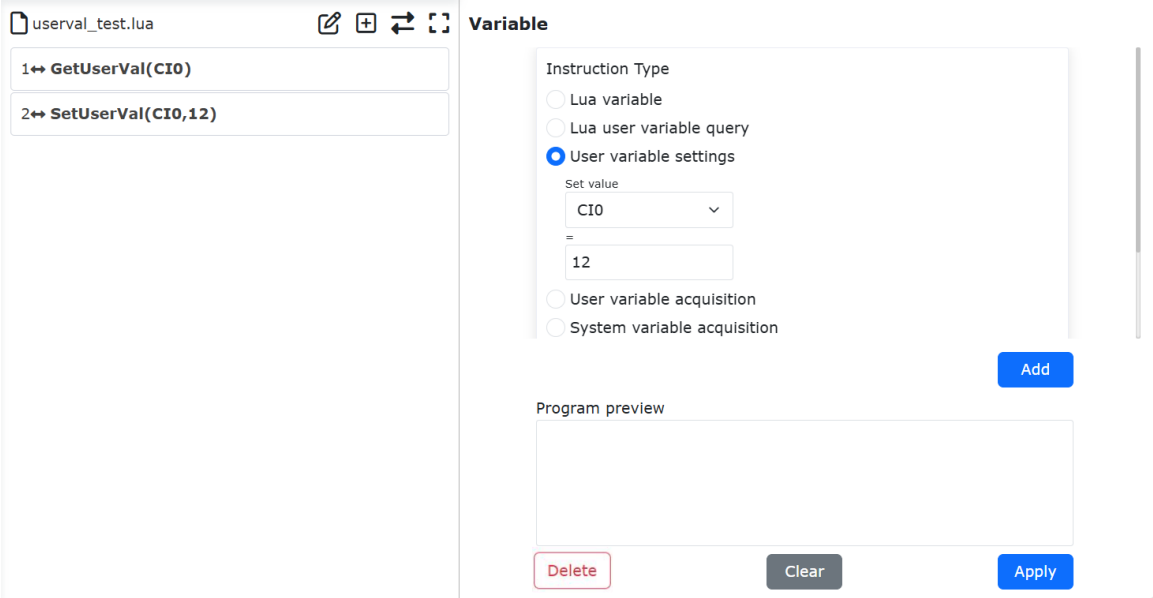


Figure 9.19-8 Use the user variable write interface

1.3.9.20 XY horizontal constant

1.3.9.20.1 Overview

The principle of lateral constant force grinding in the XY direction is as follows: Lateral constant force grinding refers to applying a grinding tool (such as a grinding wheel, grinding disc, etc.) with a constant force on a specified workpiece surface, and controlling the movement of the tool along the XY direction to maintain a constant grinding force at the contact point.

1.3.9.20.2 Operation process of lateral constant force grinding function in XY direction

To perform constant force grinding using a force sensor, it is necessary to install a grinding tool under the force sensor and configure the tool coordinate system. Click on the “Initial” -> “Base” -> “Coordinate” -> “TCP” button in sequence to enter the “Tool coordinate system settings” interface. Select the coordinate system to be set in the “Coordinate system name” (taking toolcoord 0 coordinate system as an example), and set it according to the size of the end tool.

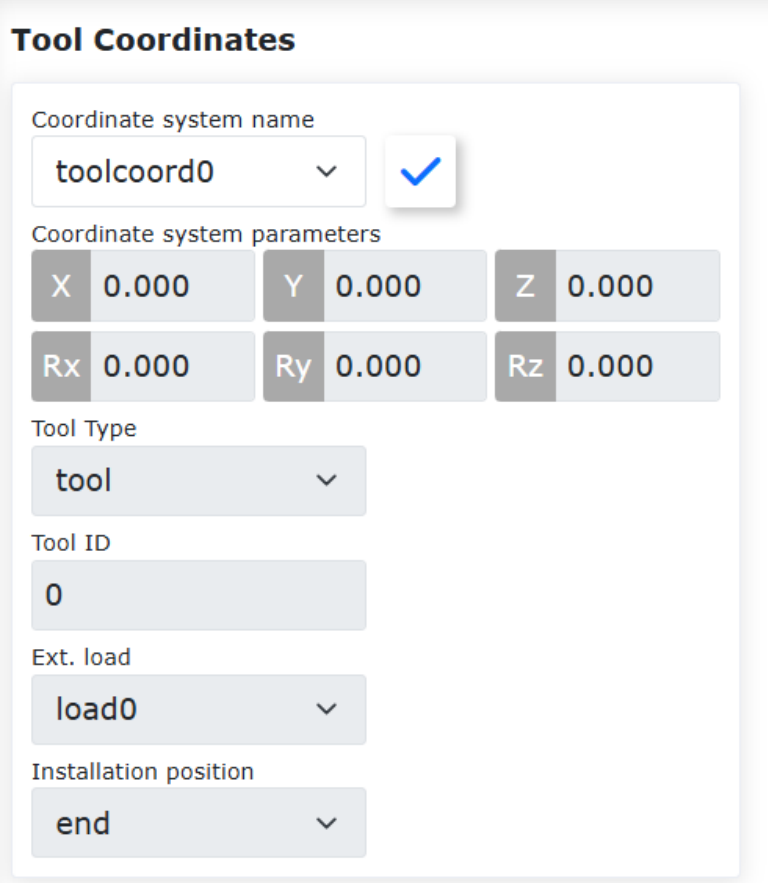


Figure 9.20-1 Set tool coordinate system

Force control reference coordinate system setting. In the web interface, click on “FT” -> “Reference coord.”, select “Custom coordinates”, and set each parameter to “0”. When the force sensor is working, different reference coordinate systems will affect the magnitude of the external force obtained by the sensor.

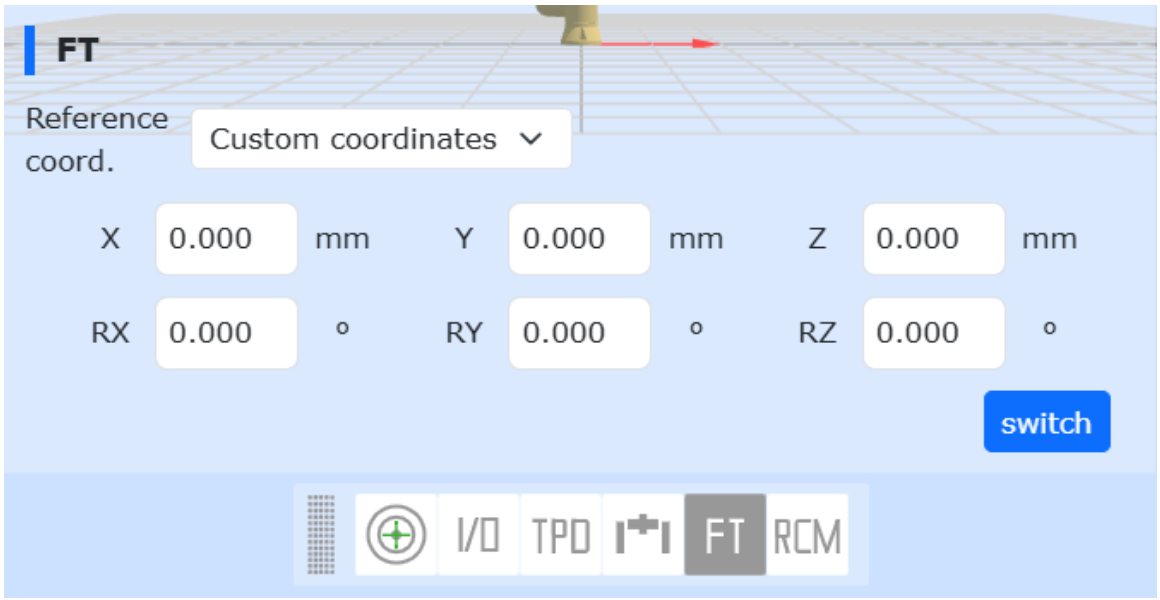


Figure 9.20-2 Set reference coordinate

Fix the flat plate to be polished in the robot workspace, and the plate should not shake. Place the end of the tool approximately perpendicular to the polishing plate and teach the starting and ending points.



Figure 9.20-3 Polishing layout diagram

Click on the “Program” -> “Coding” -> “F/T” button in sequence, and add the “FT_Control” instruction. The “FT_Control” command is a force controlled motion command that allows the robot to move around the set force.



Figure 9.20-4 Add FT_Control command



Figure 9.20-5 Example of FT_Control polishing instruction

The specific function of parameters:

Coordinate system name:The name corresponding to setting the sensor coordinate system;

Check the direction of force detection and set the detection threshold:Select the direction of the control force. In horizontal polishing, check Fx and Fy and set the corresponding expected constant force;

PID parameters:Set the PID proportional coefficients for force and torque, generally setting the F_P_gain to 0.001;

Maximum adjustment distance:corresponding to the maximum movement distance in the X, Y, and Z directions;

Maximum adjustment angle:corresponding to the maximum rotation angle of RX, RY, RZ;

Grinding disc radius:determined by the actual radius of the end grinding tool.

1.3.9.21 Automatic Singularity Avoidance Trajectory

1.3.9.21.1 Overview

When the robot encounters a singular range that cannot be passed by the robot in the LIN and ARC command tracks, the robot will report an error or prompt the next digit to be strange or a singular warning appears.

If you want to be able to reach the next waypoint that will pass through the singular range, you can use this function to avoid the singularity point through joint space or Cartesian space to reach the next target pose.

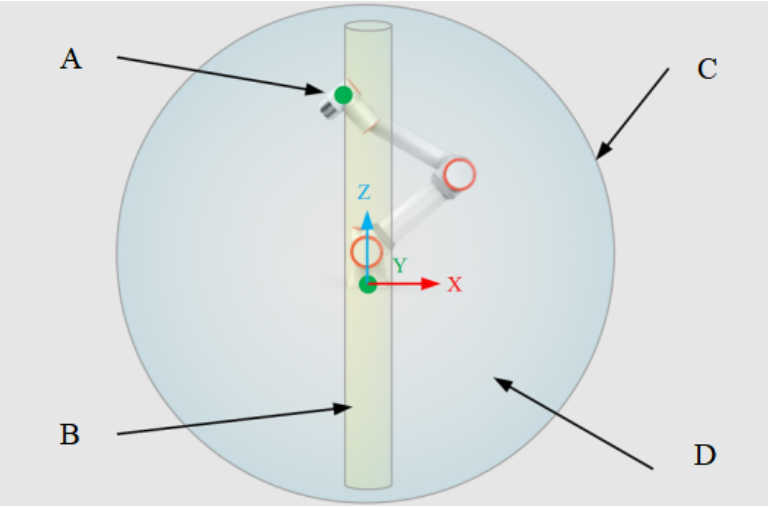


Figure 9.21-1 A simple schematic diagram of a robot's singularity

The above figure is a schematic diagram of the robot singularity, the robot singularity includes three kinds of singularity: shoulder, elbow and wrist, and A is the 5 joint center WCP (Wrist Center Point), which is used to judge the shoulder singularity; B is the singular range of the shoulder, which resembles a cylinder, and its radius is the length of the robot DH parameter d4, and the robot enters the singular state when the WCP enters the cylinder B; C is the elbow singularity boundary of the robot, and the robot is in the elbow singularity state when J3=0 or 180°; D is the internal space, and it is in the wrist singular state when J5=0 or 180° at any position in the internal space.

Note: It should be noted that singularity is a motion characteristic determined by the physical structure of the robot, which should be avoided as much as possible during actual operation, and it will lead to changes in the terminal posture and speed and even configuration configuration when it is avoided by algorithms, and it is necessary to consider whether the side effects of avoidance affect the requirements before making a choice.

1.3.9.21.2 The trajectory automatically avoids the operation process of the singularity function

In the new program, click Add Robot LIN/ARC type motion command.

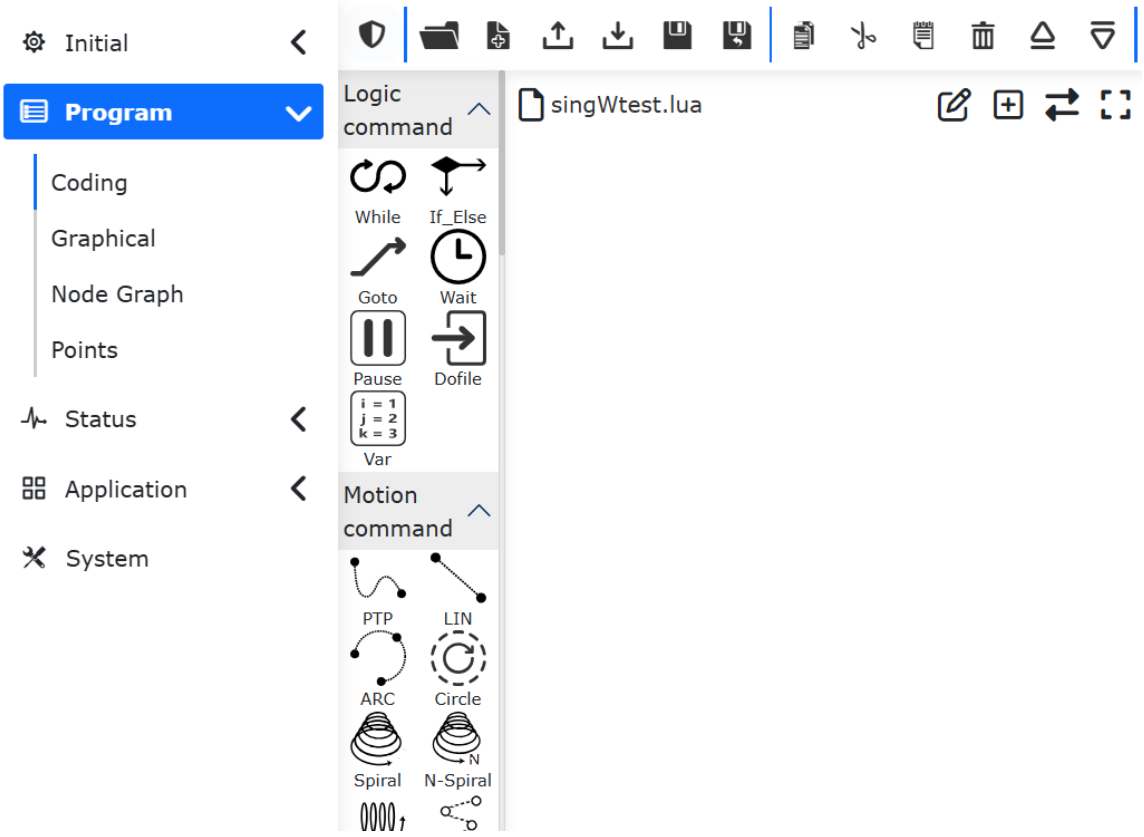


Figure 9.21-2 Add LIN/ARC LUA command

2.Click the “LIN” command, select the waypoint of the robot singularity, and click the “Singularity Avoidance” button in the “Motion Protection” sub-option of the command parameter configuration interface.

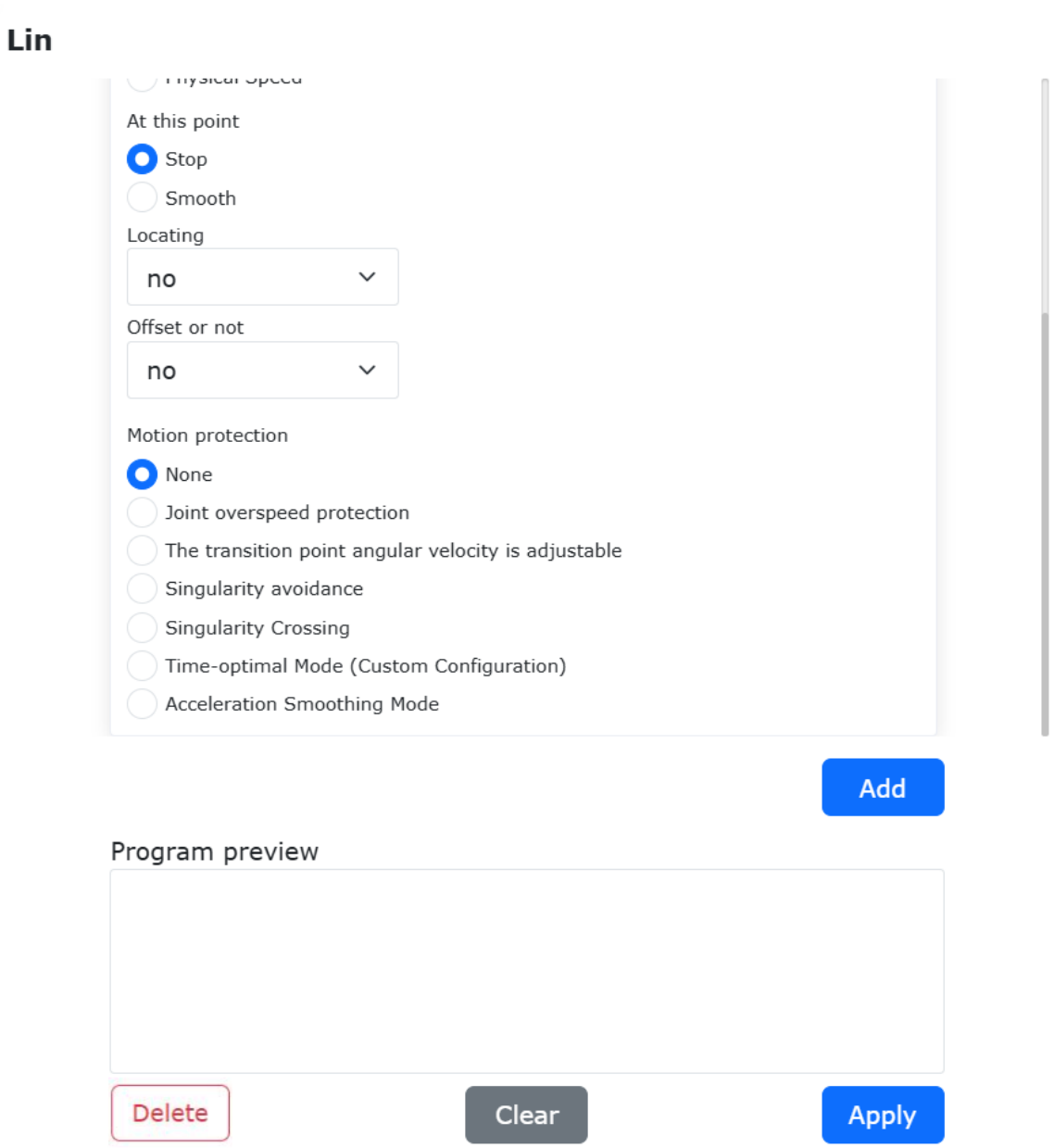


Figure 9.21-3 Turn on singularity avoidance switch

3.Singularity Avoidance parameters include Guard Mode, Shoulder Singularity, Elbow Singularity, and Wrist Singularity. Among them, the "protection mode" is divided into "joint mode" and "Cartesian mode", which means that the robot can cross the singular from the joint space or bypass the singular from the Cartesian space; The parameters of the "Singular Adjustment" specify the maximum deviation between the range of the determination of singularity and the singularity avoidance, which is mm for shoulder and elbow singularity units, and ° for wrist singular units.

Note: The joint space will select the nearest trajectory between the joints, so there will be no limit situation, and the joint limit may occur when the Cartesian space is avoided, so you need to pay attention to and adjust it during teaching.

After selecting and setting the singularity avoidance parameters, you can click the "Add" button to add the command, and then click "Apply" to add the LUA command to the program.

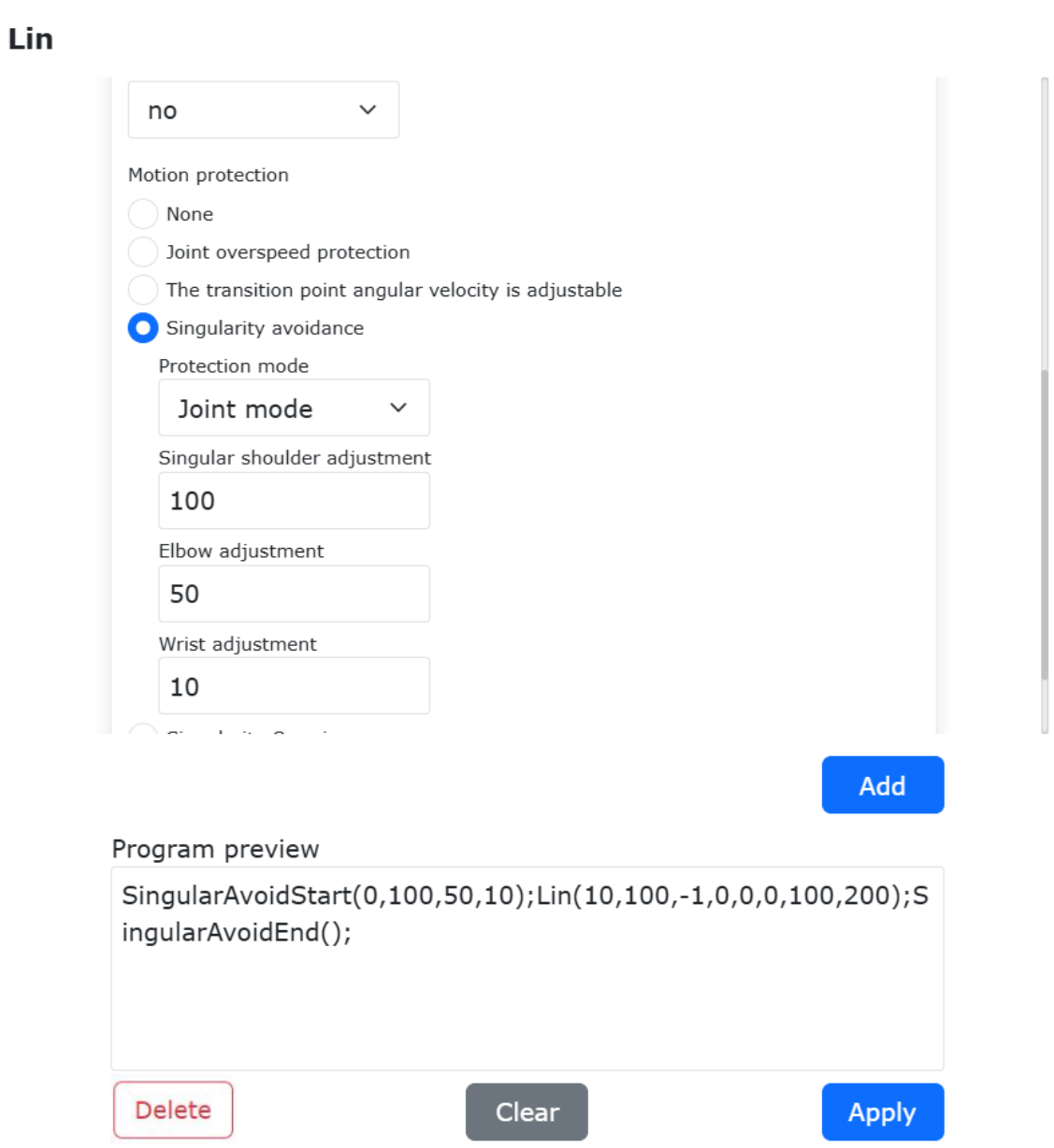


Figure 9.21-4 Configure singularity avoidance parameters and add the lua command

The teach-in completes a typical LIN singular avoidance movement lua procedure as follows:



Figure 9.21-5 lua program that contains singularity avoidance instructions

The effect of achieving avoidance is as follows, and the red is the trajectory line at the end of the robot:

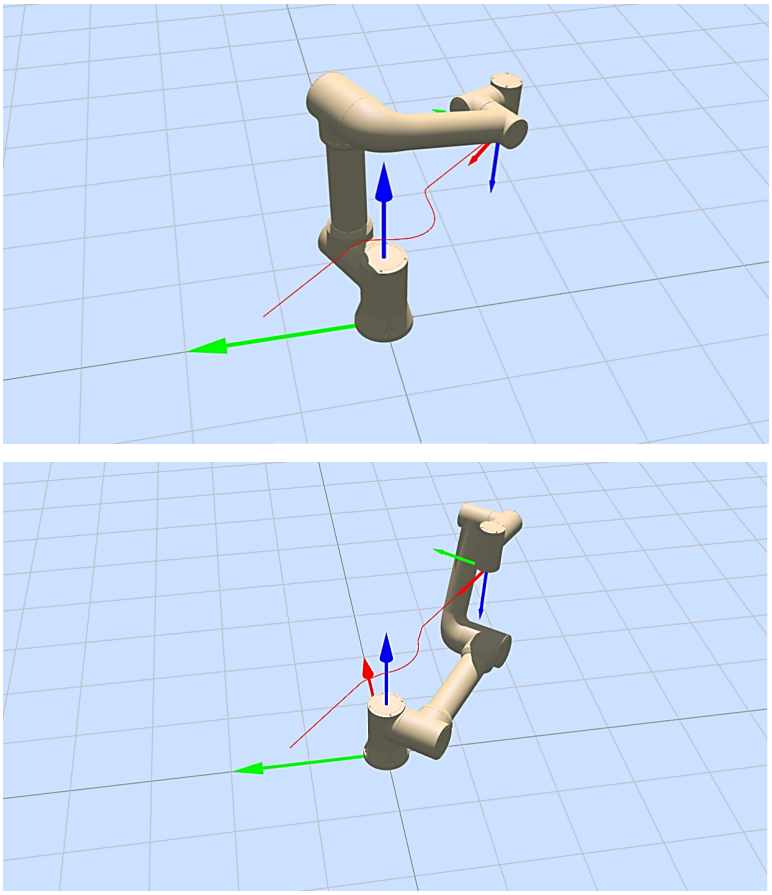


Figure 9.21-6 Example of a shoulder singular avoidance trajectory(top: Cartesian space, bottom: joint space)

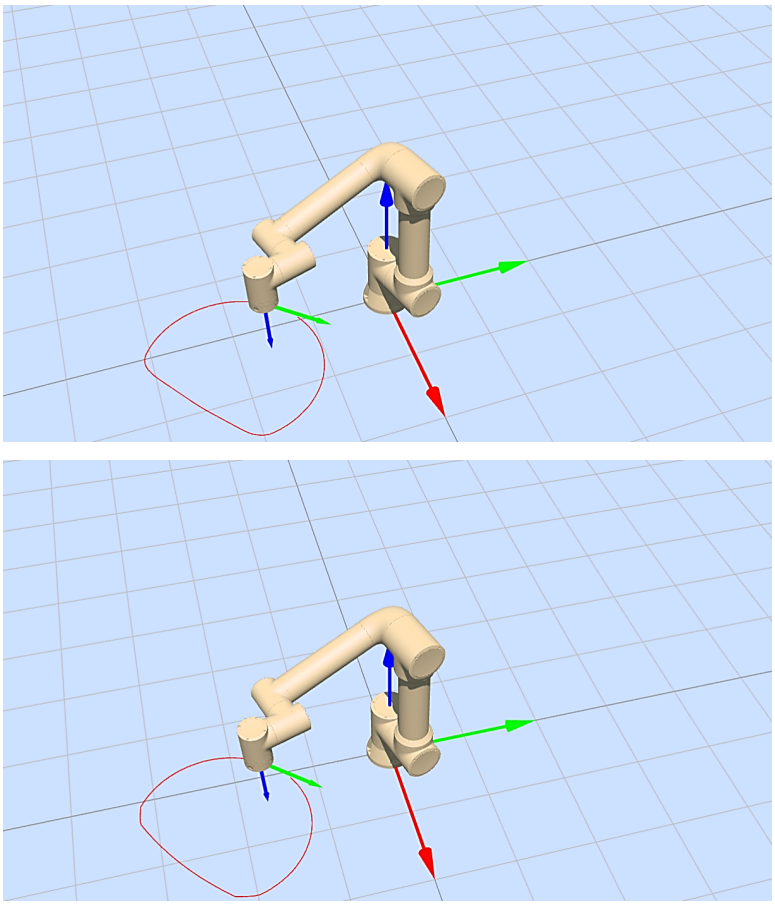


Figure 9.21-7 Example of an elbow singular avoidance trajectory(top: Cartesian space, bottom: joint space)

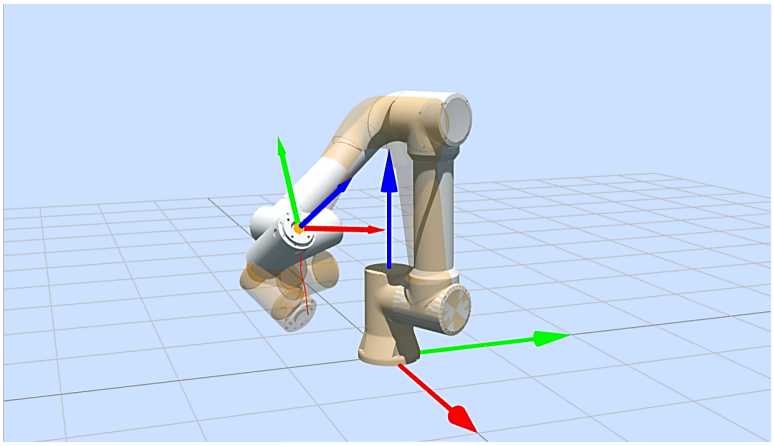


Figure 9.21-8 Example of a wrist singular avoidance trajectory (joint space)

6.If the start and end points of the movement are within the set singular range, when more than one singularity occurs during the movement, or even when two or more singular situations occur at the same time, the interface will display a pop-up window of “[Warning] Singular Pose” to indicate that the current singular situation cannot be avoided.

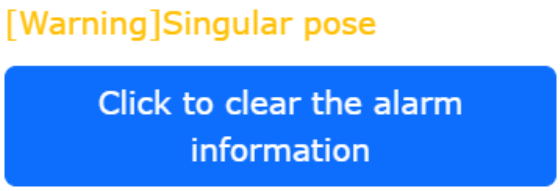


Figure 9.21-9 The current singular situation cannot be avoided

1.3.9.22 Singularity crossing function in automatic mode

1.3.9.22.1 Overview

When the robot executes LIN or ARC commands and passes through a singularity, its speed may fluctuate drastically, leading to unstable motion control and even potential damage to the equipment. By utilizing the singularity crossing function, the robot can smoothly pass through singularities. This manual takes the example of the LIN command passing through a wrist singularity to illustrate how to use the singularity crossing function in automatic mode.

1.3.9.22.2 Operation Flow

1. Teach the robot to execute the LIN command with two motion control points (named wristlin1 and wristlin2 in this manual).
2. Click the “Program” button, then click “Coding”, select the “PTP” command under “Motion command”, and add the first motion point.

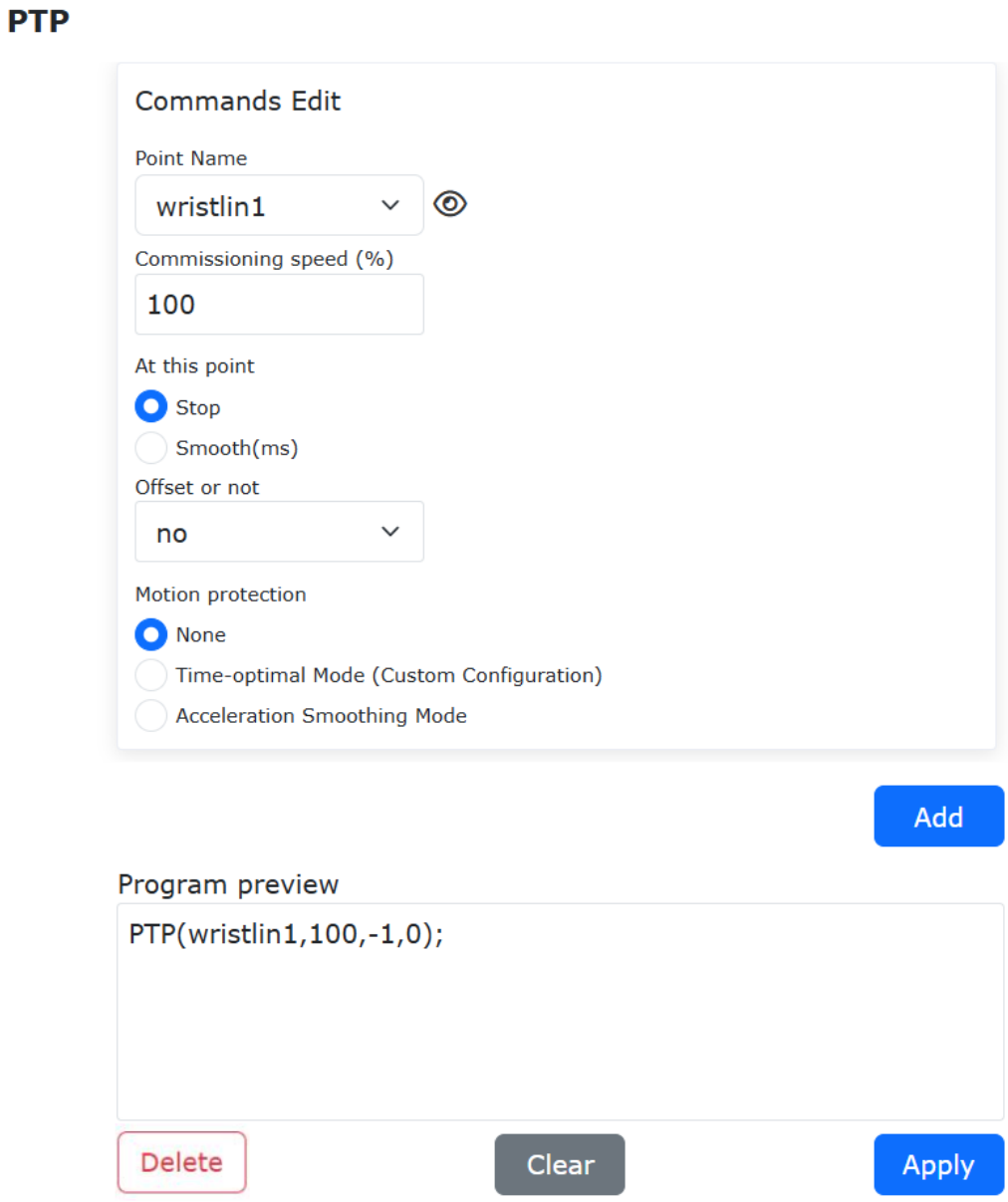


Figure 9.22-1 Add the first motion point

3. Select the “LIN” command under “Motion command”, and add the second motion point. In the “Motion protection” section, select “Singularity Crossing,” and set the adjustment ranges for shoulder singularity, elbow singularity, and wrist singularity.

Lin

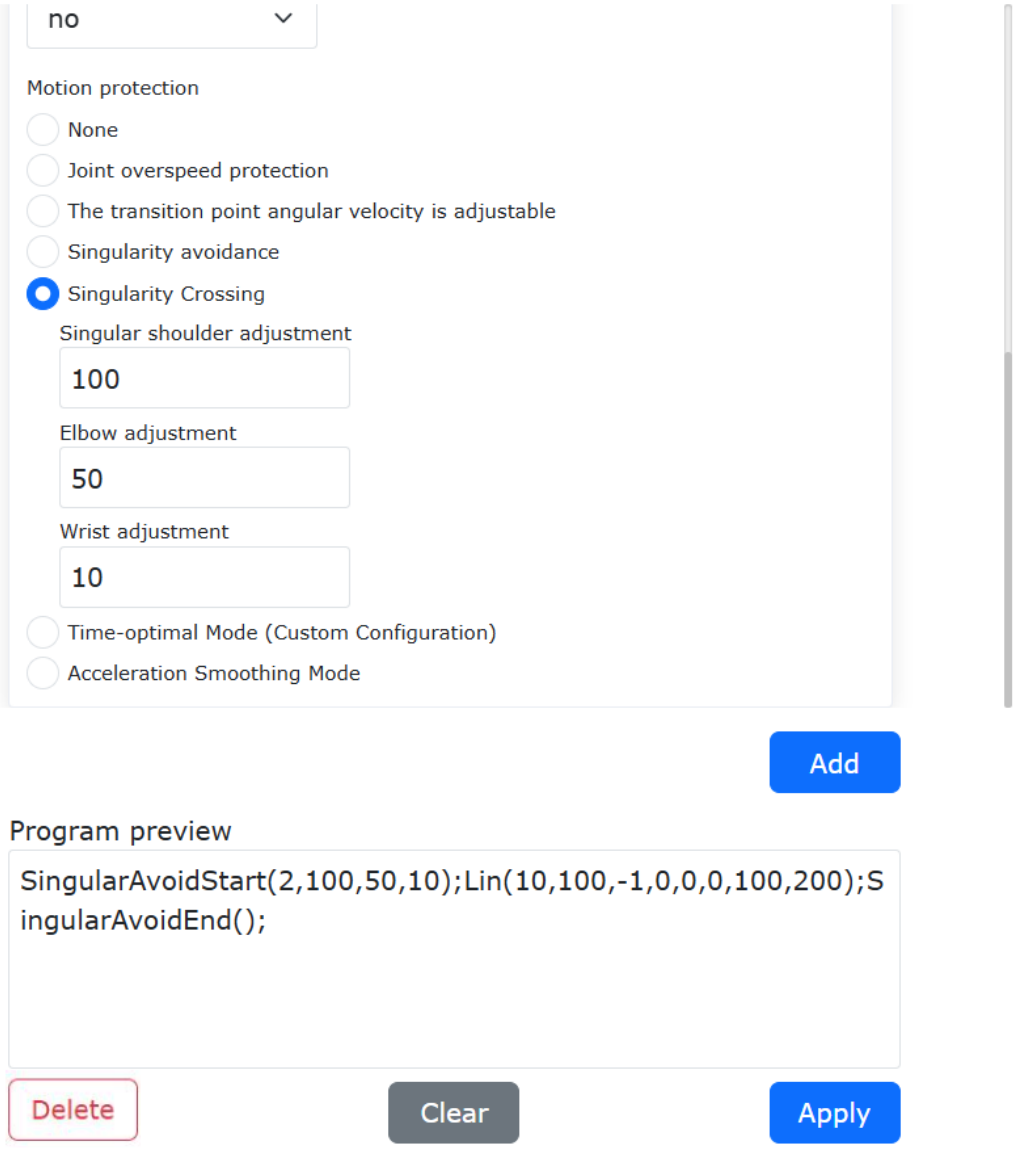


Figure 9.22-2 Set the singularity crossing parameters

4. Generate and run the Lua program. The typical LIN command program for singularity crossing in automatic mode.



Figure 9.22-3 Typical singularities crossing LIN instructions

5. Observe the robot's motion results, and adjust the robot's movement speed and singularity settings range to achieve different motion accuracy and impact.

1.3.9.22.3 Precision Impact Table

1. Wrist singularity is the most easily triggered singularity type for the robot. A table comparing the precision and impact of wrist singularity for LIN and ARC commands has been compiled. The comparison table for LIN and ARC commands are shown below, (indicates that a collision warning was triggered).

Table 9.22-3-1 Error of wrist singularity LIN command (Unit: mm

Range / Velocity	2	20	40	60	80	100
2 mm	0.19	0.20	0.20	0.21		
4 mm	0.14	0.14	0.14	0.14	0.14	0.14
6 mm	0.40	0.40	0.41	0.41	0.42	0.42
8 mm	0.82	0.83	0.83	0.84	0.83	0.84
10 mm	1.38	1.38	1.39	1.39	1.39	1.41

Table 9.22-3-2 Linear jerk of wrist singularity LIN command (Unit: m/s3)

Range / Velocity	2	20	40	60	80	100
2 mm	0.605	12.040	11.370	2743.000		
4 mm	0.916	34.620	110.900	241.300	303.900	400.700
6 mm	0.906	59.700	139.600	343.700	445.600	582.900
8 mm	1.073	67.480	199.600	438.300	553.400	623.900
10 mm	1.013	69.490	195.800	556.600	649.300	953.300

Table 9.22-3-3 Angular jerk of wrist singularity LIN command (Unit: °/s^3)

Range / Velocity	2	20	40	60	80	100
2 mm	1122	25140	24780	54890		
4 mm	305	9035	26030	39330	60510	80330
6 mm	219	8161	19450	84700	109300	143400
8 mm	478	6651	19780	121600	150500	162100
10 mm	281	5296	14470	161600	177300	256000

Table 9.22-3-4 Error of wrist singularity ARC command (Unit: mm)

Range / Velocity	2	20	40	60	80	100
2 mm	1.06	1.06	1.05	1.05		
4 mm	1.58	1.59	1.60	1.62		
6 mm	3.31	3.34	3.35	3.32	3.39	3.33
8 mm	5.81	5.83	5.87	5.87	5.87	5.96
10 mm	9.06	9.09	9.12	9.17	9.17	9.22

Table 9.22-3-5 Linear jerk of wrist singularity ARC command (Unit: m/s3)

Range / Velocity	2	20	40	60	80	100
2 mm	13.970	643.000	2230.000	3408.000		
4 mm	0.635	24.850	42.480	76.990		
6 mm	3.000	19.960	45.350	57.120	77.050	59.800
8 mm	1.494	27.830	90.290	124.200	148.400	168.000
10 mm	0.460	31.870	112.600	211.000	229.300	117.500

Table 9.22-3-6 Angular jerk of wrist singularity ARC command (Unit: $°/s^3$)

Range / Velocity	2	20	40	60	80	100
2 mm	3378	85380	228600	351900		
4 mm	1098	31360	71460	104800		
6 mm	390	15770	43650	79330	93930	124200
8 mm	315	10270	28770	57000	75840	94050
10 mm	504	6108	21470	34920	47280	97160

2. Since shoulder singularity and elbow singularity correspond to the robot's minimum and maximum working boundaries, respectively, precision cannot be used as an evaluation metric. Therefore, an impact comparison table for shoulder singularity is compiled, as shown in Tab, and an impact comparison table for elbow singularity is compiled, as shown below (where indicates that a collision warning was triggered).

Table 9.22-3-7 Linear jerk of shoulder singularity (Unit: m/s3)

Range / Velocity	2	20	40	60	80	100
40 mm	1.166	99.730	253.200	273.500		
70 mm	1.047	92.440	328.900	634.500	878.400	1499.000
100 mm	1.060	90.250	273.900	506.600	926.300	1555.000

Table 9.22-3-8 Angular jerk of shoulder singularity (Unit: $°/s^3$)

Range / Velocity	2	20	40	60	80	100
40 mm	396	89.83	824	348		
70 mm	428	121	681	167	1783	35690
100 mm	440	151	473	246	1495	39280

Table 9.22-3-9 Linear jerk of elbow singularity (Unit: m/s3)

Range / Velocity	2	20	40	60	80	100
40 mm	0.905	14.430	52.080	87.380	129.400	657.000
70 mm	1.144	24.320	79.580	270.300	793.300	1478.000
100 mm	1.852	27.930	112.700	328.100	583.000	758.600

Table 9.22-3-10 Angular jerk of elbow singularity (Unit: °/s^3)

Range / Velocity	2	20	40	60	80	100
40 mm	347	128	148	142	63	38050
70 mm	424	132	141	21780	56190	95610
100 mm	46	1443	6194	19940	35170	46770

1.3.9.23 Real-time forward trajectory planning function

1.3.9.23.1 Overview

Real-time forward-looking trajectory planning is based on current and future path information, dynamically adjusting the robot's motion parameters such as speed and acceleration to ensure smoothness, continuity and accuracy of motion. By predicting the robot's future position and posture, forward-looking control can respond before the key points of the path, thereby avoiding unstable motion or trajectory errors due to sudden changes in speed and acceleration.

1.3.9.23.2 Operation process

Step1: Prepare a trajectory point file in "txt" format, in which each trajectory point is represented by a Cartesian pose.

Step2: Click the "Teaching Program"->"Programming" button in sequence, select the "Trajectory Forward" command in "Motion Command", and import and delete the trajectory file in "Command Configuration".

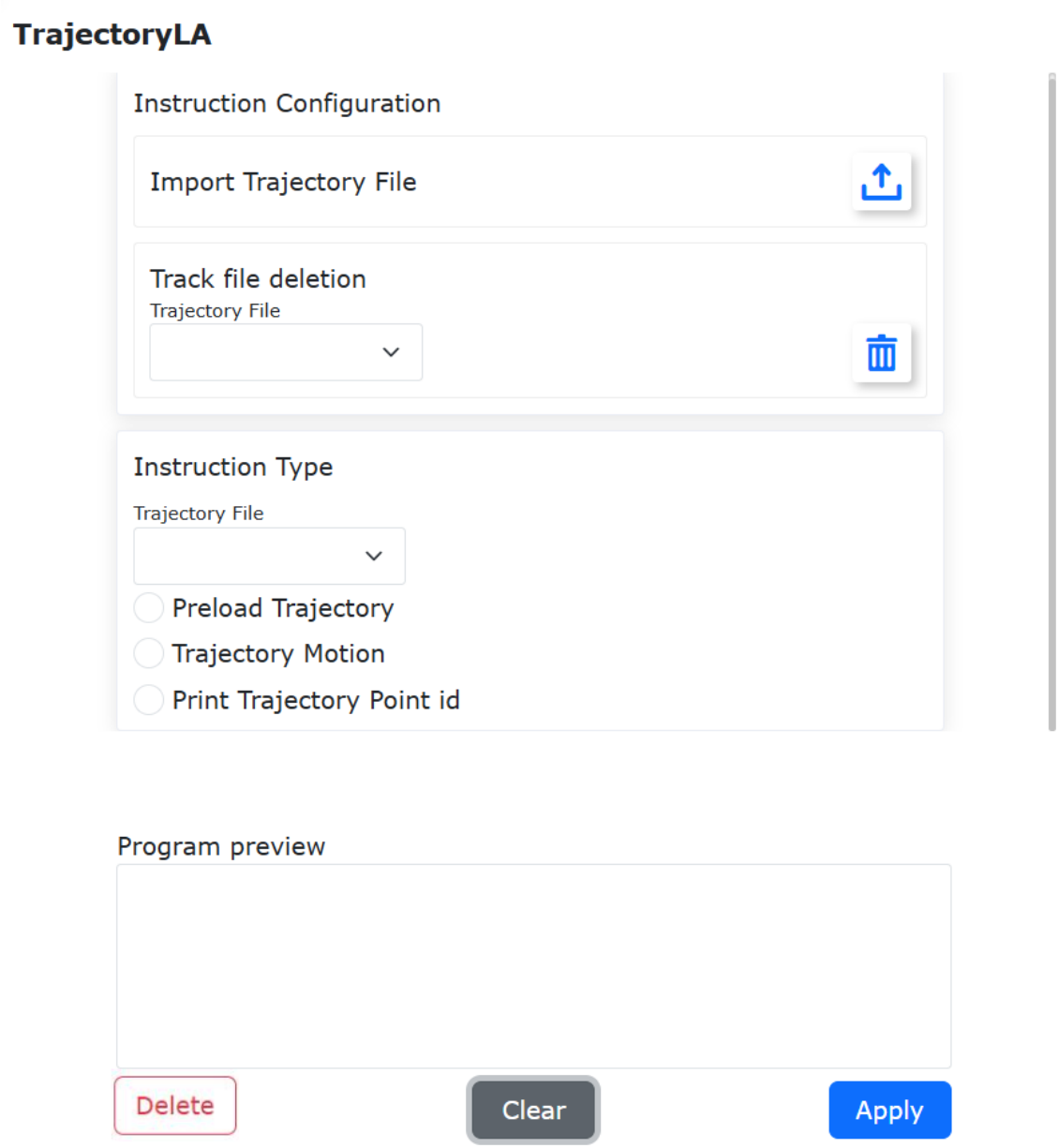


Figure 9.23-1 Import and delete trajectory files

Step3: Select the trajectory file to run and add the "Trajectory Preload" command:

First, choose the fitting method for trajectory points in "Curve Fitting Method", including "Linear Connection", "Linear Fitting", "B-spline Curve", "Polynomial Optimization Method", etc. When selecting "Linear Fitting", additional error limits need to be set, which is not required for other methods.

Then set the smoothing method and smoothing precision. Finally, configure the maximum velocity, maximum acceleration and maximum jerk during operation. The "Uniform Motion" option can be enabled for constant velocity lookahead, which will make the robot perform lookahead at constant speed when activated.

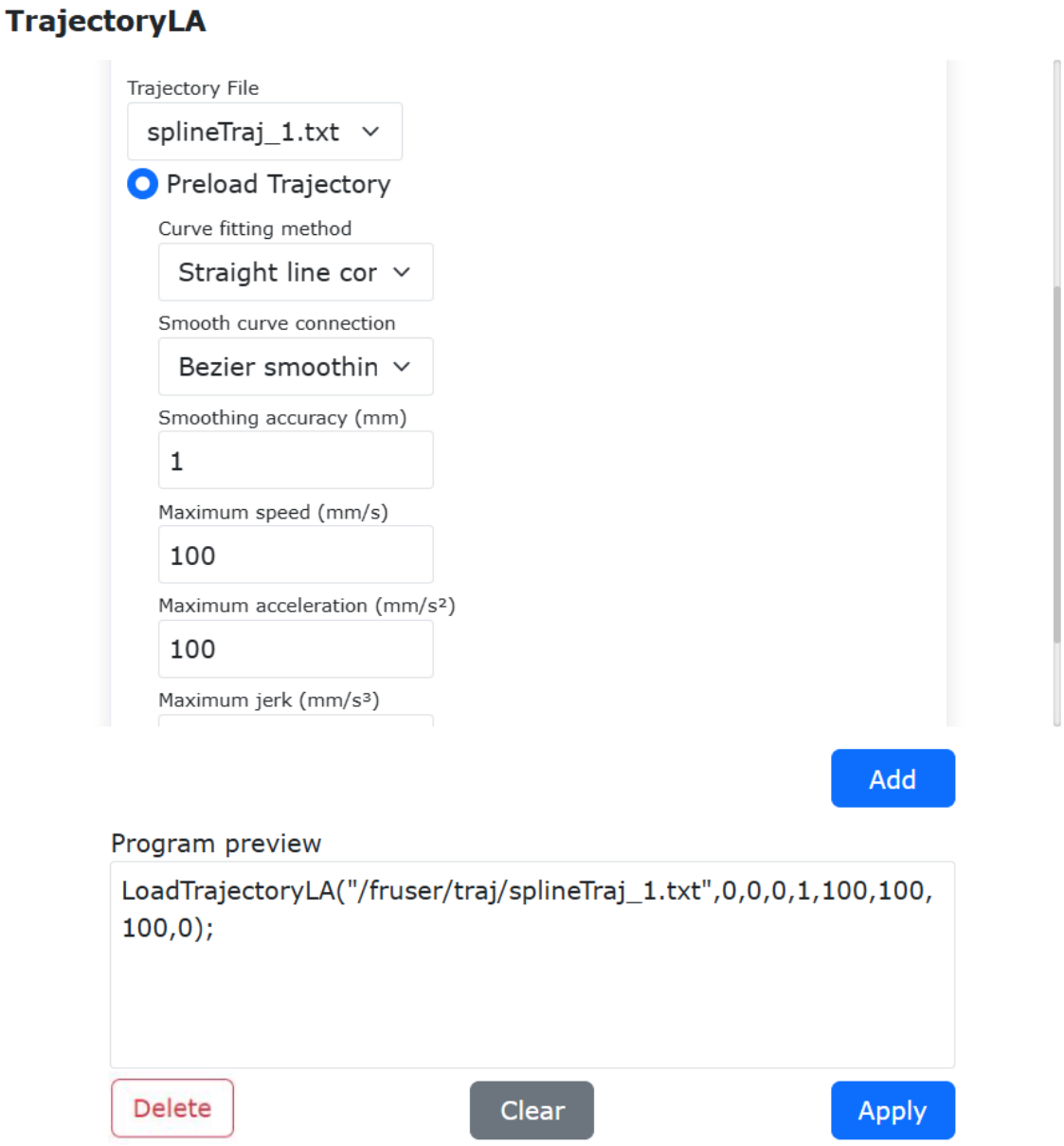


Figure 9.23-2 Setting “Linear Fitting” Parameters for Trajectory Preloading

```
1↔ LoadTrajectoryLA("/fruser/traj/horse.txt",2,0,0,1,100,100,100,1)
2↔ startPose = GetTrajectoryStartPose("/fruser/traj/horse.txt")
3↔ tool_num = GetActualTCPNum()
4↔ wobj_num = GetActualWObjNum()
5↔ MoveCart(startPose,tool_num,wobj_num,100,100,100,-1,-1)
6↔ MoveTrajectoryLA("/fruser/traj/horse.txt")
```

Figure 9.23-3 Setting Trajectory Preloading Parameters

Step4: Add the "Trajectory Motion" command and generate Lua program. The imported trajectory file can be processed with real-time lookahead trajectory planning by running the Lua program. A typical real-time lookahead trajectory planning program is shown below.

```
1↔ LoadTrajectoryLA("/fruser/traj/horse.txt",2,0,0,1,100,100,100,1)
2↔ startPose = GetTrajectoryStartPose("/fruser/traj/horse.txt")
3↔ tool_num = GetActualTCPNum()
4↔ wobj_num = GetActualWObjNum()
5↔ MoveCart(startPose,tool_num,wobj_num,100,100,100,-1,-1)
6↔ MoveTrajectoryLA("/fruser/traj/horse.txt")
```

Figure 9.23-4 Typical Program for Real-time Lookahead Trajectory Planning (B-spline Curve)

Step 5: For the "LoadTrajectory" command line in the lua program, click the edit button to modify the setting parameters to achieve different trajectory planning effects.

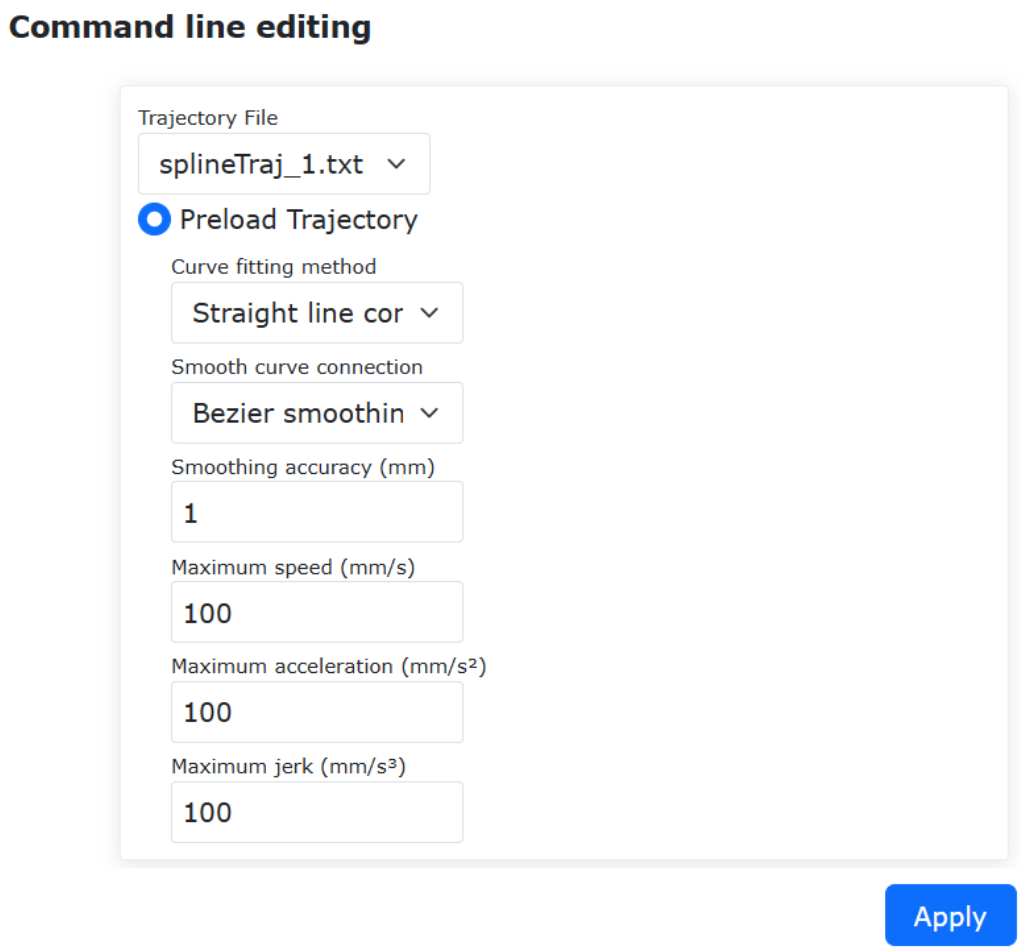


Figure 9.23-5 Modify the setting parameters

1.3.9.24 Swing amplitude monotonous gradual arc tracking function

Swinging motion can realize two swinging amplitude switching modes: “sudden change” and “gradual change”.

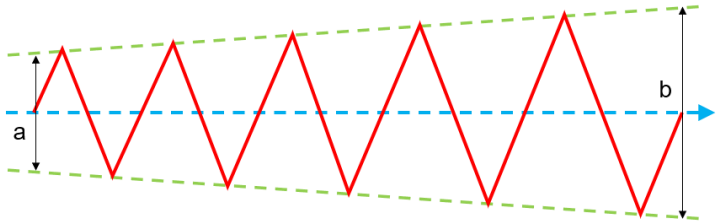
The “sudden change” mode refers to the mode of directly switching the swinging parameters of the previous section with the swinging parameters of the next section. It can be realized by setting two adjacent swinging motions with different parameters, or by issuing a new swinging number in real time during the swinging motion (see the manual section corresponding to the function for details, which will not be repeated here).

The “gradual change” mode means that in the current swinging motion, the swinging amplitude set at the beginning gradually changes to the swinging amplitude set at the end.

The swinging parameter gradual switching mode only supports the process of linear swinging.

1.3.9.24.1 Introduction

The swinging motion trajectory of monotonic gradual swinging amplitude is shown in the figure below.



Among them, the blue line is the swing direction, a is the swing amplitude of the starting point, and b is the swing amplitude of the ending point. The swing amplitude gradually changes during the movement.

Note: Please note that currently only the starting point and the end point are of the same type, the swing amplitude is different (from a to b), and the other parameters are consistent. It is recommended to check the swing parameters before executing the swing.

The operation process of setting a swing amplitude gradual swing is as follows:

Step 1: Click “Teaching Program”, “Programming”, select and click the “Swing” button under “Motion Instructions” to enter the swing instruction configuration page.

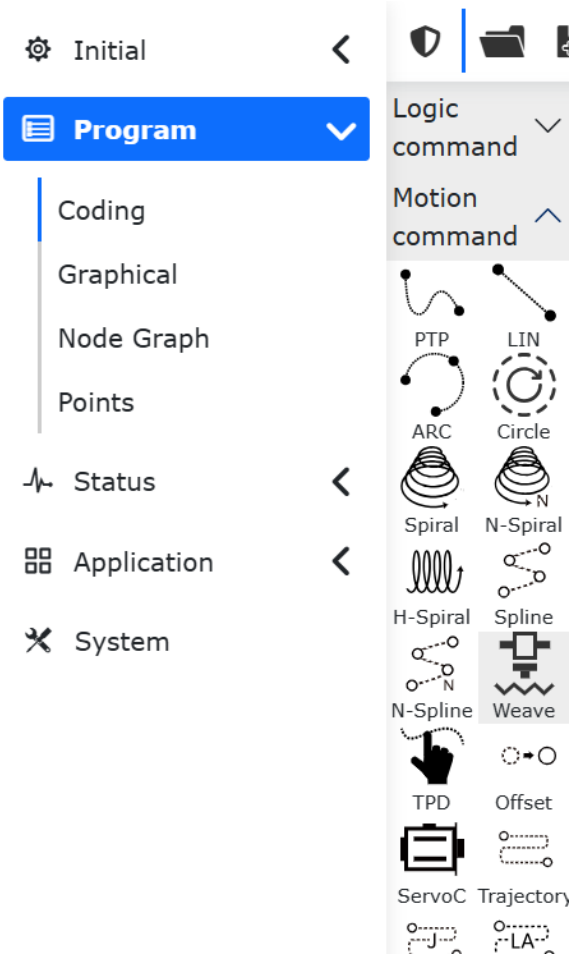


Figure 9.24-1 Click the swing function button

Step 2: Select the swing parameter number at the start of the swing in the command editor, click “Start Swing” and then click the “Add” button.

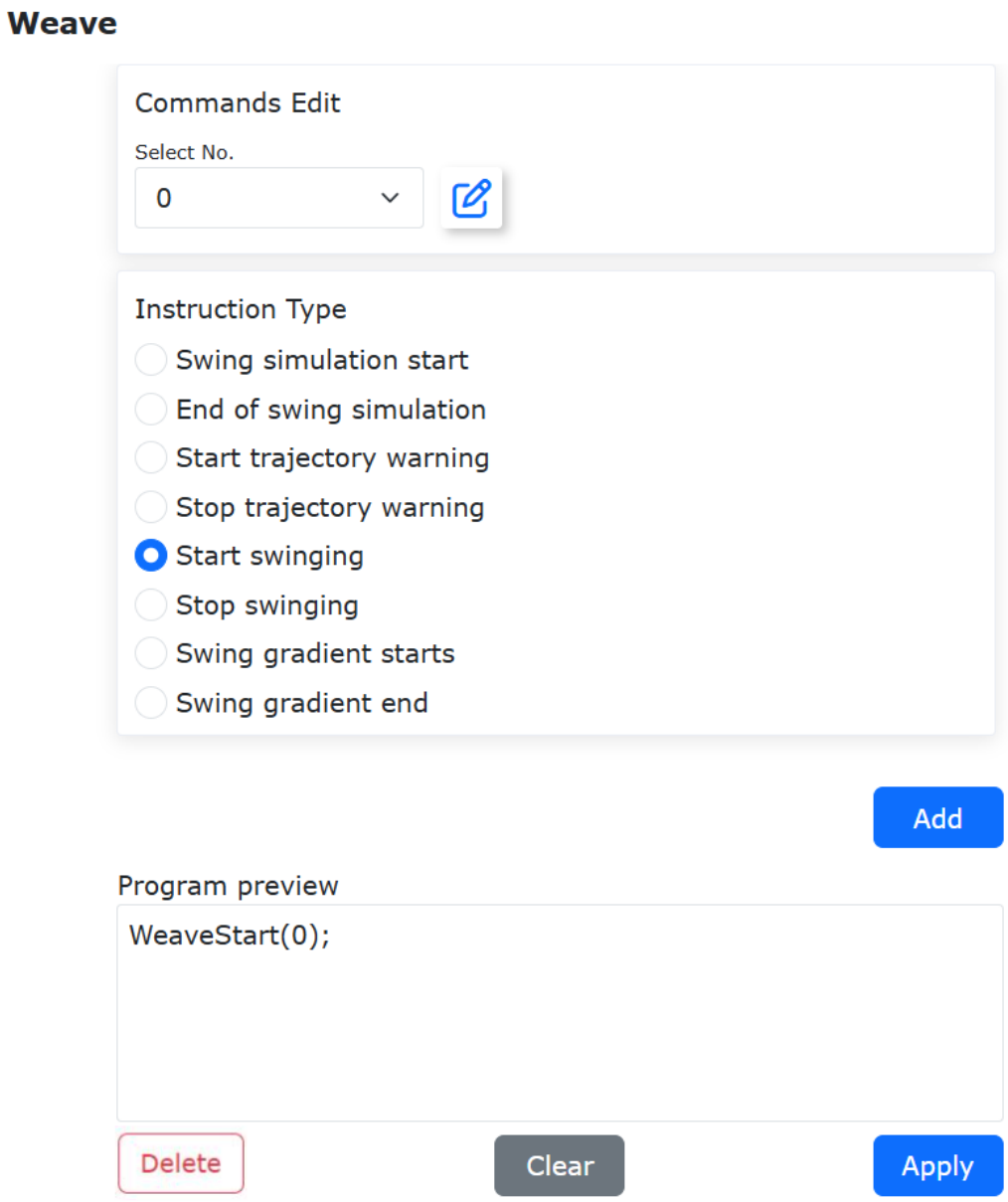


Figure 9.24-2 Add starting swing parameters

Step 3: Select the target number of the swing gradient, click “Swing gradient start”, and click the “Add” button.

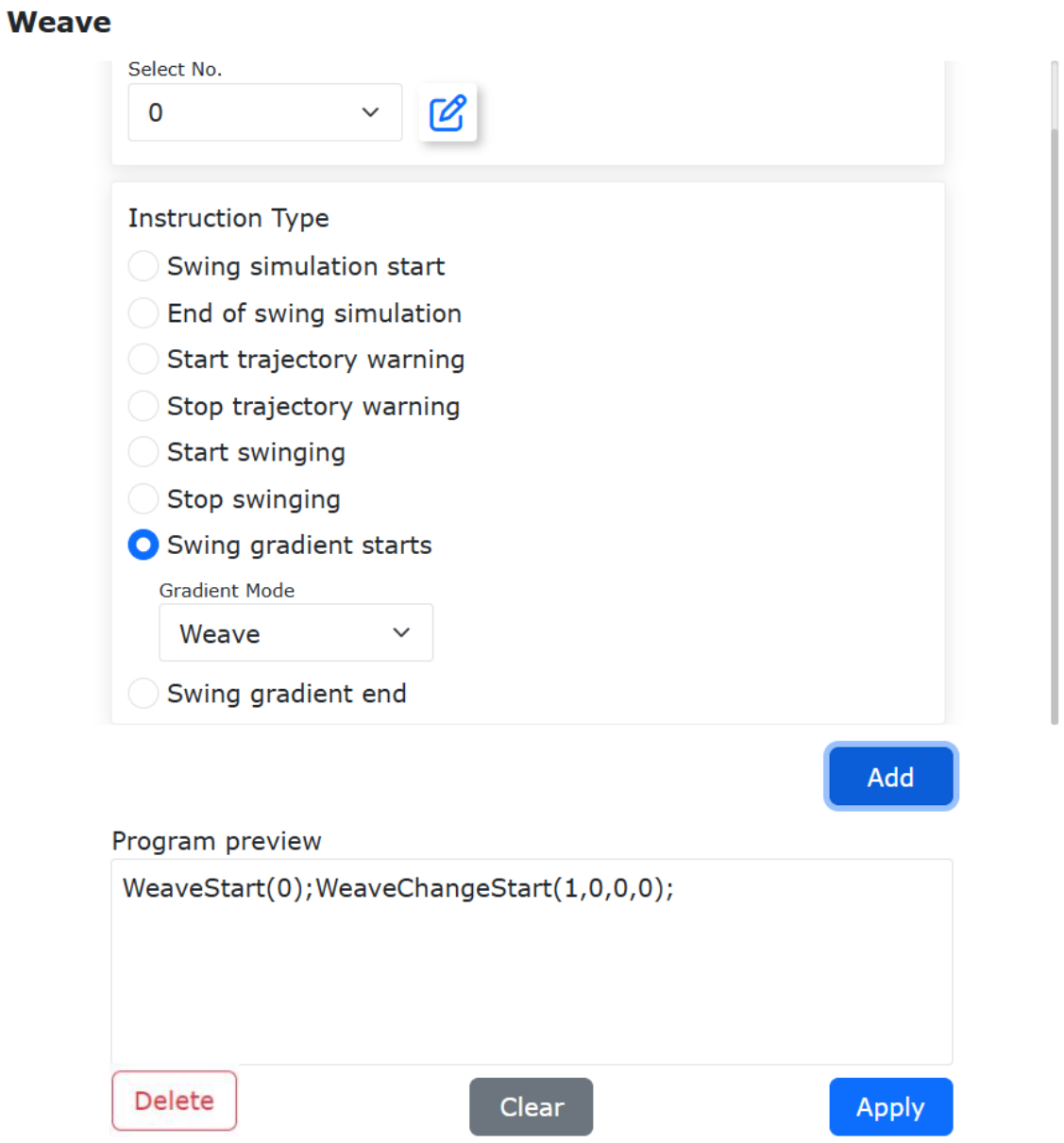


Figure 9.24-3 Add gradual swing parameters

Step 4: After adding the corresponding linear motion, click “Swing Gradient End” and click Add, then click “Stop Swing” and add, complete the setting of a gradual swing motion, and click “Apply” to add it to the LUA program.

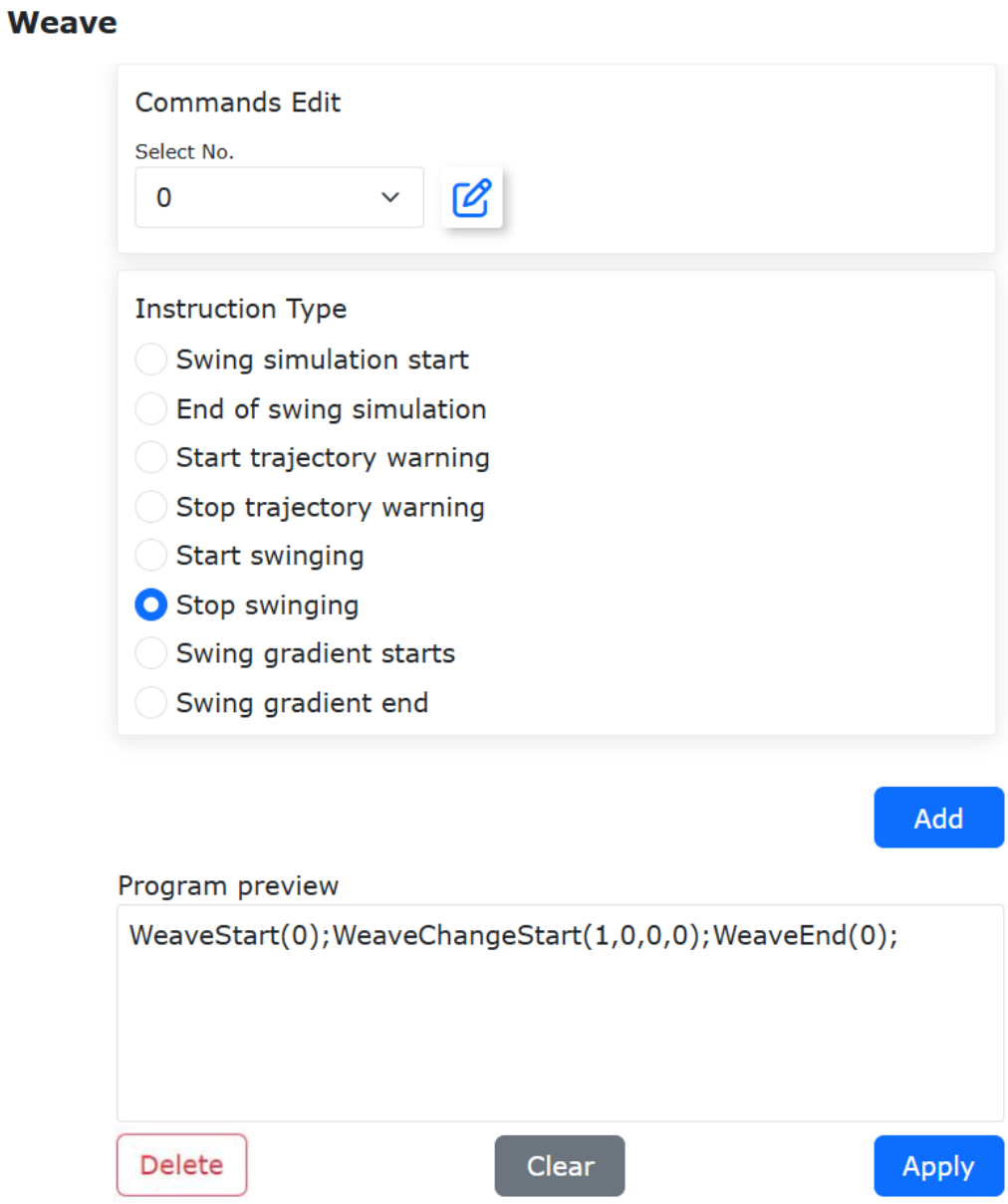


Figure 9.24-4 Implement a complete swing amplitude gradient motion LUA instruction

1.3.9.25 Offset arc tracking function

During the arc tracking welding process, the robot defaults to adjusting the welding gun swing center to track the workpiece groove center according to the current information to keep consistent, but some process requirements require that the welding gun swing center has a certain offset relative to the workpiece groove center.

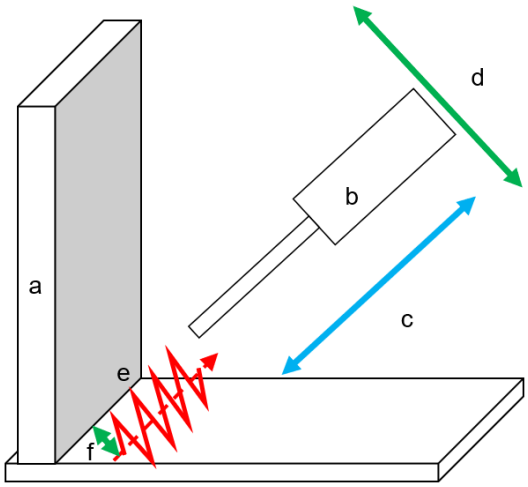


Figure 9.25-1 Typical scenarios of bias arc tracking

Typical scenarios of the bias arc tracking function include: a. welding workpiece (welding groove is right angle or acute angle), b. welding gun, e. groove center line. The arc tracking function realizes the following for the welding groove: c. up and down (depth) direction tracking and d. left and right (center) direction tracking, f. left and right direction tracking offset distance.

To realize bias arc tracking, two methods of setting left and right offsets can be selected, namely “sampling” and “percentage” adjustment methods.

1.3.9.25.1 Sampling bias arc tracking

The sampling method is to collect the left and right current values during the swinging within a certain cycle as a reference after the swing welding arc is started, and compare the sampling current with the reference current in the subsequent welding process to obtain the tracking direction.

The sampling method requires teaching the swing welding starting position to the required offset, the offset shall not be greater than the swing amplitude, and the weld needs to cover the splicing groove.

The sampling bias instruction setting process is as follows:

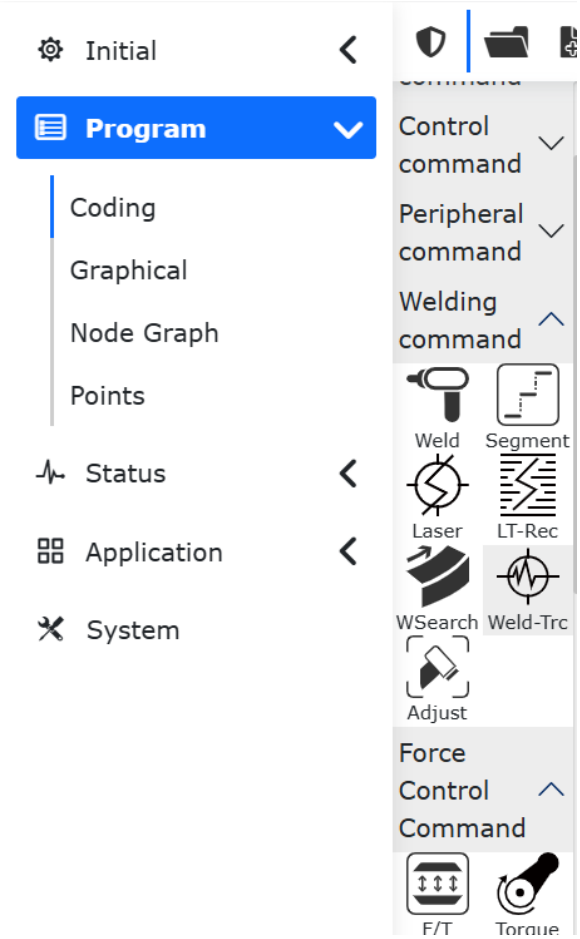


Figure 9.25-2 Click the arc tracking command button

Step 1: Click “Teaching Program”, “Programming”, select and click the “Arc Tracking” button under “Welding Command” to enter the arc tracking command configuration page.

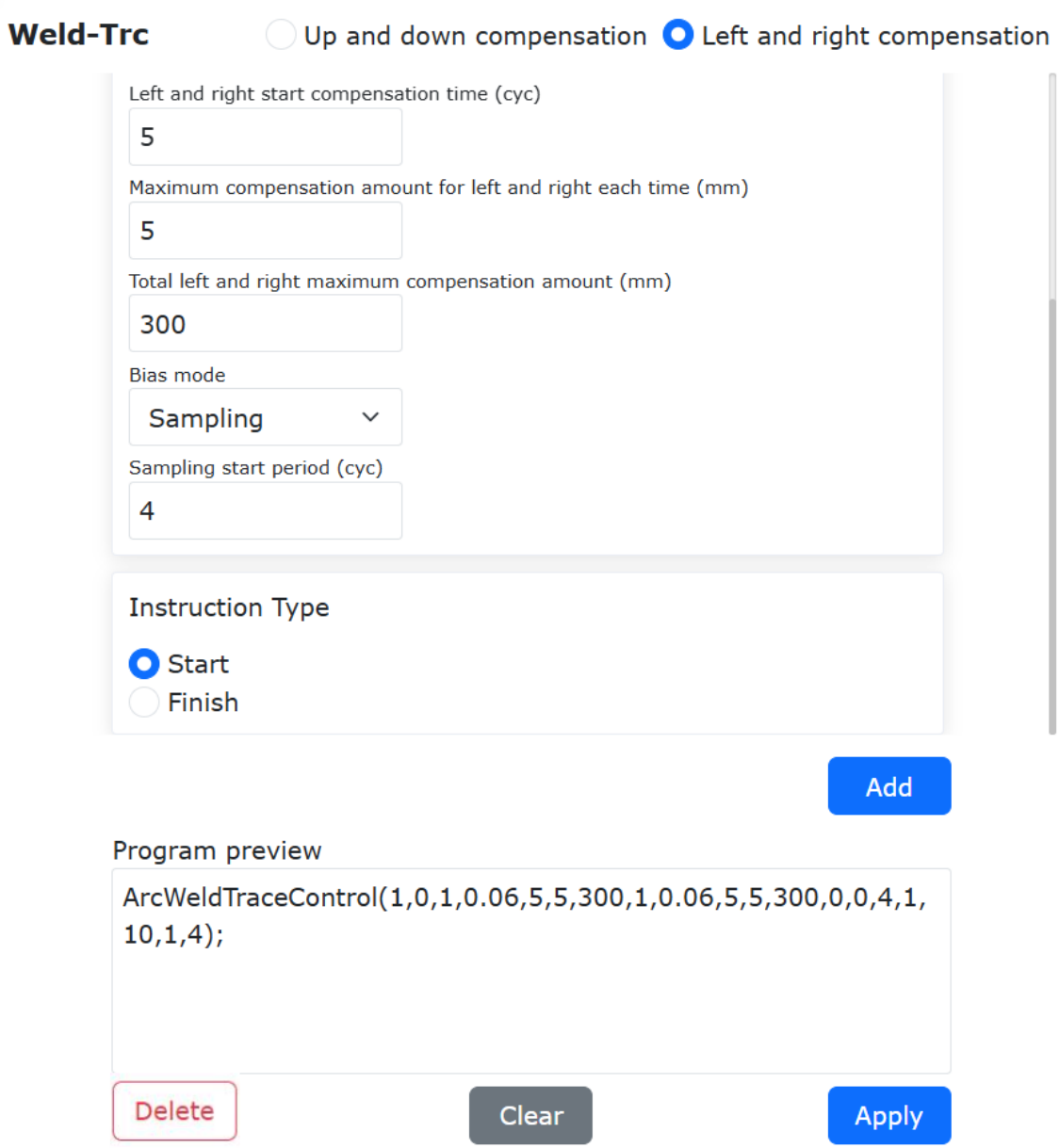


Figure 9.25-3 Sampling bias arc tracking configuration page

Step 2: Bias arc tracking acts on left and right compensation. Click the "Left and Right Compensation" subpage, pull down and select "Sampling" in the bias mode, set the sampling start period (the sampling start period must be less than the left and right compensation start time), select "Start" for the instruction type, and click the Add button to generate the LUA instruction.

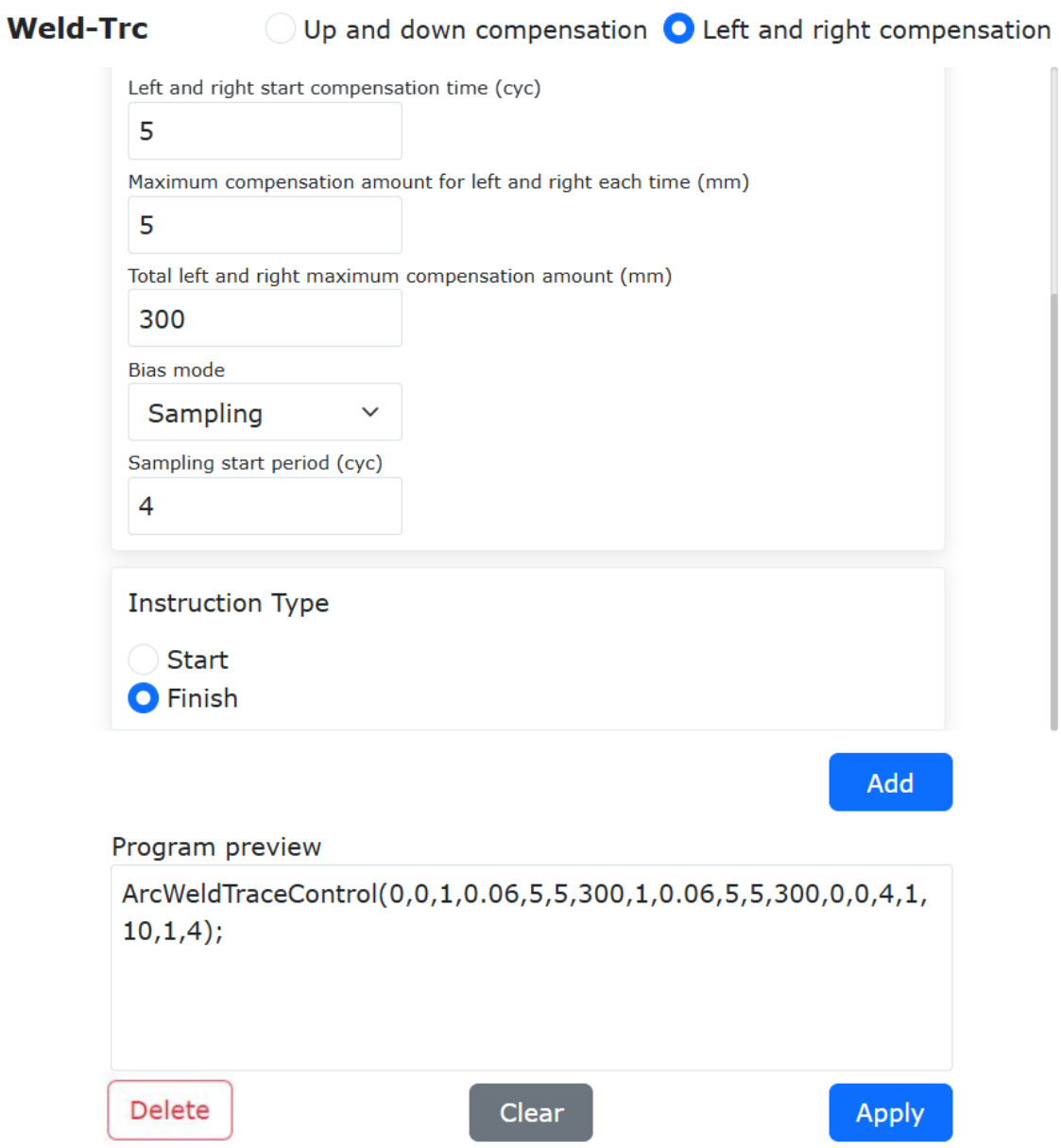


Figure 9.25-4 Add sampling bias arc tracking end instruction

Step 3: After adding the swing motion instruction, click to select the arc tracking instruction type “End”, and click Add to generate the corresponding LUA instruction.

1.3.9.25.2 Percentage bias arc tracking

The percentage bias is to gain the input sampling current by percentage during the arc tracking process, so that the current of the left and right swing cycles will deviate, and the robot will automatically compensate for the deviation signal.

Note: It should be noted that the smaller the swing amplitude and the larger the groove angle, the smaller the deviation of the left and right currents, and the smaller the adjustment percentage. It is recommended to debug with every 1% as the adjustment interval.

The percentage bias instruction setting process is as follows:

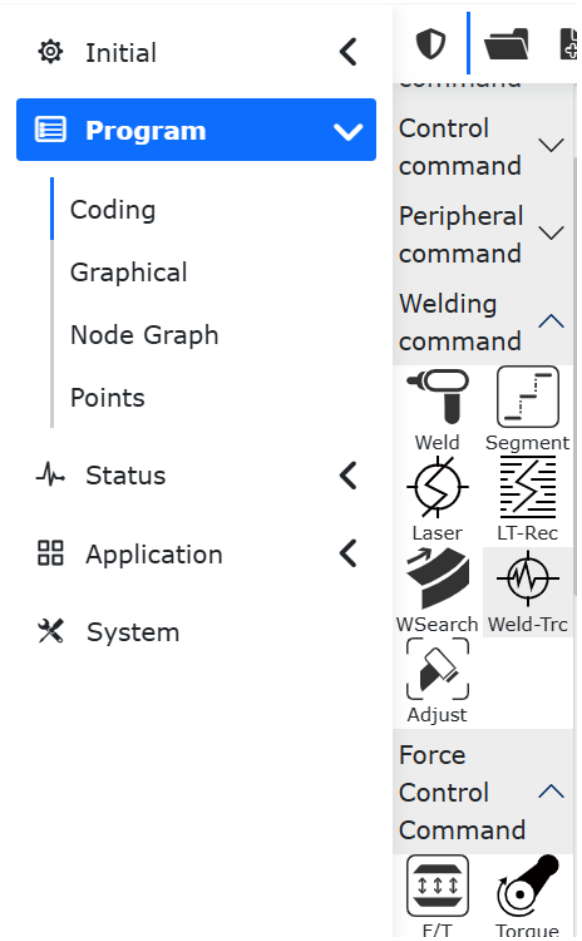


Figure 9.25-5 Click the arc tracking command button

Step 1: Click “Teaching Program”, “Programming”, select and click the “Arc Tracking” button under “Welding Command” to enter the arc tracking command configuration page.

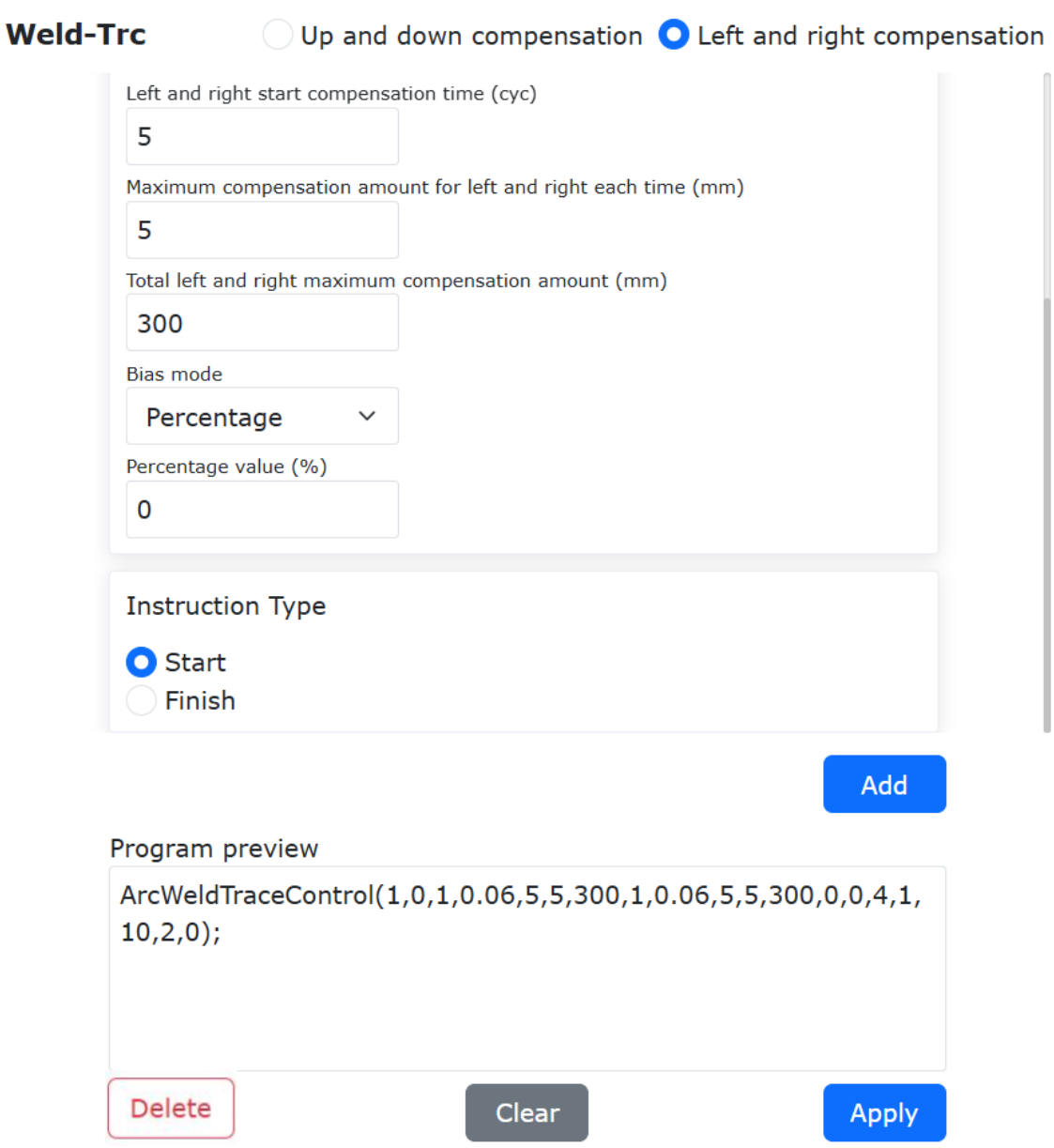


Figure 9.25-6 Sampling bias arc tracking configuration page

Step 2: Bias arc tracking acts on left and right compensation. Click the "Left and Right Compensation" subpage, pull down and select "Percent" in the bias mode, set the percentage value (positive value compensates the current gain of the first half cycle and the direction of the second half cycle, negative value is the opposite), select "Start" for the instruction type, and click the Add button to generate the LUA instruction.

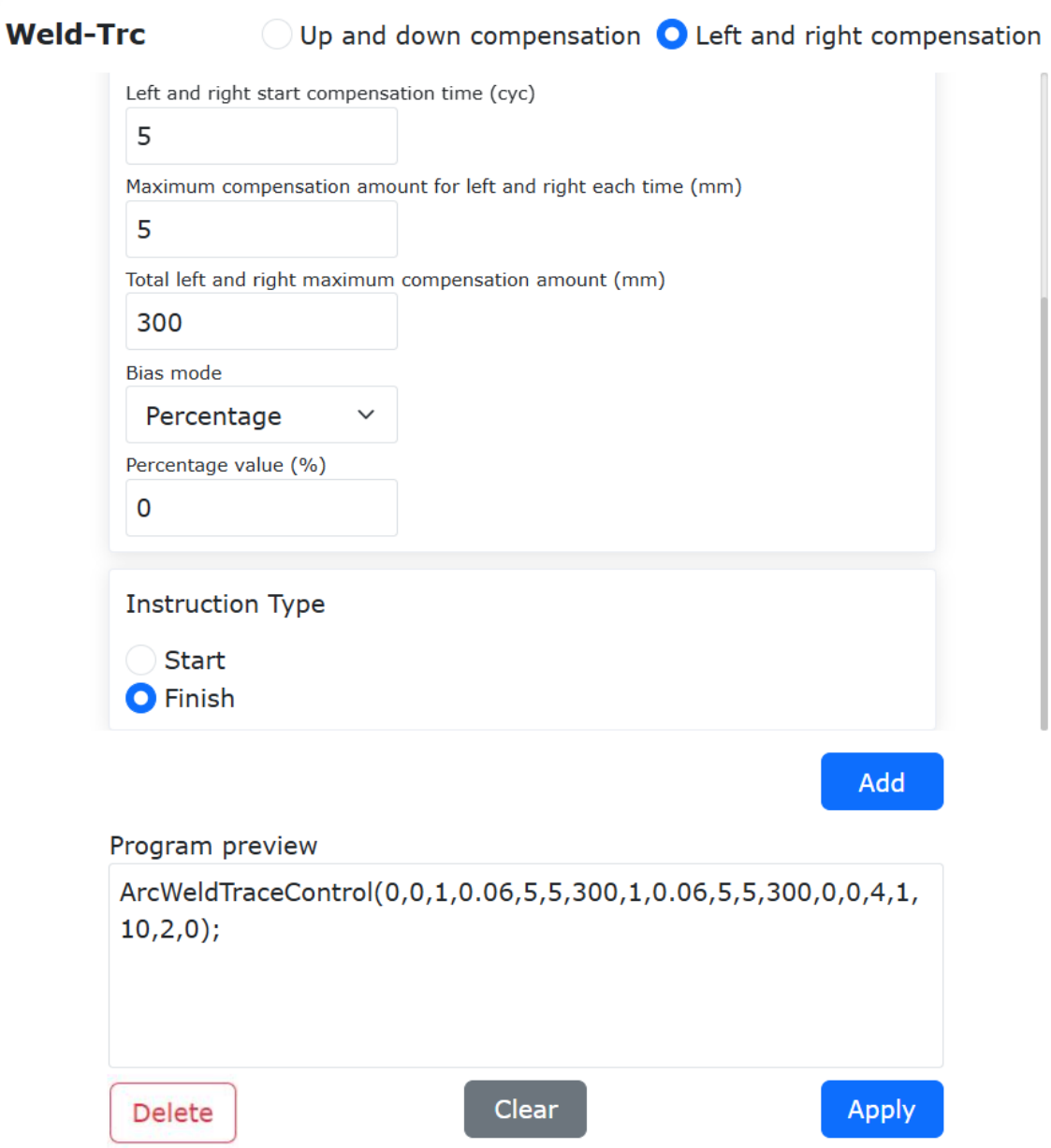


Figure 9.25-7 Add percentage bias arc tracking end instruction

Step 3: After adding the swing motion instruction, click to select the arc tracking instruction type “End”, and click Add to generate the corresponding LUA instruction.

The typical LUA program structure of a bias tracking is as follows:

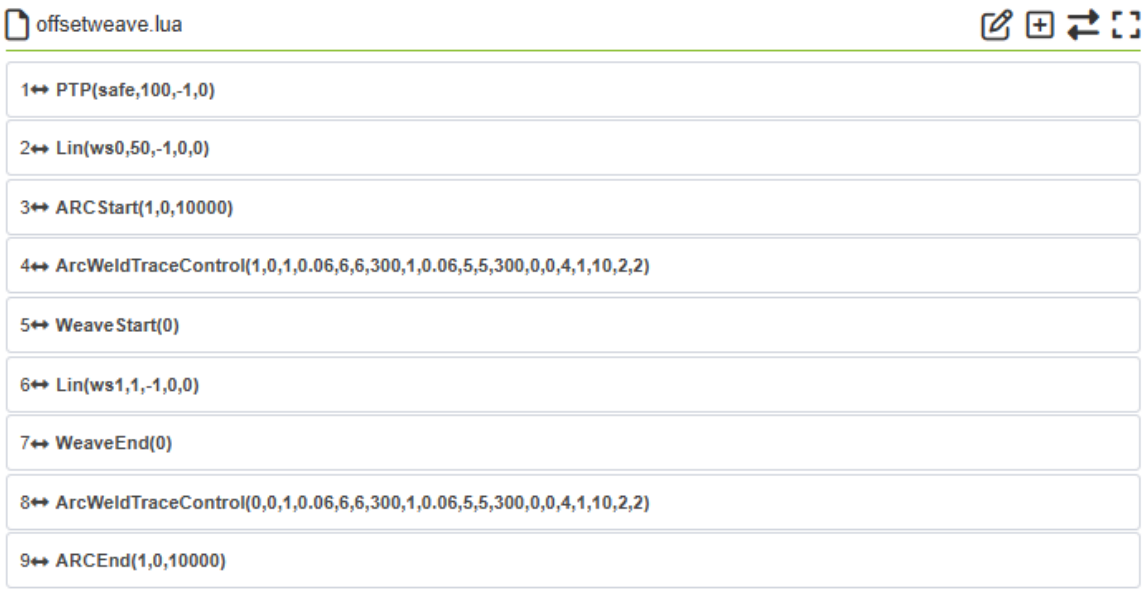


Figure 9.25-8 A typical bias arc tracking LUA program

1.3.9.26 Custom collision detection threshold function

1.3.9.26.1 Overview

The custom collision detection threshold function is a perfection of the current manual setting of the collision level function. If the current collision level setting does not meet the usage scenario, the user can set a custom collision detection threshold according to the actual situation. The collision detection threshold is divided into joint detection threshold and TCP detection threshold.

1.3.9.26.2 Function setting description

Step1: Click “Teaching Program”, select “Programming”, and open the corresponding interface.

Step2: Click the “New” button above, enter “example”, select “empty.lua”, and create a new lua script, as shown in Figure 1.

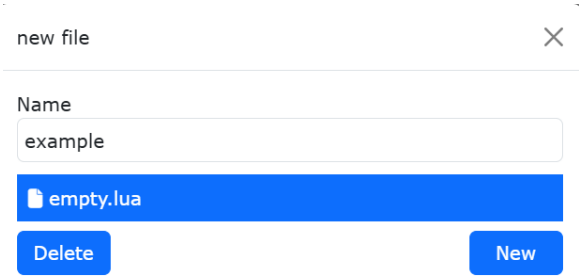


Figure 9.26-1 Create a new lua script

1.3.9.26.2.1 Joint detection threshold function setting instructions

1.3.9.26.2.2 Parameter setting instructions

Step1: In the control command interface, select the "collision detection" function, as shown in Figure 2. Click "collision detection on", and select "joint only" under the detection status bar. According to actual needs, modify the input values of J1-J6, the value range is, the unit is NM. In this mode, modifying the TCP threshold in the X-RZ direction does not take effect. According to actual needs, select "non-blocking" or "blocking" for blocking. Click the Add button to complete the opening instruction addition.

Step2: Click "collision detection off", click the Add button, and complete the closing instruction addition. The program preview interface is shown in Figure 3, click the "Apply" button to complete the function addition.

Note: The custom collision detection threshold function is a set of instructions, which needs to be closed in time after opening.

Step3: In the collision detection function, add the corresponding motion instructions, as shown in Figure 4.

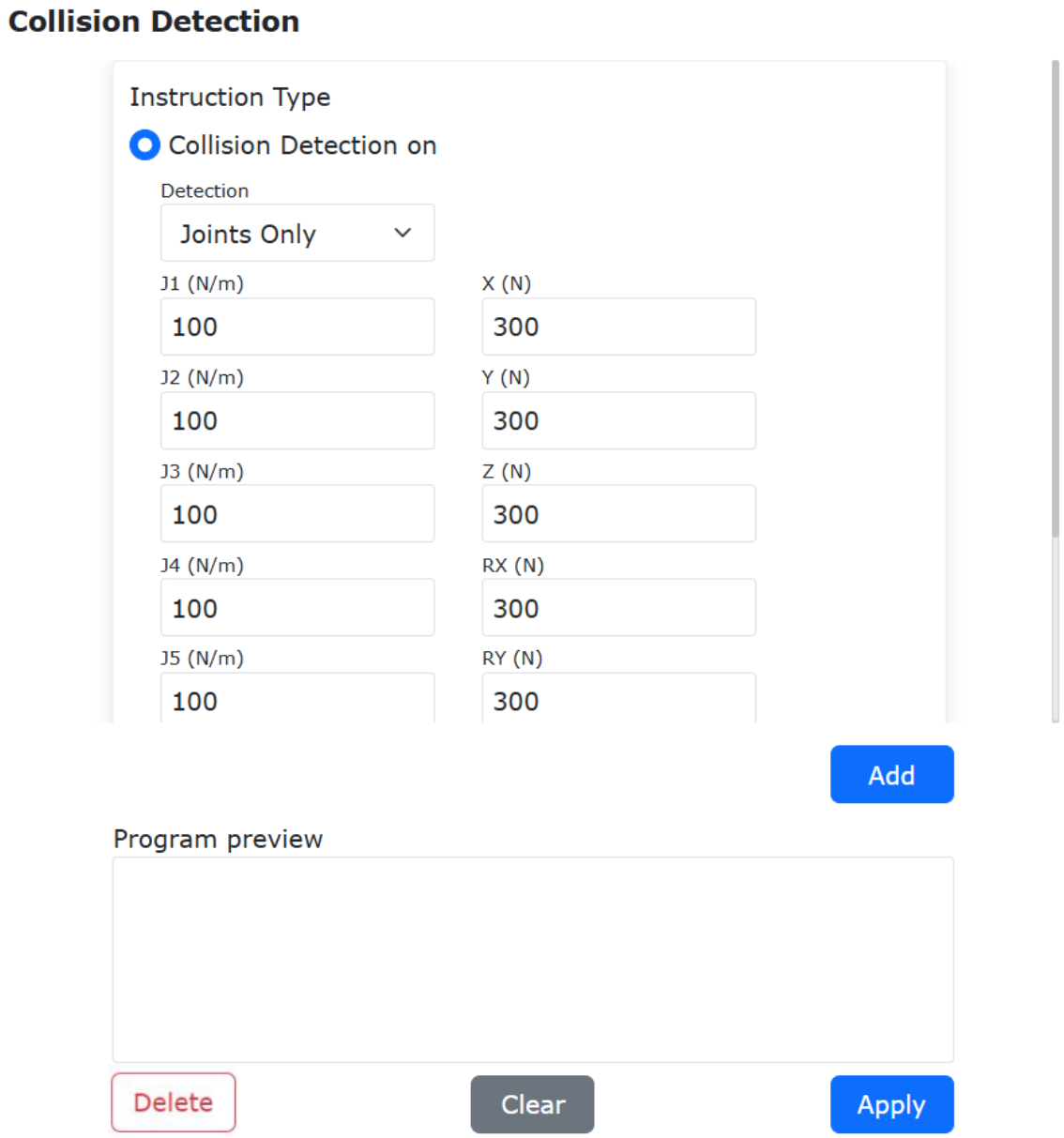


Figure 9.26-2 Joint detection threshold setting interface

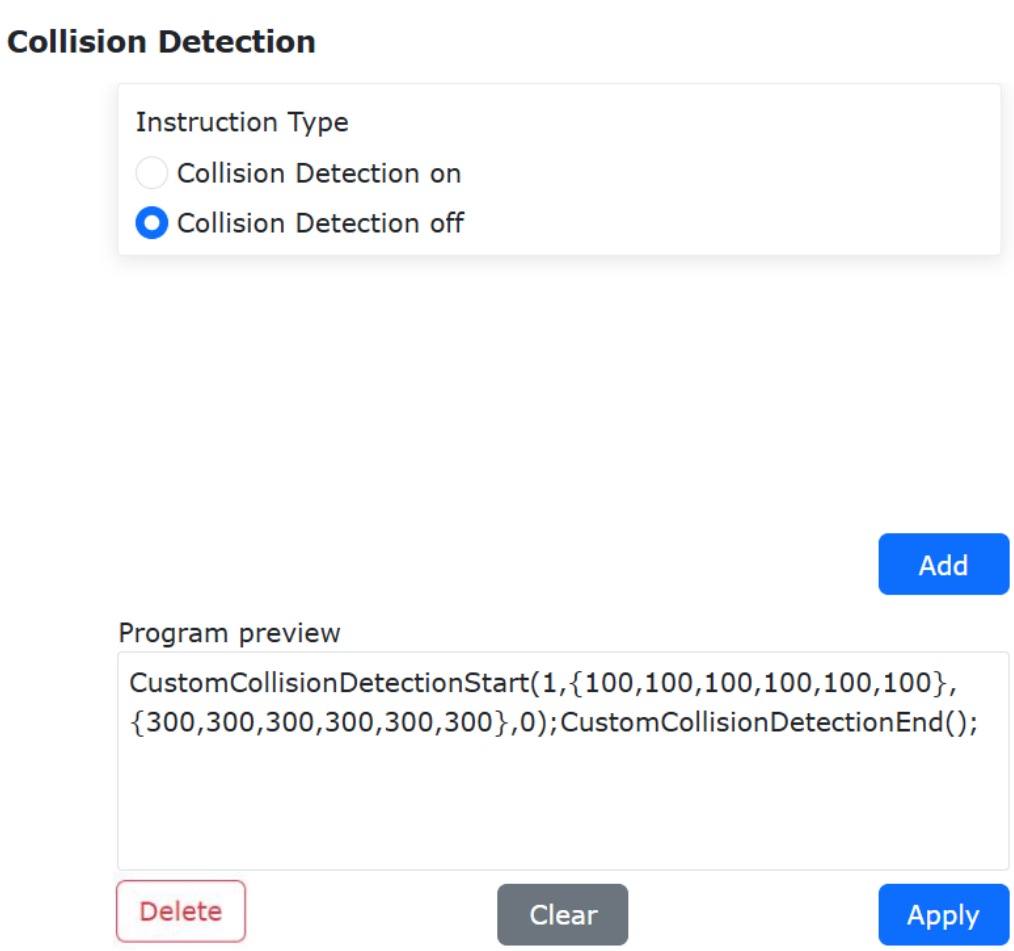


Figure 9.26-3 Program preview interface

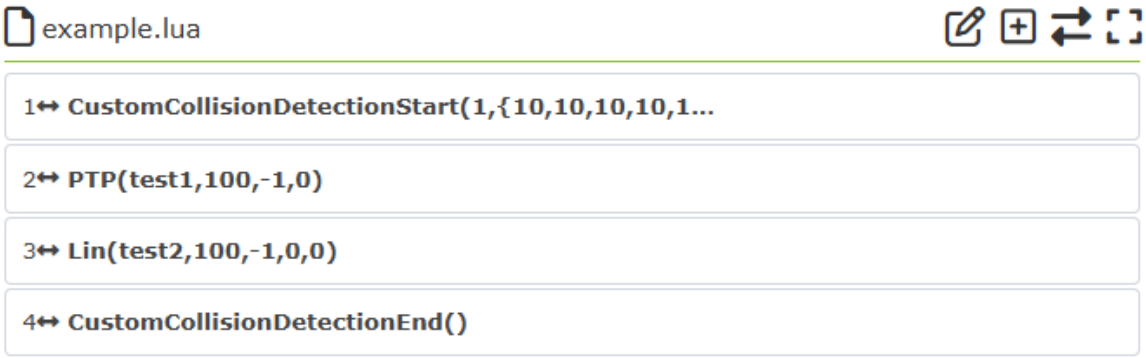


Figure 9.26-4 lua script program example interface

1.3.9.26.2.3 TCP detection threshold function setting instructions

1.3.9.26.2.4 Parameter setting instructions

Step1: In the control command interface, select the "collision detection" function, as shown in Figure 5. Click "collision detection on", and select "TCP only" under the detection status bar. According to actual needs, modify the input value of the X-RZ direction, the value range is, and the unit is N. In this mode, modifying the joint threshold of J1-J6 does not take effect. According to actual needs, select "non-blocking" or "blocking" for blocking. Click the Add button to complete the opening instruction addition.

Step2: Click "collision detection off", click the Add button, and complete the closing instruction addition. The program preview interface is shown in Figure 6, click the "Apply" button to complete the function addition.

Note: The custom collision detection threshold function is a set of instructions, which needs to be closed in time after opening.

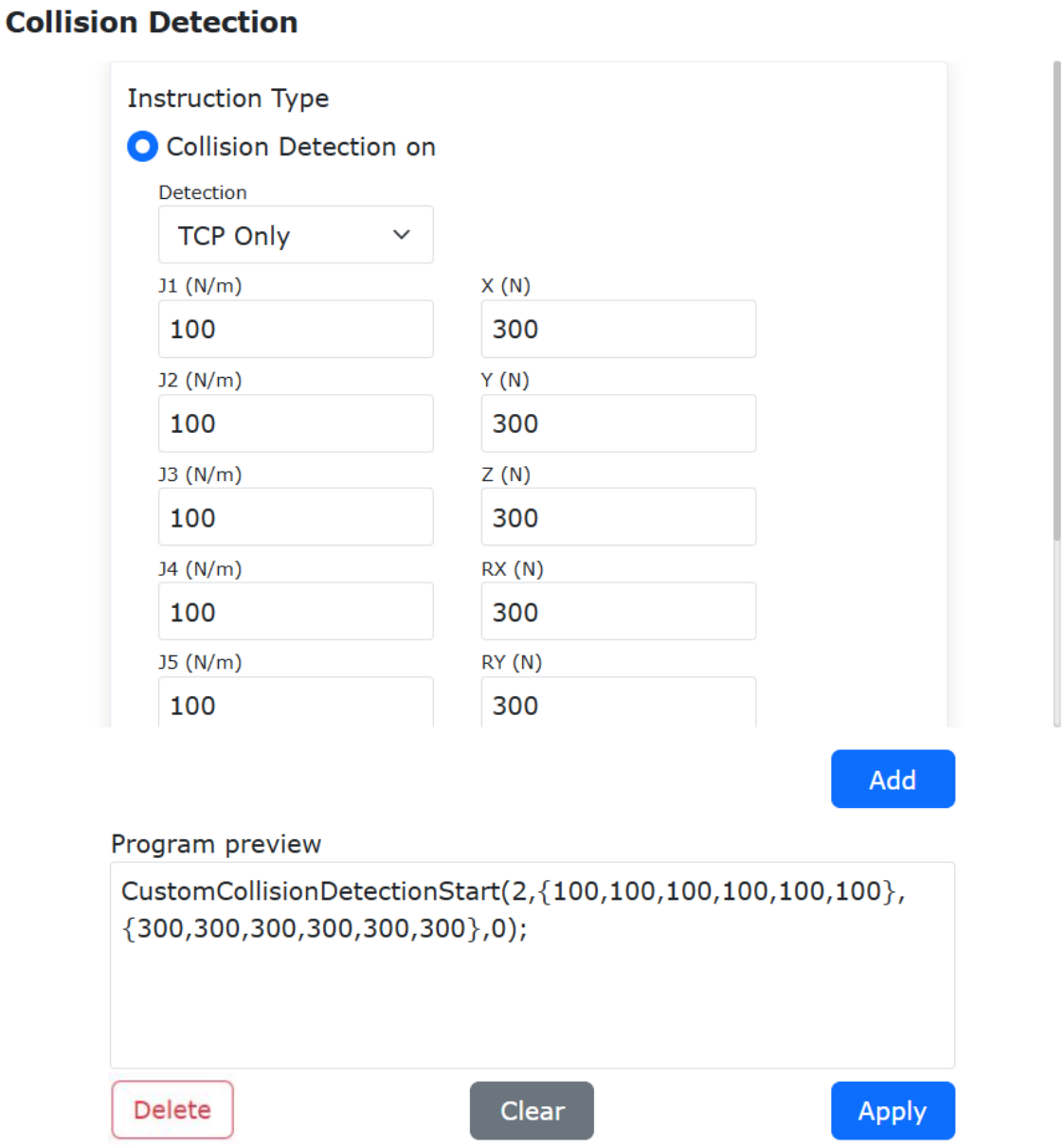


Figure 9.26-5 TCP detection threshold setting interface

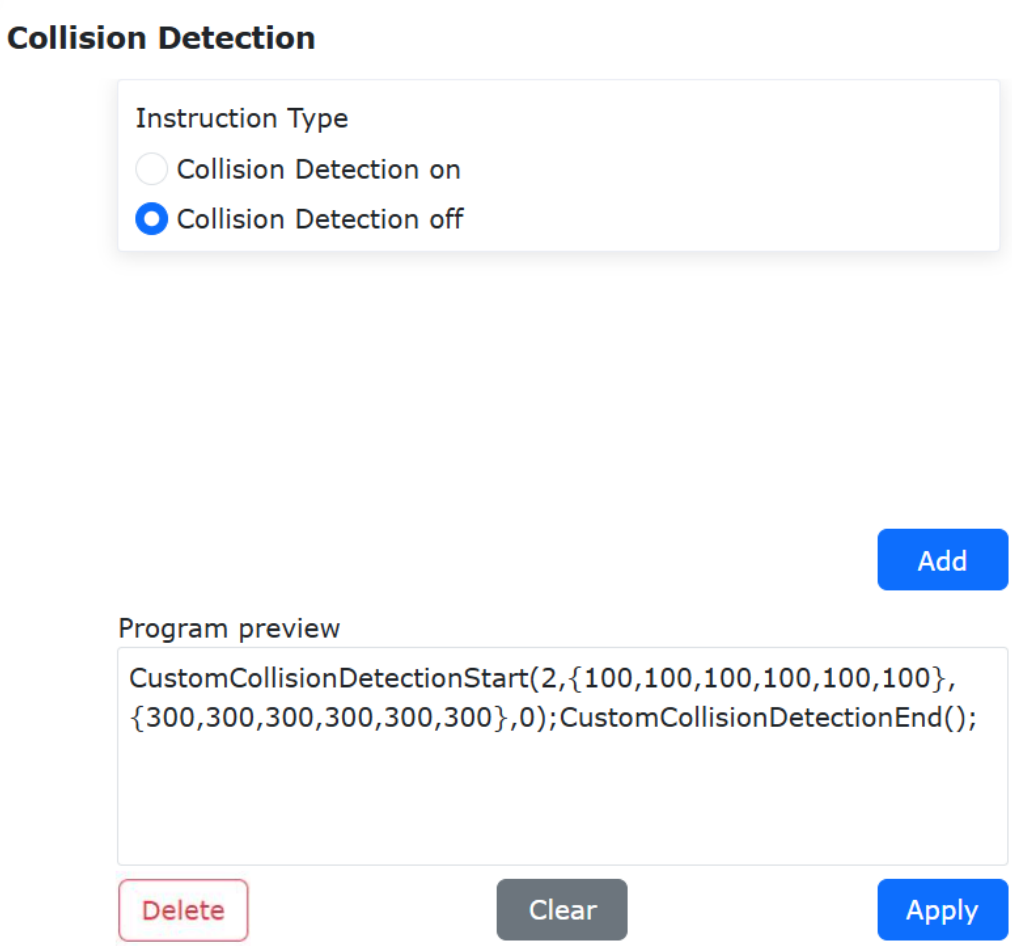


Figure 9.26-6 Program preview interface

Step 3: Add corresponding motion instructions in the collision detection function, as shown in Figure 7.

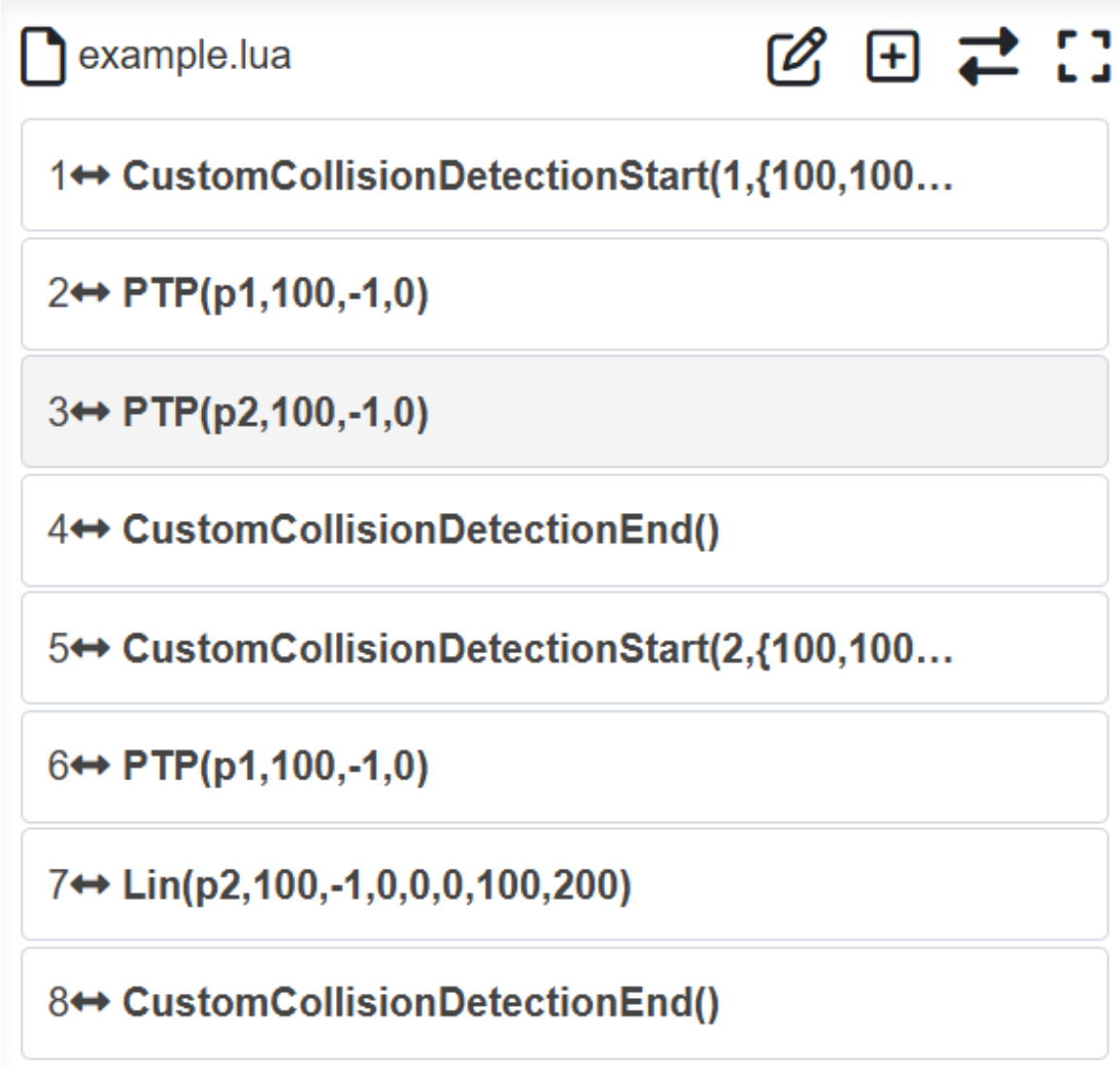


Figure 9.26-7 lua script program example interface

1.3.9.26.2.5 Joint and TCP detection threshold function setting instructions

1.3.9.26.2.6 Parameter setting instructions

Step 1: In the control instruction interface, select the "collision detection" function, as shown in Figure 8. Click "collision detection on", and select "joint and TCP" under the detection status bar. According to actual needs, modify the input values of J1-J6 and X-RZ directions. The value range of J1-J6 is, the unit is NM; the input value of X-RZ direction is, the unit is N. According to actual needs, select "non-blocking" or "blocking" for blocking. Click the Add button to complete the instruction addition.

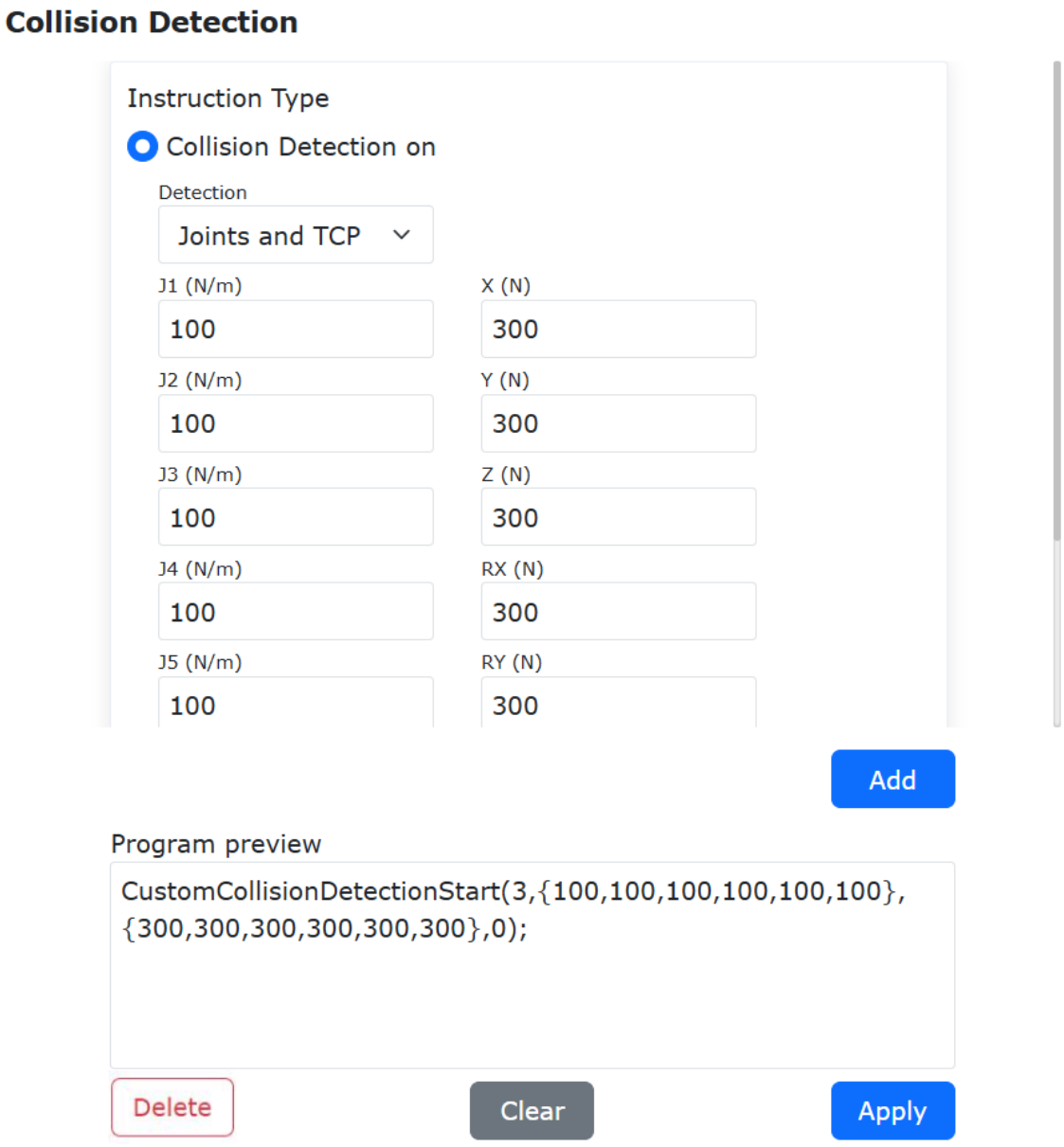


Figure 9.26-8 Joint and TCP detection threshold setting interface

Step2: Click "Close Collision Detection", click the Add button, and the closing command is added. The program preview interface is shown in Figure 9. Click the "Apply" button to complete the function addition.

Note: The custom collision detection threshold function is a set of instructions, which needs to be closed in time after opening.

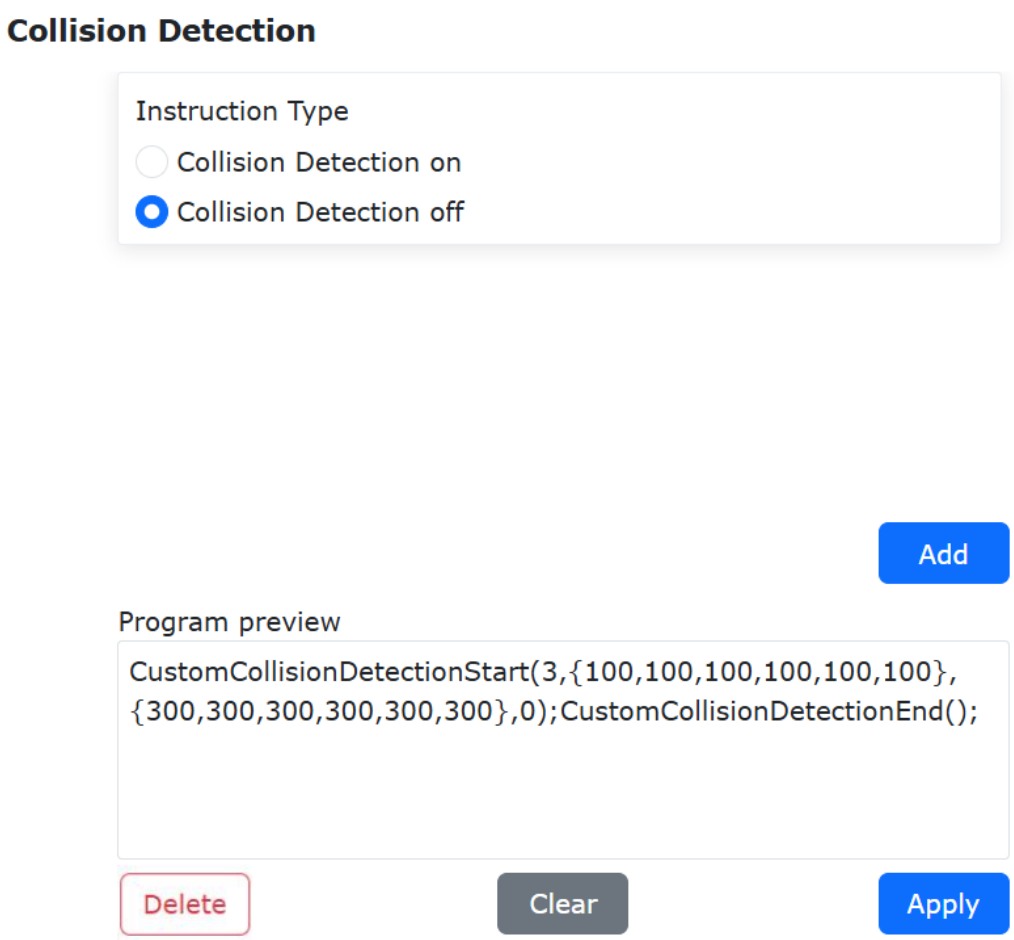


Figure 9.26-9 Program preview interface

Step 3: Add corresponding motion instructions in the collision detection function, as shown in Figure 10.



Figure 9.26-10 lua script program example interface

1.3.9.26.2.7 Recommended settings for detection thresholds

1.3.9.26.2.8 Joint detection thresholds

The recommended joint detection threshold is equivalent to setting the collision level to level 10. The larger the value, the less sensitive the collision detection. The value range is in NM. The data in the table is for reference only. The actual value needs to be adjusted according to the robot's running speed and load conditions.

Table 9.26-1 Recommended joint thresholds

Robot Type	J1	J2	J3	J4	J5	J6
FR3	0.4	0.7	0.6	0.3	0.3	0.3
FR3-WMS	0.4	0.7	0.6	0.3	0.3	0.3
FR3-WML	0.4	0.7	0.6	0.3	0.3	0.3
FR3-C	0.4	0.7	0.6	0.3	0.3	0.3
FR5	0.6	1	0.8	0.3	0.3	0.3
FR10	2.5	3.6	0.8	0.6	0.6	0.6
FR16	2.5	3.6	0.8	0.6	0.6	0.6
FR20	5	8	4.5	0.9	0.9	0.9
FR30	5	8	4.5	0.9	0.9	0.9

1.3.9.26.2.9 TCP detection threshold

The larger the TCP detection threshold, the less sensitive the collision detection. The value range is, and the unit is N. The data in the table is for reference only. The actual value needs to be adjusted according to the robot's running speed and load conditions.

Table 9.26-2 TCP detection threshold

Robot Type	X	Y	Z	RX	RY	RZ
FR3	300	300	300	20	20	20
FR3-WMS	300	300	300	20	20	20
FR3-WML	300	300	300	20	20	20
FR3-C	300	300	300	20	20	20
FR5	300	300	300	20	20	20
FR10	500	500	500	35	35	35
FR16	500	500	500	35	35	35
FR20	800	800	800	60	60	60
FR30	800	800	800	60	60	60

1.3.9.27 T-Shape Velocity Characteristic Optimization + Blending Smoothing Function

1.3.9.27.1 Overview

Blending between two trajectory segments avoids frequent start-stop issues caused by complete stops, thereby improving robot motion efficiency. This function primarily performs blending between PTP, LIN, ARC, and CIRCLE instructions, achievable in two ways: using Lua instruction mode or motion configuration switch mode.

1.3.9.27.2 Operation Procedures

1.3.9.27.2.1 PTP-PTP Blending

1.3.9.27.2.2 Using Lua Instruction Mode

Step 1: Select teaching points for PTP-PTP function execution. This manual uses "A0" to "A5" as teaching point names.

Step 2: Click "Teach Program" → "Program Programming" button, select "Point-to-Point" in "Motion Instructions", choose teaching points in "Instruction Editing", set debug speed, select "Acceleration Smooth Mode" for motion protection, and configure "Smooth Transition" parameter at points requiring smoothing.

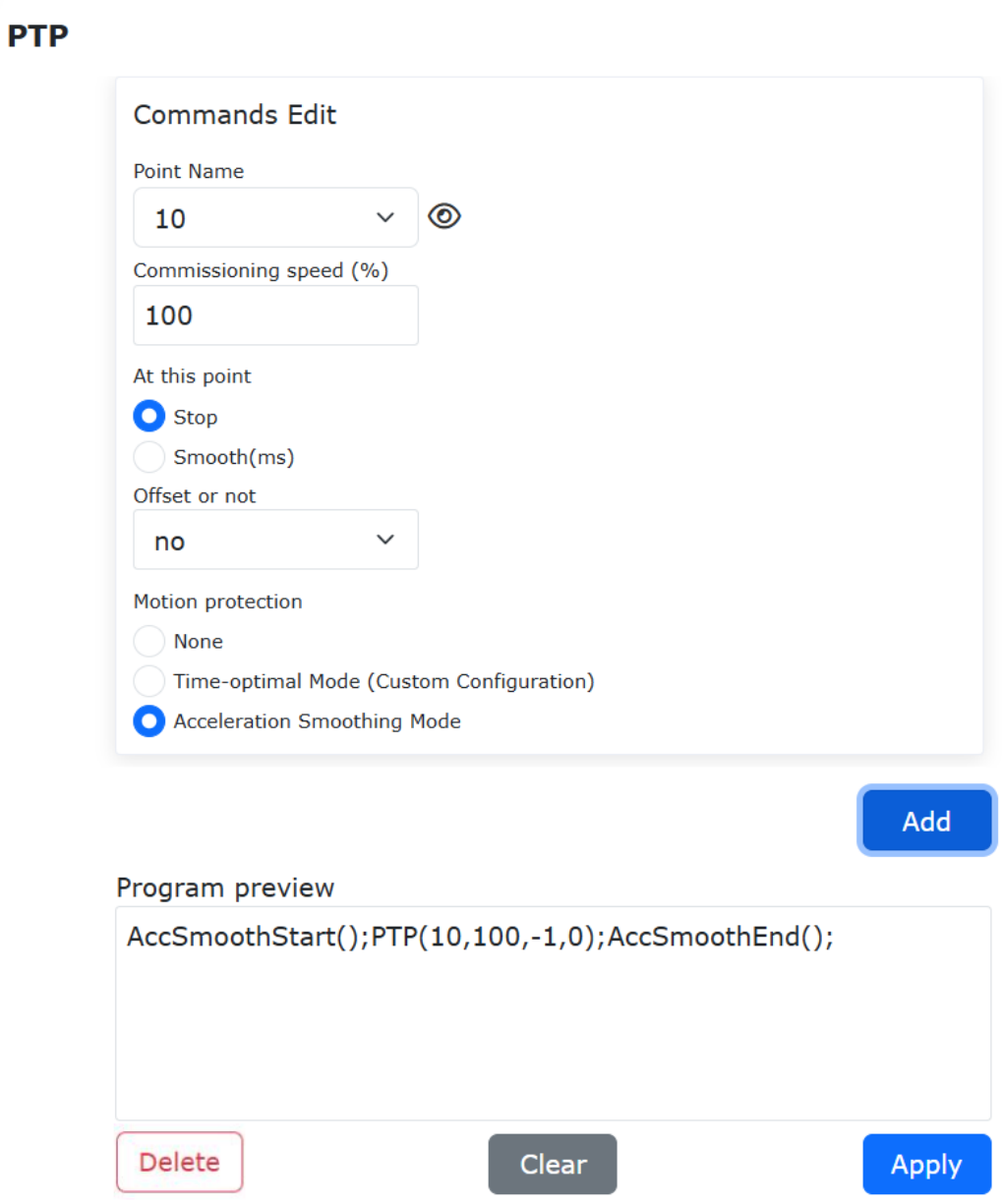


Figure 9.27-1 Blending Instruction Settings for Accelerated Smooth PTP Instructions

Step 3: Add multiple PTP instructions, generate and run a Lua program to implement PTP-PTP blending. This mode uses optimized T-shape velocity motion only for instructions between AccSmoothStart() and AccSmoothEnd(), with original T-shape velocity for others.

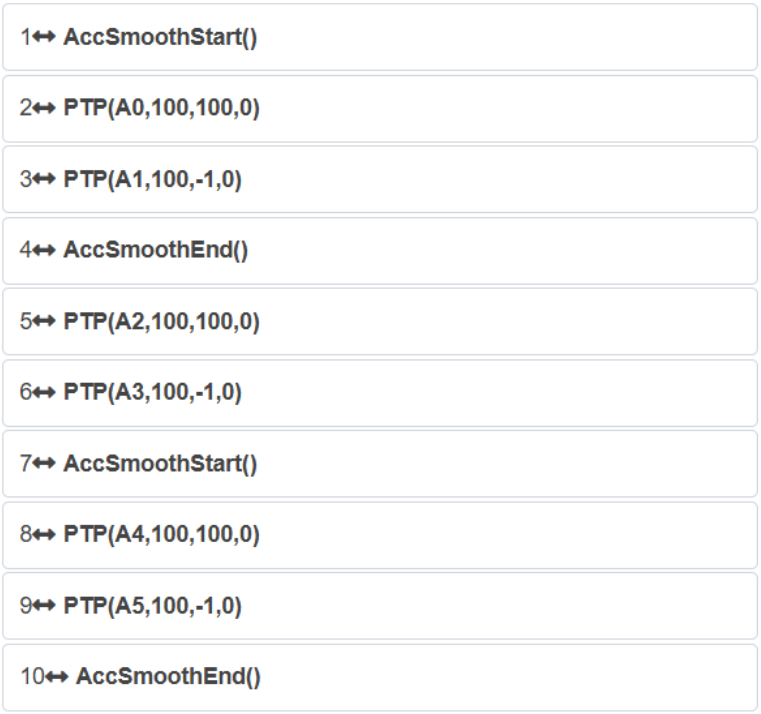


Figure 9.27-2 Typical PTP-PTP Blending Program in Lua Instruction Mode

1.3.9.27.2.3 Using Motion Configuration Switch Mode

Step 1: Click "Initial Settings" → "Safety" → "Motion Configuration" button to enable the "Acceleration Smooth Mode" switch.

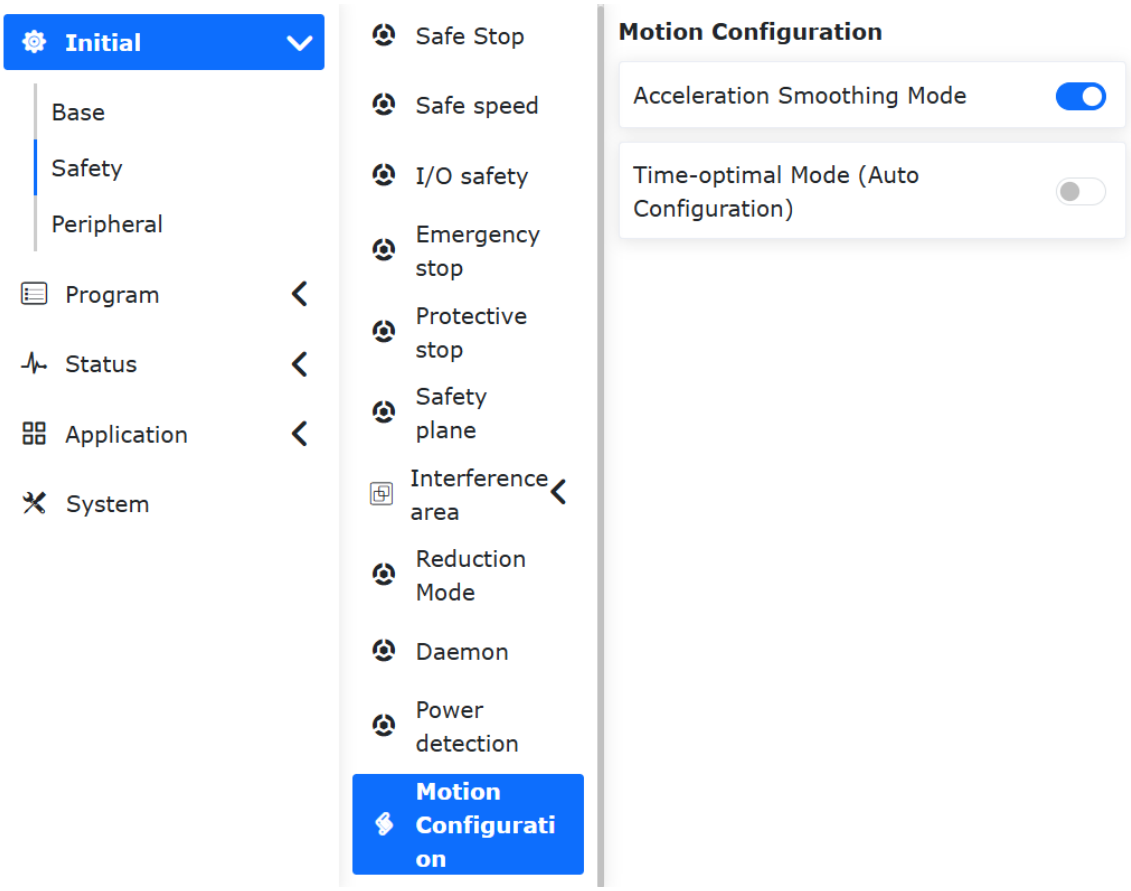


Figure 9.27-3 Acceleration Smooth Mode Configuration Switch Settings

Step 2: Select teaching points for PTP-PTP function execution, using “A0” to “A5” as names.

Step 3: Click “Teach Program” → “Program Programming” button, select “Point-to-Point” in “Motion Instructions”, choose teaching points, set debug speed, select “None” for motion protection, and configure “Smooth Transition” parameter at points requiring smoothing.

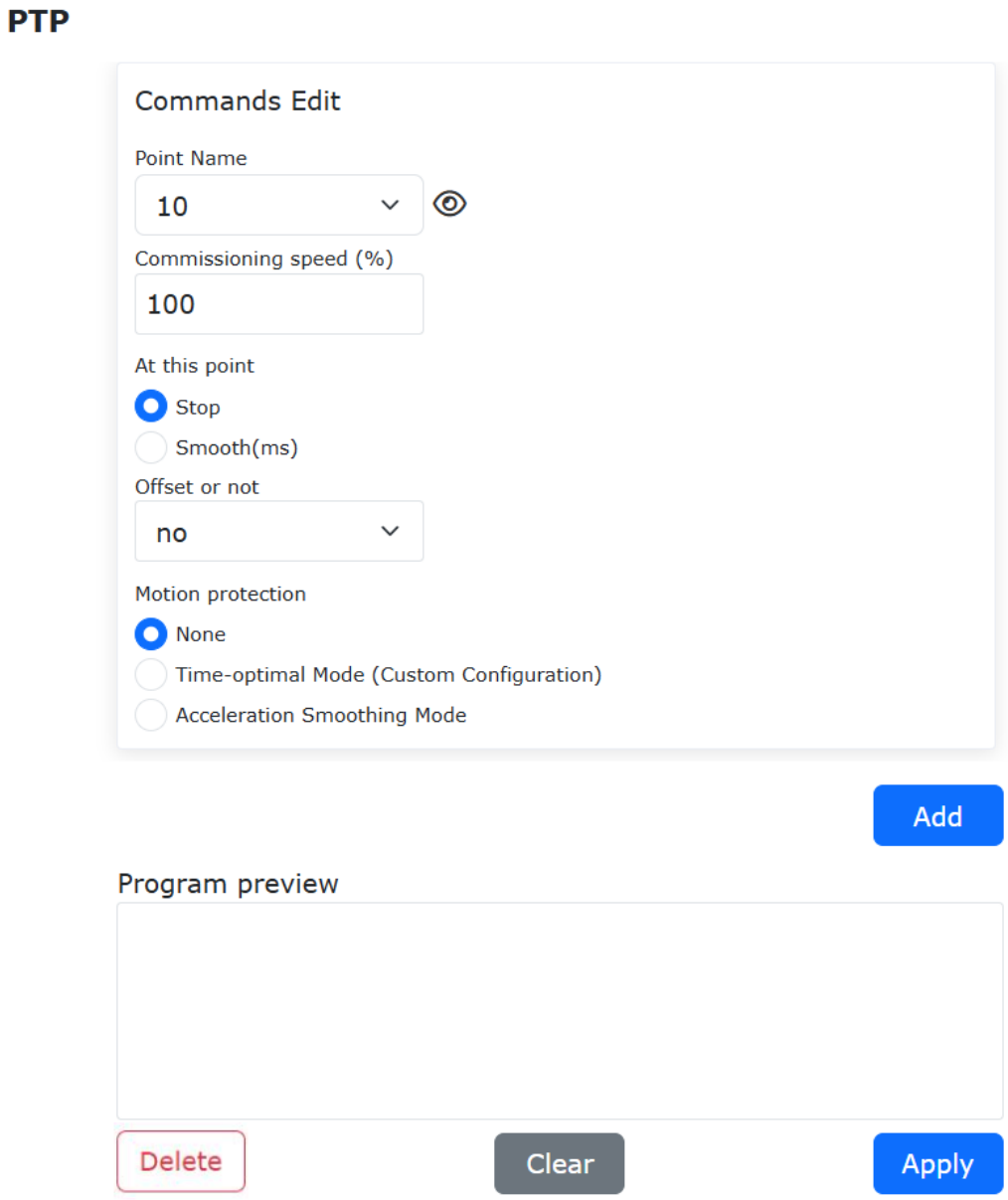


Figure 9.27-4 Blending Instruction Settings for Conventional PTP Instructions

Step 4: Add multiple PTP instructions, generate and run a Lua program to implement PTP-PTP blending. The typical program mirrors conventional PTP-PTP programs, applying optimized T-shape velocity motion to all instructions.



Figure 9.27-5 Typical PTP-PTP Blending Program Using Configuration Switch

1.3.9.27.2.4 PTP-LIN Blending

1.3.9.27.2.5 Using Lua Instruction Mode

Step 1: Select teaching points for PTP-LIN function execution, using "A0" to "A5" as names.

Step 2: Click "Teach Program" → "Program Programming" button, select "Point-to-Point" in "Motion Instructions", choose teaching points, set debug speed, select "Acceleration Smooth Mode", and configure "Smooth Transition" parameter.

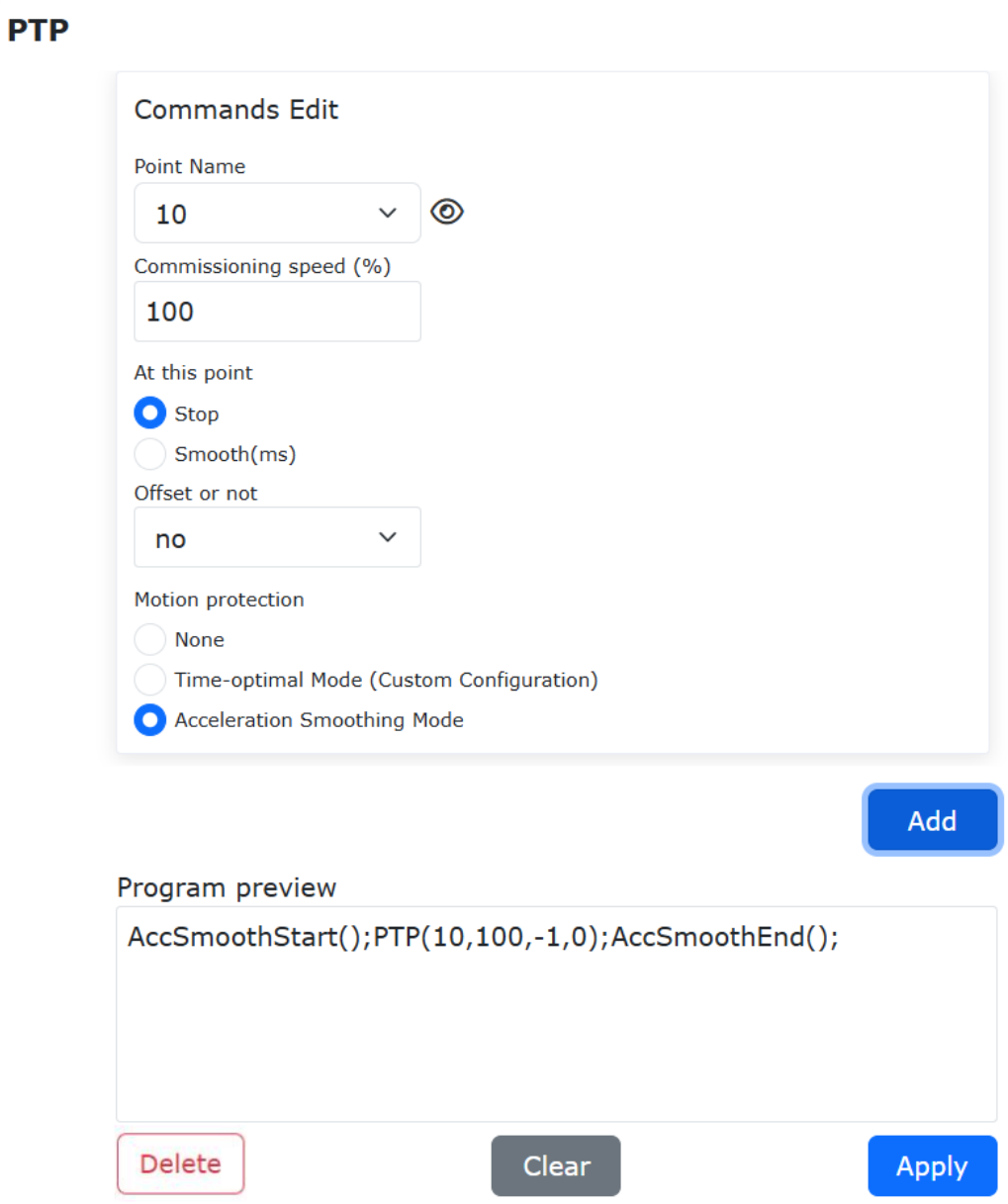


Figure 9.27-6 Blending Instruction Settings for Accelerated Smooth PTP Instructions

Step 3: Add multiple PTP and LIN instructions, generate Lua program to implement PTP-LIN blending. Only instructions between AccSmoothStart() and AccSmoothEnd() use optimized T-shape velocity.



Figure 9.27-7 Typical PTP-LIN Blending Program in Lua Mode

1.3.9.27.2.6 Using Motion Configuration Switch Mode

Step 1: Enable “Acceleration Smooth Mode” in “Initial Settings” → “Safety” → “Motion Configuration”.

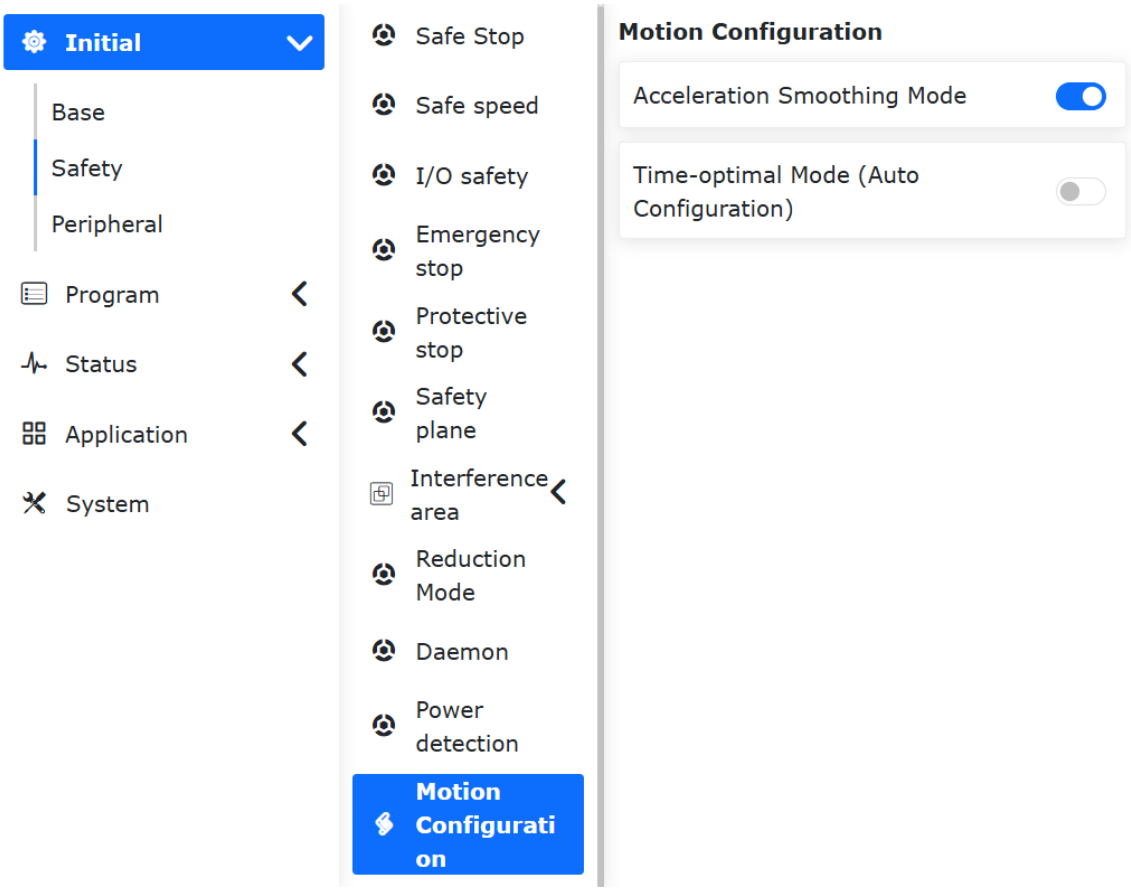


Figure 9.27-8 Acceleration Smooth Mode Switch Configuration

Step 2: Select teaching points for PTP-LIN, using “A0” to “A5”.

Step 3: Choose “Point-to-Point” instruction, set debug speed, select “None” for motion protection, configure “Smooth Transition”.

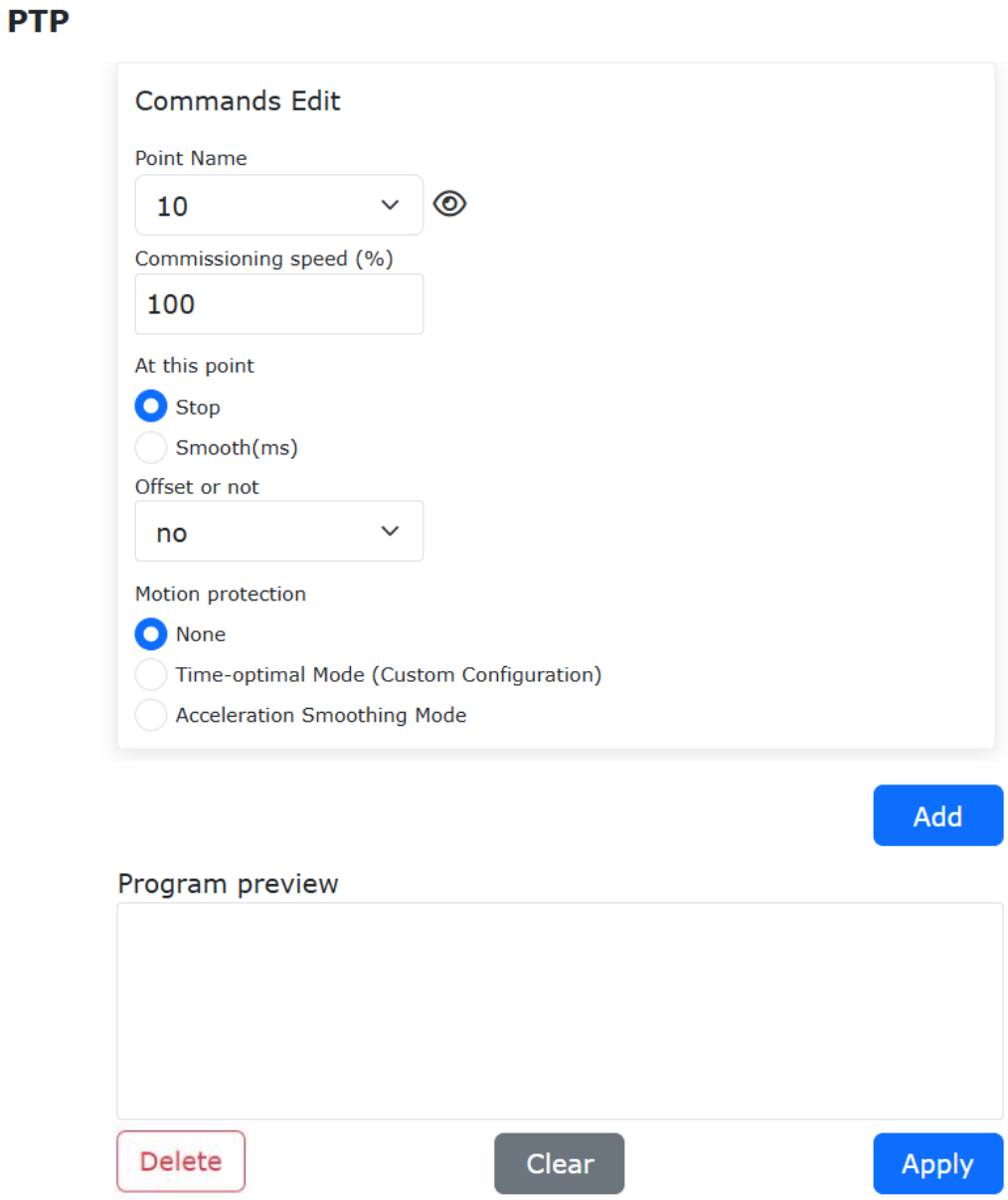


Figure 9.27-9 Blending Settings for Conventional PTP Instructions

Step 4: Add PTP and LIN instructions, run Lua program. All instructions use optimized T-shape velocity.



Figure 9.27-10 Typical PTP-LIN Blending Program via Configuration Switch

1.3.9.27.2.7 PTP-ARC Blending

1.3.9.27.2.8 Using Lua Instruction Mode

Step 1: Select teaching points “A0” to “A8” for PTP-ARC blending.

Step 2: Insert “Point-to-Point” instruction, set “Acceleration Smooth Mode” and “Smooth Transition”.

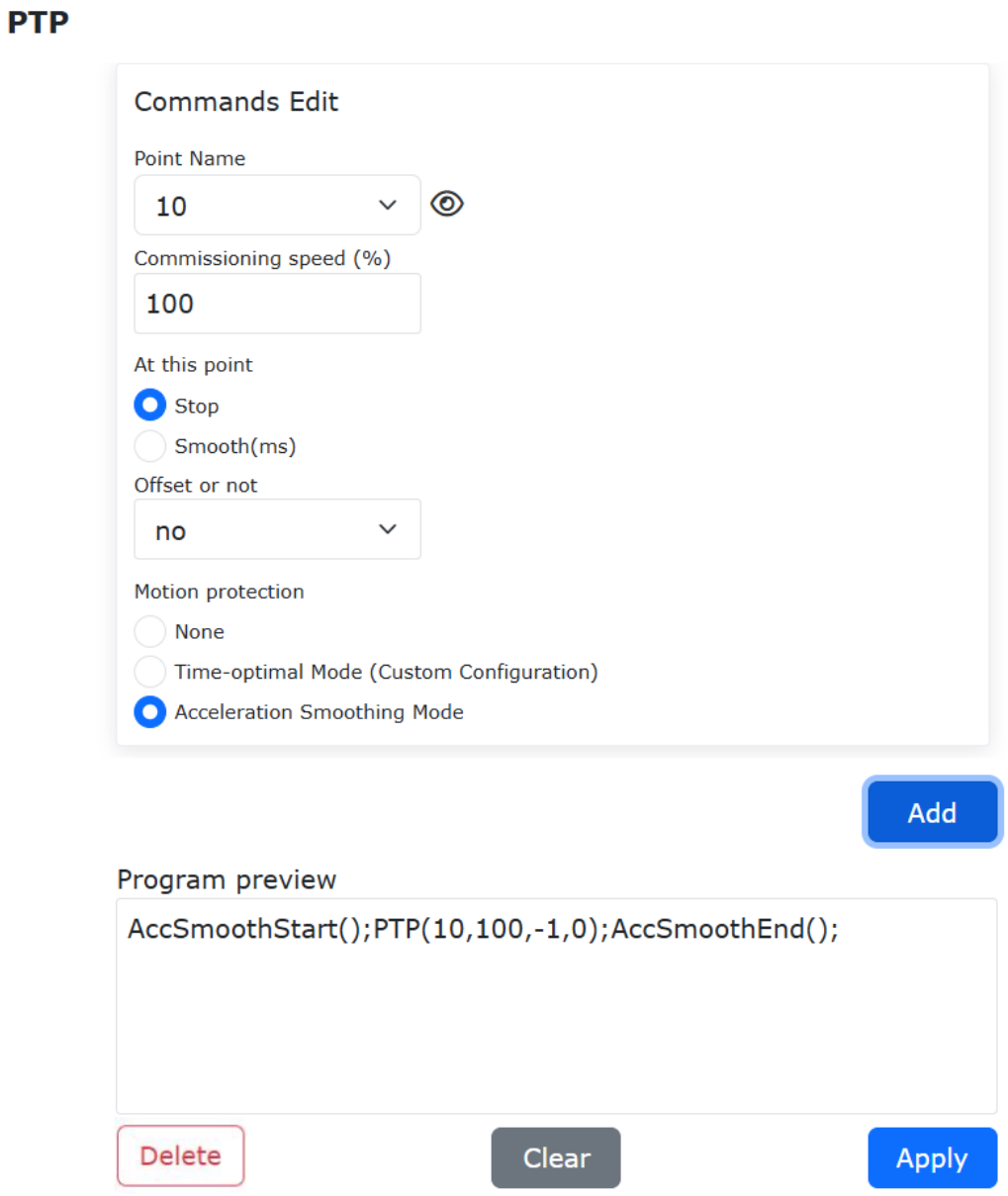


Figure 9.27-11 Blending Settings for Accelerated Smooth PTP

Step 3: Add PTP and ARC instructions. Only AccSmoothStart/End-enclosed instructions use optimized velocity.



Figure 9.27-12 Typical PTP-ARC Blending Program in Lua Mode

1.3.9.27.2.9 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode in motion configuration.

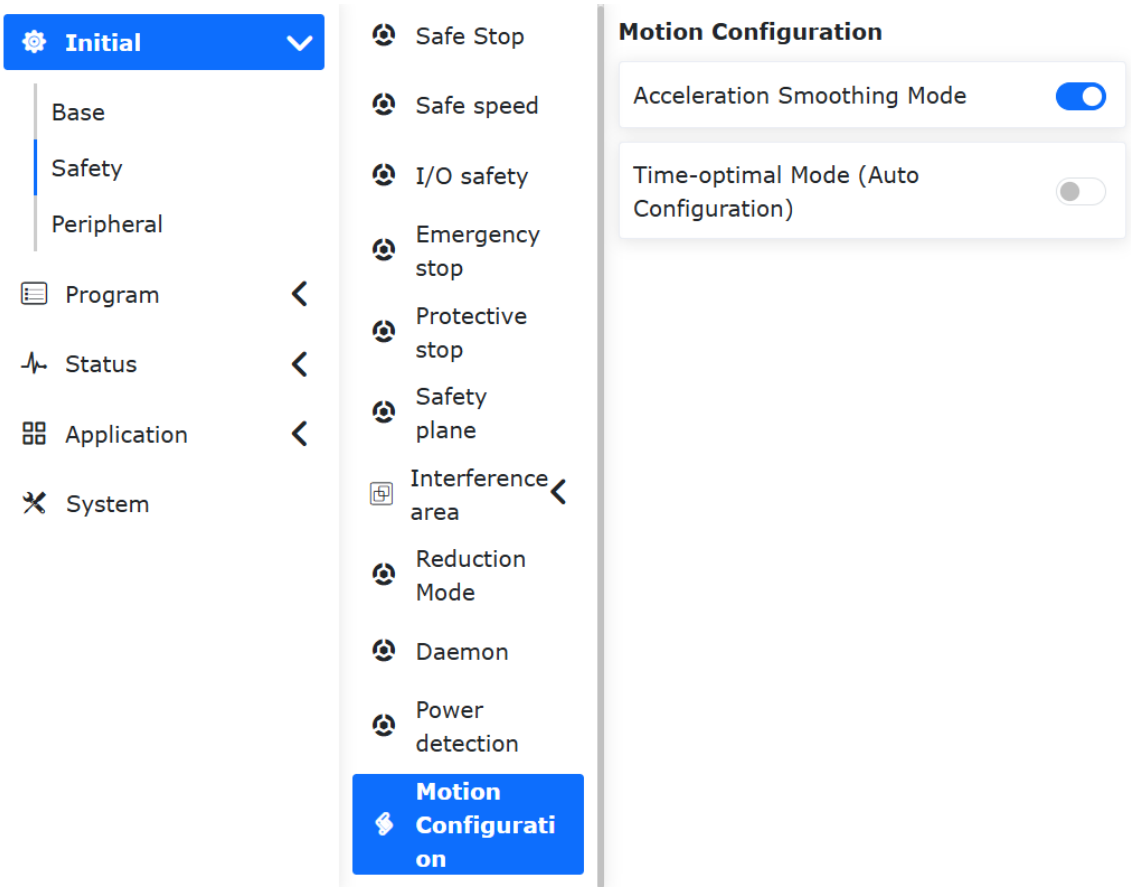


Figure 9.27-13 Acceleration Smooth Mode Switch Setup

Step 2: Select “A0” to “A8” as teaching points.

Step 3: Configure “Point-to-Point” instruction with “Smooth Transition”, motion protection “None”.

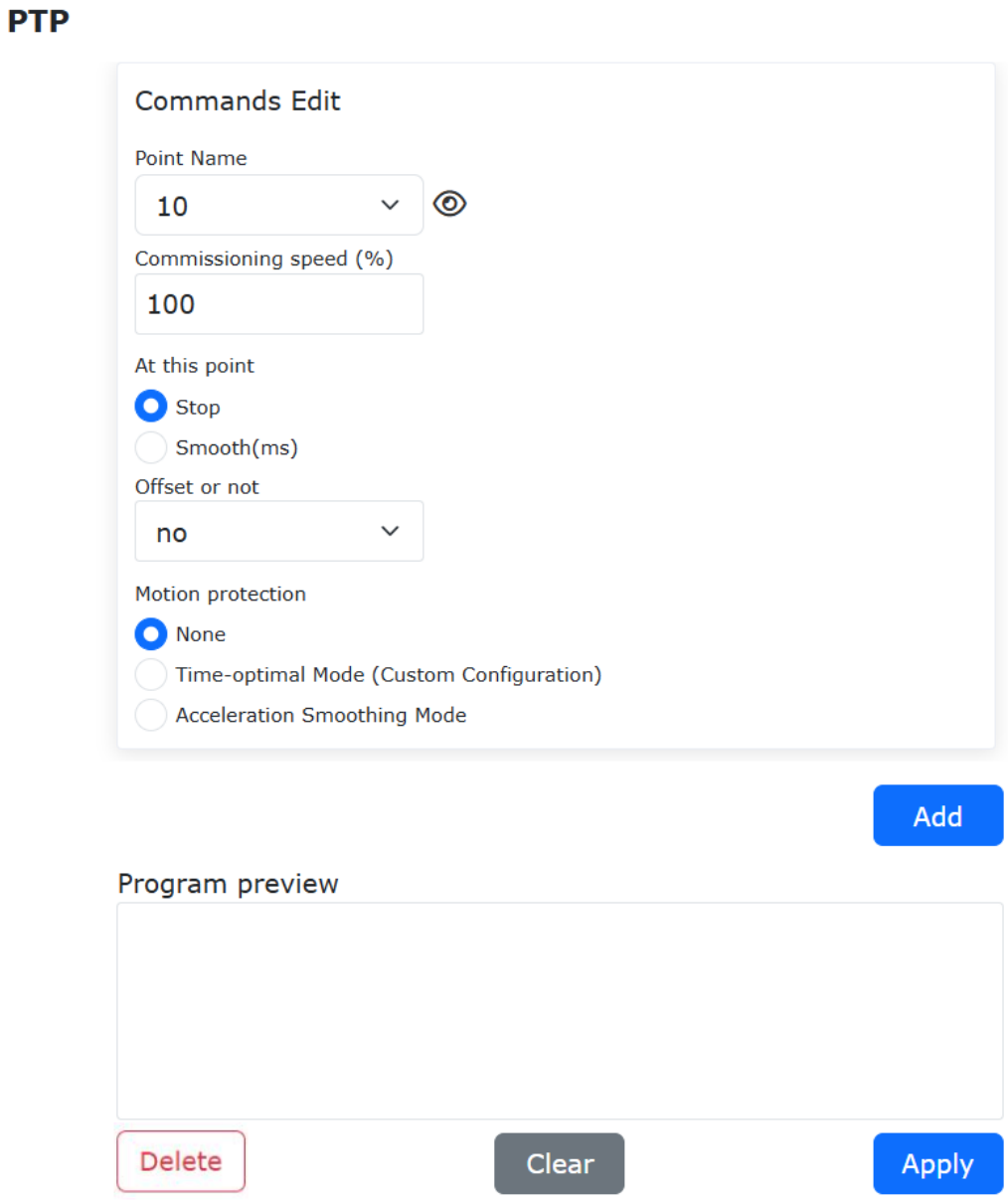


Figure 9.27-14 Blending Settings for Conventional PTP

Step 4: Generate program; all instructions use optimized T-shape velocity.

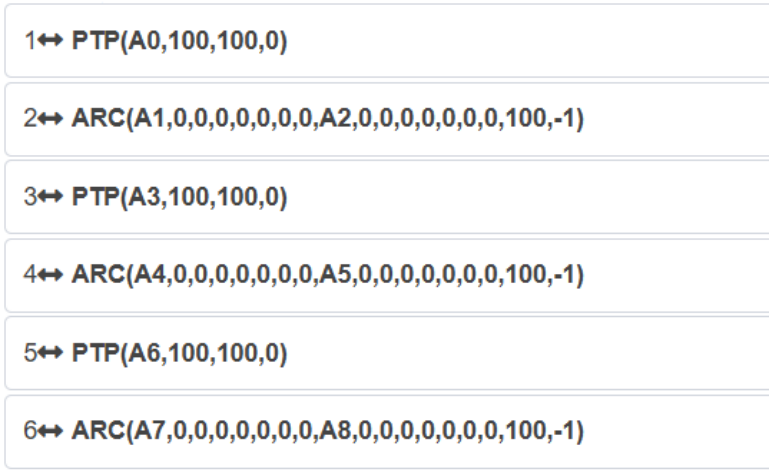


Figure 9.27-15 Typical PTP-ARC Blending via Configuration Switch

1.3.9.27.2.10 PTP-CIRCLE Blending

1.3.9.27.2.11 Using Lua Instruction Mode

Step 1: Select teaching points “A0” to “A8” for PTP-CIRCLE.

Step 2: Set up “Point-to-Point” instruction with acceleration smooth mode.

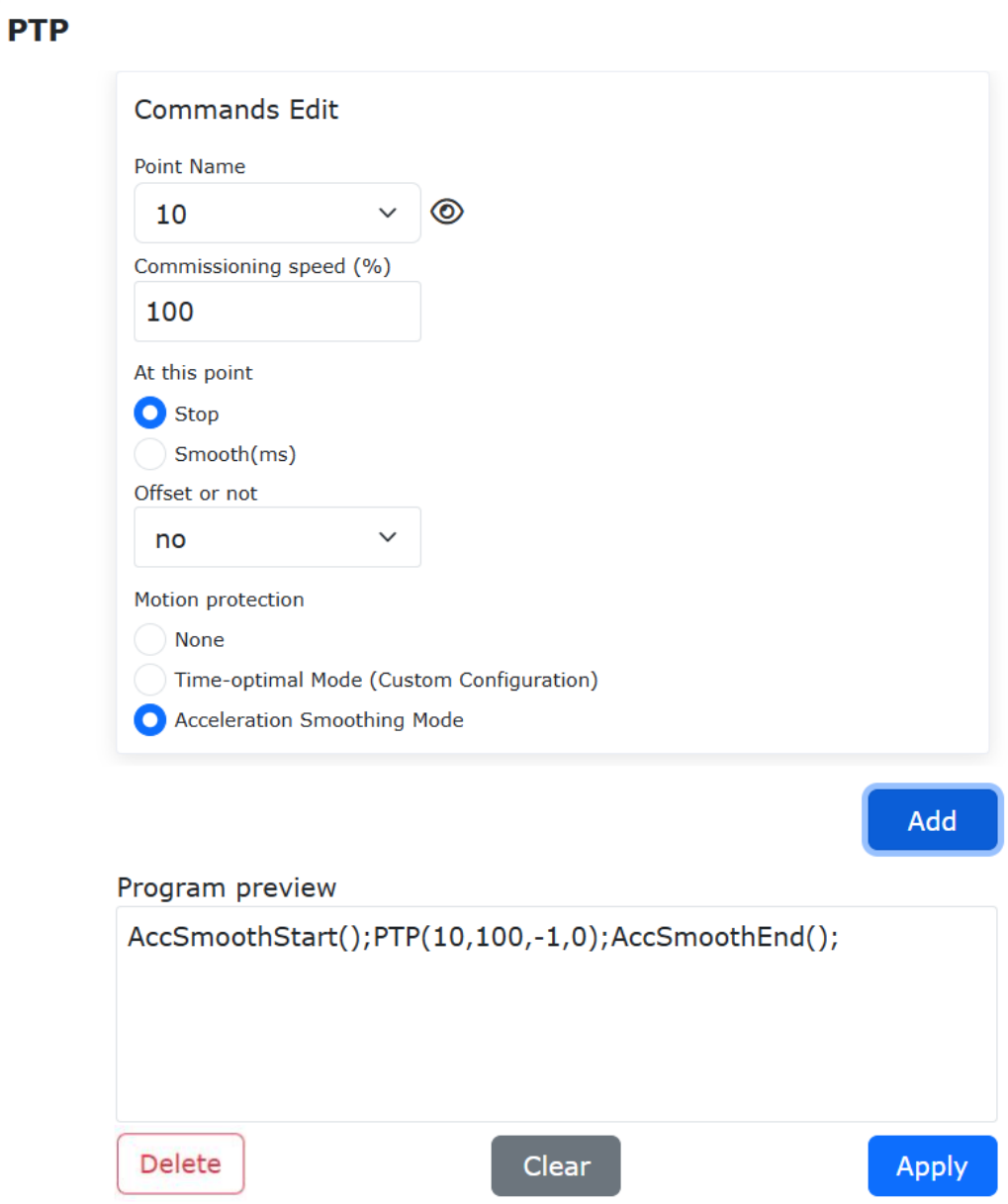


Figure 9.27-16 Blending Settings for Accelerated Smooth PTP

Step 3: Add PTP and CIRCLE instructions, run between AccSmoothStart/End.

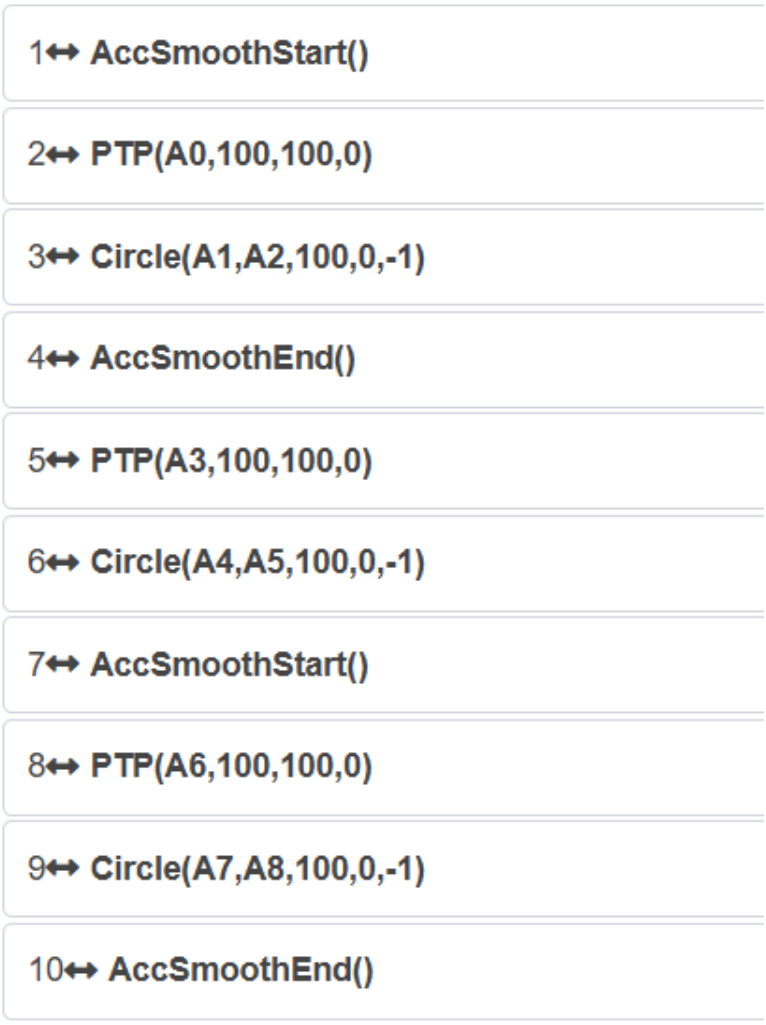


Figure 9.27-17 Typical PTP-CIRCLE Blending in Lua Mode

1.3.9.27.2.12 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode switch.

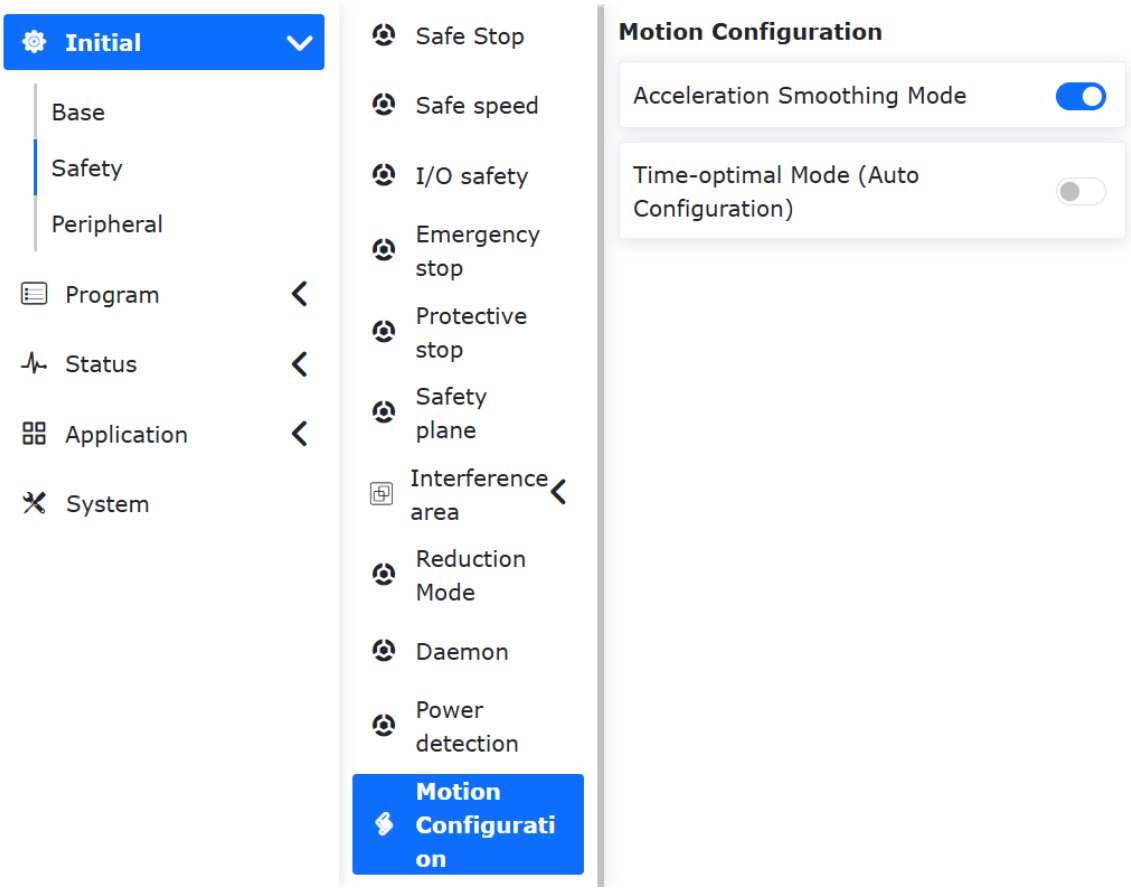


Figure 9.27-18 Acceleration Smooth Mode Configuration

Step 2: Choose “A0” to “A8” as teaching points.

Step 3: Configure PTP instruction with “Smooth Transition”, no motion protection.

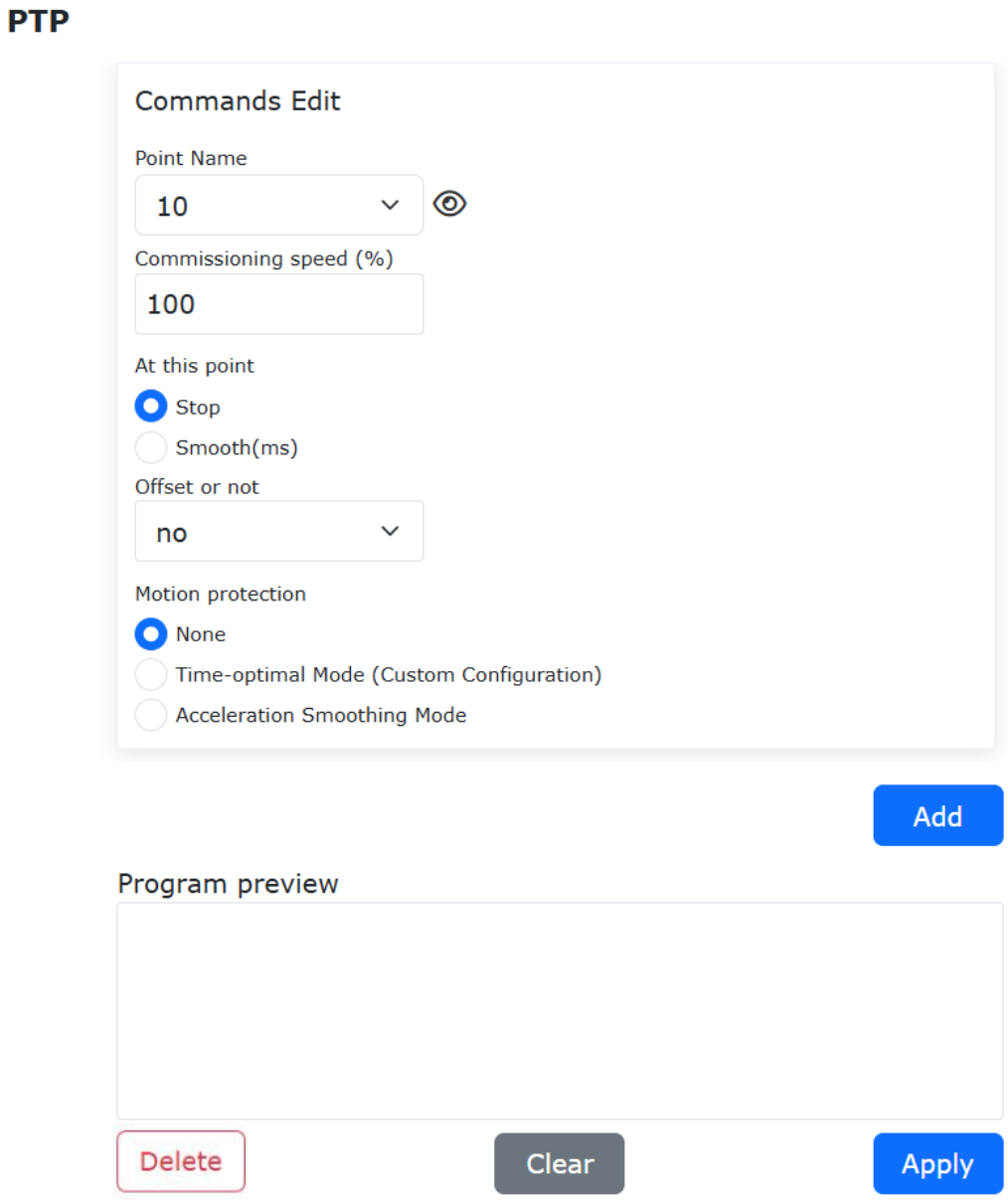


Figure 9.27-19 Blending Settings for Conventional PTP

Step 4: Generate program for all-instruction optimized velocity.

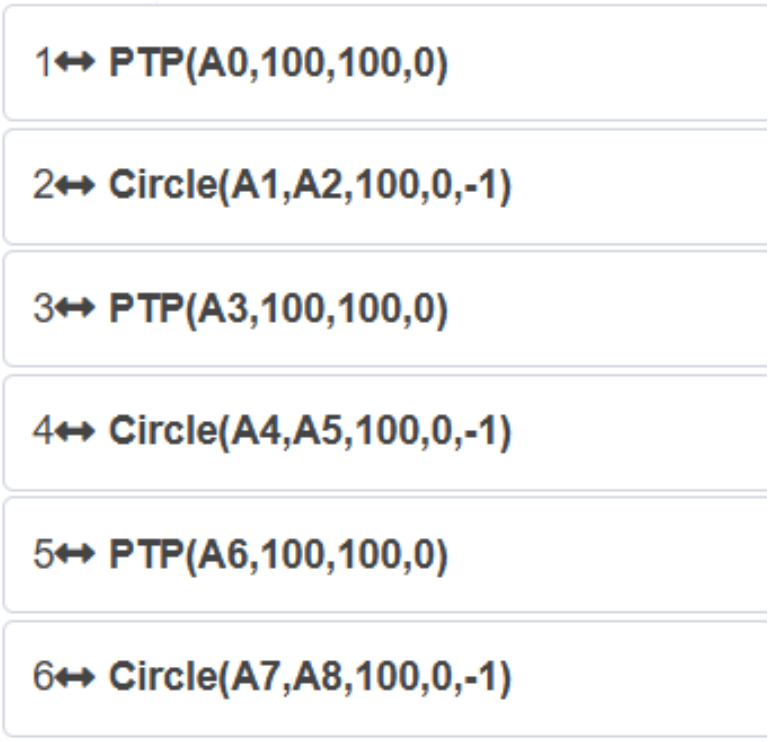


Figure 9.27-20 PTP-CIRCLE Blending via Configuration Switch

1.3.9.27.2.13 LIN-PTP Blending

1.3.9.27.2.14 Using Lua Instruction Mode

Step 1: Select “A0” to “A5” for LIN-PTP blending.

Step 2: Insert “Linear” instruction, set “Acceleration Smooth Mode”, “Transition Radius” and “Transition Mode”.

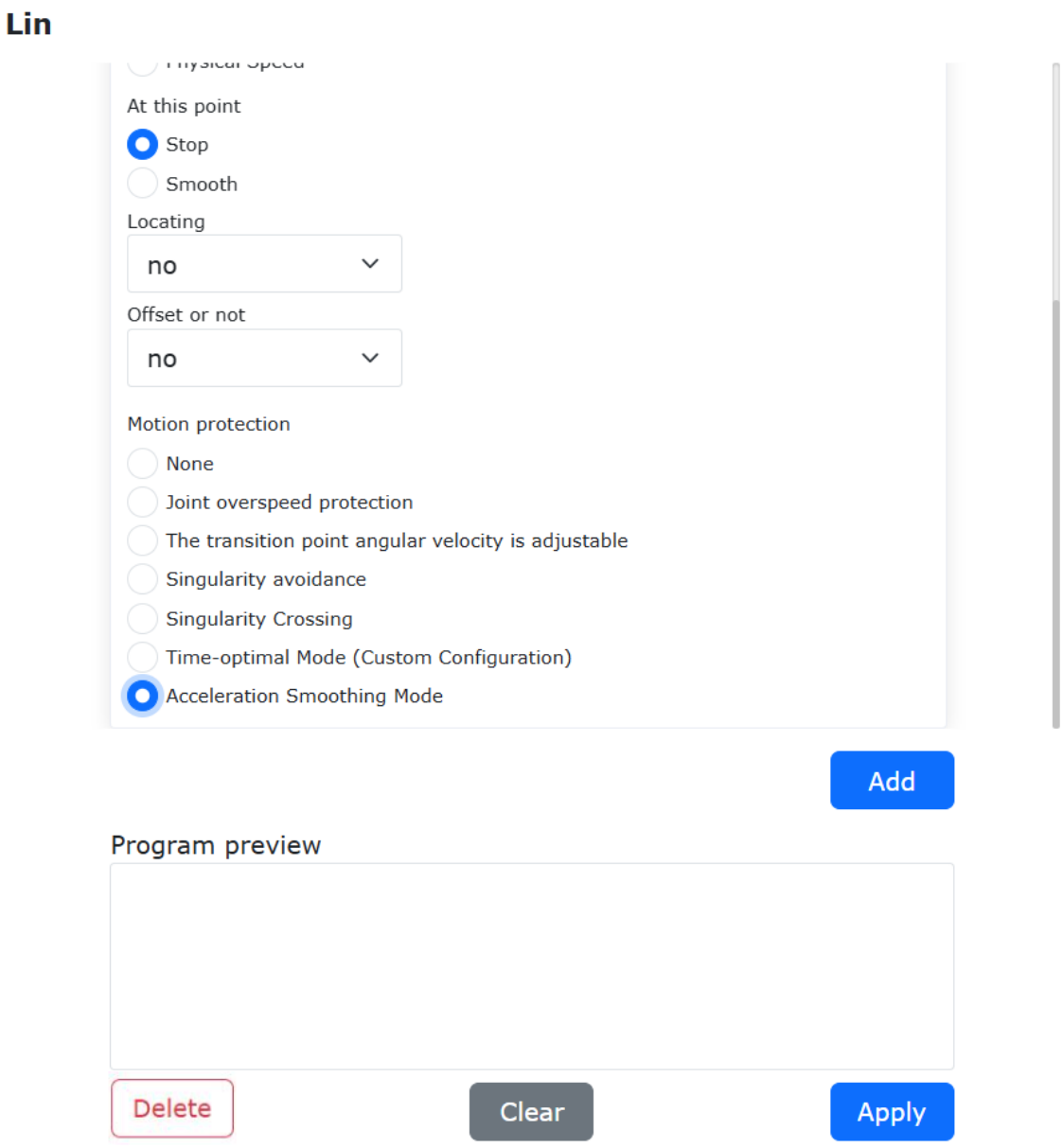


Figure 9.27-21 Blending Settings for Accelerated Smooth LIN

Step 3: Add LIN and PTP instructions, run within AccSmoothStart/End.



Figure 9.27-22 Typical LIN-PTP Blending in Lua Mode

1.3.9.27.2.15 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode in settings.

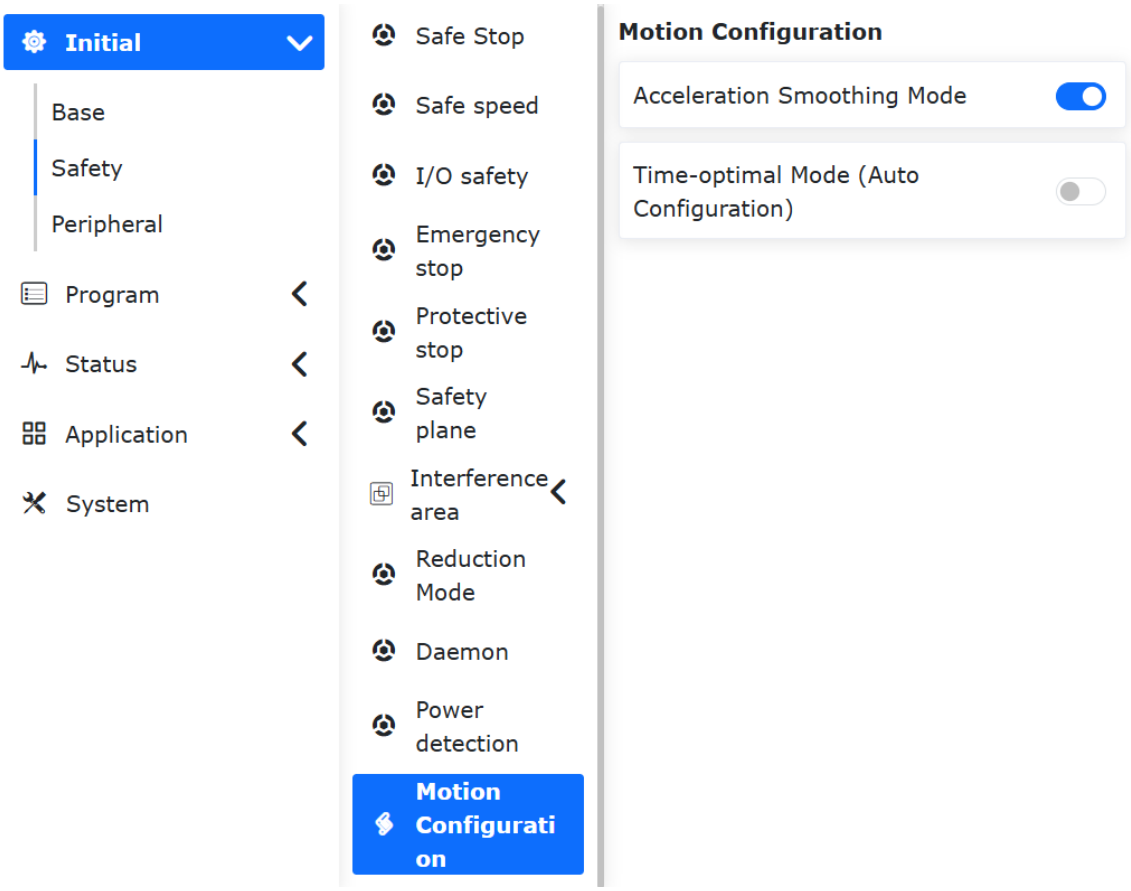


Figure 9.27-23 Acceleration Smooth Mode Switch Setup

Step 2: Select “A0” to “A5” as teaching points.

Step 3: Configure LIN instruction with transition parameters, no motion protection.

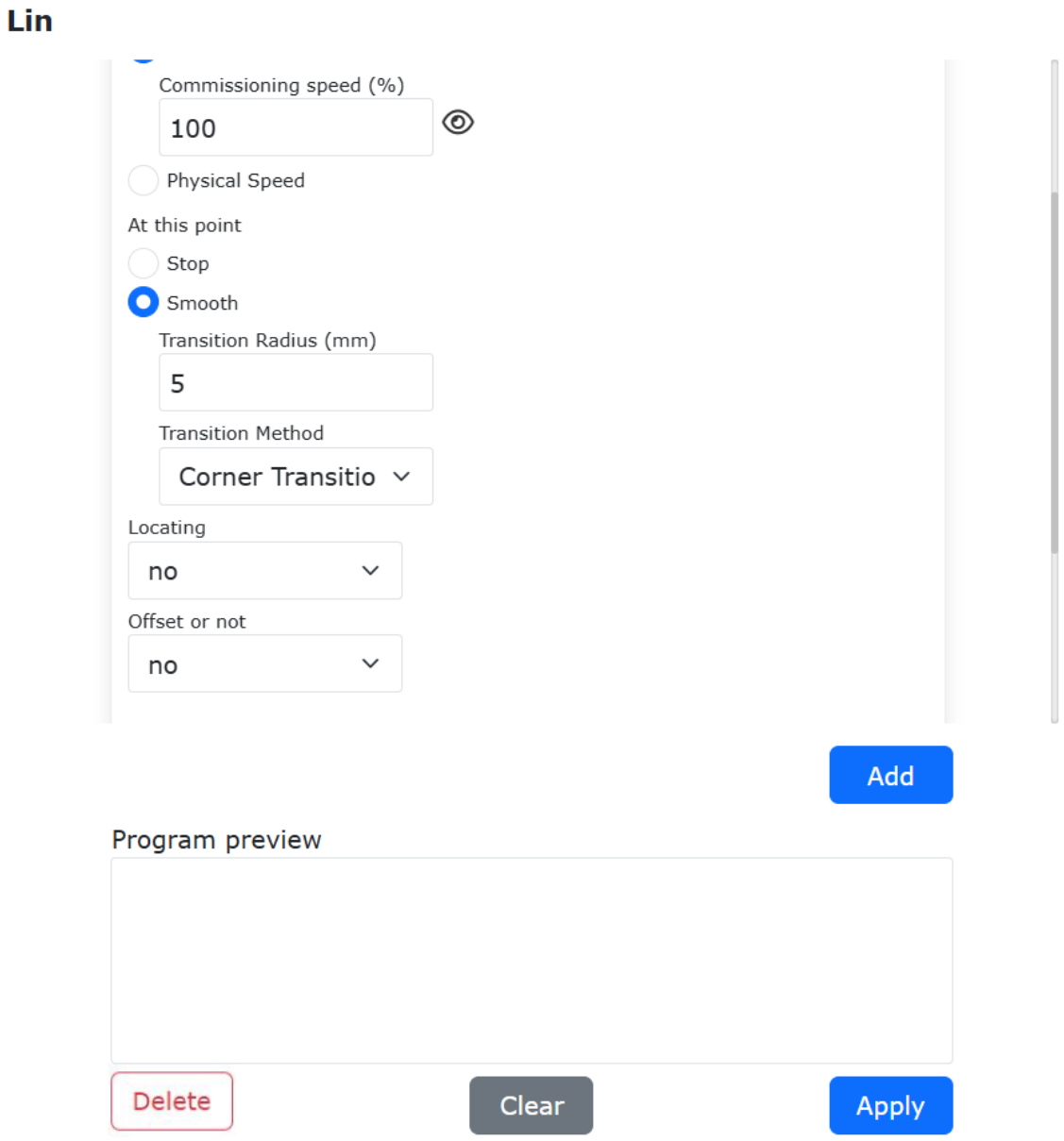


Figure 9.27-24 Blending Settings for Conventional LIN

Step 4: Generate program for all-instruction optimized velocity.



Figure 9.27-25 LIN-PTP Blending via Configuration Switch

1.3.9.27.2.16 LIN-LIN Blending

1.3.9.27.2.17 Using Lua Instruction Mode

Step 1: Select “A0” to “A5” for LIN-LIN blending.

Step 2: Set up LIN instruction with acceleration smooth mode and transition parameters.

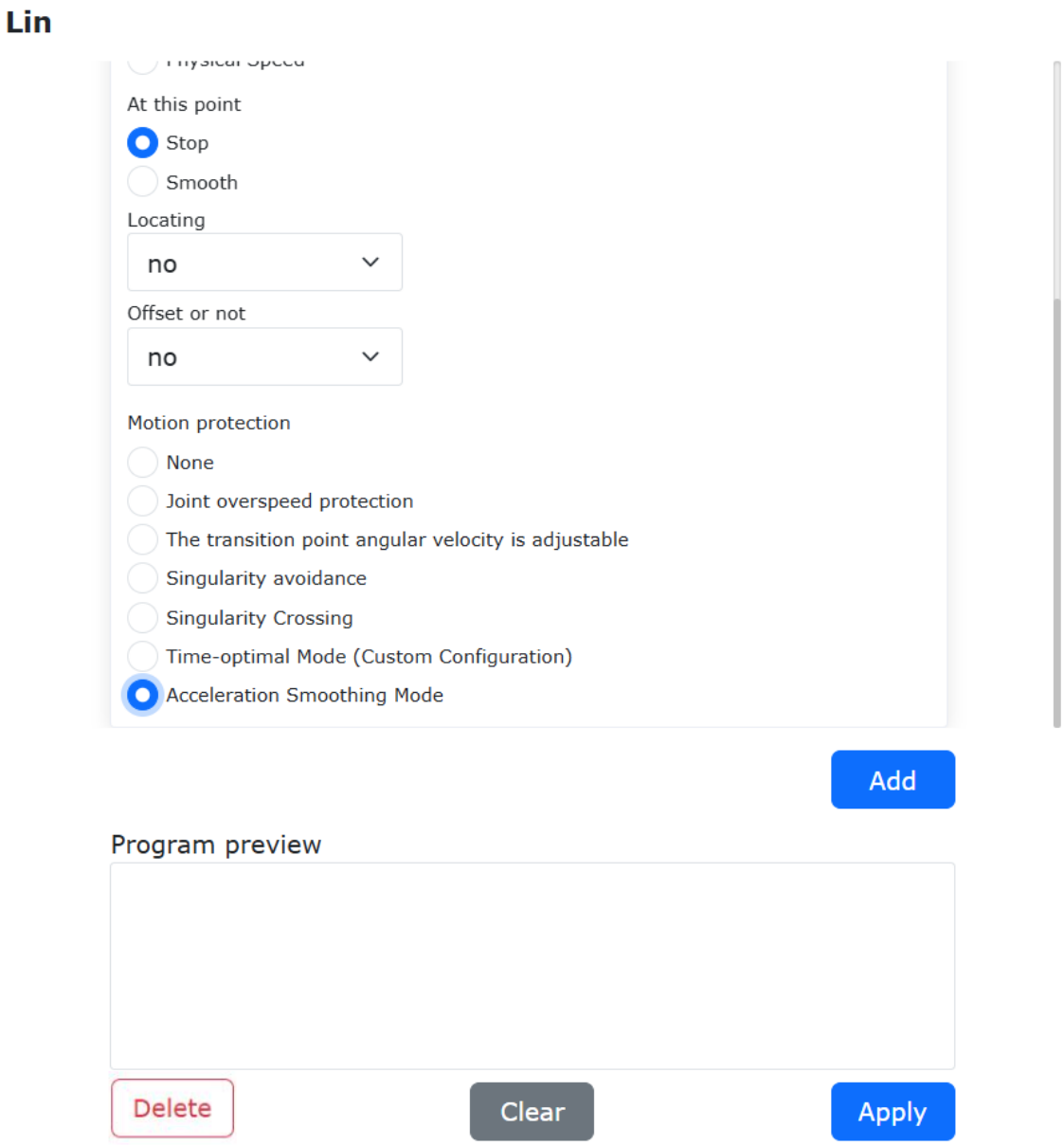


Figure 9.27-26 Blending Settings for Accelerated Smooth LIN

Step 3: Add multiple LIN instructions, run between AccSmoothStart/End.



Figure 9.27-27 Typical LIN-LIN Blending in Lua Mode

1.3.9.27.2.18 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode switch.

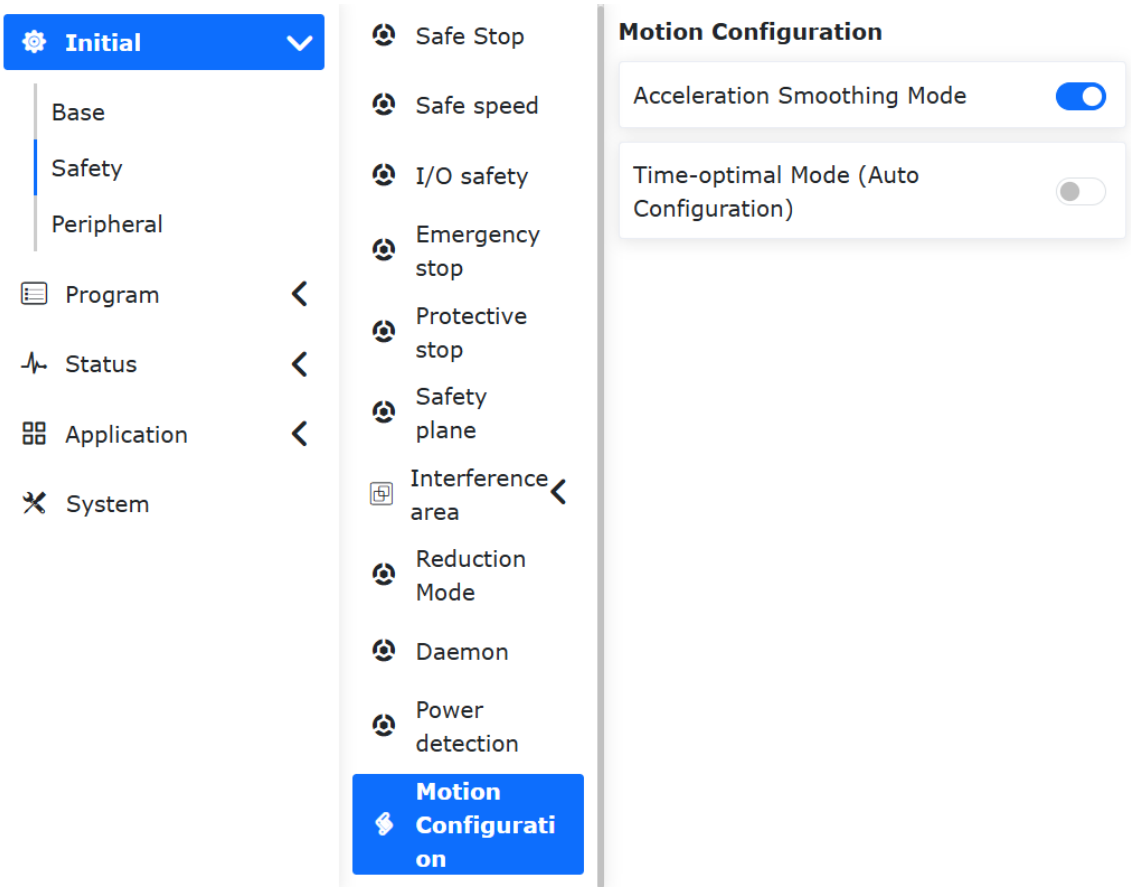


Figure 9.27-28 Acceleration Smooth Mode Configuration

Step 2: Choose “A0” to “A5” as teaching points.

Step 3: Configure LIN instruction with transition radius and mode.

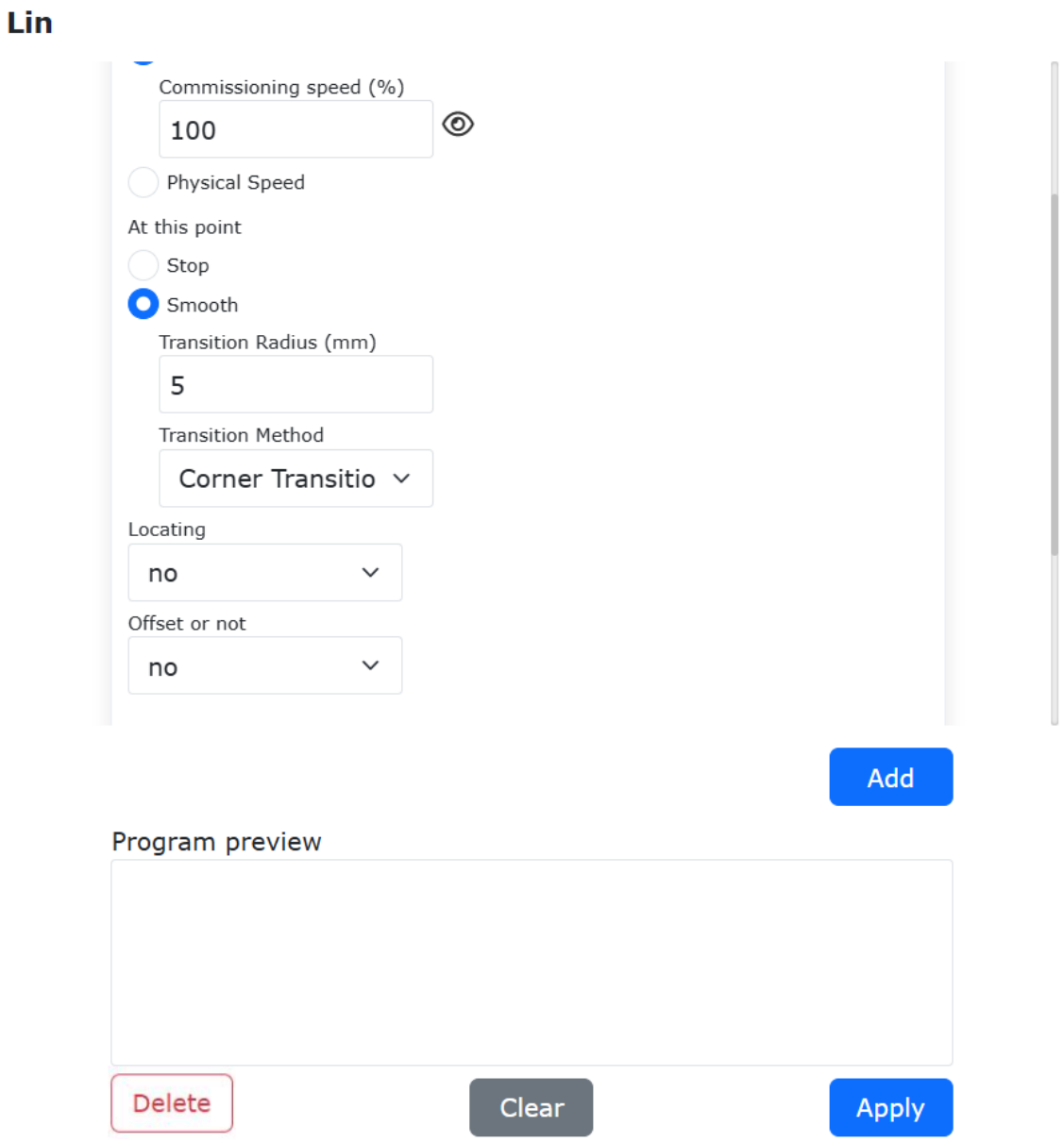


Figure 9.27-29 Blending Settings for Conventional LIN

Step 4: Generate program for optimized velocity across all instructions.



Figure 9.27-30 LIN-LIN Blending via Configuration Switch

1.3.9.27.2.19 LIN-ARC Blending

1.3.9.27.2.20 Using Lua Instruction Mode

Step 1: Select “A0” to “A8” for LIN-ARC blending.

Step 2: Configure LIN instruction with acceleration smooth mode and transition parameters.

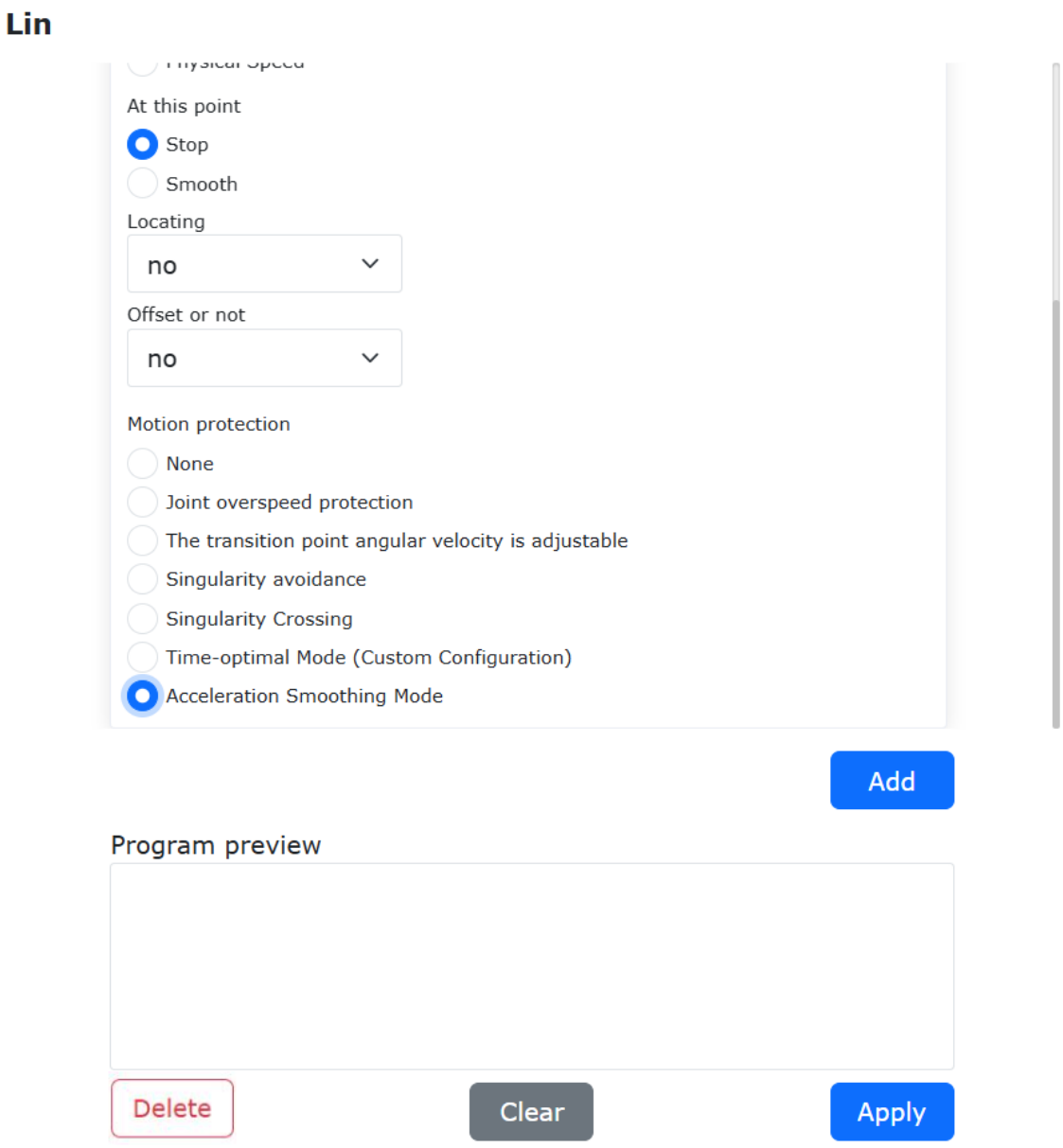


Figure 9.27-31 Blending Settings for Accelerated Smooth LIN

Step 3: Add LIN and ARC instructions within AccSmoothStart/End.

```
1↔ AccSmoothStart()
2↔ Lin(A0,100,10,0,0)
3↔ ARC(A1,0,0,0,0,0,0,0,A2,0,0,0,0,0,0,0,100,10)
4↔ AccSmoothEnd()
5↔ Lin(A3,100,10,0,0)
6↔ ARC(A4,0,0,0,0,0,0,0,A5,0,0,0,0,0,0,0,100,10)
7↔ AccSmoothStart()
8↔ Lin(A6,100,10,0,0)
9↔ ARC(A7,0,0,0,0,0,0,0,A8,0,0,0,0,0,0,0,100,10)
10↔ AccSmoothEnd()
```

Figure 9.27-32 Typical LIN-ARC Blending in Lua Mode

1.3.9.27.2.21 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode in settings.

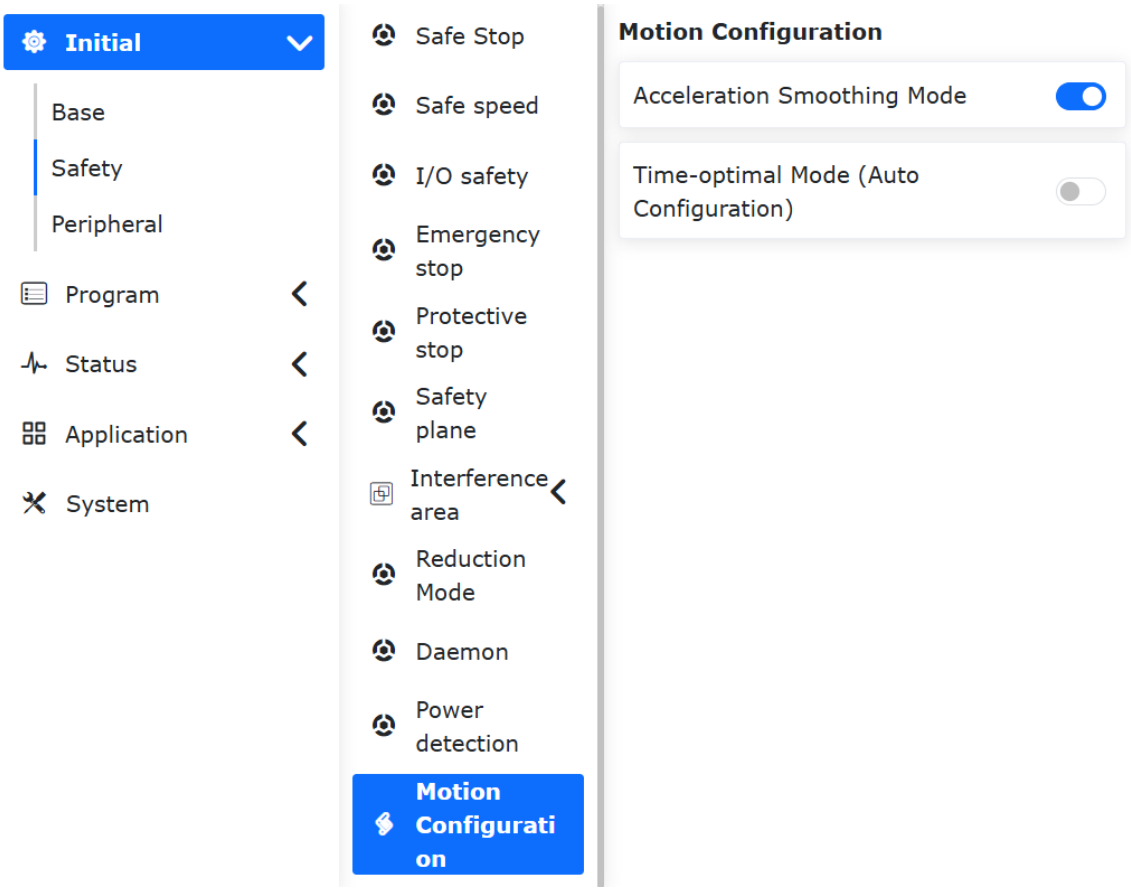


Figure 9.27-33 Acceleration Smooth Mode Switch Setup

Step 2: Select “A0” to “A8” as teaching points.

Step 3: Set LIN instruction with transition radius and mode, no motion protection.

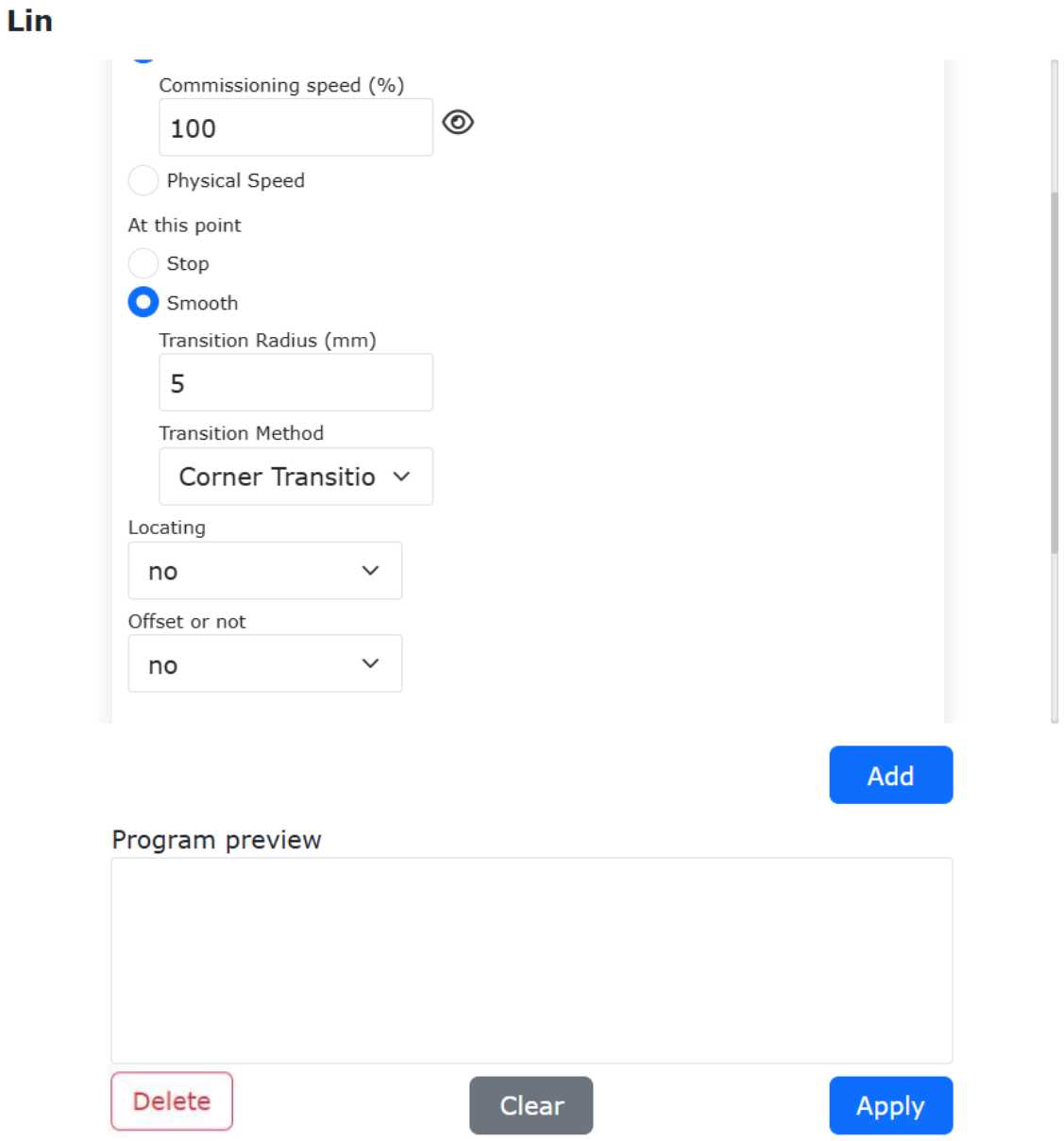


Figure 9.27-34 Blending Settings for Conventional LIN

Step 4: Generate program for all-instruction optimized velocity.

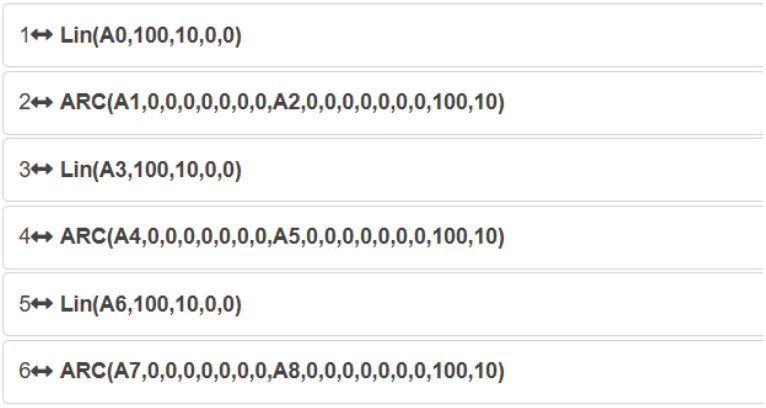


Figure 9.27-35 LIN-ARC Blending via Configuration Switch

1.3.9.27.2.22 LIN-CIRCLE Blending

1.3.9.27.2.23 Using Lua Instruction Mode

Step 1: Select “A0” to “A8” for LIN-CIRCLE blending.

Step 2: Configure LIN instruction with acceleration smooth mode and transition parameters.

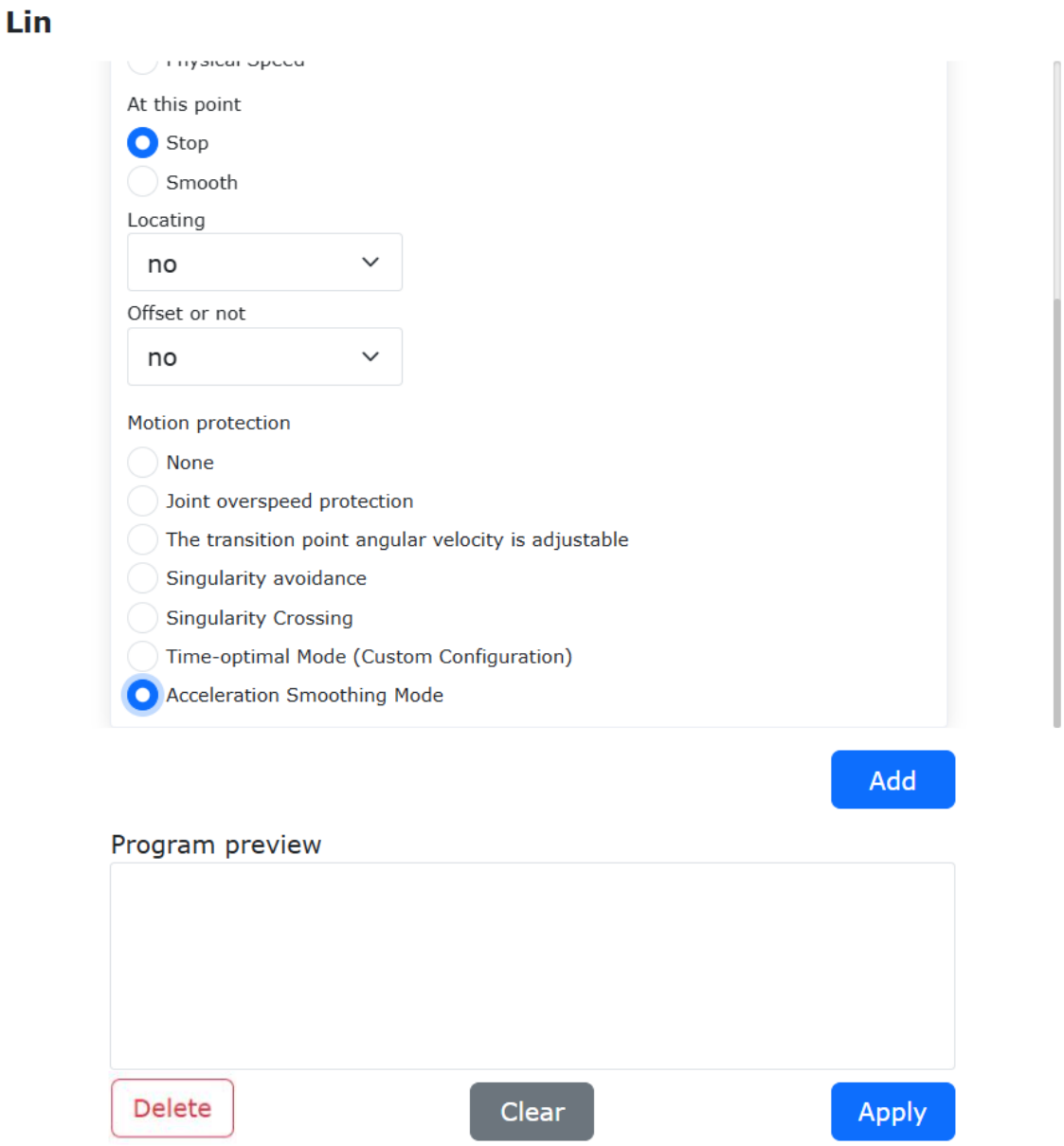


Figure 9.27-36 Blending Settings for Accelerated Smooth LIN

Step 3: Add LIN and CIRCLE instructions between AccSmoothStart/End.



Figure 9.27-37 Typical LIN-CIRCLE Blending in Lua Mode

1.3.9.27.2.24 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode switch.

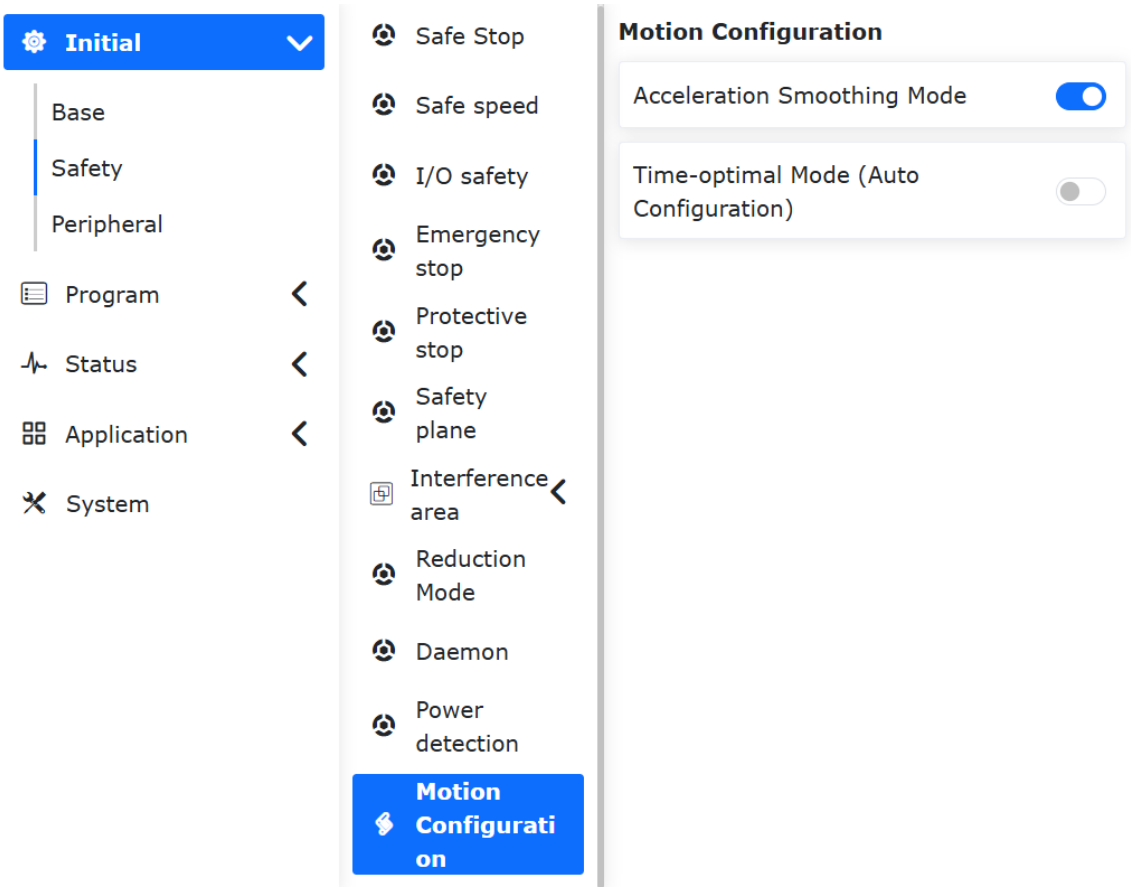


Figure 9.27-38 Acceleration Smooth Mode Configuration

Step 2: Choose “A0” to “A8” as teaching points.

Step 3: Set LIN instruction with transition radius and mode, no motion protection.

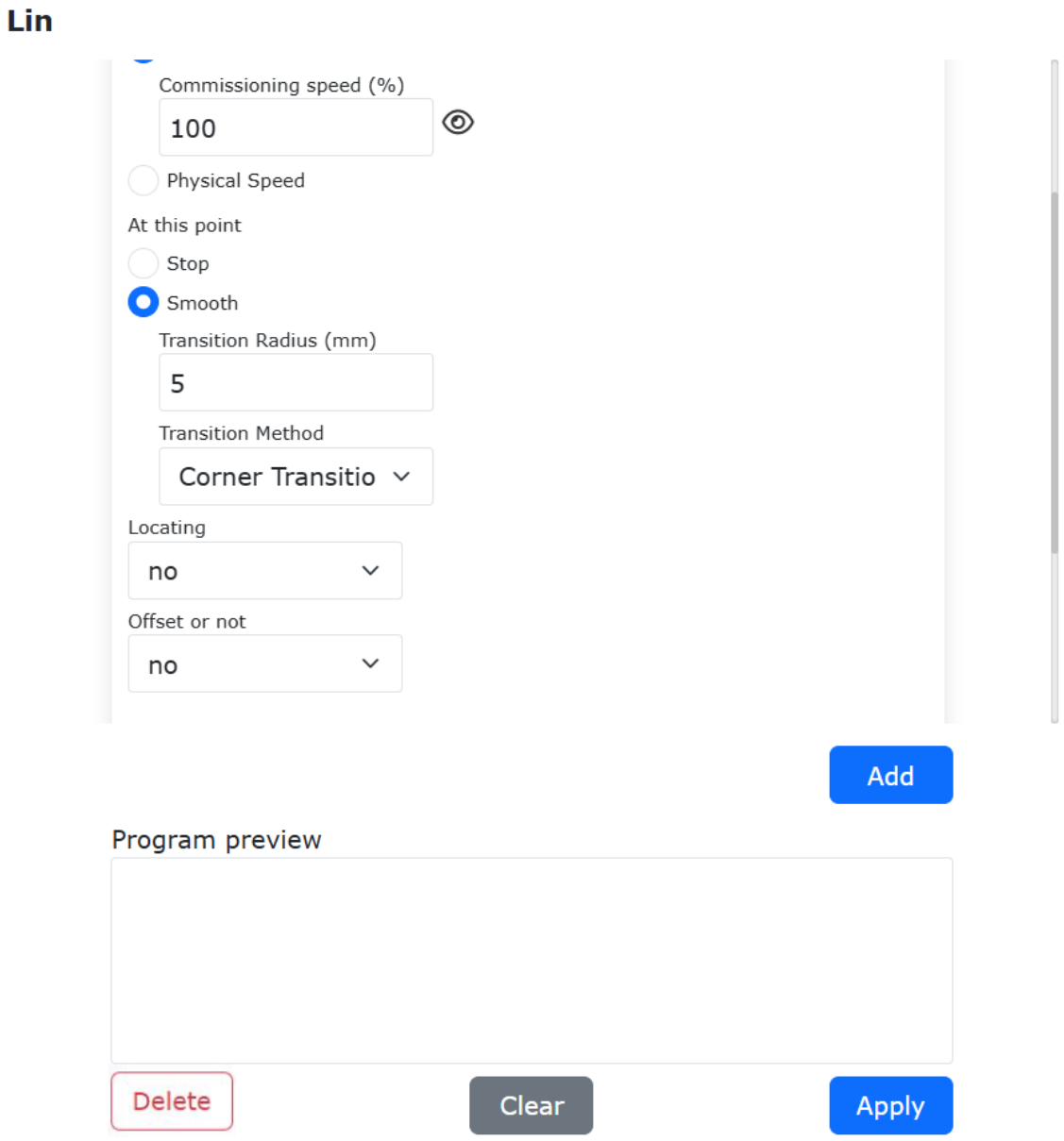


Figure 9.27-39 Blending Settings for Conventional LIN

Step 4: Generate program for optimized velocity across all instructions.

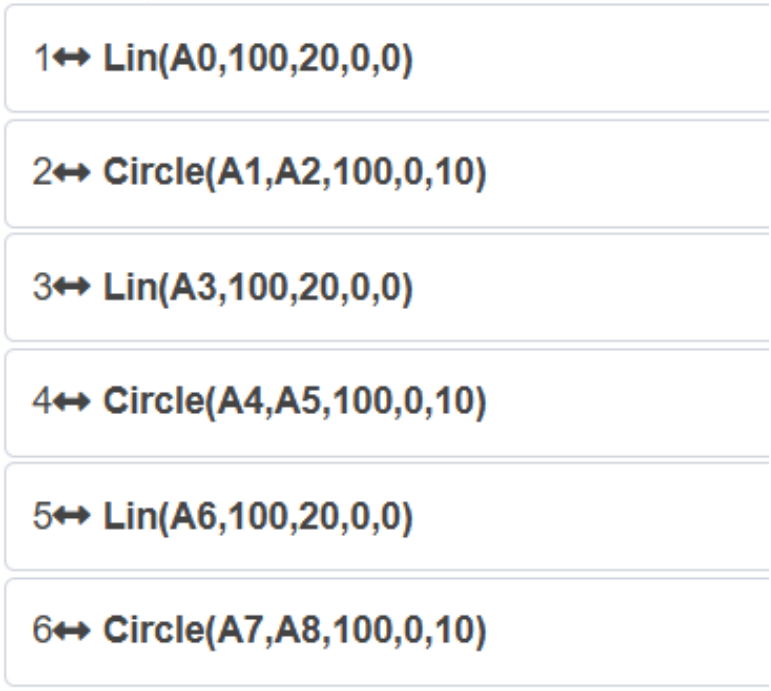


Figure 9.27-40 LIN-CIRCLE Blending via Configuration Switch

1.3.9.27.2.25 ARC-PTP Blending

1.3.9.27.2.26 Using Lua Instruction Mode

Step 1: Select “A0” to “A9” for ARC-PTP blending.

Step 2: Configure “Arc” instruction with acceleration smooth mode and “Smooth Transition”.

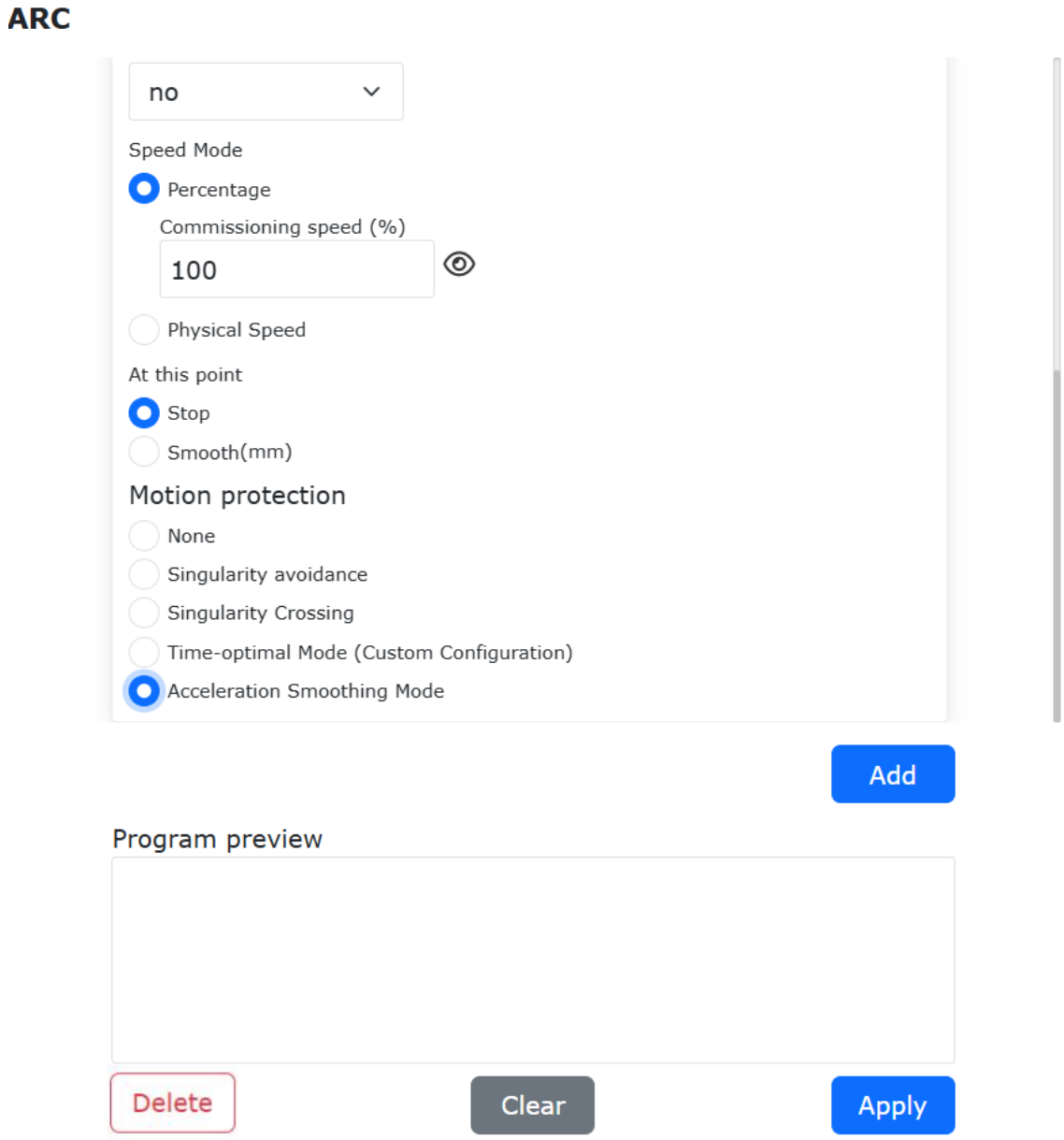


Figure 9.27-41 Blending Settings for Accelerated Smooth ARC

Step 3: Add ARC and PTP instructions within AccSmoothStart/End.



Figure 9.27-42 Typical ARC-PTP Blending in Lua Mode

1.3.9.27.2.27 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode in settings.

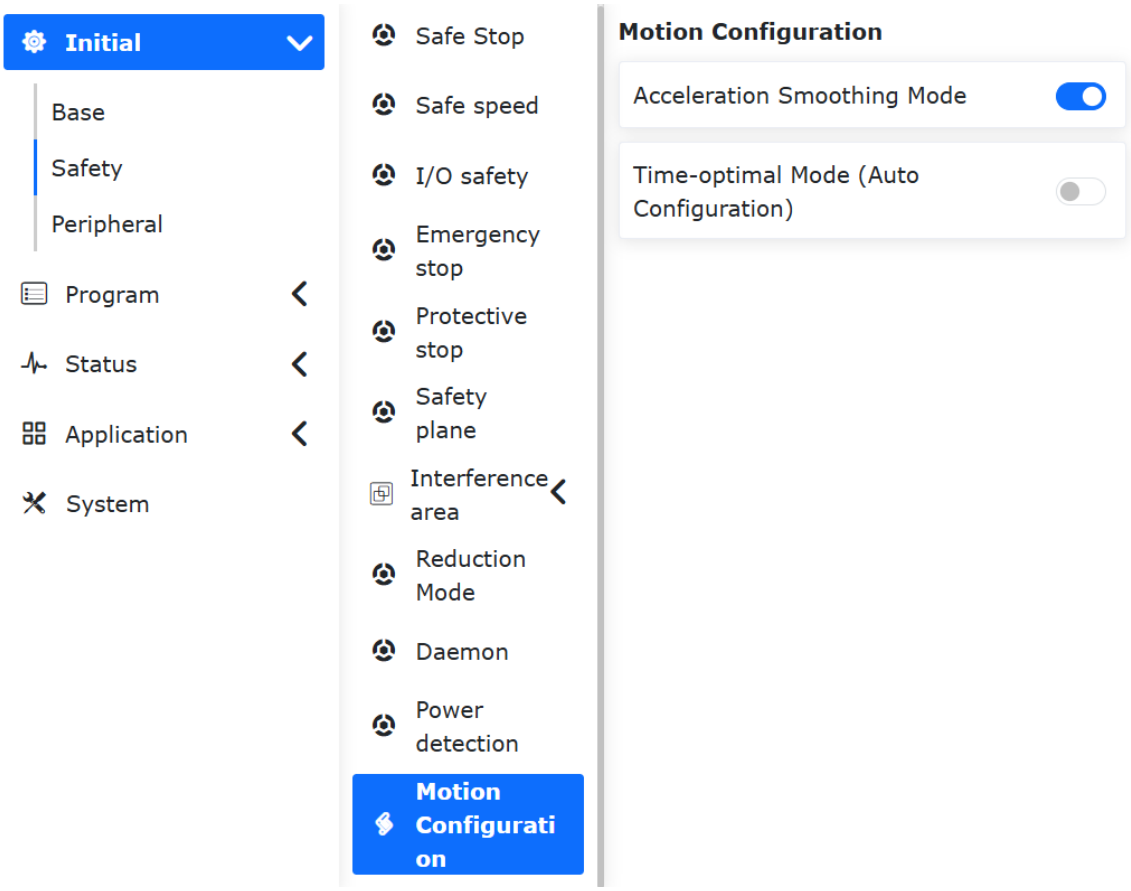


Figure 9.27-43 Acceleration Smooth Mode Switch Setup

Step 2: Select “A0” to “A9” as teaching points.

Step 3: Set Arc instruction with “Smooth Transition”, no motion protection.

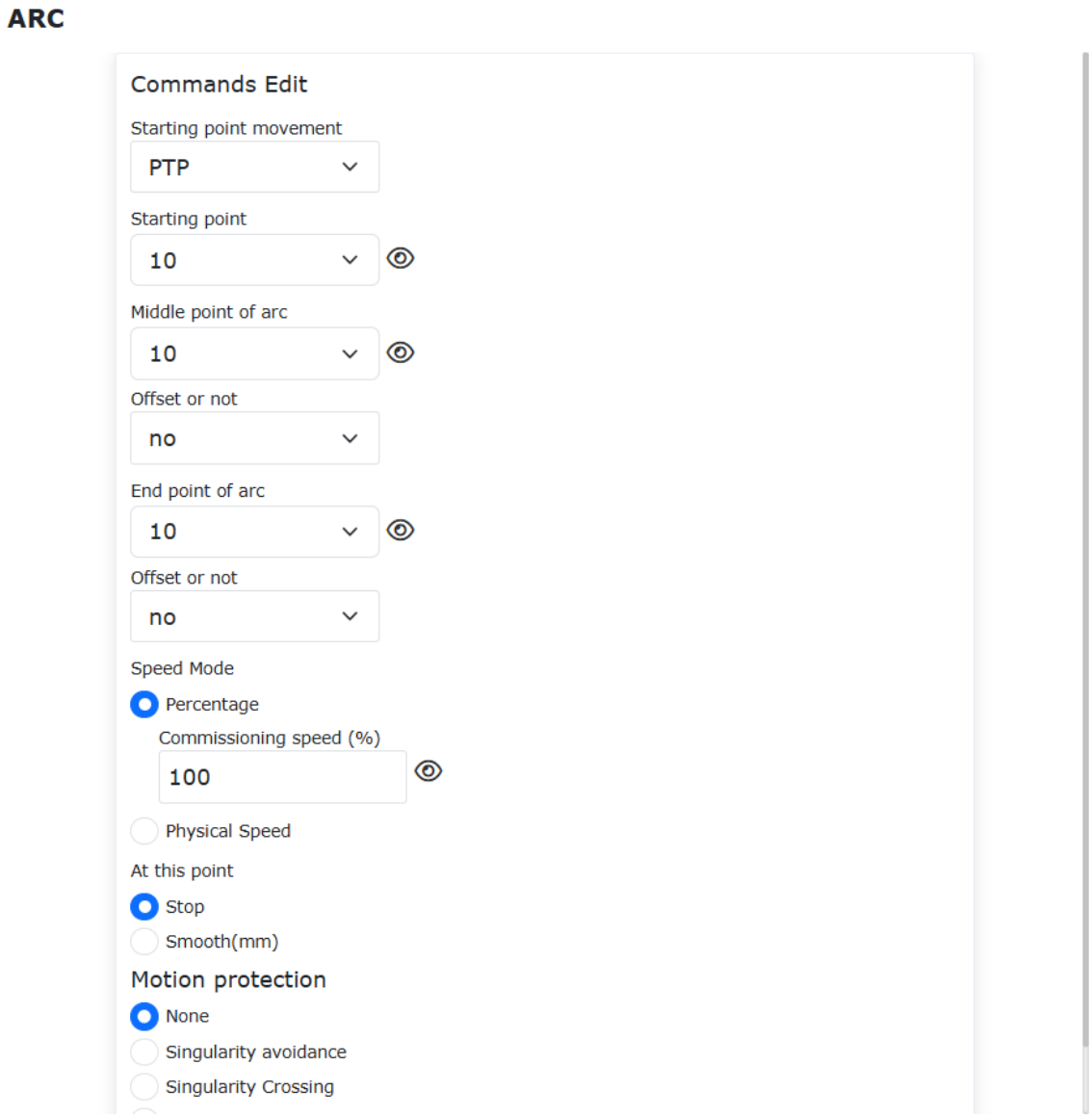


Figure 9.27-44 Blending Settings for Conventional ARC

Step 4: Generate program for all-instruction optimized velocity.

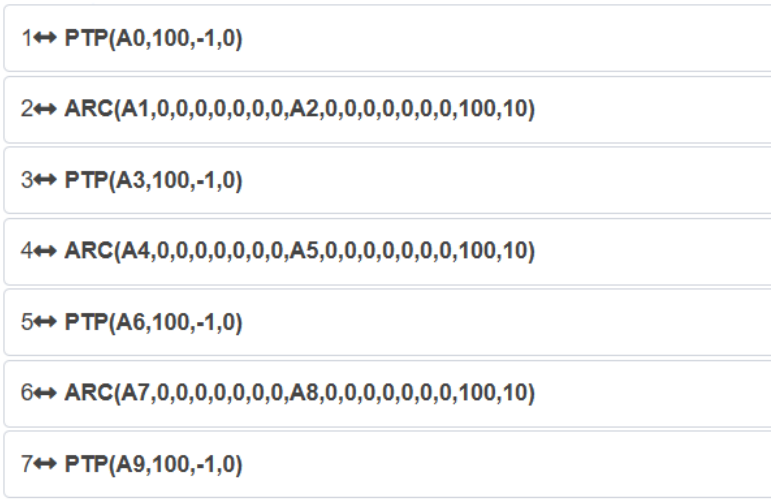


Figure 9.27-45 ARC-PTP Blending via Configuration Switch

1.3.9.27.2.28 ARC-LIN Blending

1.3.9.27.2.29 Using Lua Instruction Mode

Step 1: Select “A0” to “A9” for ARC-LIN blending.

Step 2: Configure Arc instruction with acceleration smooth mode and “Smooth Transition”.

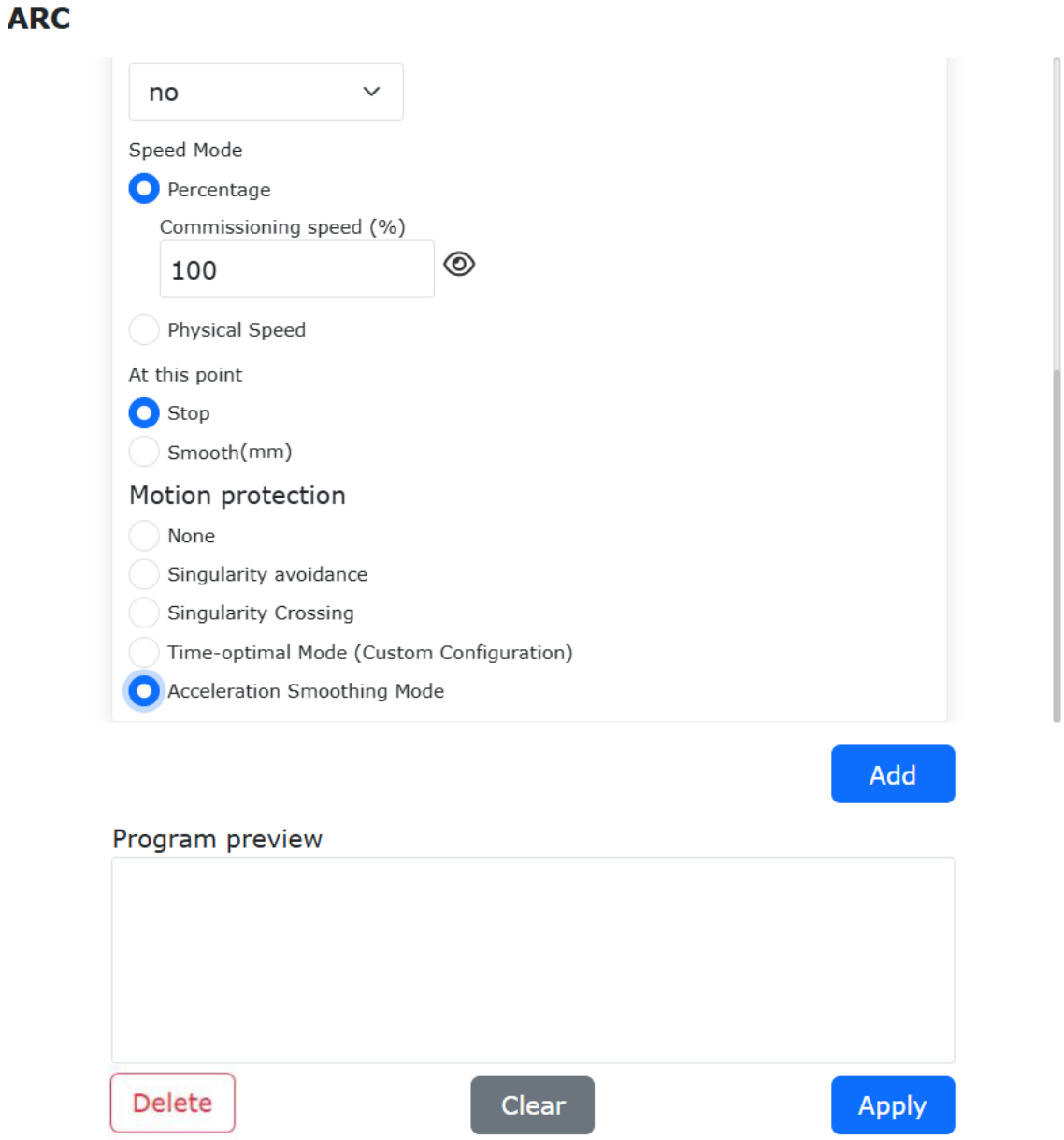


Figure 9.27-46 Blending Settings for Accelerated Smooth ARC

Step 3: Add ARC and LIN instructions between AccSmoothStart/End.

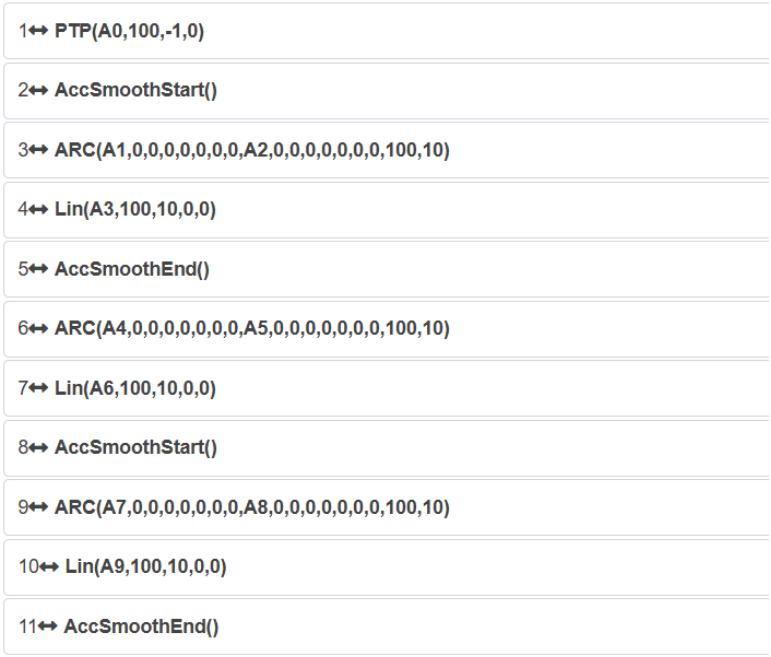


Figure 9.27-47 Typical ARC-LIN Blending in Lua Mode

1.3.9.27.2.30 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode switch.

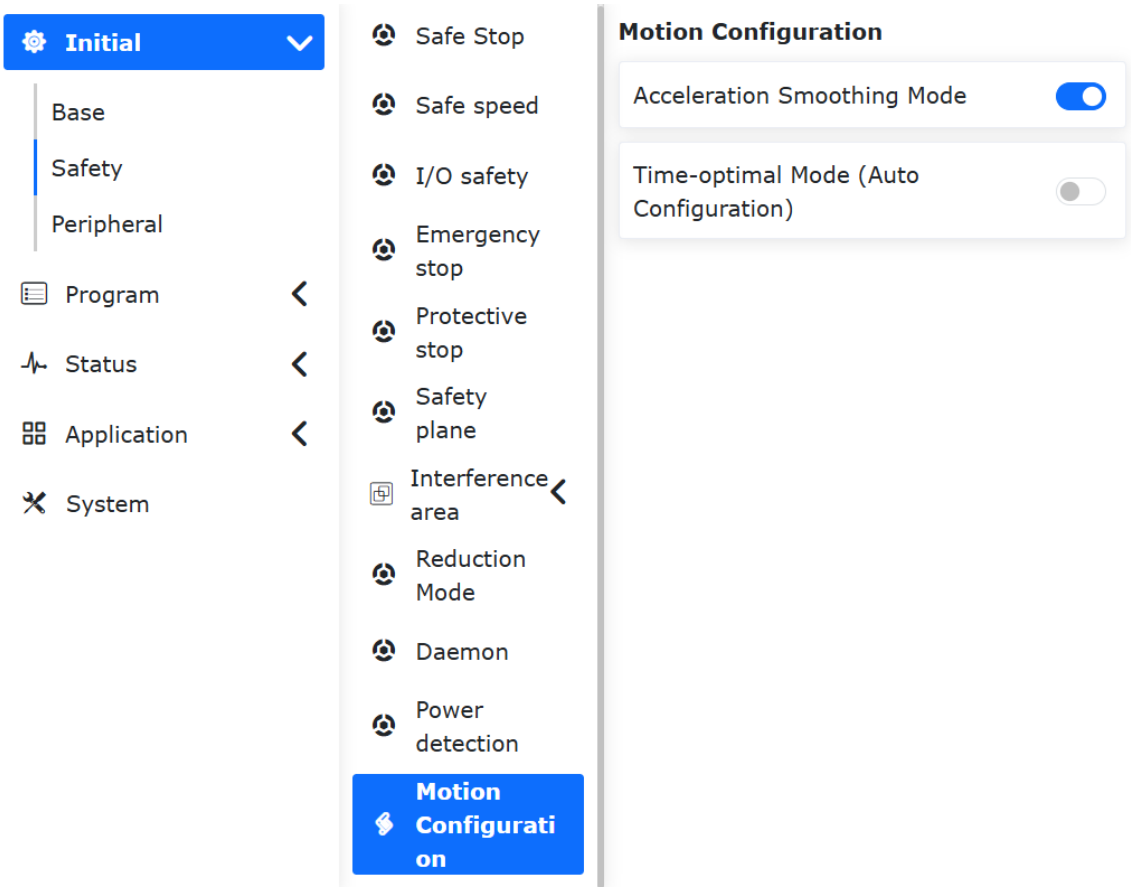


Figure 9.27-48 Acceleration Smooth Mode Configuration

Step 2: Choose “A0” to “A9” as teaching points.

Step 3: Set Arc instruction with “Smooth Transition”, no motion protection.

ARC

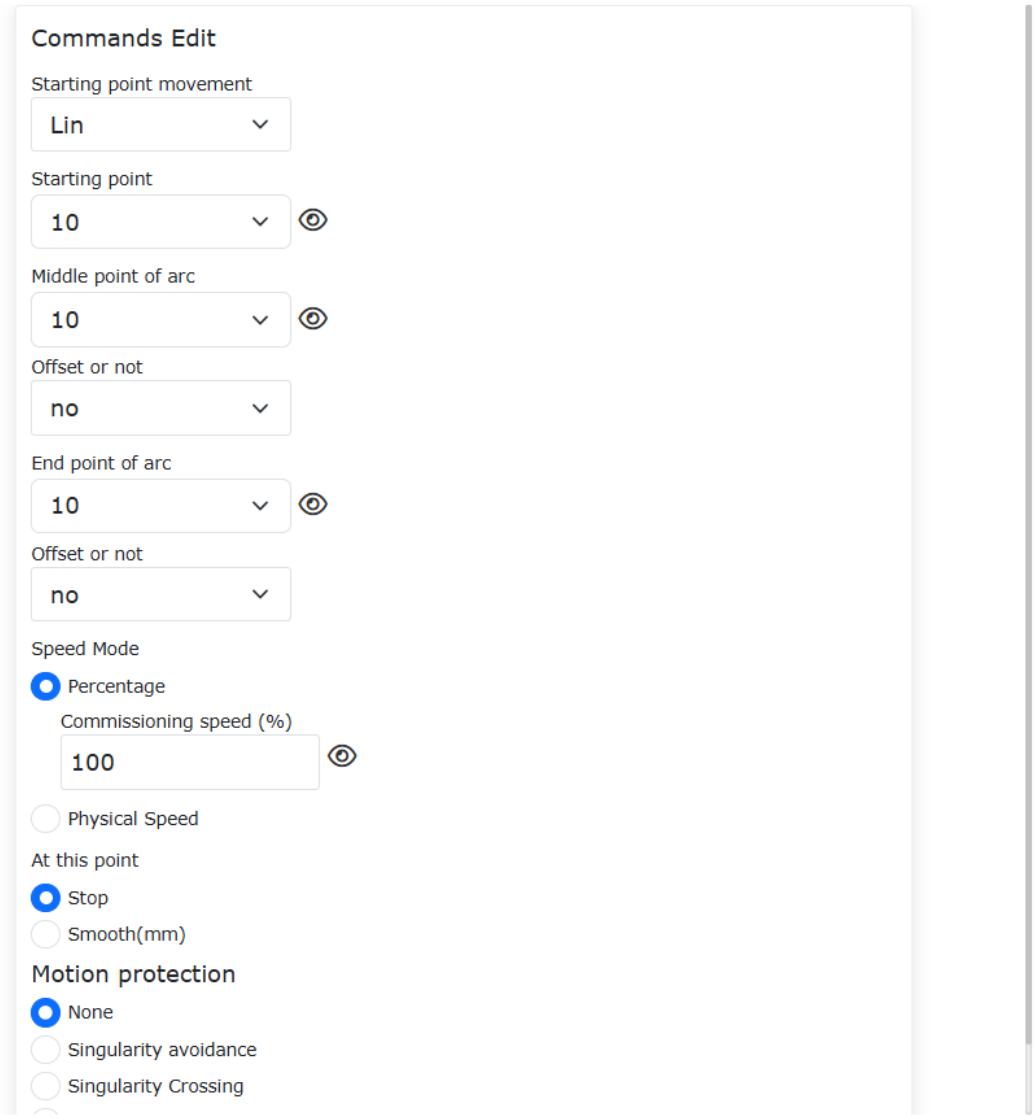


Figure 9.27-49 Blending Settings for Conventional ARC

Step 4: Generate program for optimized velocity across all instructions.

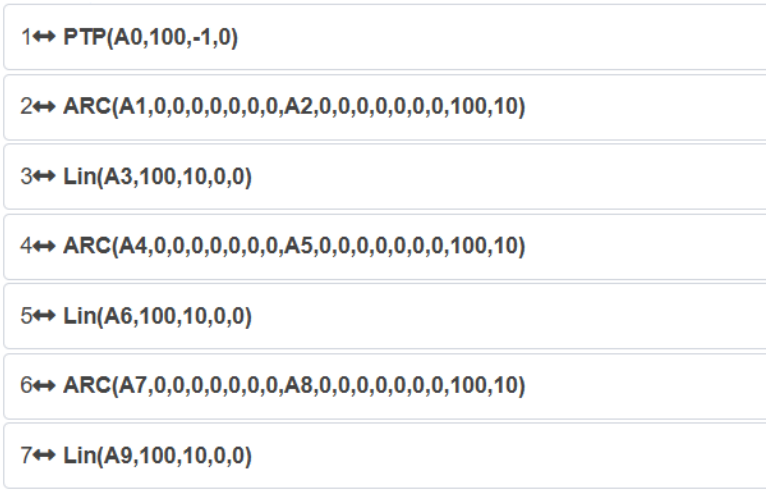


Figure 9.27-50 ARC-LIN Blending via Configuration Switch

1.3.9.27.2.31 ARC-ARC Blending

1.3.9.27.2.32 Using Lua Instruction Mode

Step 1: Select “A0” to “A12” for ARC-ARC blending.

Step 2: Configure Arc instruction with acceleration smooth mode and “Smooth Transition”.

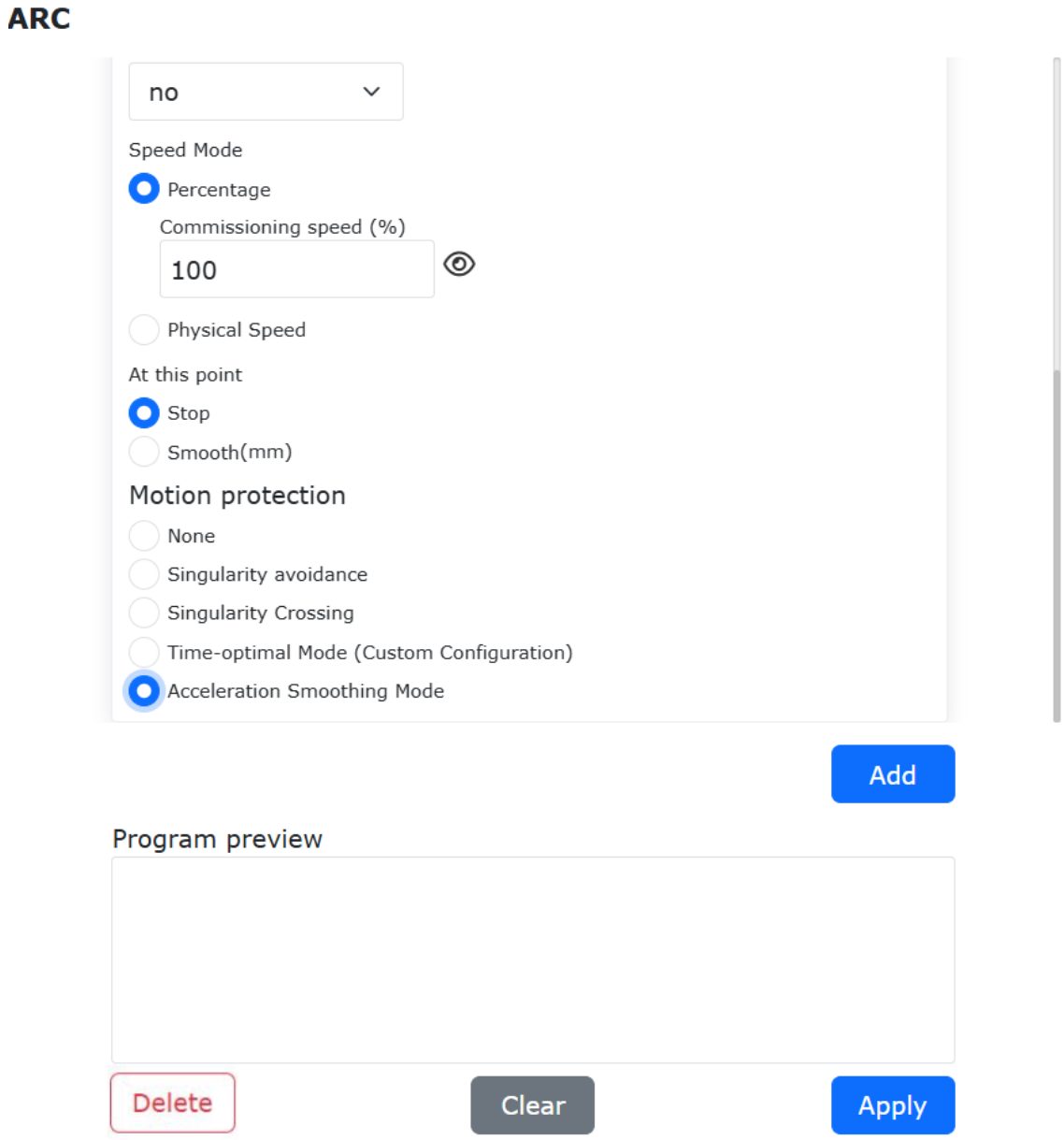


Figure 9.27-51 Blending Settings for Accelerated Smooth ARC

Step 3: Add multiple Arc instructions within AccSmoothStart/End.



Figure 9.27-52 Typical ARC-ARC Blending in Lua Mode

1.3.9.27.2.33 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode in settings.

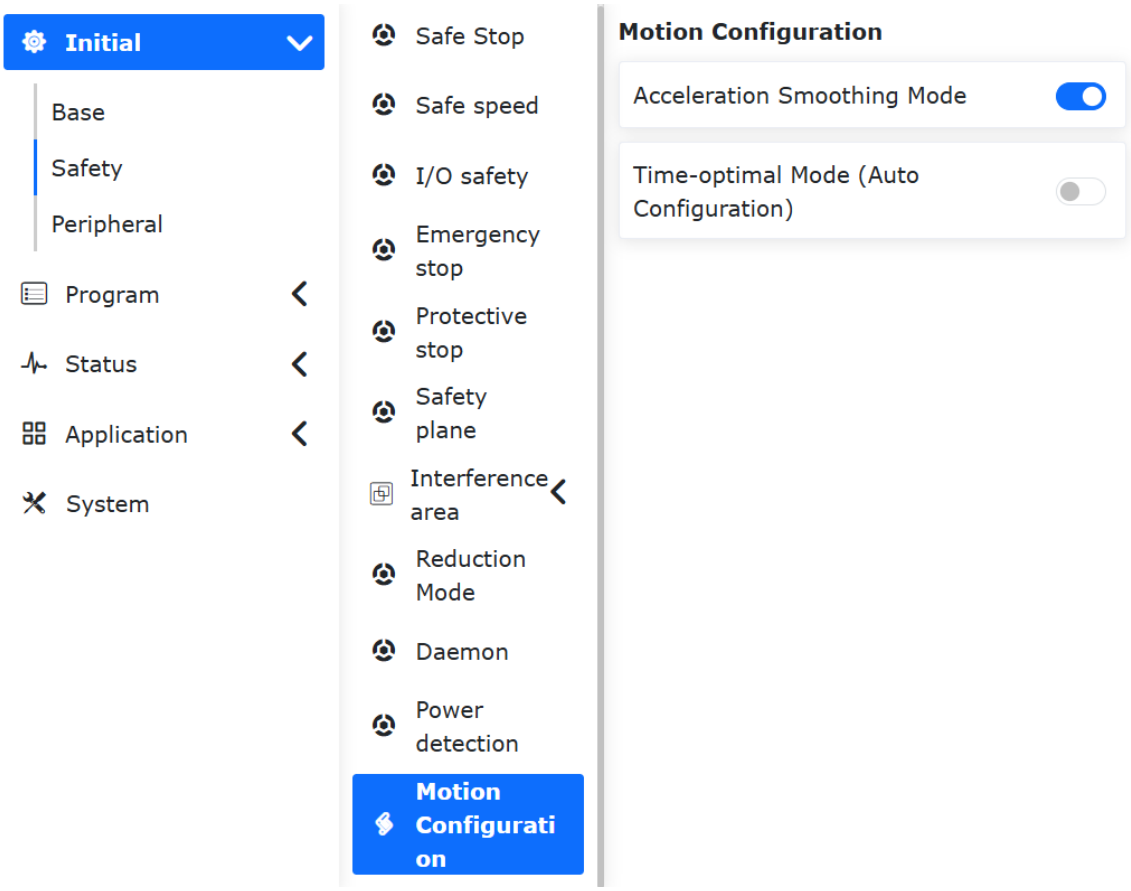


Figure 9.27-53 Acceleration Smooth Mode Switch Setup

Step 2: Select “A0” to “A12” as teaching points.

Step 3: Set Arc instruction with “Smooth Transition”, no motion protection.

ARC

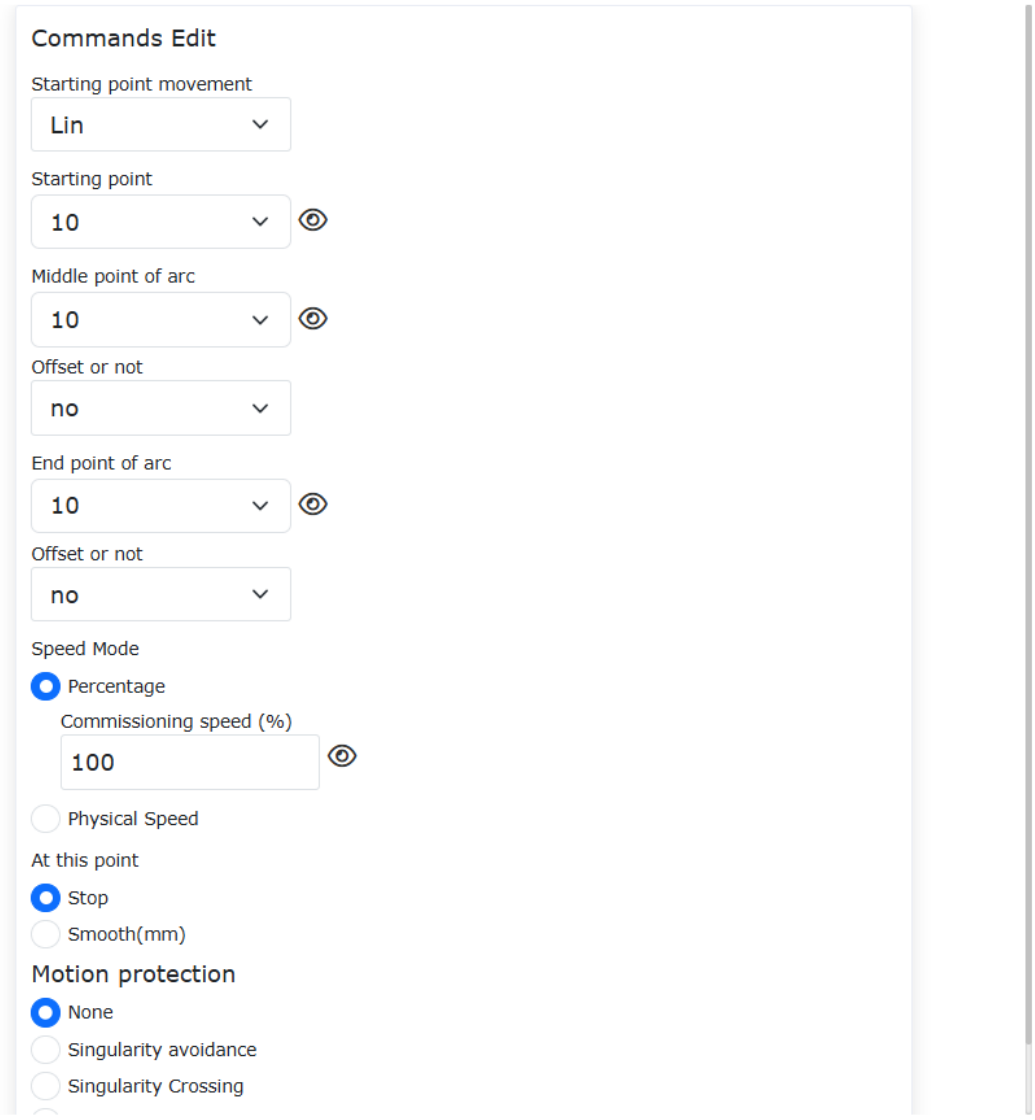


Figure 9.27-54 Blending Settings for Conventional ARC

Step 4: Generate program for all-instruction optimized velocity.

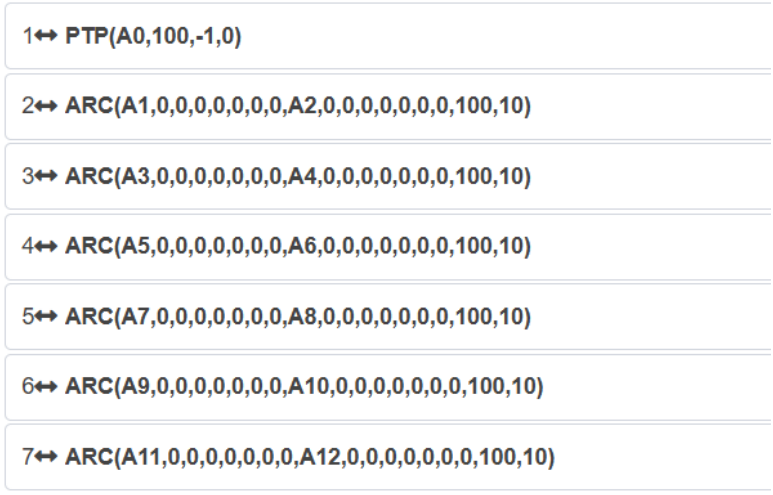


Figure 9.27-55 ARC-ARC Blending via Configuration Switch

1.3.9.27.2.34 ARC-CIRCLE Blending

1.3.9.27.2.35 Using Lua Instruction Mode

Step 1: Select “A0” to “A12” for ARC-CIRCLE blending.

Step 2: Configure Arc instruction with acceleration smooth mode and “Smooth Transition”.

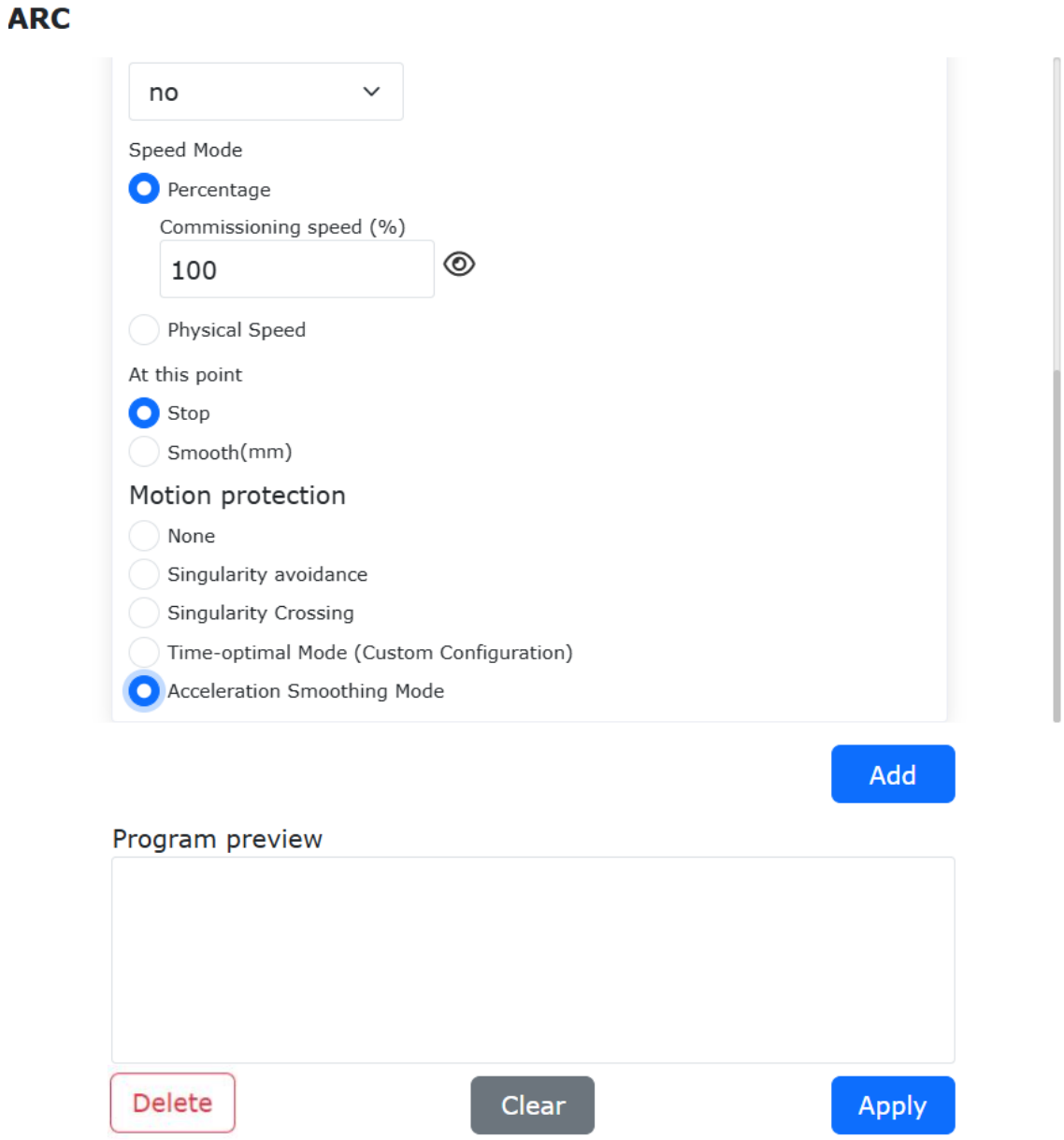


Figure 9.27-56 Blending Settings for Accelerated Smooth ARC

Step 3: Add Arc and CIRCLE instructions between AccSmoothStart/End.

```
1↔ PTP(A0,100,-1,0)
2↔ AccSmoothStart()
3↔ ARC(A1,0,0,0,0,0,0,0,A2,0,0,0,0,0,0,0,100,10)
4↔ Circle(A3,A4,100,0,10)
5↔ AccSmoothEnd()
6↔ ARC(A5,0,0,0,0,0,0,0,A6,0,0,0,0,0,0,0,100,10)
7↔ Circle(A7,A8,100,0,10)
8↔ AccSmoothStart()
9↔ ARC(A9,0,0,0,0,0,0,0,A10,0,0,0,0,0,0,0,100,10)
10↔ Circle(A11,A12,100,0,10)
11↔ AccSmoothEnd()
```

Figure 9.27-57 Typical ARC-CIRCLE Blending in Lua Mode

1.3.9.27.2.36 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode switch.

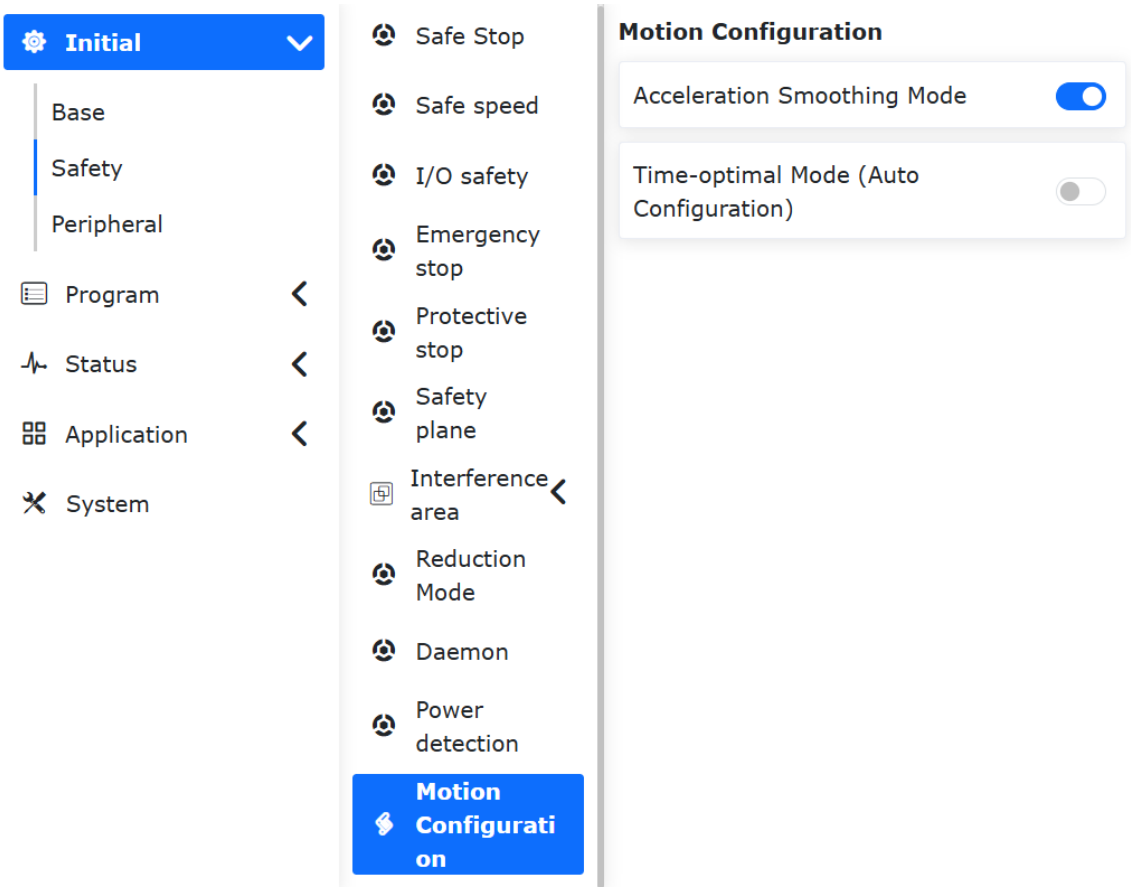


Figure 9.27-58 Acceleration Smooth Mode Configuration

Step 2: Choose “A0” to “A12” as teaching points.

Step 3: Set Arc instruction with “Smooth Transition”, no motion protection.

ARC

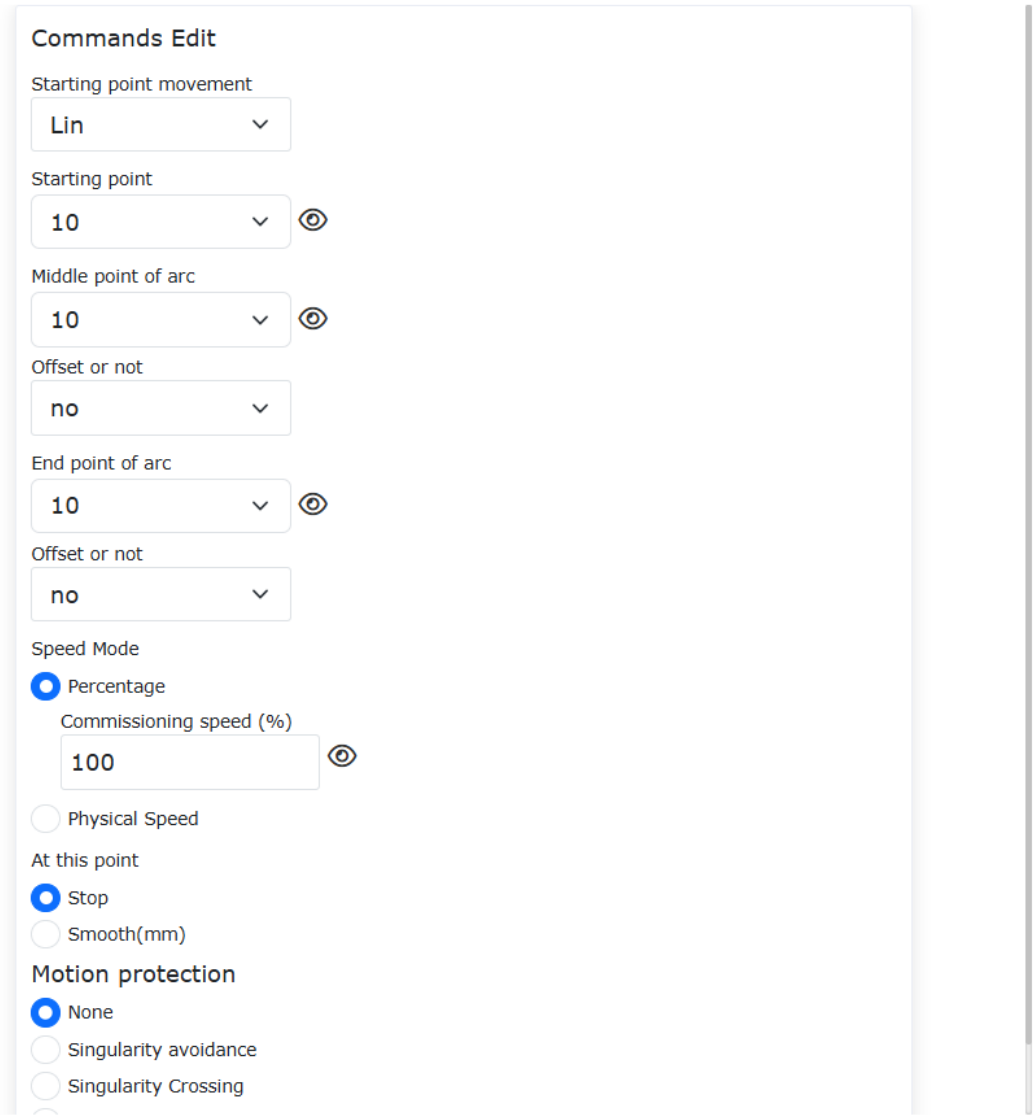


Figure 9.27-59 Blending Settings for Conventional ARC

Step 4: Generate program for optimized velocity across all instructions.

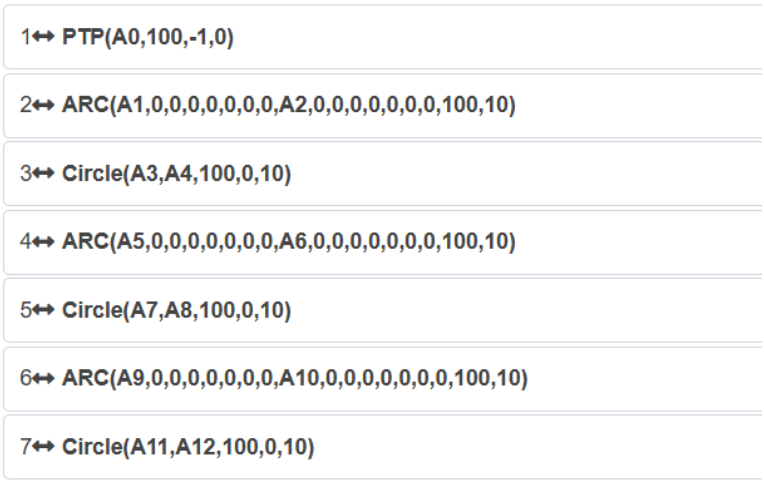


Figure 9.27-60 ARC-CIRCLE Blending via Configuration Switch

1.3.9.27.2.37 CIRCLE-PTP Blending

1.3.9.27.2.38 Using Lua Instruction Mode

Step 1: Select “A0” to “A9” for CIRCLE-PTP blending.

Step 2: Configure “Full Circle” instruction with acceleration smooth mode and “Smooth Transition”.

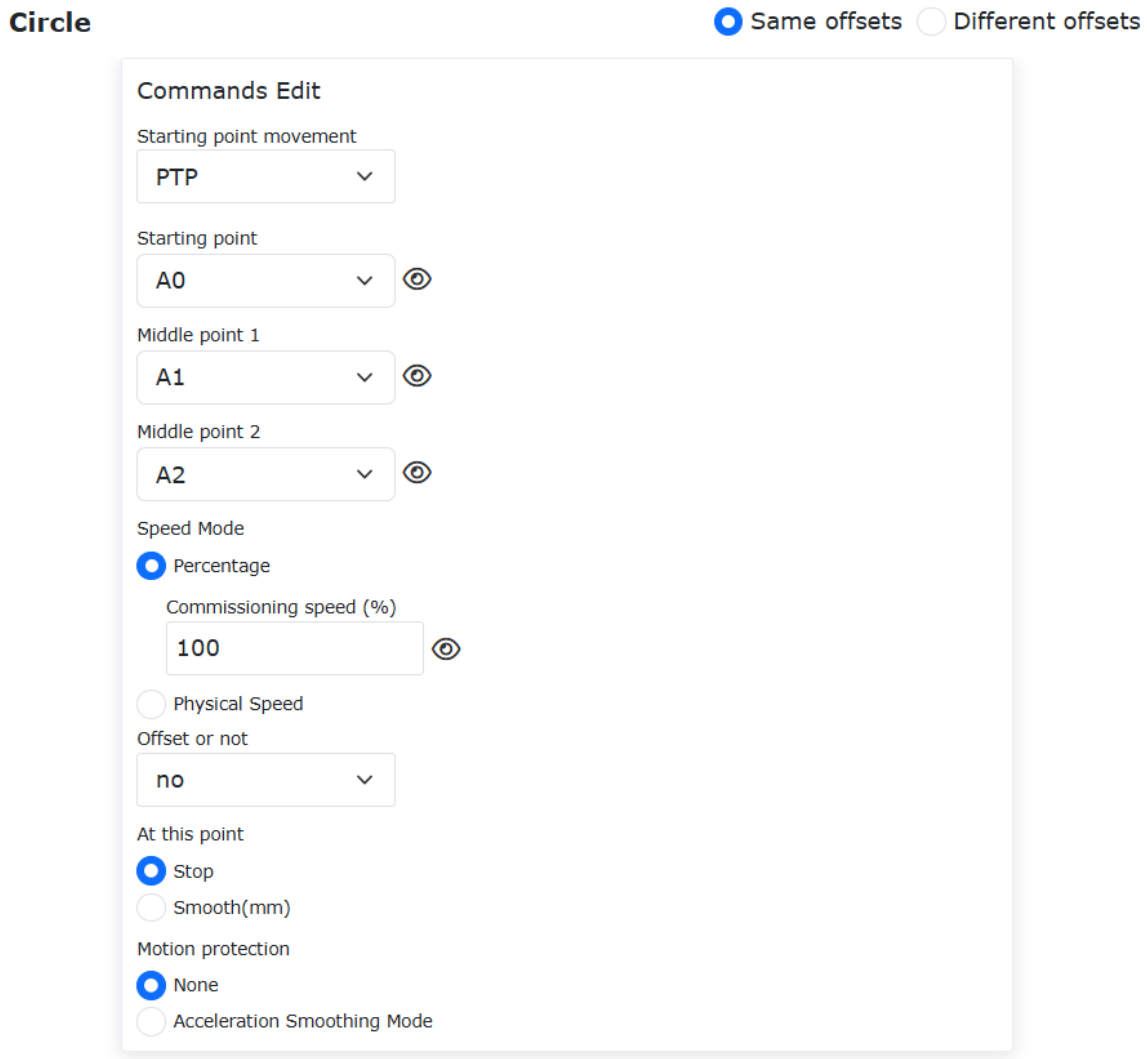


Figure 9.27-61 Blending Settings for Accelerated Smooth CIRCLE

Step 3: Add CIRCLE and PTP instructions within AccSmoothStart/End.



Figure 9.27-62 Typical CIRCLE-PTP Blending in Lua Mode

1.3.9.27.2.39 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode in settings.

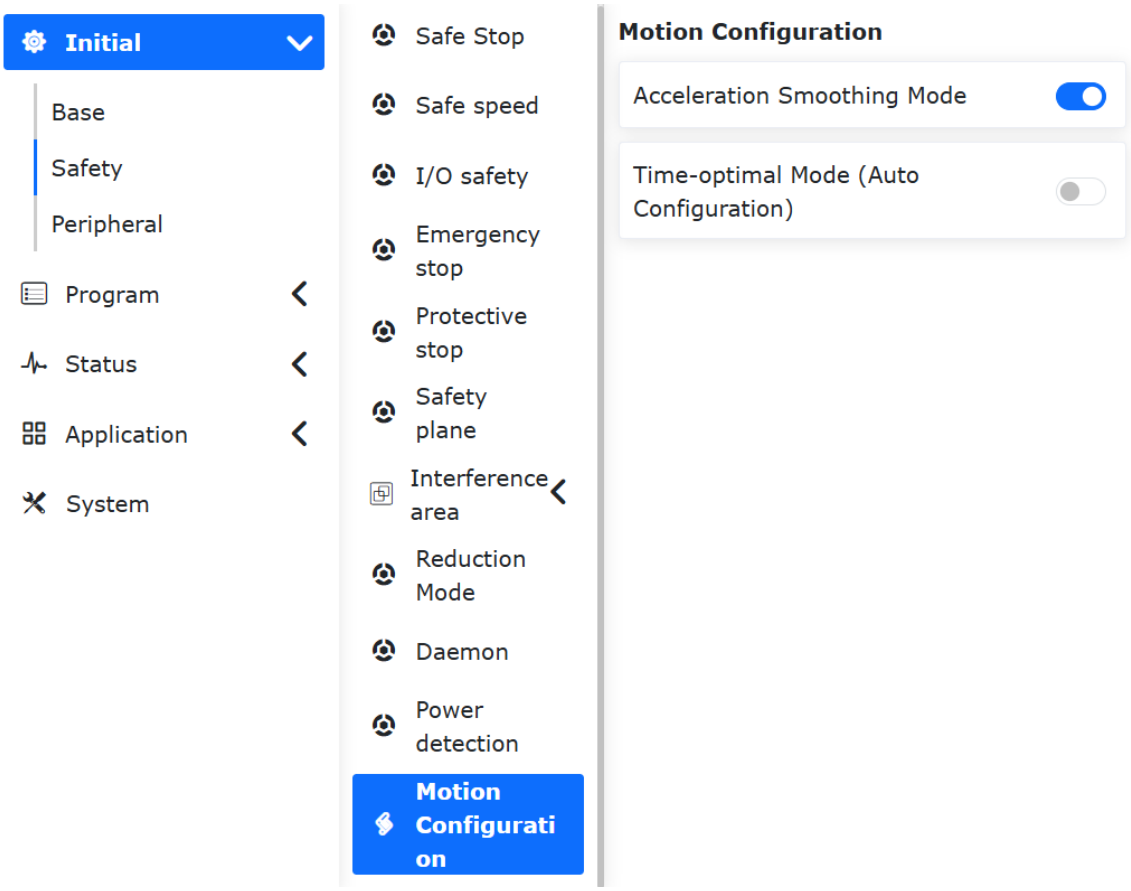


Figure 9.27-63 Acceleration Smooth Mode Switch Setup

Step 2: Select “A0” to “A9” as teaching points.

Step 3: Set Full Circle instruction with “Smooth Transition”, no motion protection.

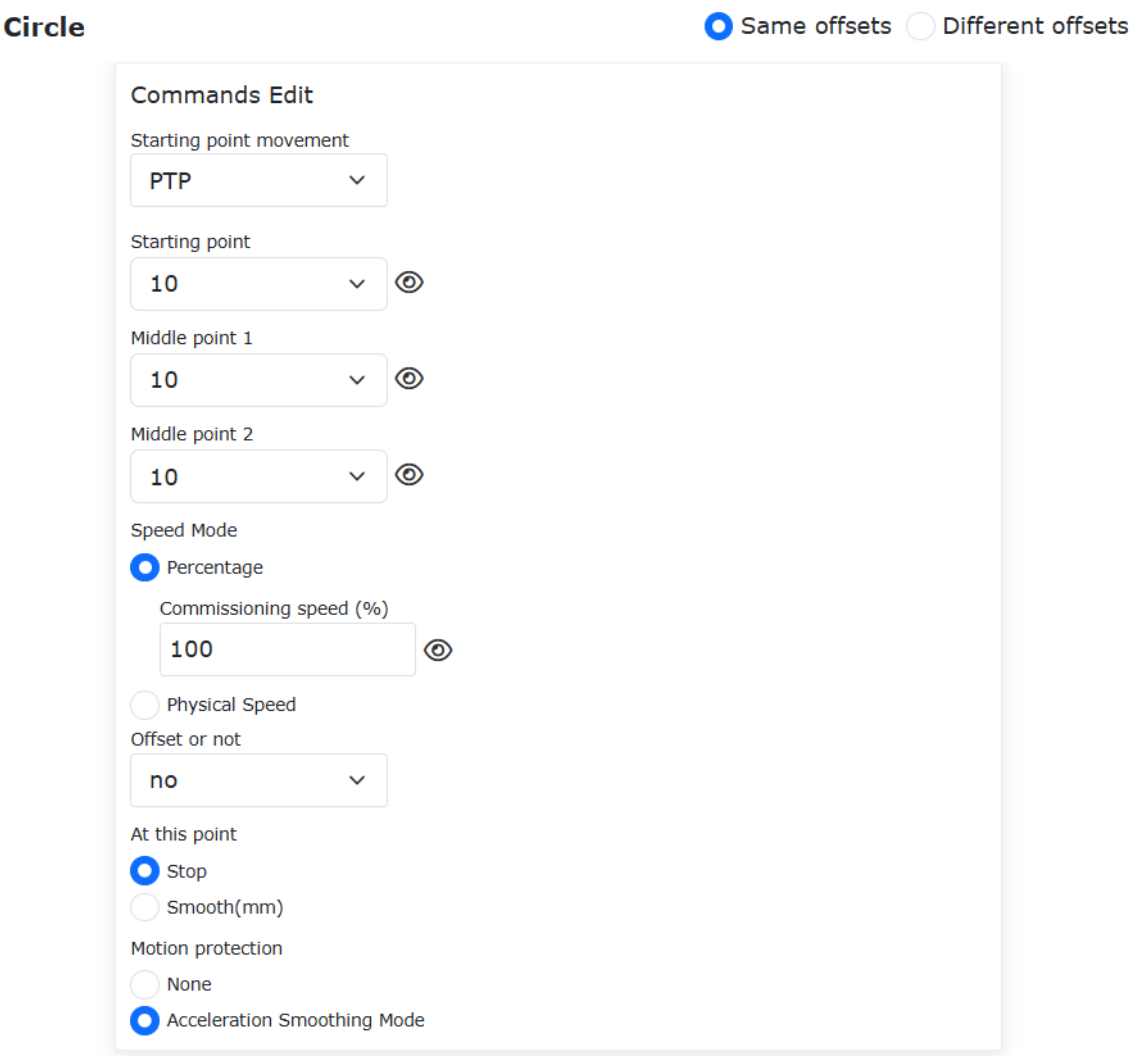


Figure 9.27-64 Blending Settings for Conventional CIRCLE

Step 4: Generate program for all-instruction optimized velocity.



Figure 9.27-65 CIRCLE-PTP Blending via Configuration Switch

1.3.9.27.2.40 CIRCLE-LIN Blending

1.3.9.27.2.41 Using Lua Instruction Mode

Step 1: Select “A0” to “A12” for CIRCLE-LIN blending.

Step 2: Configure Full Circle instruction with acceleration smooth mode and “Smooth Transition”.

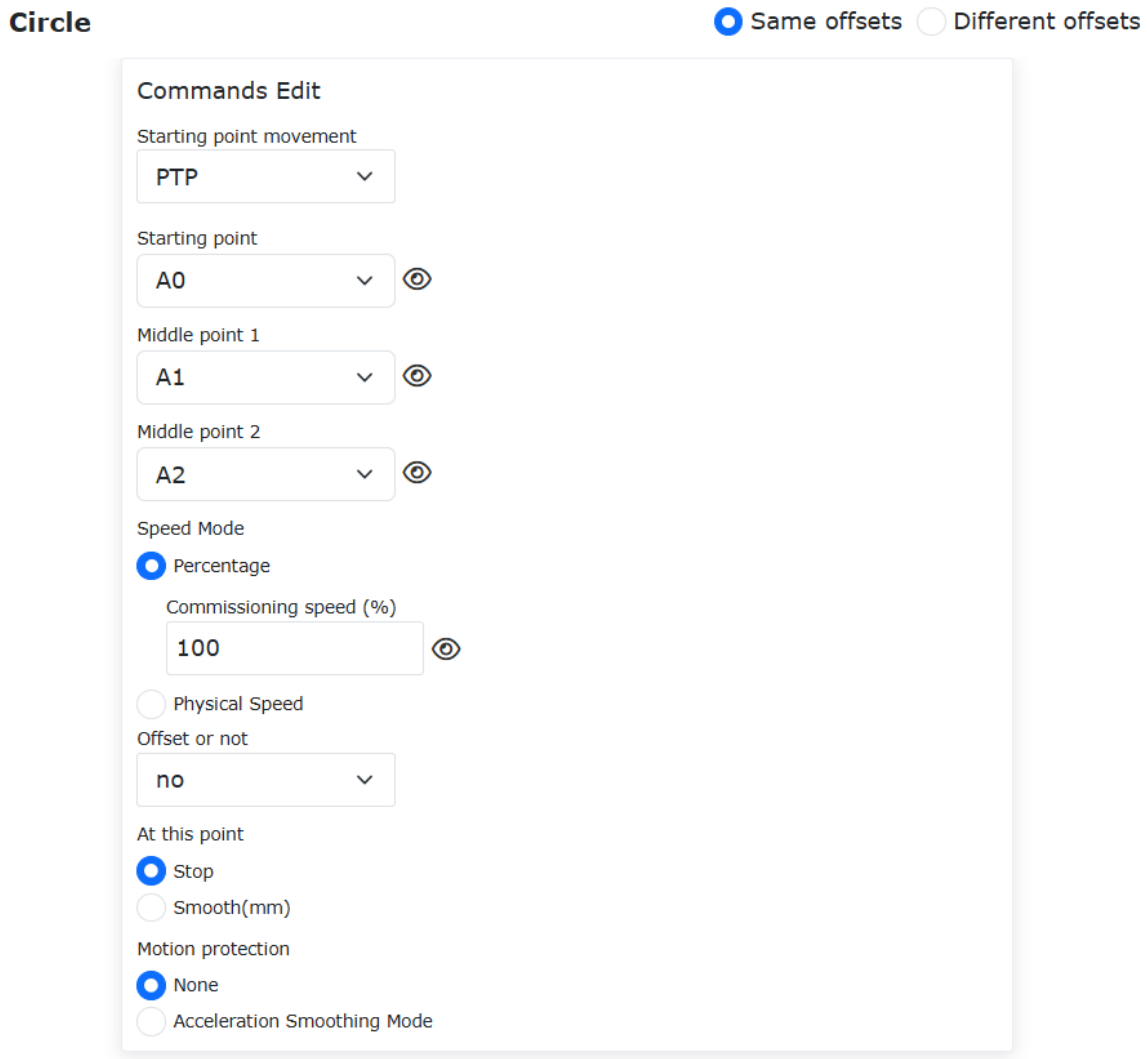


Figure 9.27-66 Blending Settings for Accelerated Smooth CIRCLE

Step 3: Add CIRCLE and LIN instructions between AccSmoothStart/End.

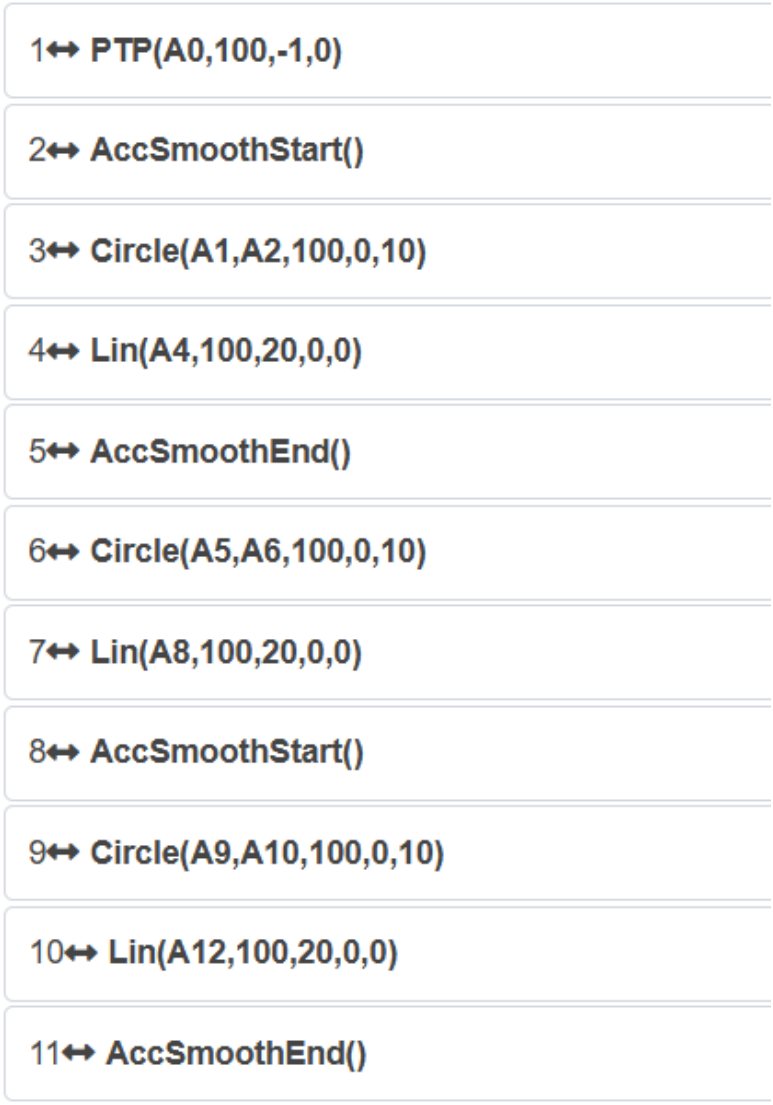


Figure 9.27-67 Typical CIRCLE-LIN Blending in Lua Mode

1.3.9.27.2.42 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode switch.

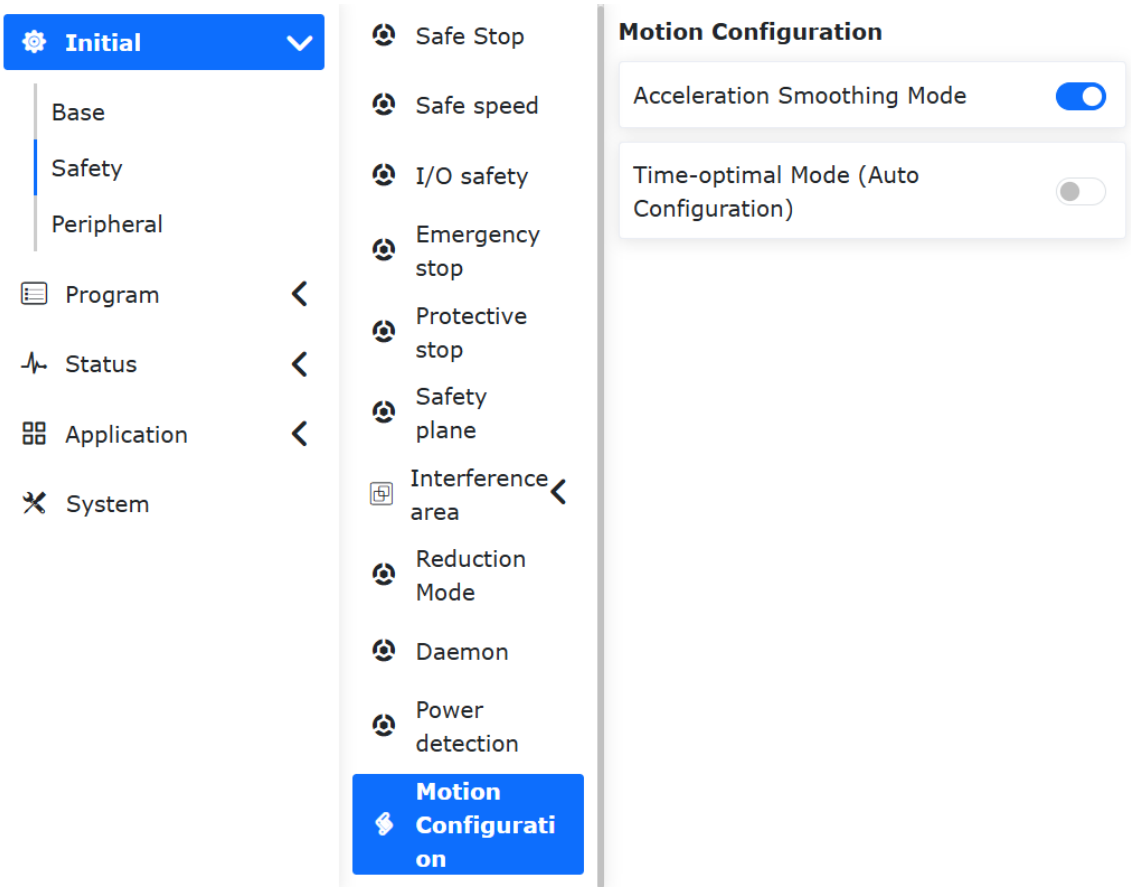


Figure 9.27-68 Acceleration Smooth Mode Configuration

Step 2: Choose “A0” to “A12” as teaching points.

Step 3: Set Full Circle instruction with “Smooth Transition”, no motion protection.

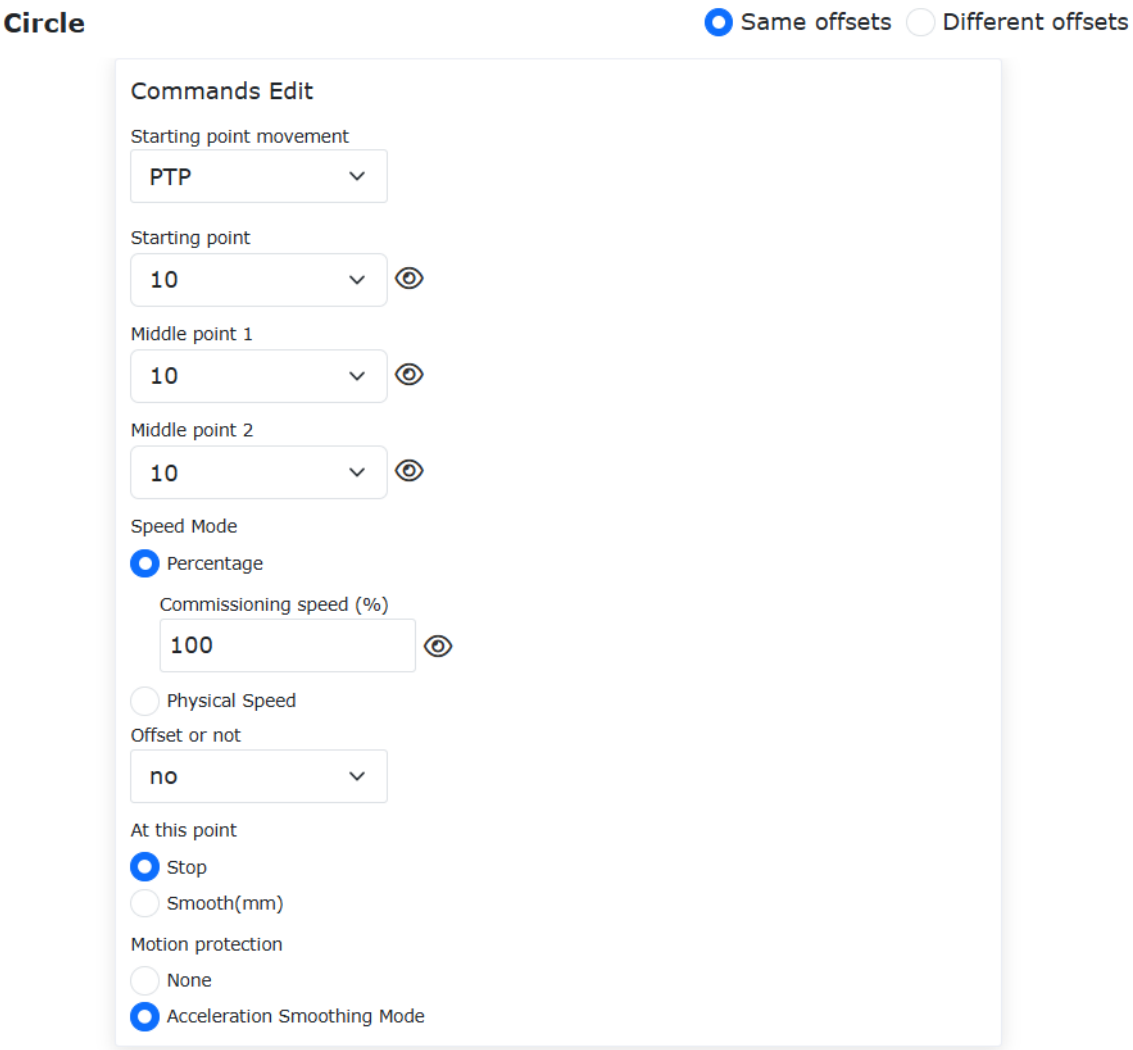


Figure 9.27-69 Blending Settings for Conventional CIRCLE

Step 4: Generate program for optimized velocity across all instructions.

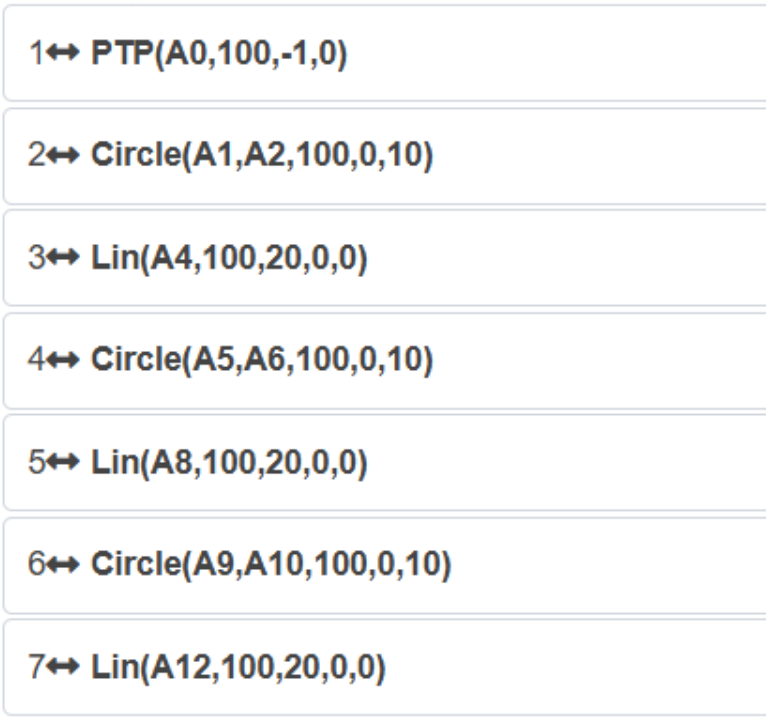


Figure 9.27-70 CIRCLE-LIN Blending via Configuration Switch

1.3.9.27.2.43 CIRCLE-ARC Blending

1.3.9.27.2.44 Using Lua Instruction Mode

Step 1: Select “A0” to “A12” for CIRCLE-ARC blending.

Step 2: Configure Full Circle instruction with acceleration smooth mode and “Smooth Transition”.

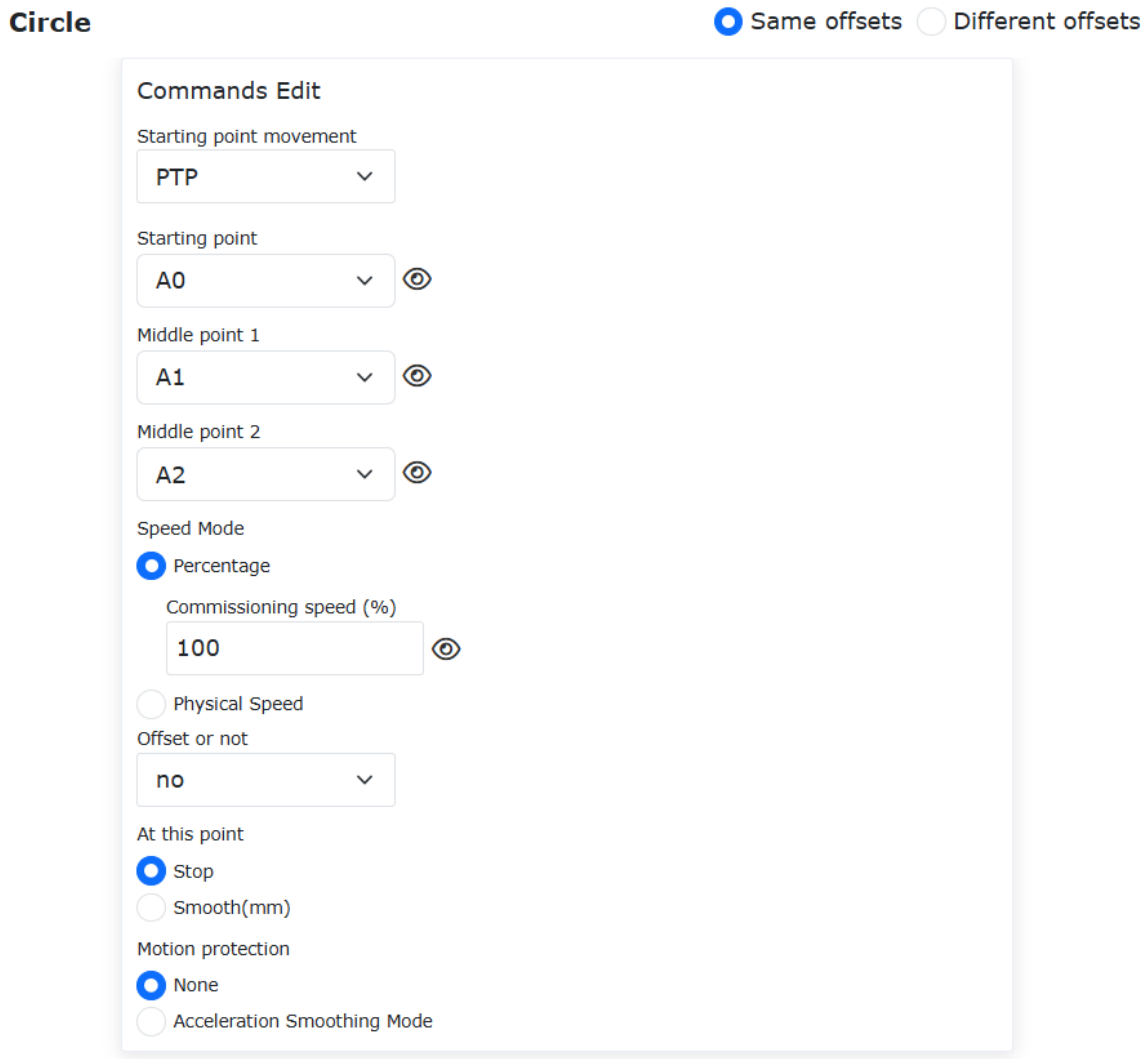


Figure 9.27-71 Blending Settings for Accelerated Smooth CIRCLE

Step 3: Add CIRCLE and Arc instructions within AccSmoothStart/End.



Figure 9.27-72 Typical CIRCLE-ARC Blending in Lua Mode

1.3.9.27.2.45 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode in settings.

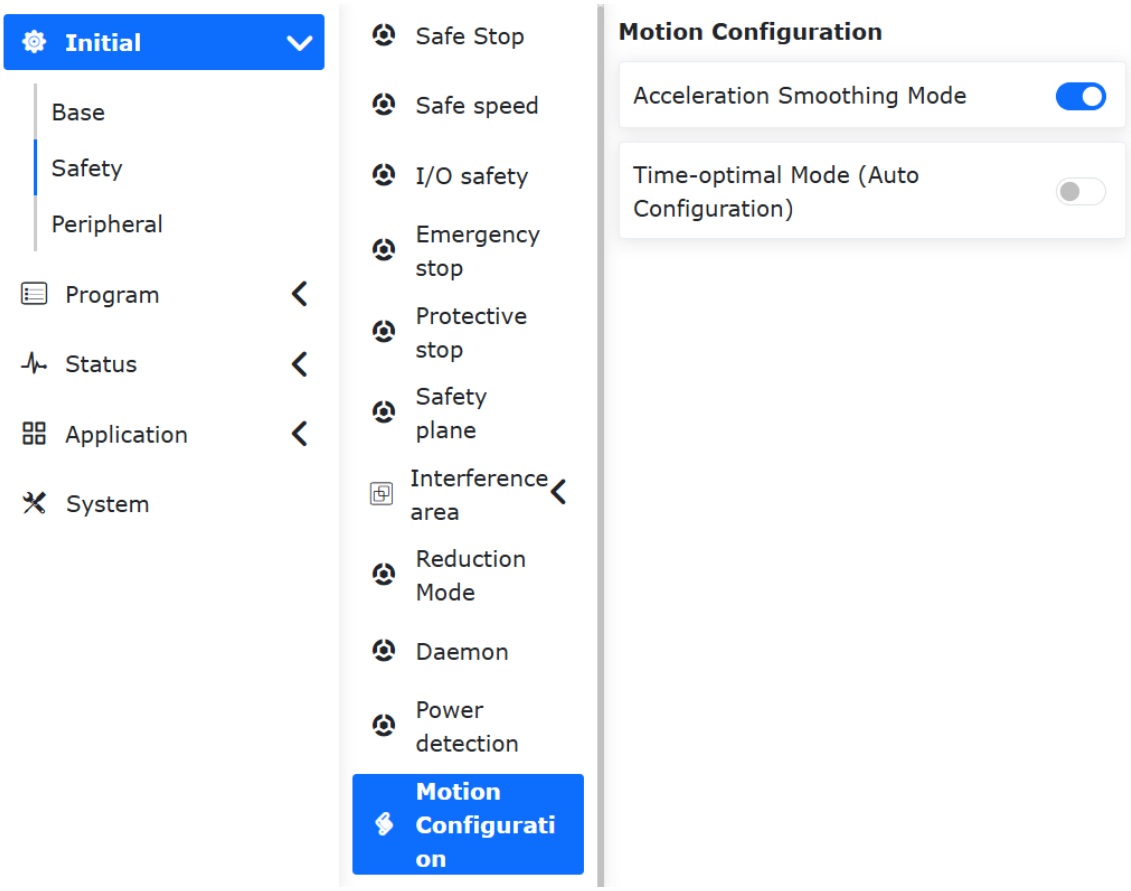


Figure 9.27-73 Acceleration Smooth Mode Switch Setup

Step 2: Select “A0” to “A12” as teaching points.

Step 3: Set Full Circle instruction with “Smooth Transition”, no motion protection.

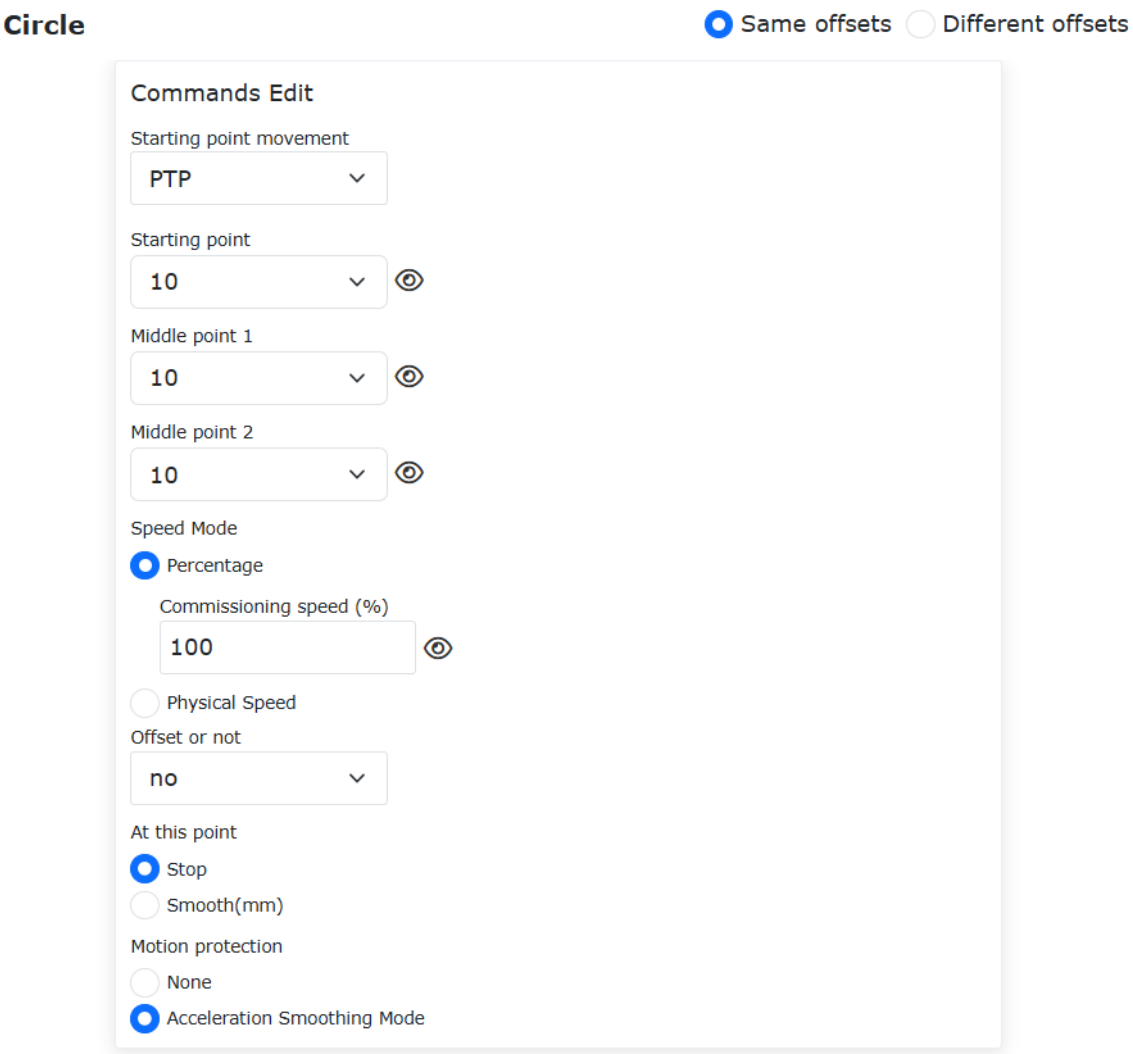


Figure 9.27-74 Blending Settings for Conventional CIRCLE

Step 4: Generate program for all-instruction optimized velocity.

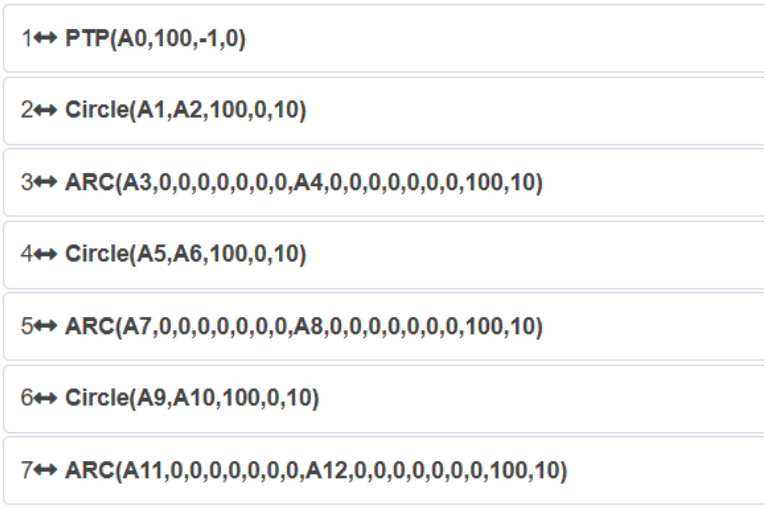


Figure 9.27-75 CIRCLE-ARC Blending via Configuration Switch

1.3.9.27.2.46 CIRCLE-CIRCLE Blending

1.3.9.27.2.47 Using Lua Instruction Mode

Step 1: Select “A0” to “A12” for CIRCLE-CIRCLE blending.

Step 2: Configure Full Circle instruction with acceleration smooth mode and “Smooth Transition”.

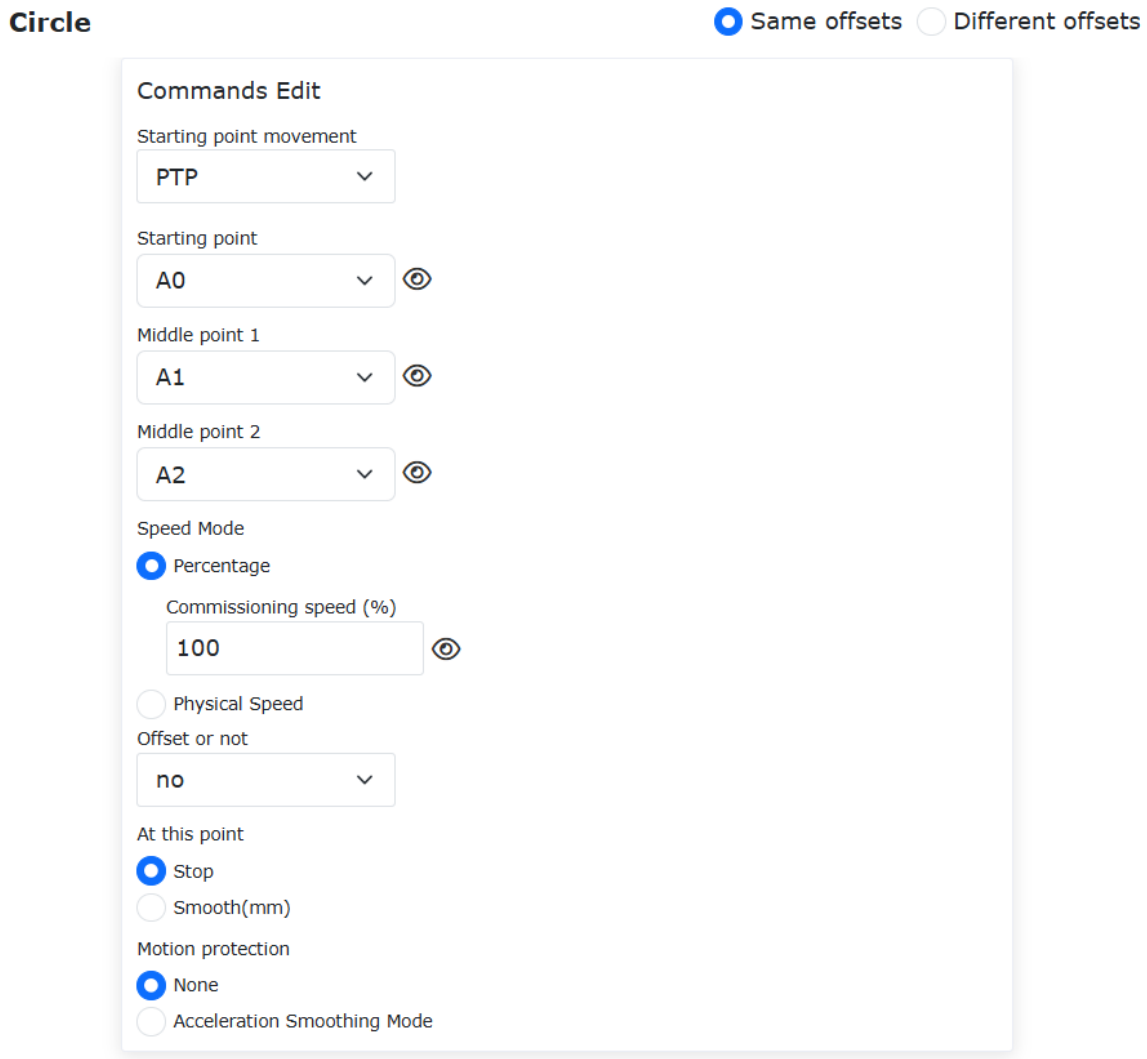


Figure 9.27-76 Blending Settings for Accelerated Smooth CIRCLE

Step 3: Add multiple CIRCLE instructions within AccSmoothStart/End.



Figure 9.27-77 Typical CIRCLE-CIRCLE Blending in Lua Mode

1.3.9.27.2.48 Using Motion Configuration Switch Mode

Step 1: Enable acceleration smooth mode switch.

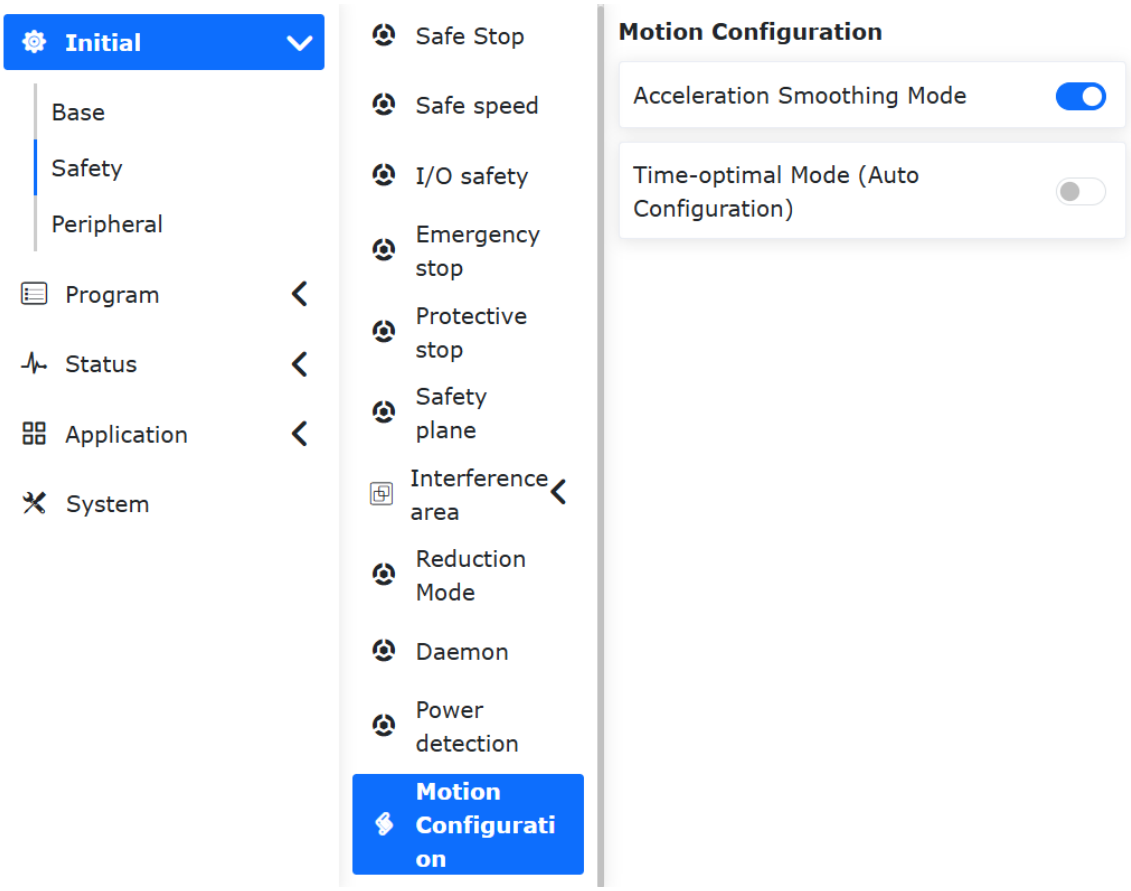


Figure 9.27-78 Acceleration Smooth Mode Configuration

Step 2: Choose “A0” to “A12” as teaching points.

Step 3: Set Full Circle instruction with “Smooth Transition”, no motion protection.

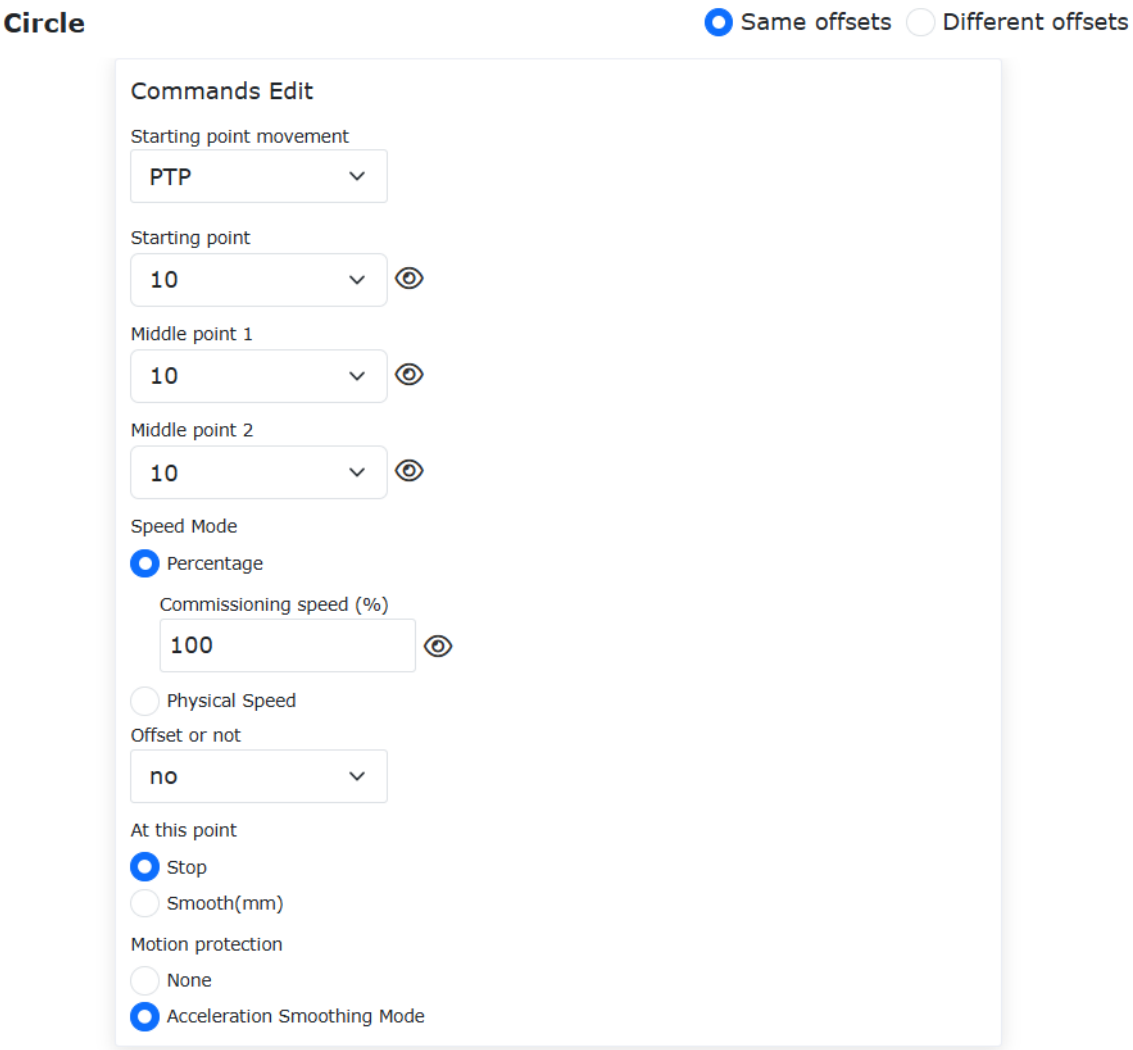


Figure 9.27-79 Blending Settings for Conventional CIRCLE

Step 4: Generate program for optimized velocity across all instructions.

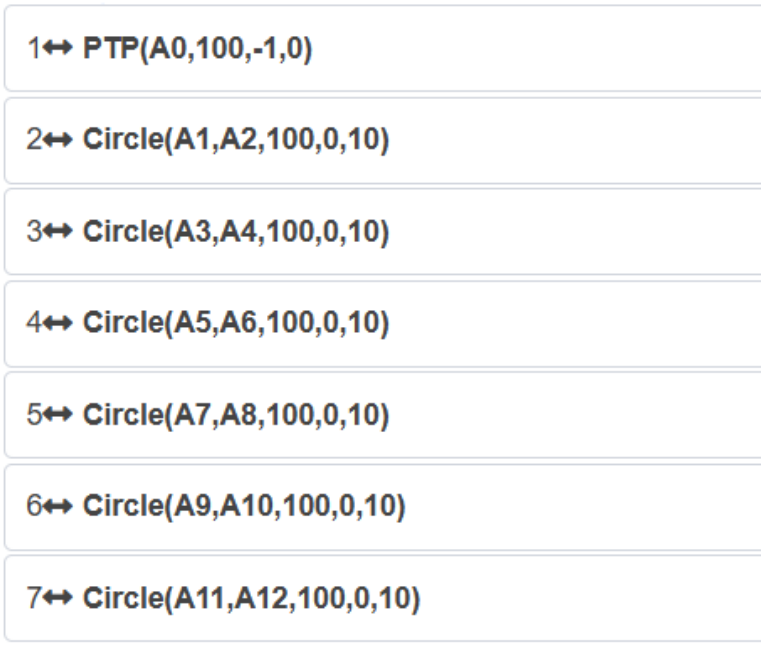


Figure 9.27-80 CIRCLE-CIRCLE Blending via Configuration Switch

1.3.9.27.2.49 Blending for Extended Axis Asynchronous Motion

1.3.9.27.2.50 Using Lua Command Method

Step1: Select the taught points for extended axis asynchronous motion blending. This manual uses "A0" to "A5" as the taught point names.

Step2: Click "Teach Program" → "Program Editing", select the "Extended Axis" command under "Peripheral Commands", set "Motion Mode" to "Asynchronous", select the taught points and configure the debug speed. For motion protection, choose "Acceleration Smoothing Mode" and set the "Smooth Transition" parameter at the points requiring smoothing.

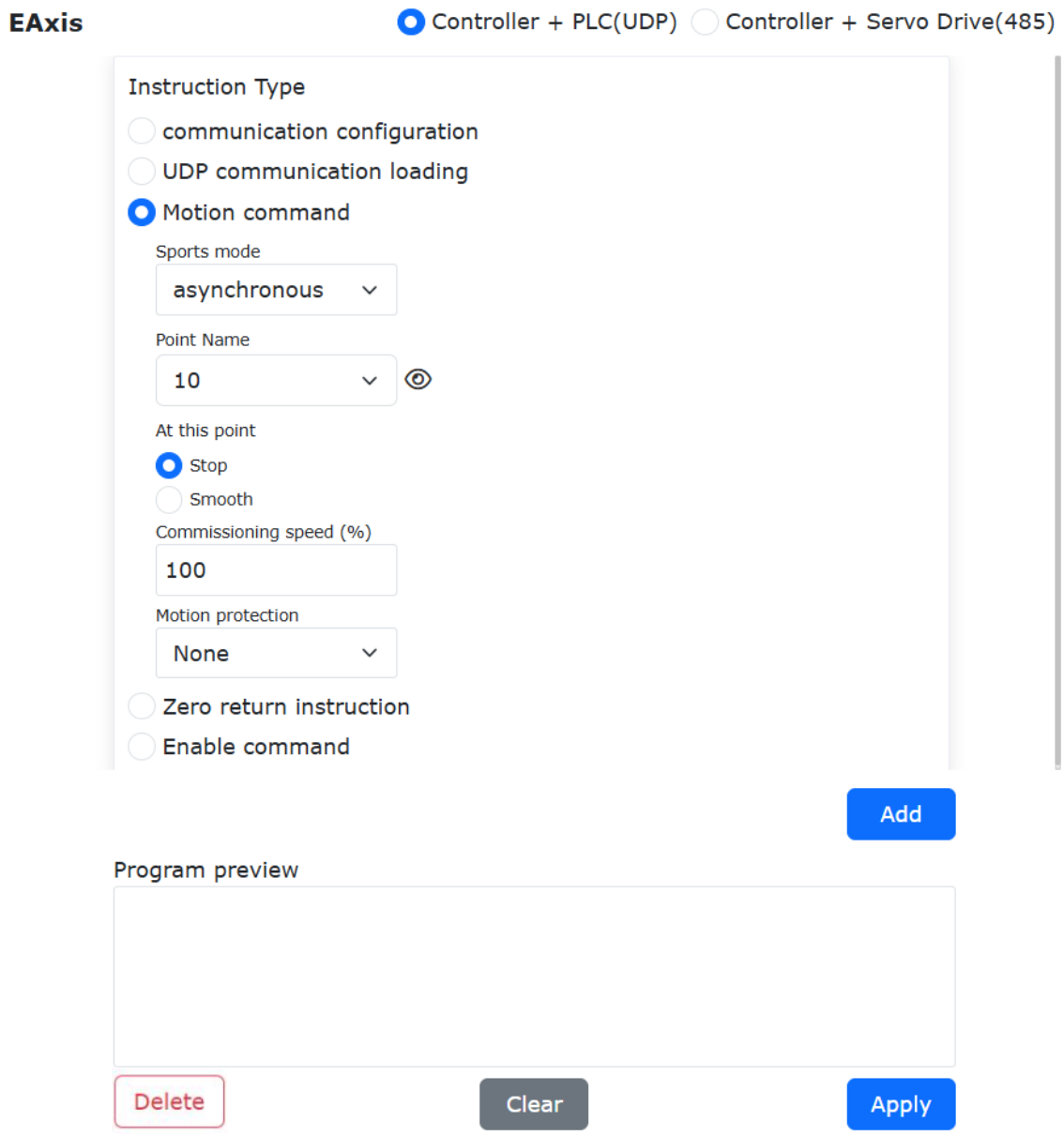


Figure 9.27-81 Extended Axis Asynchronous Motion Blending Command Settings

Step3: Add motion commands, generate and run the Lua program to achieve blending for extended axis asynchronous motion. This method only applies S-curve velocity planning and blending between *AccSmoothStart()* and *AccSmoothEnd()*, while using T-curve planning for other commands.

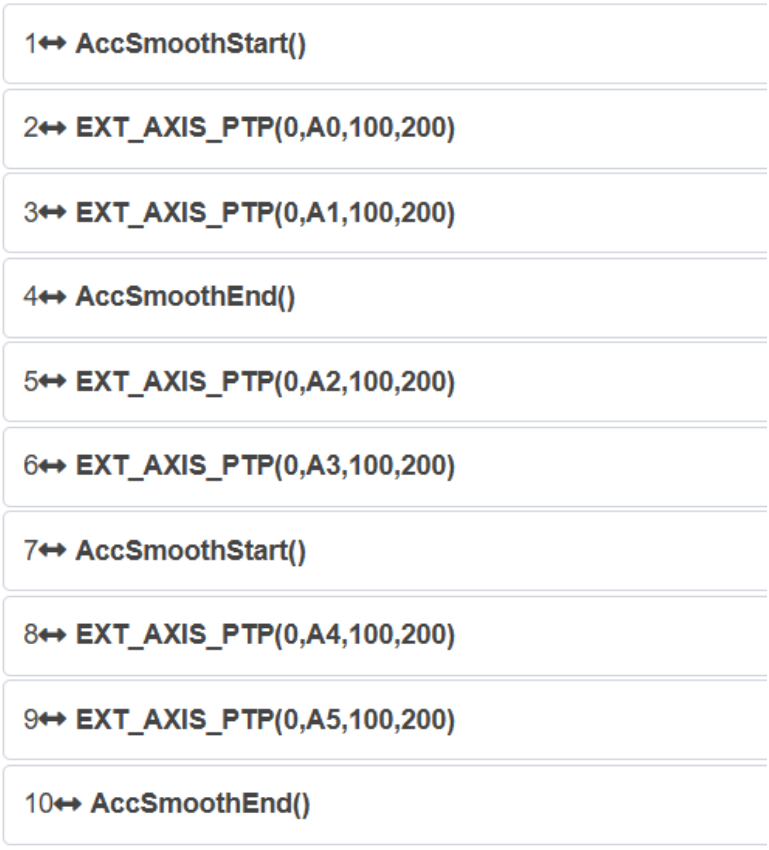


Figure 9.27-82 Typical Lua Program for Extended Axis Asynchronous Motion Blending

1.3.9.27.2.51 Using Motion Configuration Switch

Step1: Click "Initial Setup" → "Safety" → "Motion Configuration" and enable the "Acceleration Smoothing Mode" switch.

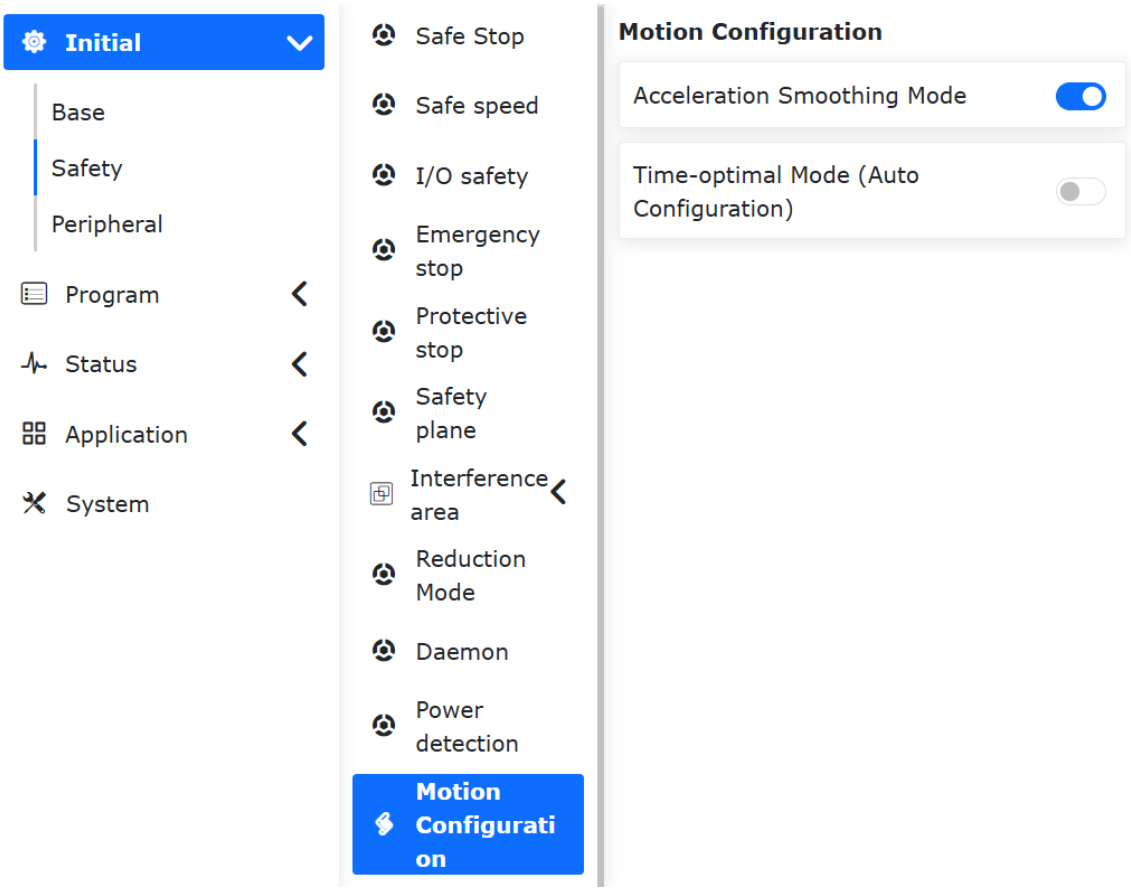


Figure 9.27-83 Acceleration Smoothing Mode Configuration Switch

Step2: Select the taught points for extended axis asynchronous motion blending. This manual uses "A0" to "A5" as the taught point names.

Step3: Click "Teach Program" → "Program Editing", select the "Extended Axis" command under "Peripheral Commands", set "Motion Mode" to "Asynchronous", select the taught points and configure the debug speed. For motion protection, choose "None" and set the "Smooth Transition" parameter at the points requiring smoothing.

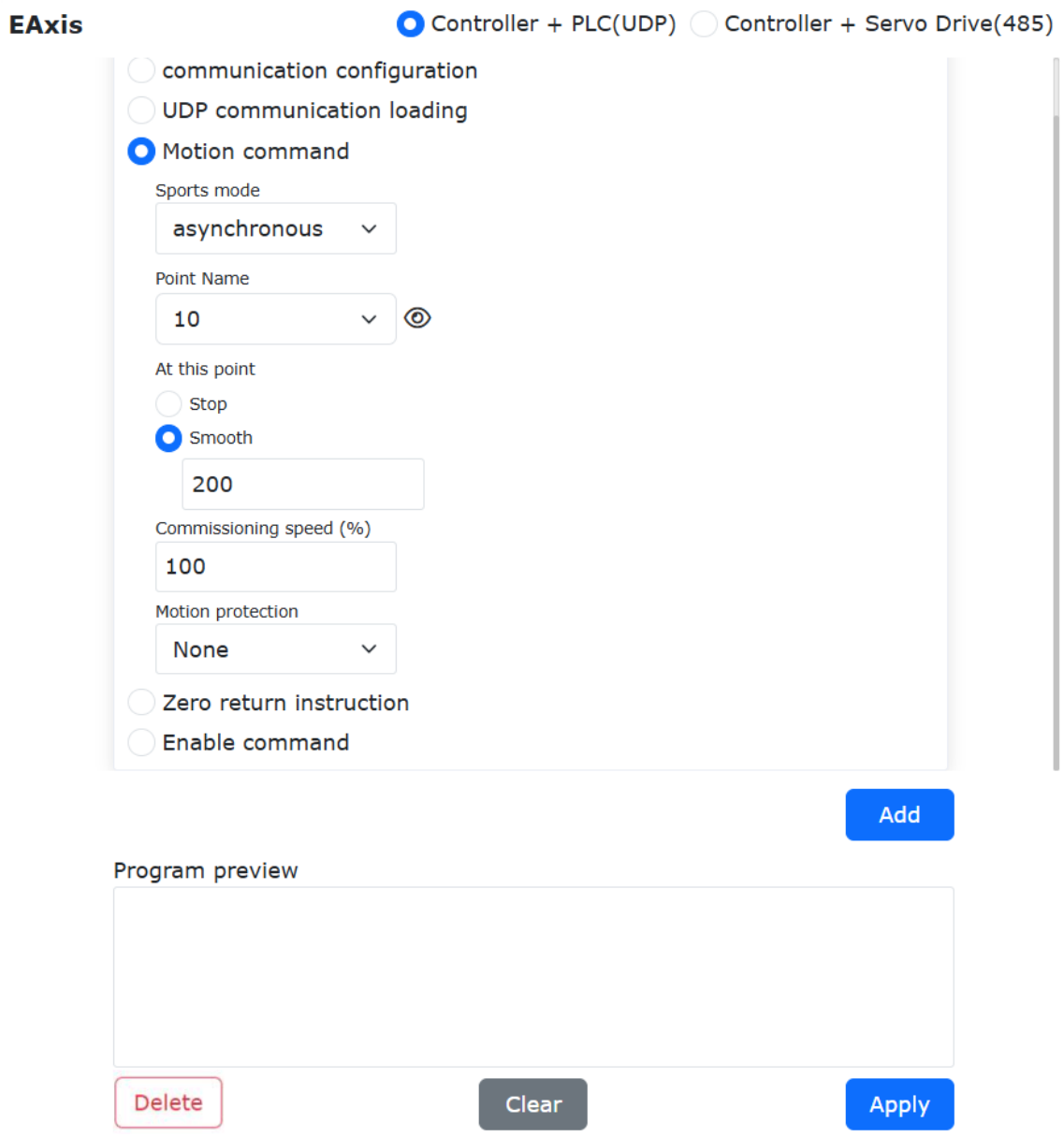


Figure 9.27-84 Standard Extended Axis Asynchronous Motion Blending Command Settings

Step4: Add multiple motion commands, generate and run the Lua program to achieve blending for extended axis asynchronous motion. The typical program is the same as standard extended axis motion programs. This method applies S-curve velocity planning and blending to all motions.

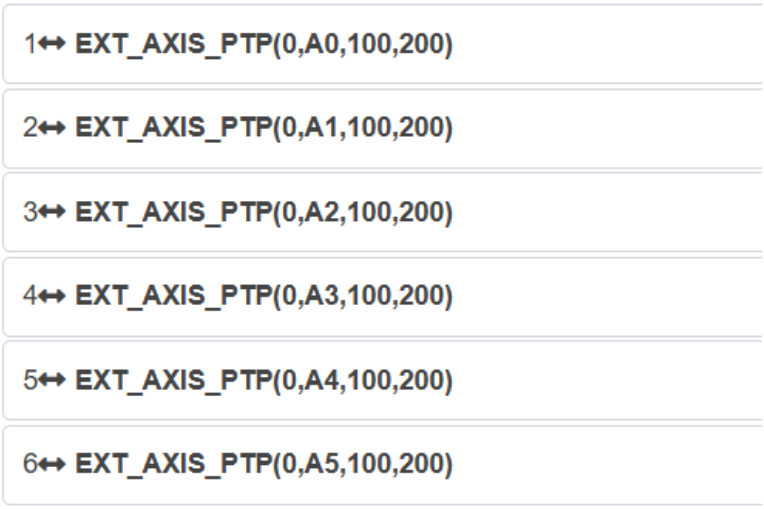


Figure 9.27-85 Typical Program for Extended Axis Asynchronous Motion Blending Using Configuration Switch

1.3.9.27.2.52 Blending for Extended Axis Synchronous Motion

1.3.9.27.2.53 Using Lua Command Method

Step1: Select the taught points for extended axis synchronous motion blending. This manual uses “A0” to “A5” as the taught point names.

Step2: Click “Teach Program” → “Program Editing”, select the “Extended Axis” command under “Peripheral Commands”, set “Motion Mode” to “Synchronous”, select the taught points and configure the debug speed. For motion protection, choose “Acceleration Smoothing Mode” and set the “Smooth Transition” parameter at the points requiring smoothing.

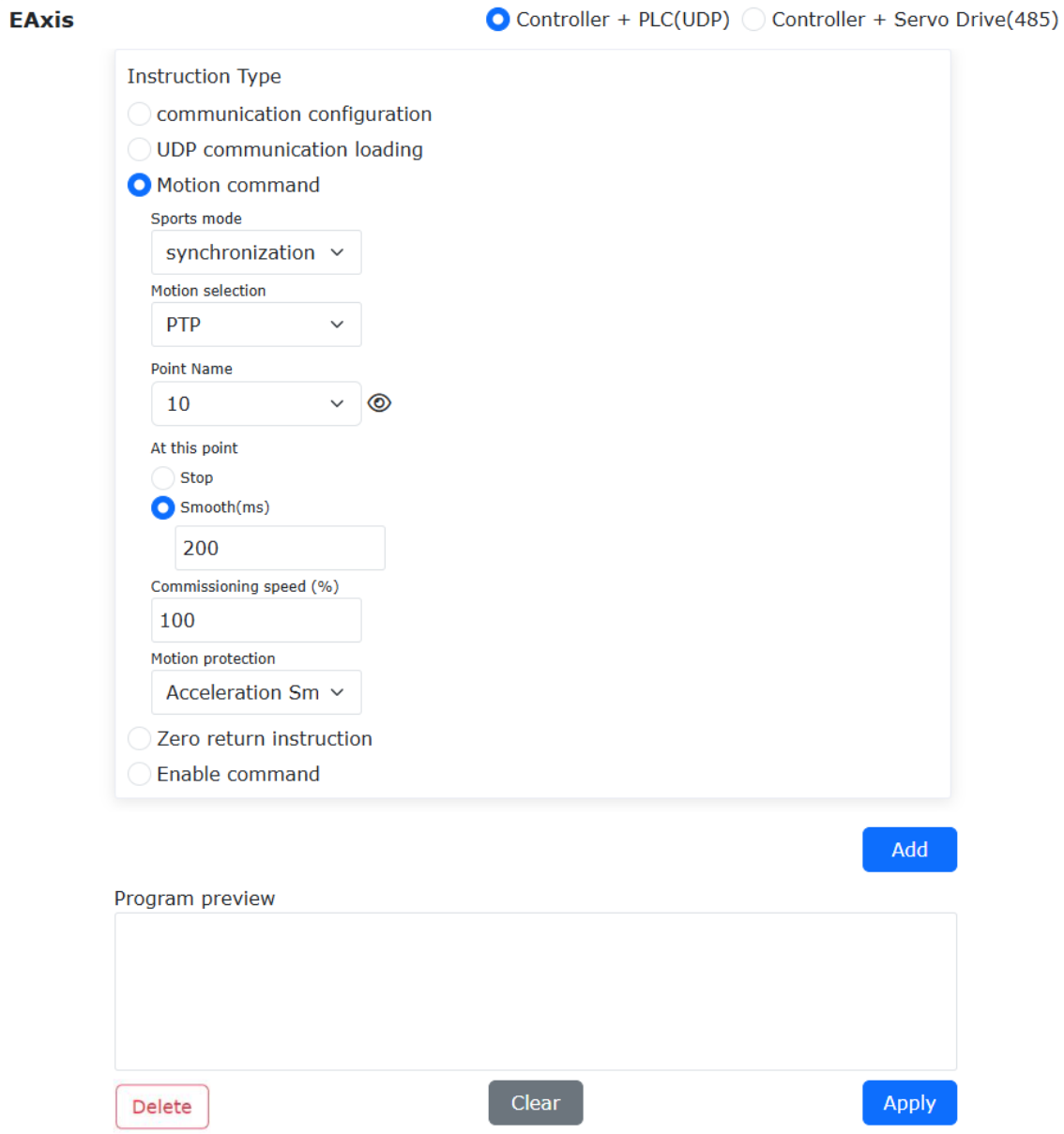


Figure 9.27-86 Extended Axis Synchronous Motion Blending Command Settings

Step3: Add motion commands, generate and run the Lua program to achieve blending for extended axis synchronous motion. This method only applies S-curve velocity planning and blending between *AccSmoothStart()* and *AccSmoothEnd()*, while using T-curve planning for other commands.



Figure 9.27-87 Typical Lua Program for Extended Axis Synchronous Motion Blending

1.3.9.27.2.54 Using Motion Configuration Switch

Step1: Click “Initial Setup” → “Safety” → “Motion Configuration” and enable the “Acceleration Smoothing Mode” switch.

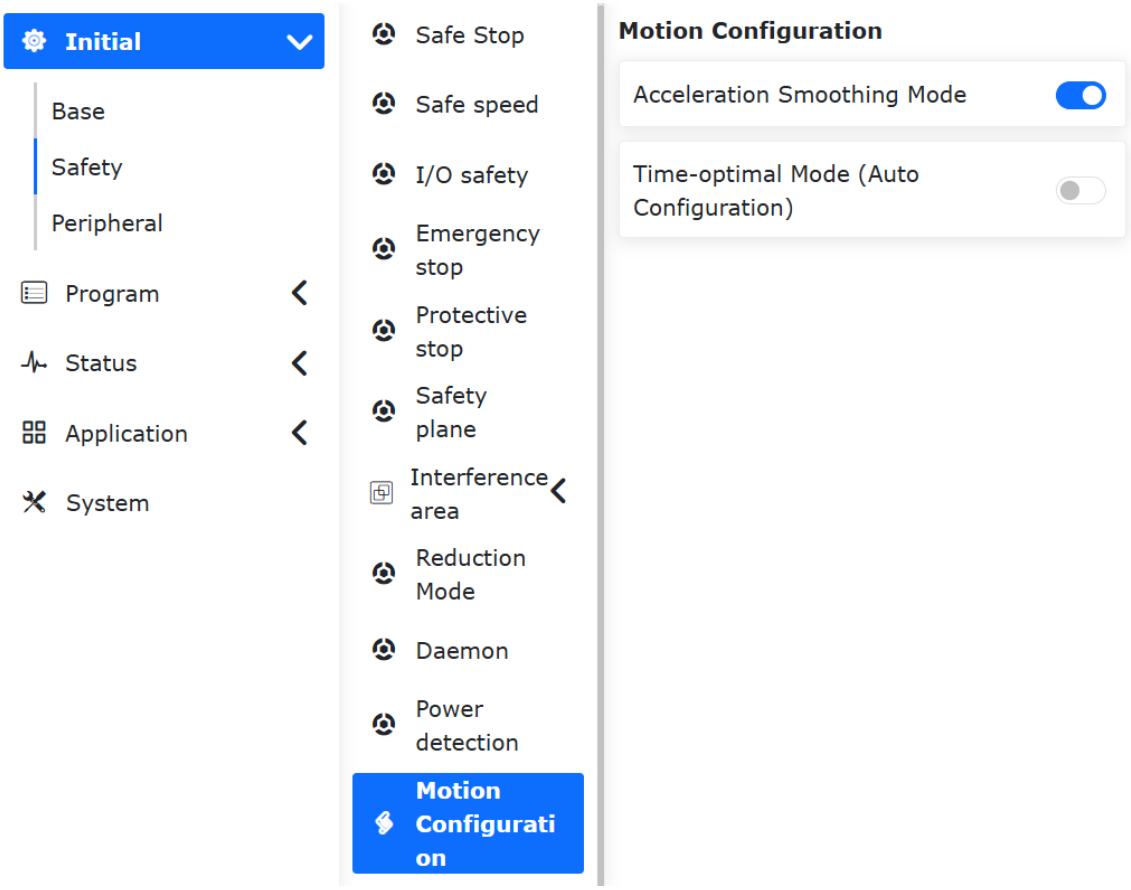


Figure 9.27-88 Acceleration Smoothing Mode Configuration Switch

Step2: Select the taught points for extended axis synchronous motion blending. This manual uses “A0” to “A5” as the taught point names.

Step3: Click “Teach Program” → “Program Editing”, select the “Extended Axis” command under “Peripheral Commands”, set “Motion Mode” to “Synchronous”, select the taught points and configure the debug speed. For motion protection, choose “None” and set the “Smooth Transition” parameter at the points requiring smoothing.

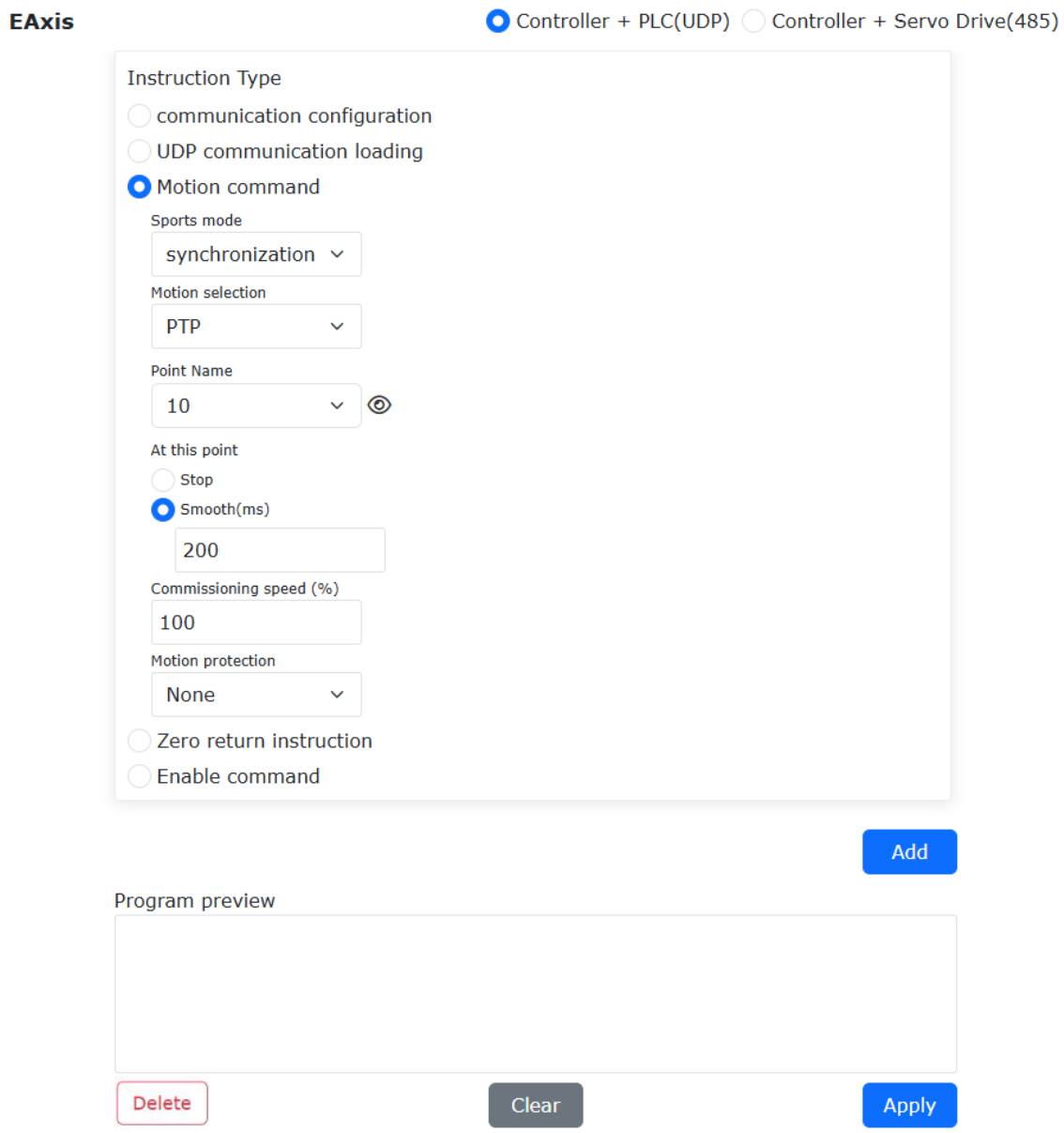


Figure 9.27-89 Standard Extended Axis Synchronous Motion Blending Command Settings

Step4: Add multiple motion commands, generate and run the Lua program to achieve blending for extended axis synchronous motion. The typical program is the same as standard extended axis motion programs. This method applies S-curve velocity planning and blending to all motions.



Figure 9.27-90 Typical Program for Extended Axis Synchronous Motion Blending Using Configuration Switch

1.3.9.28 Swing Tilt Angle Function

1.3.9.28.1 Overview

The robot swing tilt angle function allows the robot's end tool to customize the swing angle around the Rx direction of the swing coordinate system during swinging motion, reducing the difference in contact length between the fillet weld and the connected materials on both sides in operations such as lap welding.

1.3.9.28.2 Operation Procedure

On the robot's web control interface, click "Teaching Program" -> "Program Editing" to enter the "Motion Command" interface, as shown below.

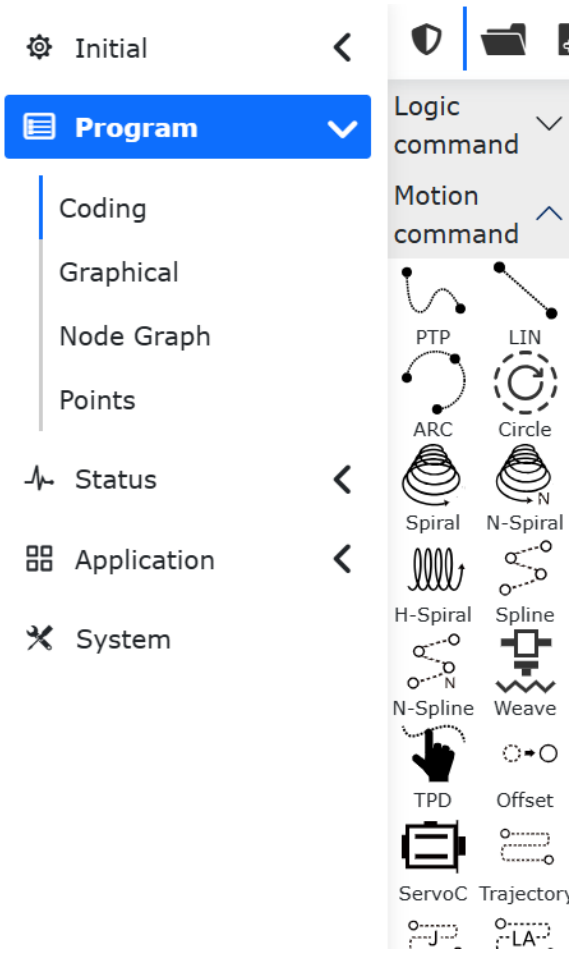


Figure 9.28-1 Motion Command Interface

In the “Motion Command” interface, click “Weave” to enter the “Weave” command editing interface.

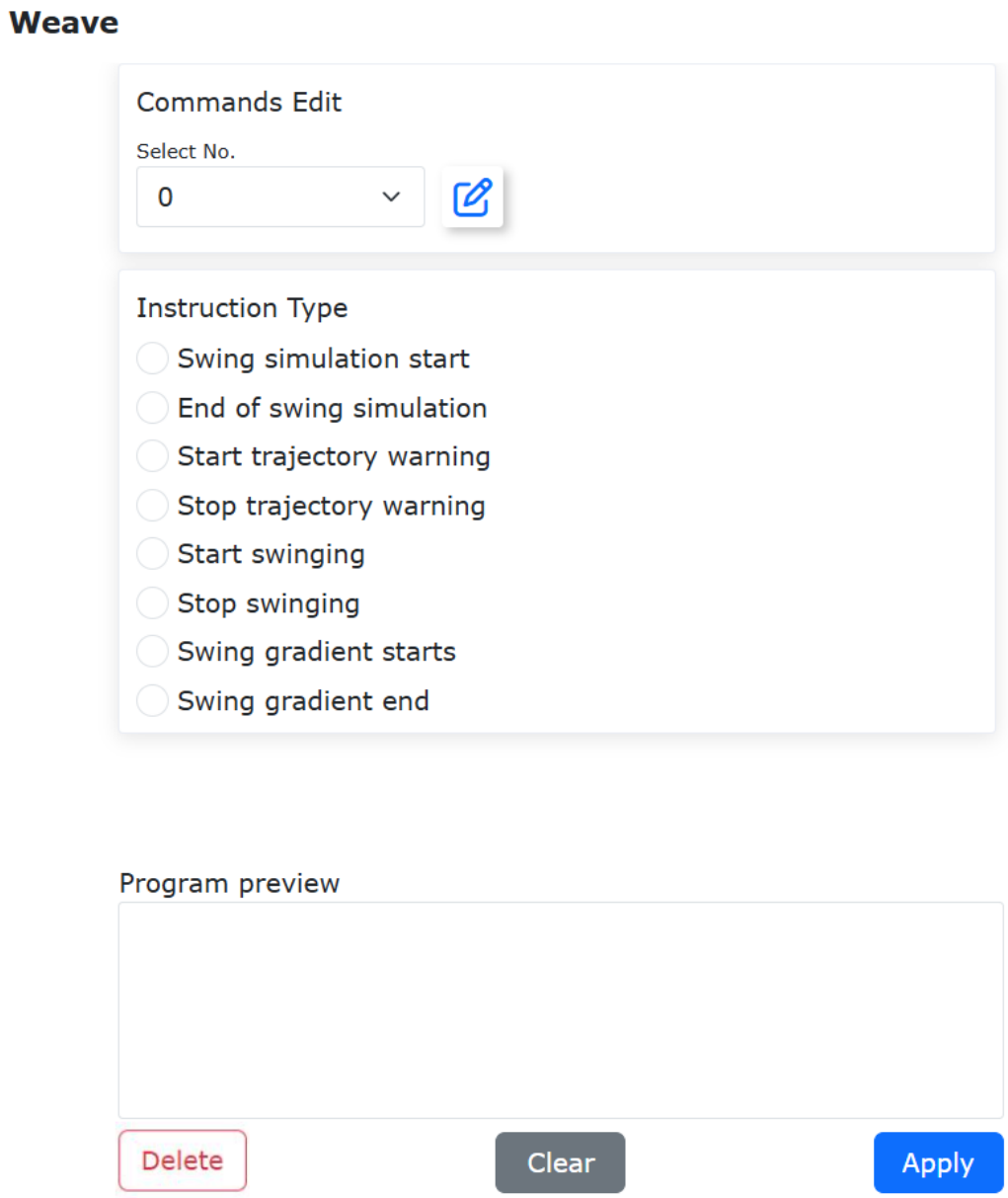


Figure 9.28-2 Weave Command Editing Interface

In the "Weave" command editing interface, click the "Selection Number" dropdown to choose different swing parameter configurations. Click the button next to the dropdown to modify the swing parameters under the selected number.

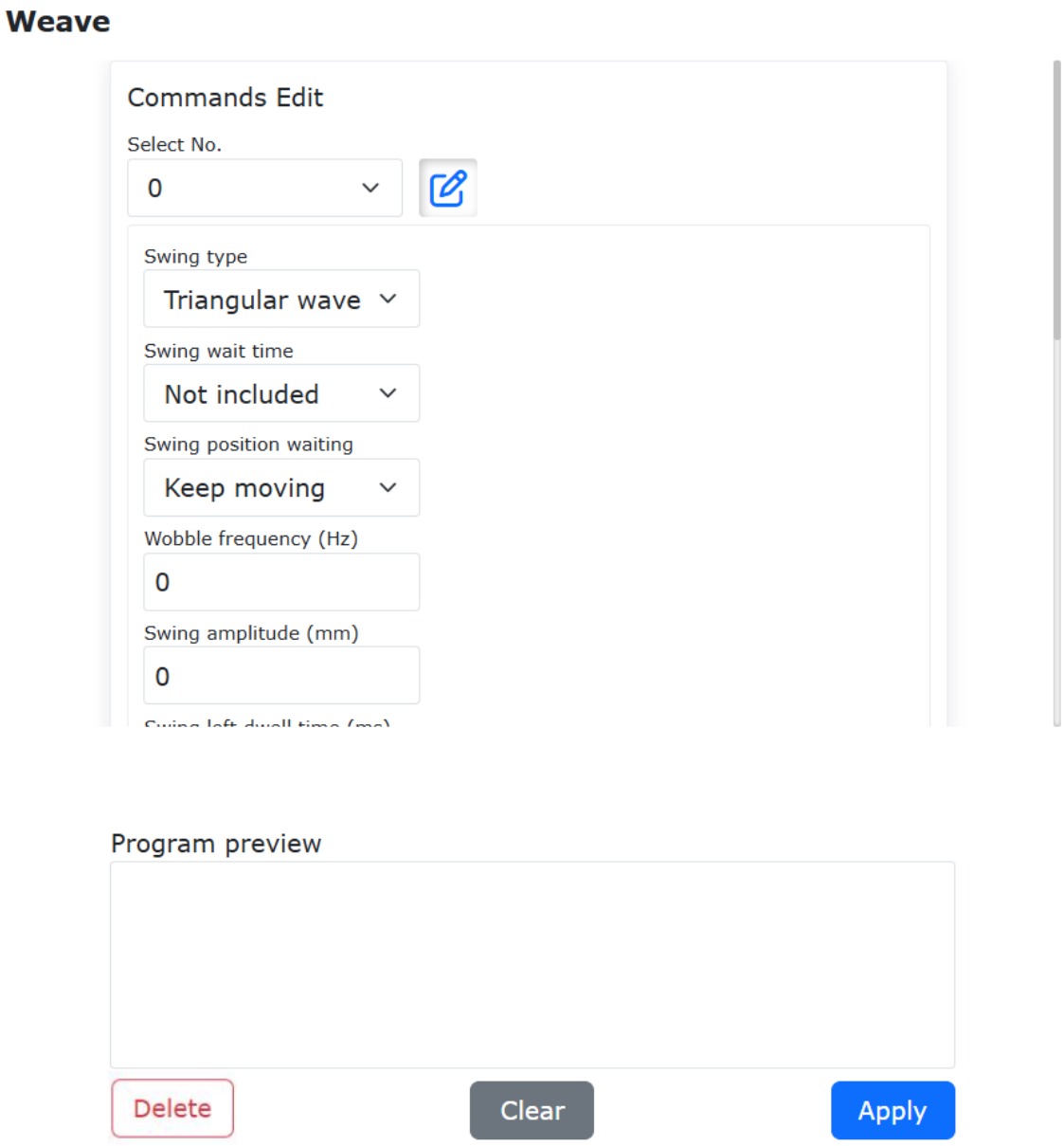


Figure 9.28-3 Swing Parameter Configuration

In the swing parameter configuration, enter a custom rotation angle around the Rx direction of the swing coordinate system in the "Swing Direction Tilt Angle" field, then click "Configure" to complete the setup.

Note: Note: The "Swing Direction Tilt Angle" parameter is applicable to "Triangle Wave Swing," "Sine Wave Swing," "Circular Swing - Clockwise," and "Circular Swing - Counterclockwise" in the "Swing Type" parameter.

The following example demonstrates the swing tilt angle function using Lin motion:

Step 1: In the "Weave" command editing interface, select a pre-configured swing parameter set from the "Selection Number" dropdown. Under "Command Type," click "Start Weave," then click "Add" to enable the swing function.

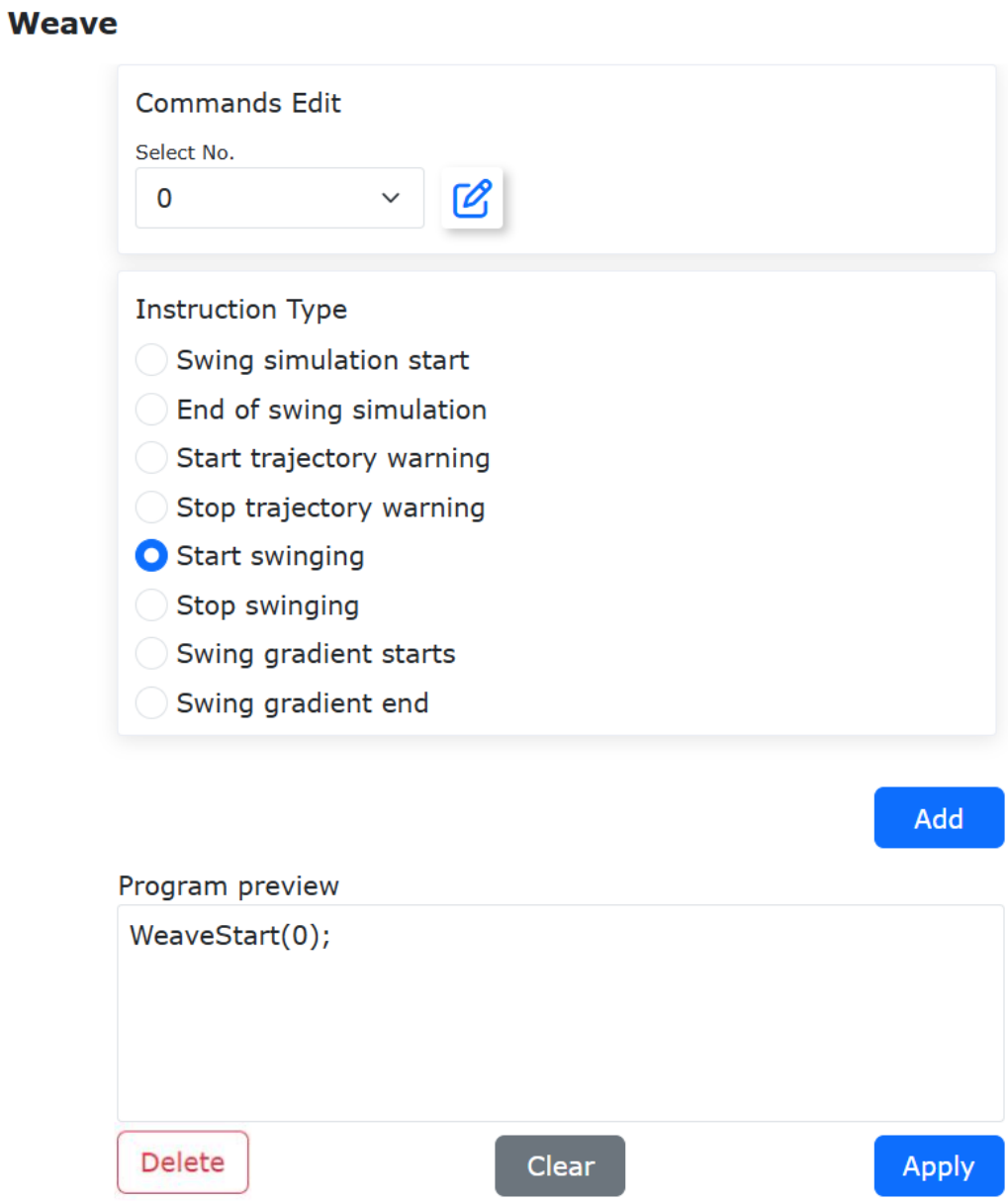


Figure 9.28-4 Adding Start Weave

Step 2: In the “Motion Command” interface, click “Linear” to create a Lin linear motion (basic motion command, not detailed here).

Step 3: In the “Weave” command editing interface, under “Command Type,” click “Stop Weave,” then click “Add” to disable the swing function.

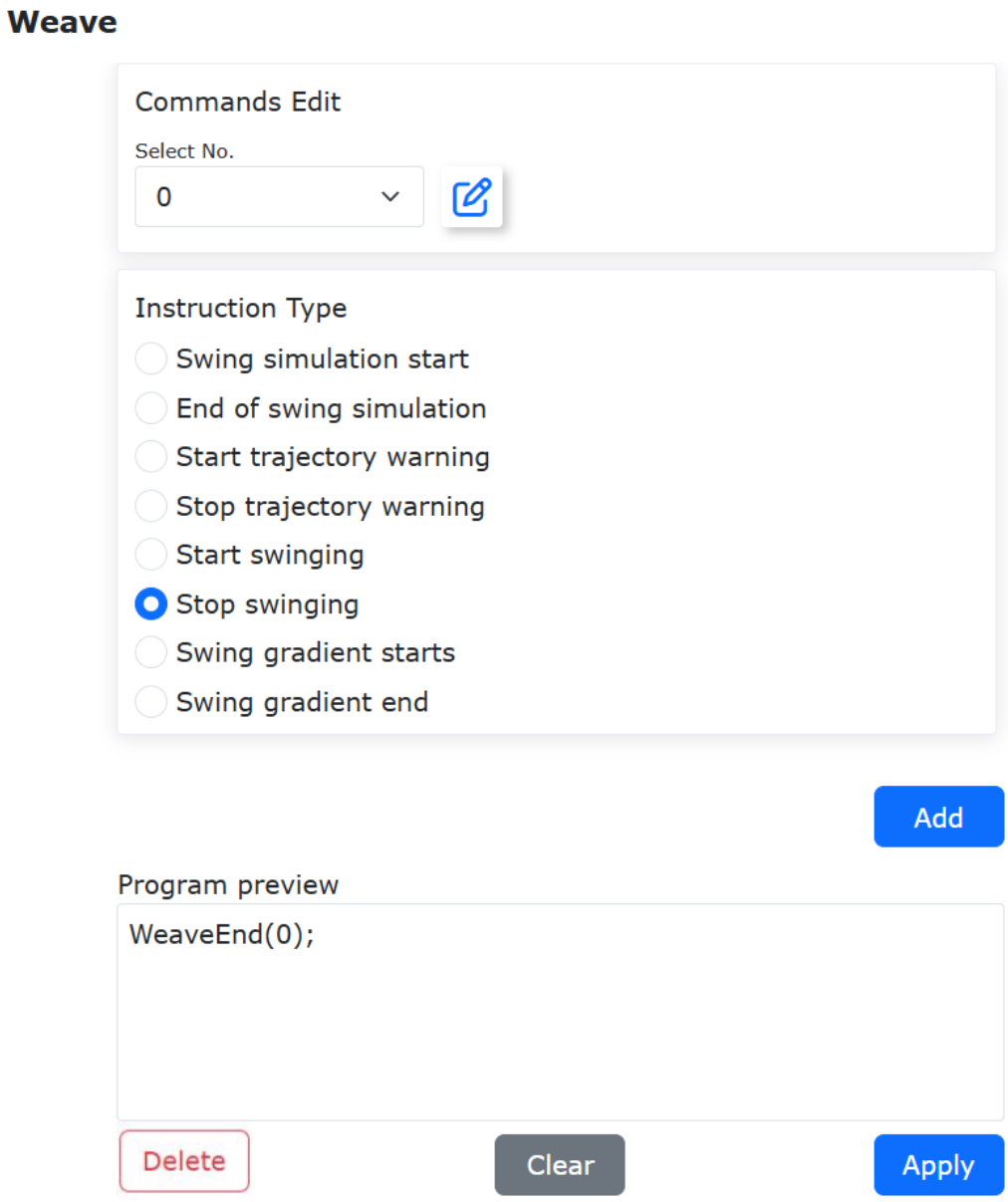


Figure 9.28-5 Adding Stop Weave

Step 4: After completing Steps 1–3, check the correctness of the settings in the "Program Preview" section of the "Weave" command editing interface.

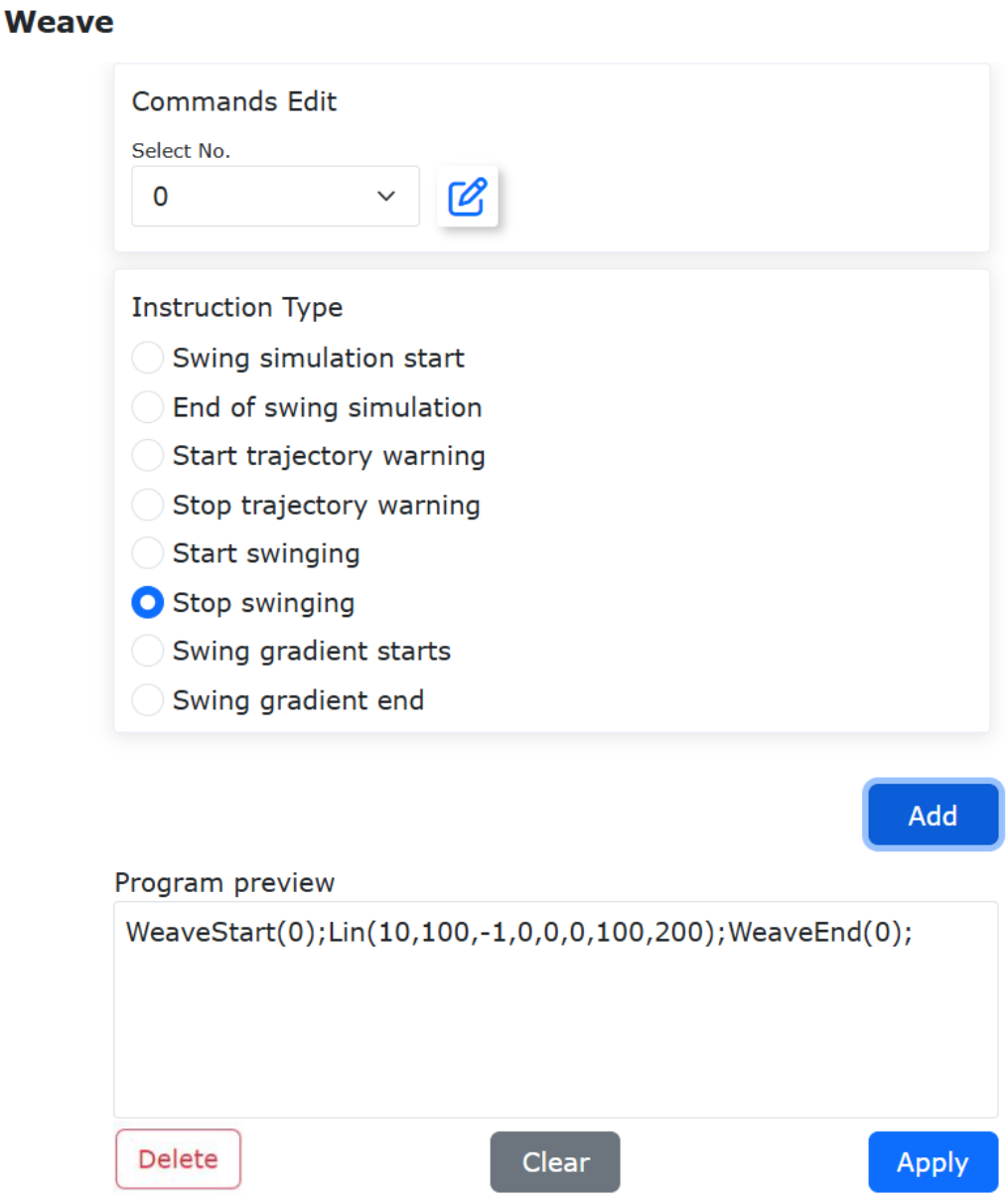


Figure 9.28-6 Swing Program Preview

Step 5: After verifying the program in the "Program Preview" section, click "Apply" to automatically generate executable LUA code.

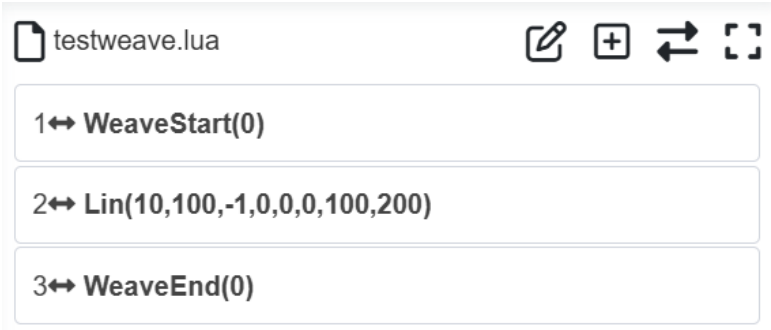


Figure 9.28-7 Example LUA Swing Motion Program

1.3.9.29 Welding process parameter gradient function (current, voltage, travel speed along weld seam)

1.3.9.29.1 Overview

The welding parameter gradual change function (current, voltage, and travel speed along the weld seam) supports customizing the variation range of process parameters during welding.

1.3.9.29.2 Current/Voltage Parameter Gradual Change Process

1.3.9.29.2.1 Current Parameter Gradual Change

On the robot web control interface, navigate to “Teach Program” -> “Program Editing” to enter the “Welding Command” interface as shown below.

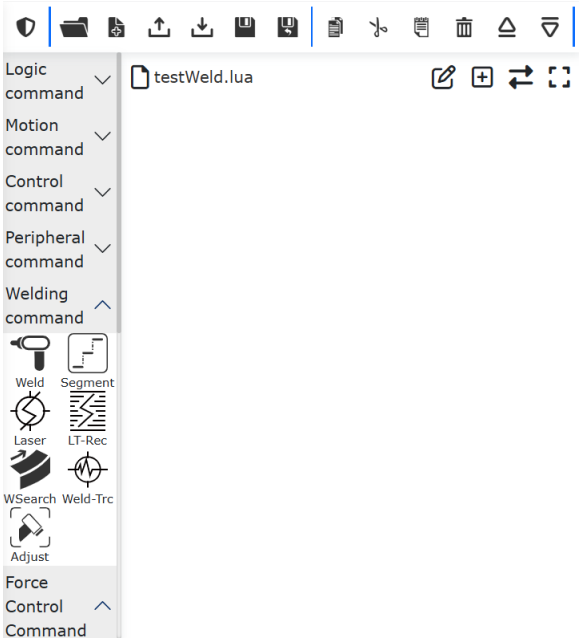


Figure 9.29-1 Welding Command Interface

In the “Welding Command” interface, click “Weld” to enter the “Weld” command configuration interface.

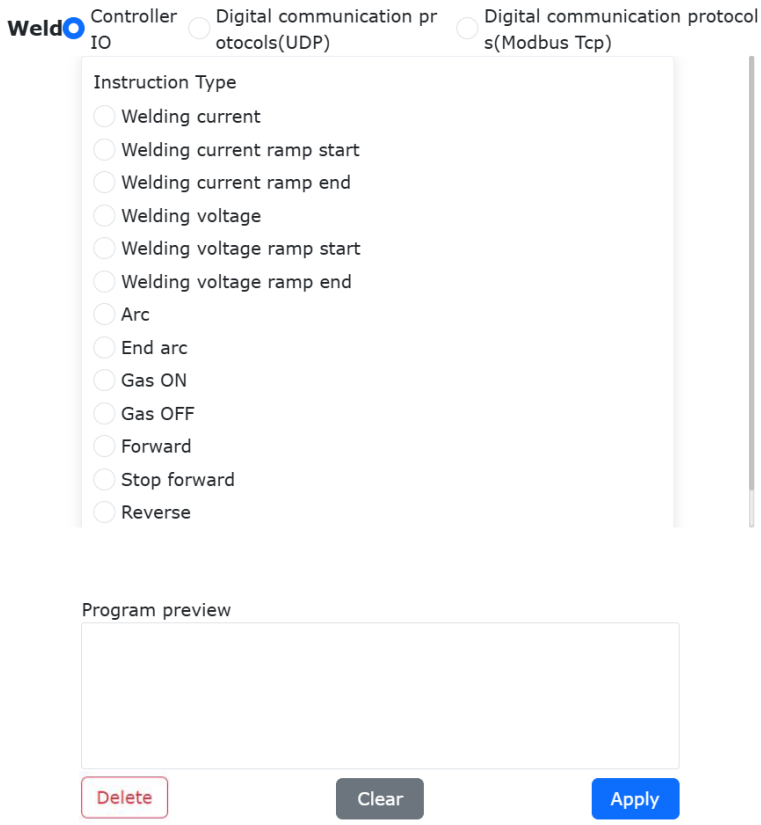


Figure 9.29-2 Weld Command Configuration Interface

In the “Command Type” section of the “Weld” command configuration interface, click “Welding Current Gradual Start” to configure parameters including “Start Current”, “End Current”, “Welding Current Control AO”, and “Smoothing Option”.

For example, configure “Start Current” as 260 A, “End Current” as 220 A, “Welding Current Control AO” as “Ctrl-AO0” control box analog channel, and “Smoothing Option” as “Break”. Click “Add” to complete configuration, then verify parameters in the “Program Preview” section for any errors.

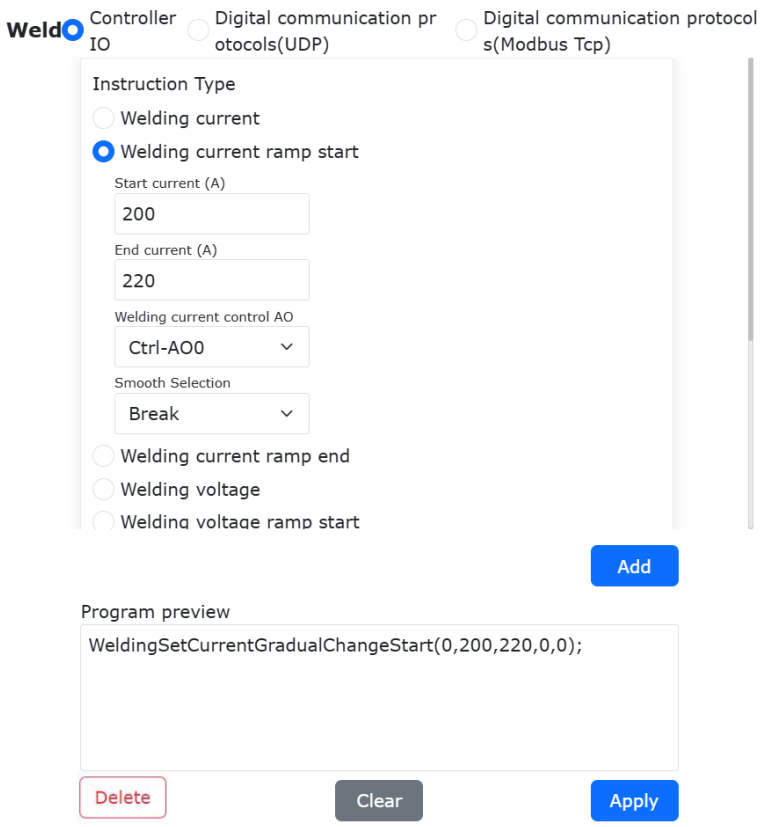


Figure 9.29-3 Welding Current Gradual Start Command Parameters

In the “Command Type” section of the “Weld” command configuration interface, click “Welding Current Gradual End”. No parameters need to be configured. Click “Add” to complete, then verify in “Program Preview”.

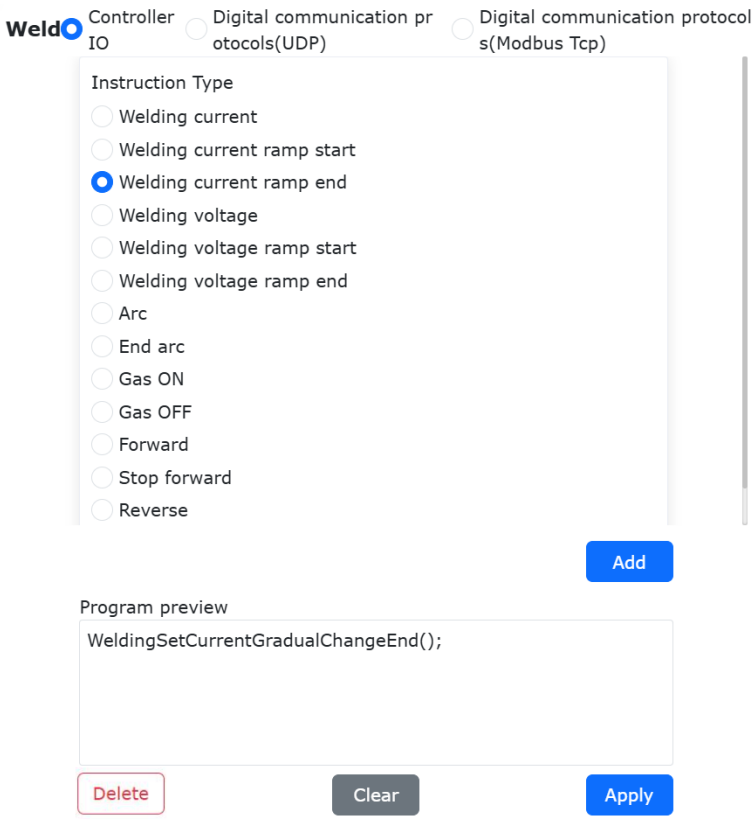


Figure 9.29-4 Welding Current Gradual End Command Parameters

After configuring both start and end commands, click “Apply” to automatically generate executable LUA program.

Note: During configuration, motion commands must be included. A typical LUA program combining arc tracking motion with current gradual change is shown below.

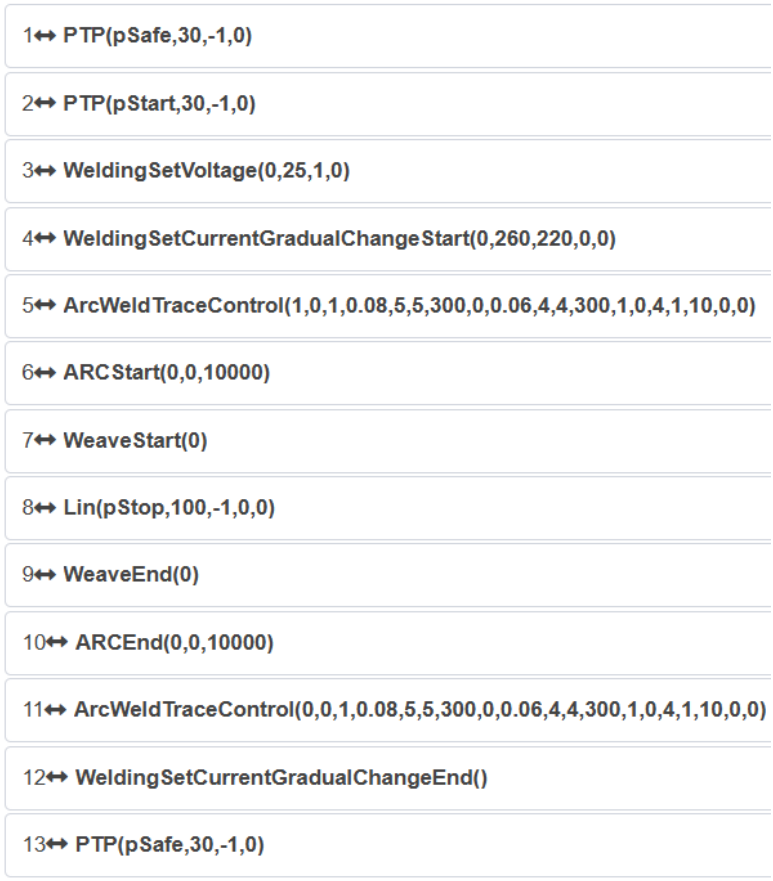


Figure 9.29-5 Typical Current Gradual Change Arc Tracking LUA Program

1.3.9.29.2.2 Voltage Parameter Gradual Change

In the “Command Type” section of the “Weld” command configuration interface, click “Welding Voltage Gradual Start” to configure parameters including “Start Voltage”, “End Voltage”, “Welding Voltage Control AO”, and “Smoothing Option”.

For example, configure “Start Voltage” as 25 V, “End Voltage” as 22 V, “Welding Voltage Control AO” as “Ctrl-AO1” control box analog channel, and “Smoothing Option” as “Break”. Click “Add” to complete configuration, then verify in “Program Preview”.

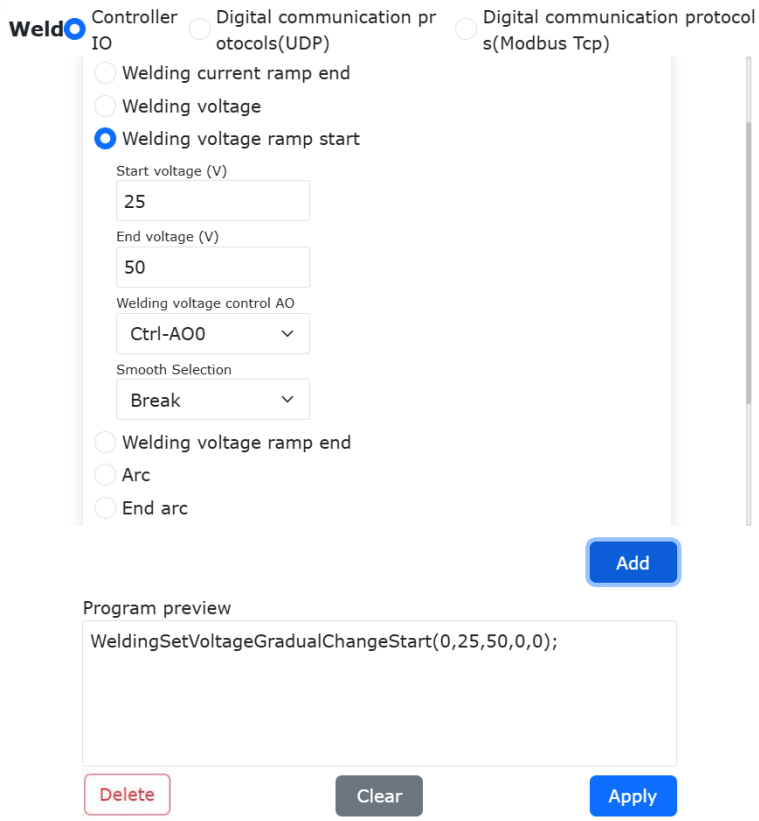


Figure 9.29-6 Welding Voltage Gradual Start Command Parameters

In the “Command Type” section, click “Welding Voltage Gradual End”. No parameters need configuration. Click “Add” then verify.

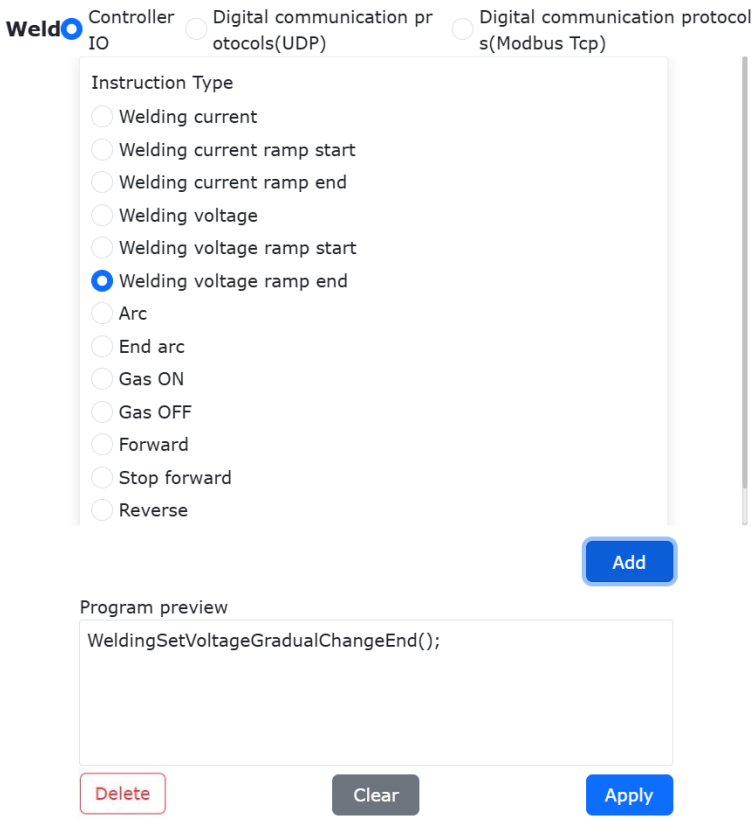


Figure 9.29-7 Welding Voltage Gradual End Command Parameters

After configuration, click “Apply” to generate LUA program.

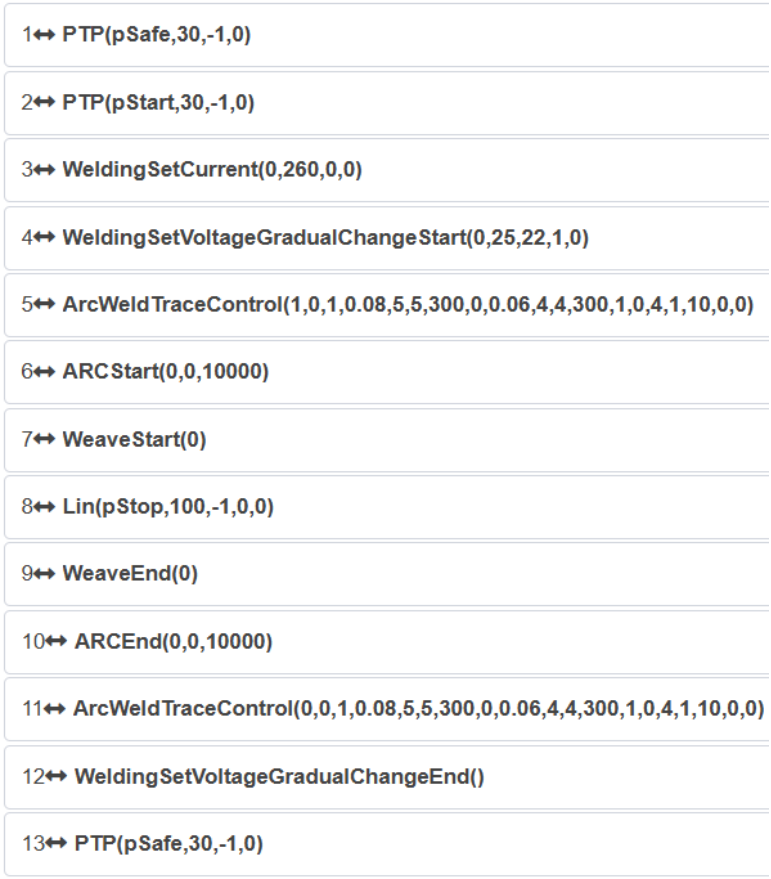


Figure 9.29-8 Typical Voltage Gradual Change Arc Tracking LUA Program

1.3.9.29.3 Travel Speed Gradual Change Process

Navigate to “Teach Program” -> “Program Editing” -> “Motion Command” interface.

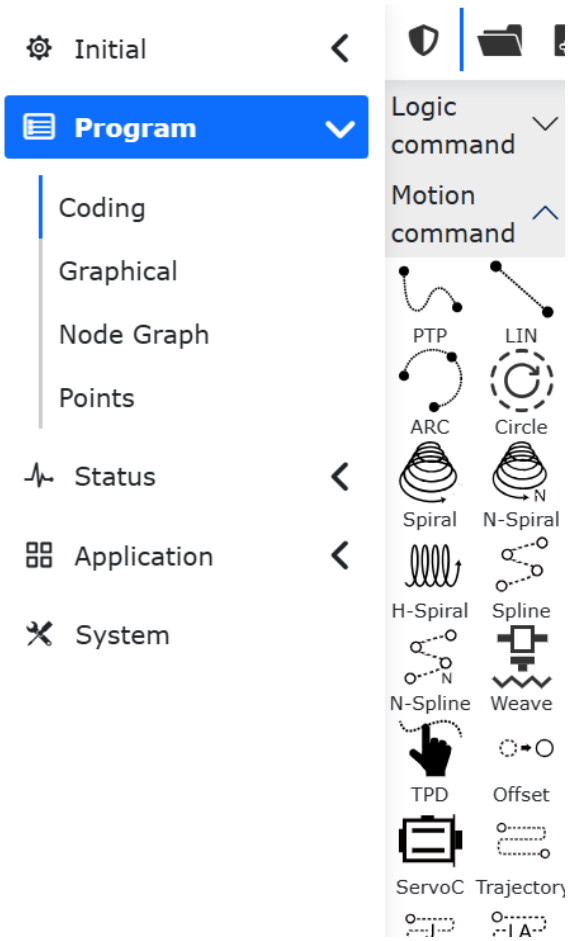


Figure 9.29-9 Motion Command Interface

Click “Weave” to enter configuration interface.

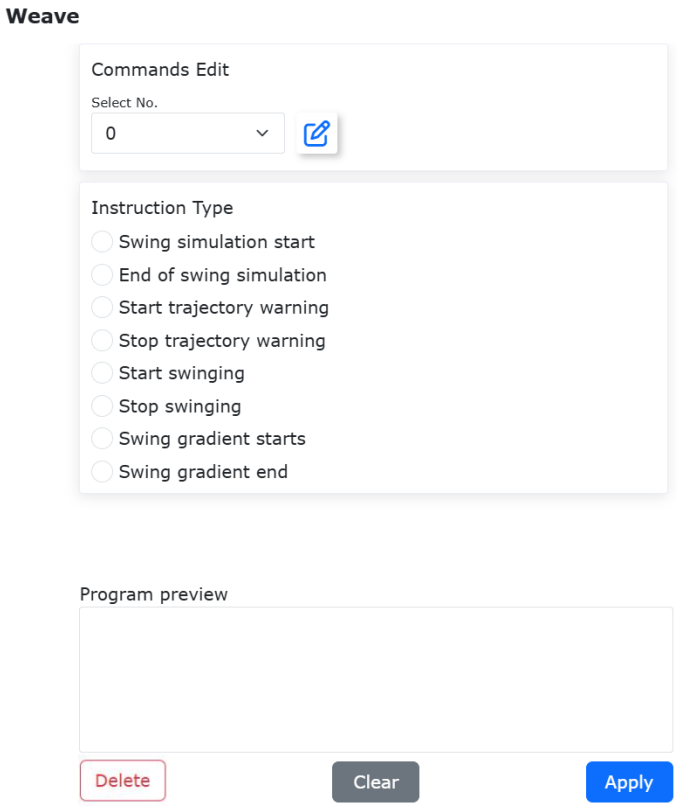


Figure 9.29-10 Weave Command Configuration Interface

In “Command Type”, click “Weave Gradual Start” to configure “Start Speed”, “End Speed”, and “Gradual Mode”.

For example, set “Gradual Mode” as “Weave+Travel Speed”, “Start Speed” as 24 cm/min, and “End Speed” as 30 cm/min. Click “Add” then verify.

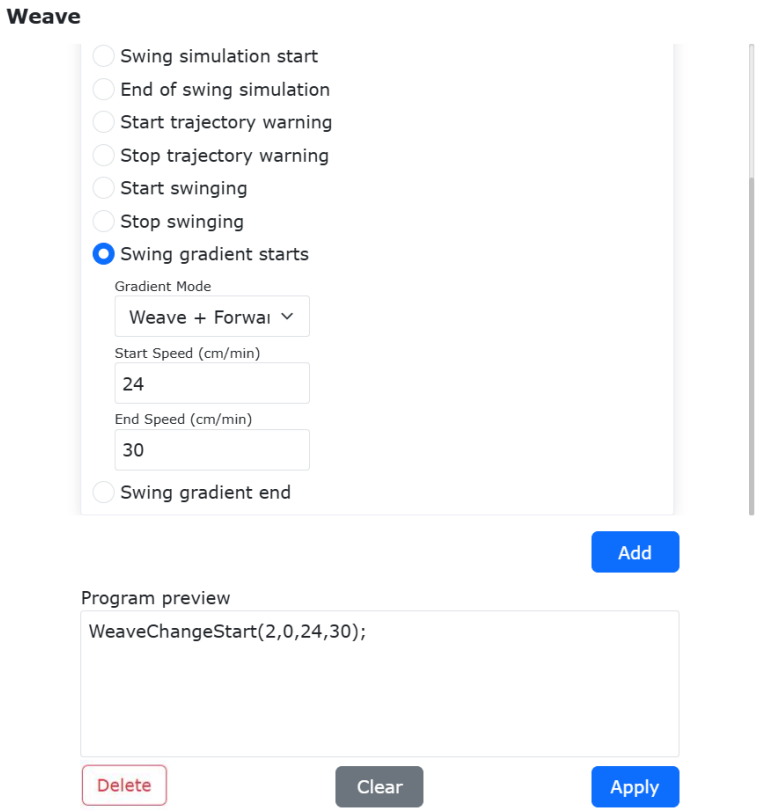


Figure 9.29-11 Weave+Travel Speed Gradual Start Parameters

Click “Weave Gradual End” (no parameters needed), then “Add” and verify.

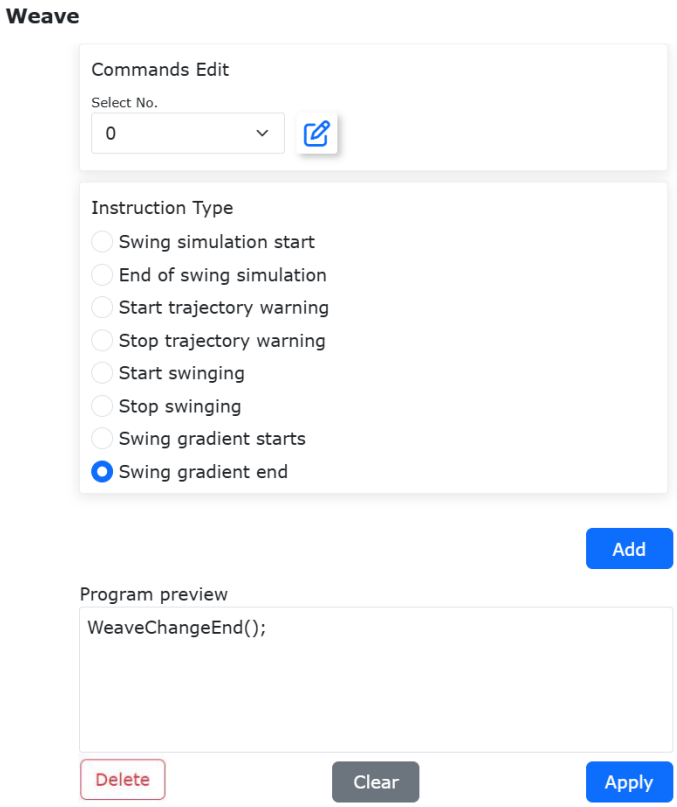


Figure 9.29-12 Weave+Travel Speed Gradual End Parameters

Click “Apply” to generate LUA program.

Note: Motion commands must be included. A sample program is shown below.



Figure 9.29-13 Typical Travel Speed Gradual Change Arc Tracking LUA Program

Note: Before configuring parameters in "Weld" or "Weave" interfaces, confirm the communication method between control box and welder (analog: "Controller I/O"; digital: "Digital Communication Protocol").

1.3.9.30 Robot ModbusRTU Communication

1.3.9.30.1 Overview

ModbusRTU is a commonly used communication protocol in industrial production. FAIR collaborative robots provide two communication methods: ModbusRTU Master and ModbusRTU Slave. The collaborative robot supports up to 8 ModbusRTU Masters communicating simultaneously with external devices, with each master supporting up to 128 registers. The robot's ModbusRTU Slave has 64 coils, 64 discrete inputs, 32 holding registers, and 32 input registers (holding registers and input registers support both signed and floating-point data types).

Additionally, some input register addresses of the robot's ModbusRTU Slave are dedicated to feedback information such as current joint positions and motion speeds. Some coil register addresses are dedicated to controlling robot operations like starting programs, stopping programs, and setting control box DOs. The robot's ModbusRTU Slave only supports connection with one master. Detailed usage methods are described below.

1.3.9.30.2 Robot ModbusRTU Master Operation Instructions

Before using the collaborative robot as a ModbusRTU Master to communicate with your device, please check the 485 hardware connection between your device and the robot. The steps to use the robot as a ModbusRTU Master are as follows: Add a master; Add registers; Communication test; Write user program; Execute user program.

1.3.9.30.2.1 Adding a ModbusRTU Master

Open WebApp, click “Teach Simulation” and “Program Teaching” in sequence, and create a new user program “testModbusRTUMaster.lua”.

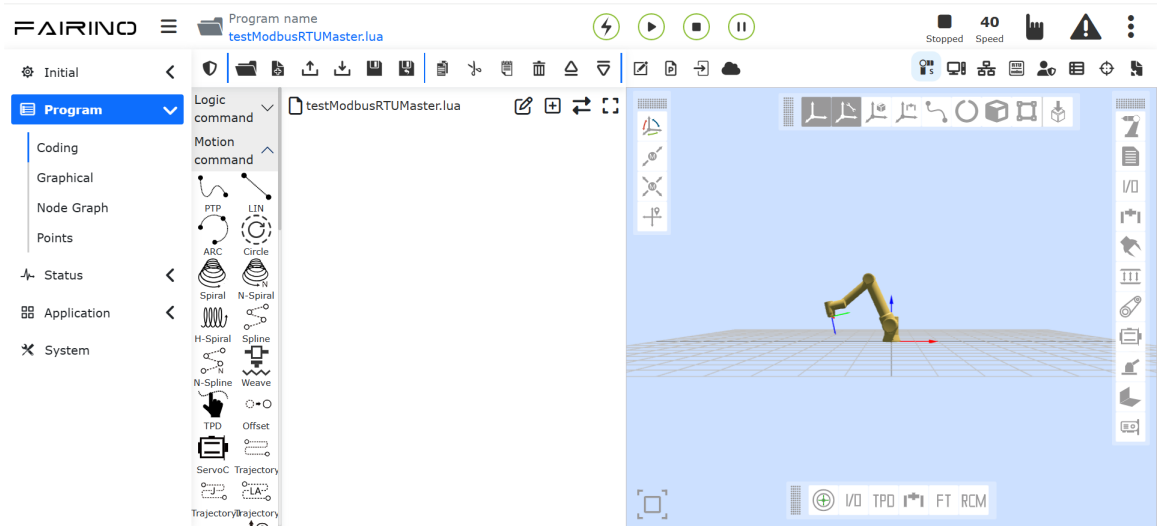


Figure 9.30-1 Creating a ModbusRTU Master User Program

Click the “ModbusRTU Settings” button to open the ModbusRTU configuration page.

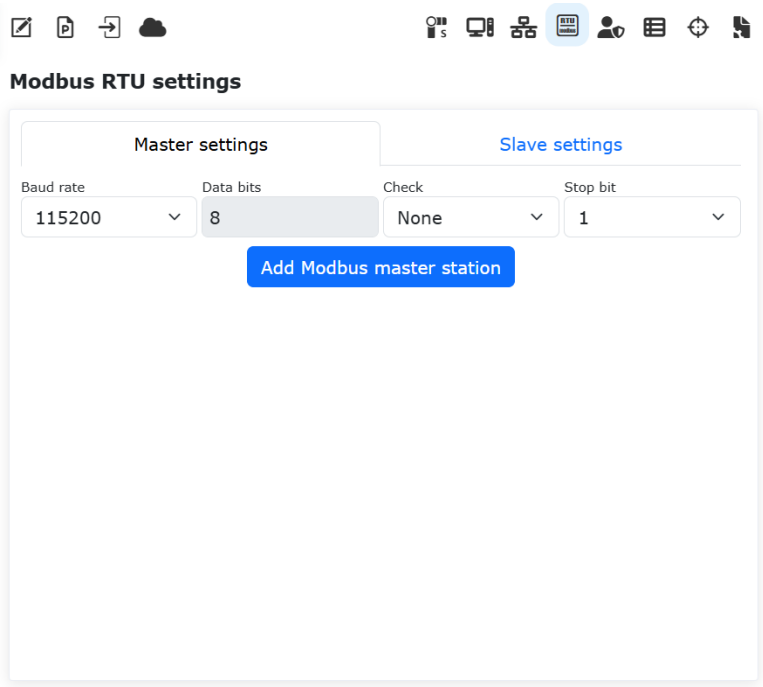


Figure 9.30-2 Opening ModbusRTU Settings

Click “Master Settings” and then “Add Modbus Master” to complete adding a ModbusRTU Master.

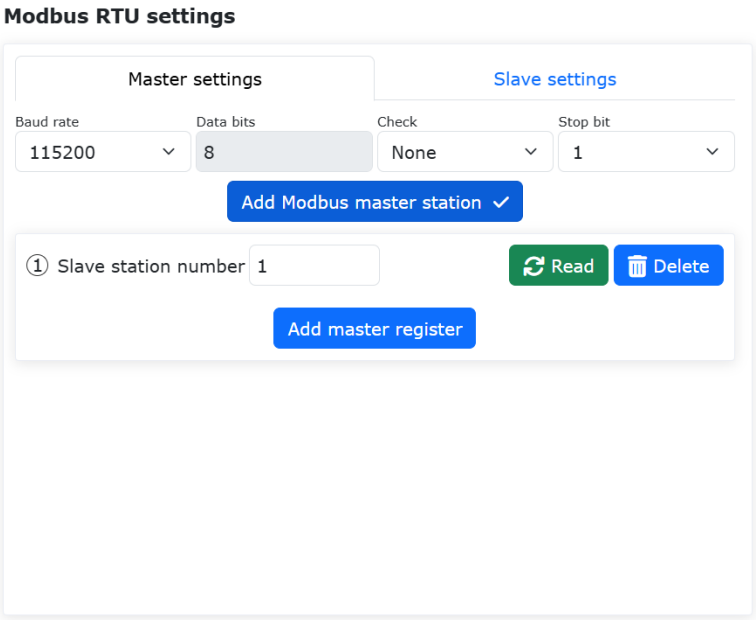


Figure 9.30-3 Adding a “ModbusRTU Master”

Select “Baud Rate”, “Data Bits”, “Parity”, and “Stop Bits” according to your slave device configuration. The meanings of these parameters are as follows:

Baud Rate: The baud rate used for ModbusRTU communication. Supported values: 9600, 14400, 19200, 38400, 56000, 67600, 115200, 128000. Default is 115200. Set to match the slave device.

Data Bits: Currently only supports 8 bits. Set to match the slave device.

Parity: Parity method. Supports None, Odd, Even. Default is None. Set to match the slave device.

Stop Bits: Supports 0.5, 1, 1.5, 2. Default is 1. Set to match the slave device.

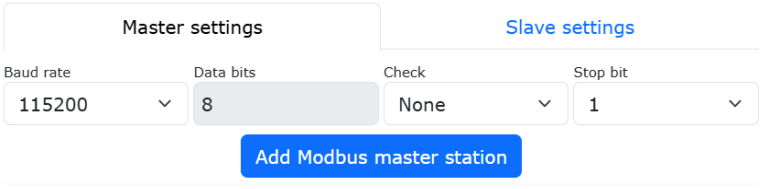


Figure 9.30-4 Setting ModbusRTU Master Parameters

After correctly entering the above parameters, the robot ModbusRTU Master can communicate with the slave device. (If you have confirmed that the ModbusRTU Master parameters are correctly configured but communication fails, please check the following:

Physical 485 connection between the robot and slave device; Check the slave device’s communication configuration and test the communication link with a serial port debugging tool first. For example, configure the same ModbusRTU parameters on the PC, create a new register in the robot web interface, and perform a 0x03 read holding register operation to see if the serial port debugging tool receives data. As shown below, reading register address 0x1000 with 0x03 instruction, the PC can receive data normally, indicating correct communication configuration.)

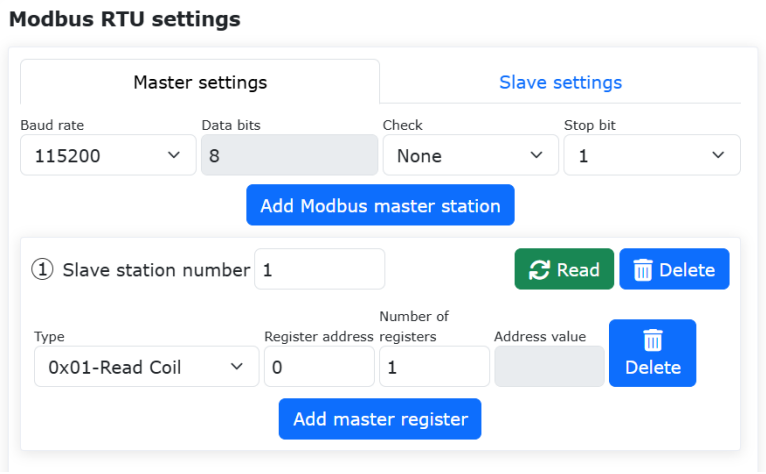


Figure 9.30-5 Verifying Modbus Connection Status

At this point, we have completed the creation of a robot ModbusRTU Master. If you click “Add Modbus Master” again, you can create another new ModbusRTU Master (Figure 2-6). The robot supports up to 8 masters communicating with external devices simultaneously. Double-click the “Delete” button in the upper right corner of the Modbus Master to delete it.

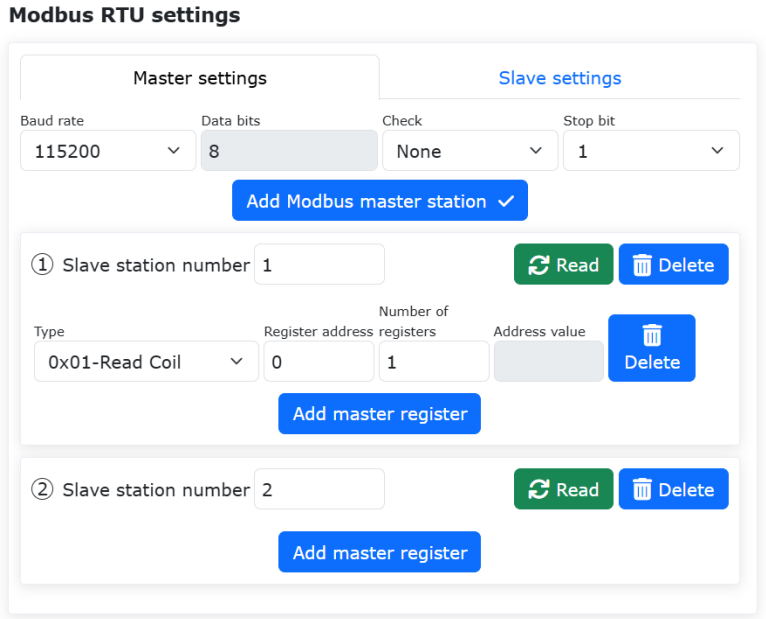


Figure 9.30-6 Adding Another ModbusRTU Master

1.3.9.30.2.2 Adding Registers to ModbusRTU Master

Click the "Add Master Register" button to add a register to the master.

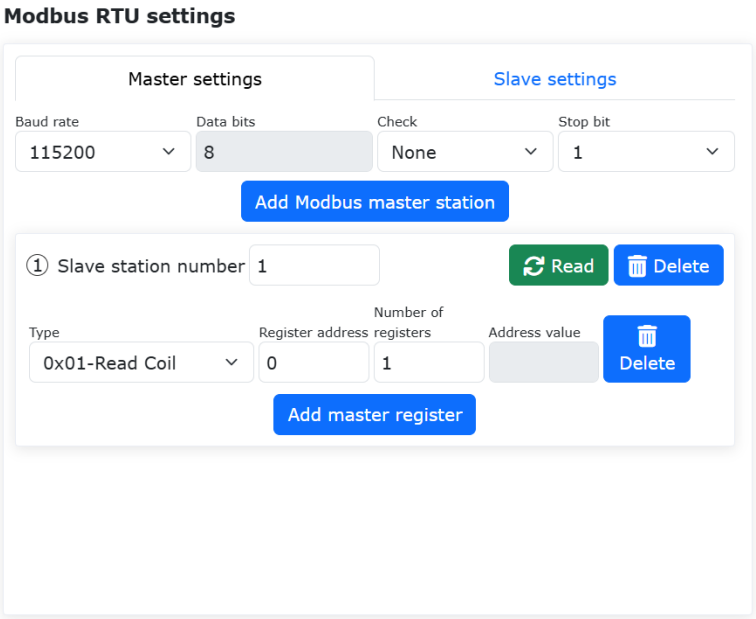


Figure 9.30-7 Adding a ModbusRTU Master Register

Select the master register type, input address number, and name in sequence. The meanings of each parameter are as follows:

Type: Modbus function code. 0x01-Read Coils; 0x02-Read Discrete Inputs; 0x03-Read Holding Registers (signed -32768-32767); 0x03-Read Holding Registers (floating-point, 32-bit data length, occupies two registers, 4 bytes); 0x04-Read Input Registers (signed -32768-32767); 0x04-Read Input Registers (floating-point, 32-bit data length, occupies two registers, 4 bytes); 0x05-Write Single Coil; 0x06-Write Single Holding Register; 0x0F-Write Multiple Coils; 0x10-Write Multiple Holding Registers. Floating-point registers are displayed in big-endian format.

Register Address: The register address of the ModbusRTU Slave to read or write.

Register Quantity: The number of registers to operate when reading or writing multiple registers (0x05, 0x06 can only be 1). Supports up to 12 registers.

Address Value: Display value for reading, or write value (use English comma "," to separate multiple values).

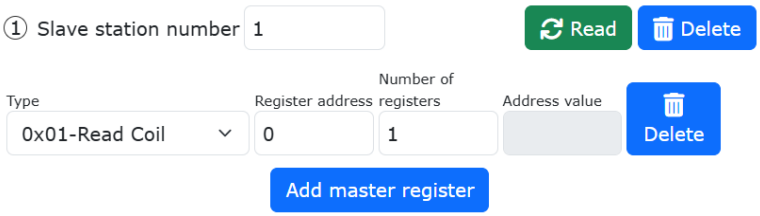


Figure 9.30-8 Setting ModbusRTU Master Register Parameters

Click the "Add Master Register" button again to add another master register. Double-click the "Delete" button on the right side of the register to delete it. The figure below shows the supported function code registers.

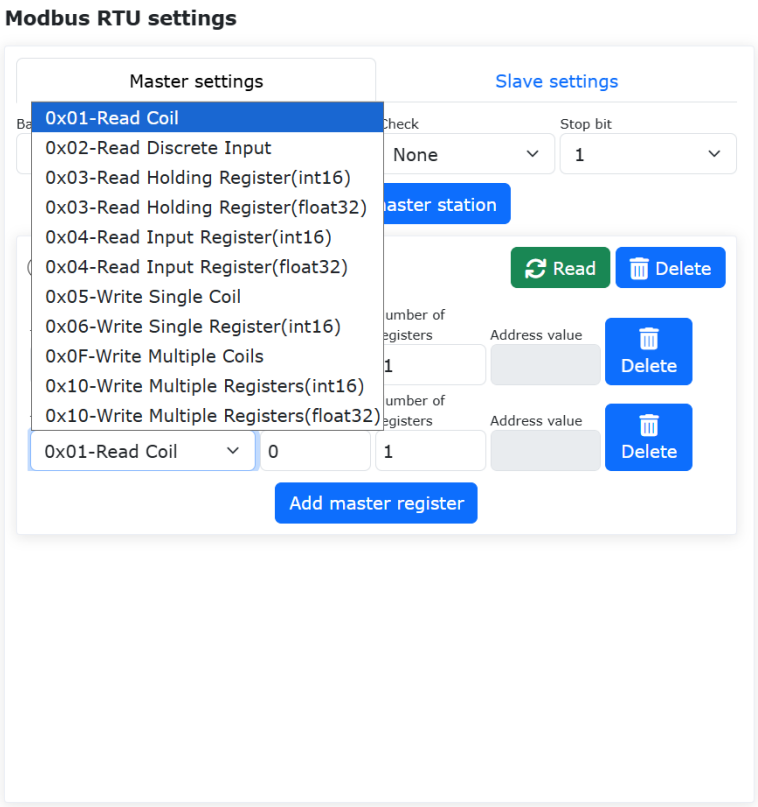


Figure 9.30-9 Adding Multiple Master Registers

1.3.9.30.2.3 ModbusRTU Master Communication Test

The robot Modbus Master register has an “Address Value” field to display the current register value. Registers of types 0x01, 0x02, 0x03, and 0x04 are read-only, and their address values are displayed in gray non-editable fields. When the value of the corresponding address in the slave changes, the robot master can read the corresponding register address value by clicking the read button and display the current value synchronously. Function codes 0x05, 0x06, 0x0F, and 0x10 are write operations, and their address values are displayed in white editable fields. You can modify the register value on the robot Modbus Master settings page.

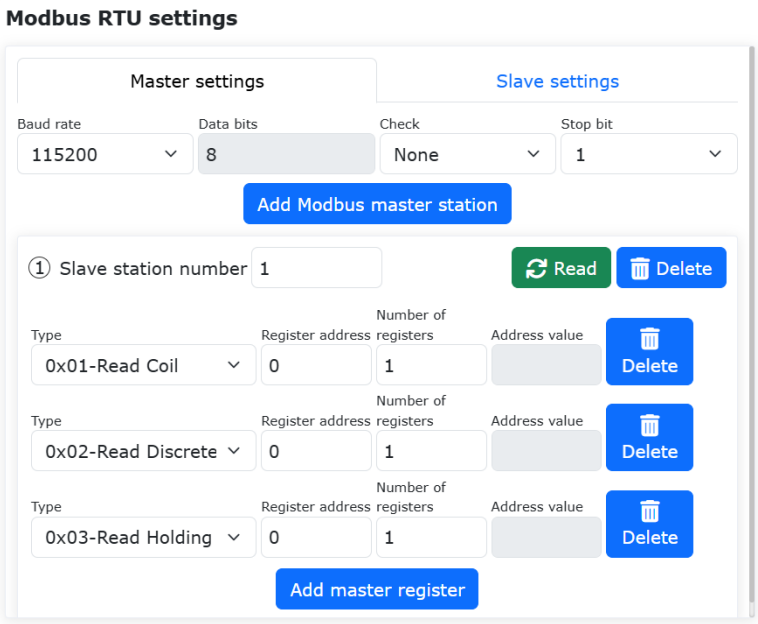


Figure 9.30-10 Modbus Master Address Value

1.3.9.30.2.4 Master Read Register Test

On the external ModbusRTU Slave device, continuously read 10 coils starting from address 0x4000, 12 discrete inputs starting from address 0x3000, 2 holding registers starting from address 0x2010 using int16, and 1 floating-point input register at address 0x1029. The corresponding register address values on the robot Modbus Master settings page will display accordingly. The sent data frames are shown below (since the register at address 0x1029 is set to floating-point type, it actually reads two 16-bit registers, 0x1029 and 0x102A, to store one floating-point number, but the read quantity is set to 1).

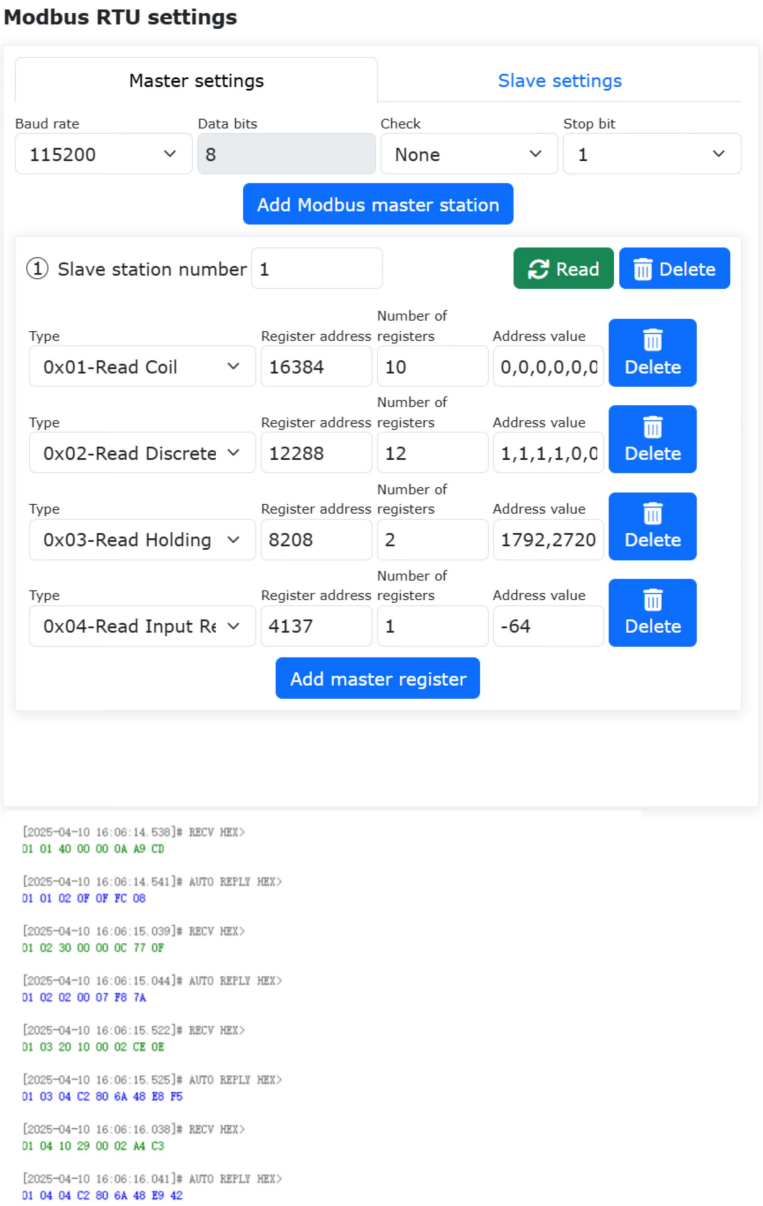


Figure 9.30-11 Modbus Master Displaying Read Register Values (Command Frame Screenshot)

1.3.9.30.2.5 Master Write Register Test

On the robot ModbusRTU Master settings page, write a single coil at address 0x1000 with a value of 1; write a single register at address 0x1001 with a value of 2001; write 5 coils starting at address 0x2000 with values 1,1,0,1,1; write 2 holding registers starting at address 0x2010 with int16 data type and values 3001, 3002; write a floating-point holding register at address 0x1029 (actually two 16-bit registers) with a value of 21.55; the corresponding register addresses on the Modbus Slave have been written with the specified values.

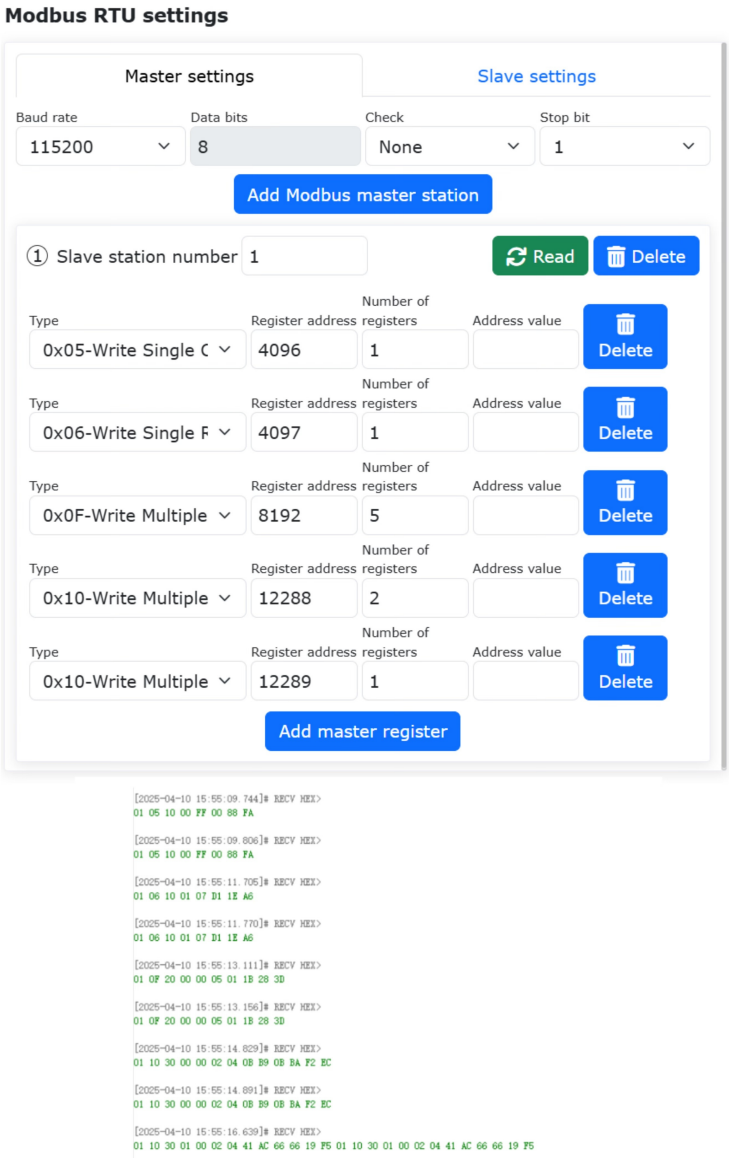


Figure 9.30-12 ModbusRTU Master Write Operation (Command Frame Screenshot)

1.3.9.30.2.6 Writing a ModbusRTU Master Program

Click “All” and then “Communication Instructions” to open the communication instruction addition page.

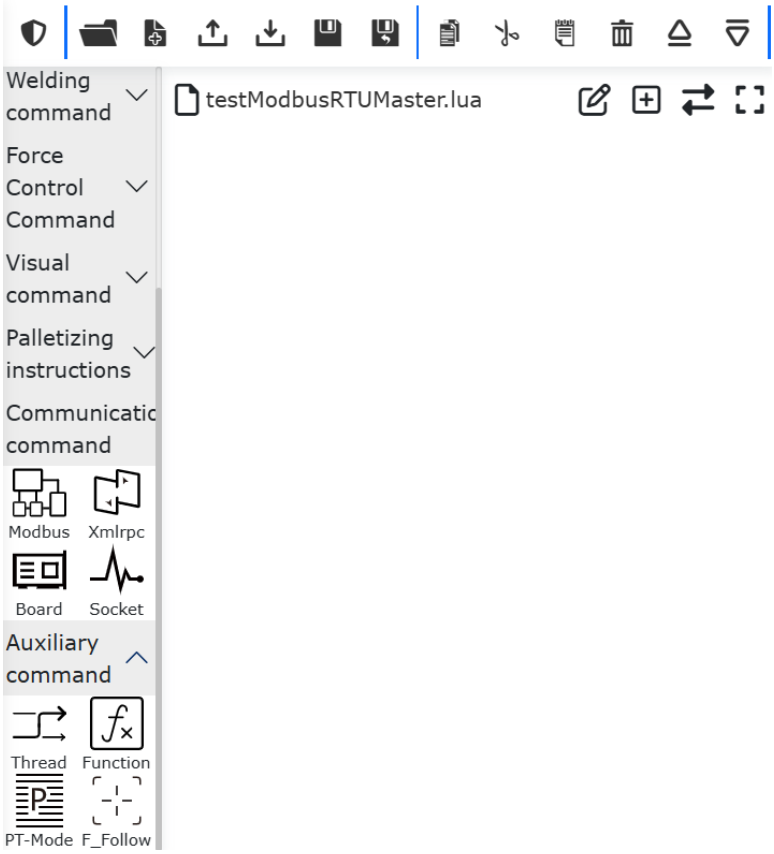


Figure 9.30-13 Opening the Communication Instruction Addition Page

Click “Modbus”.

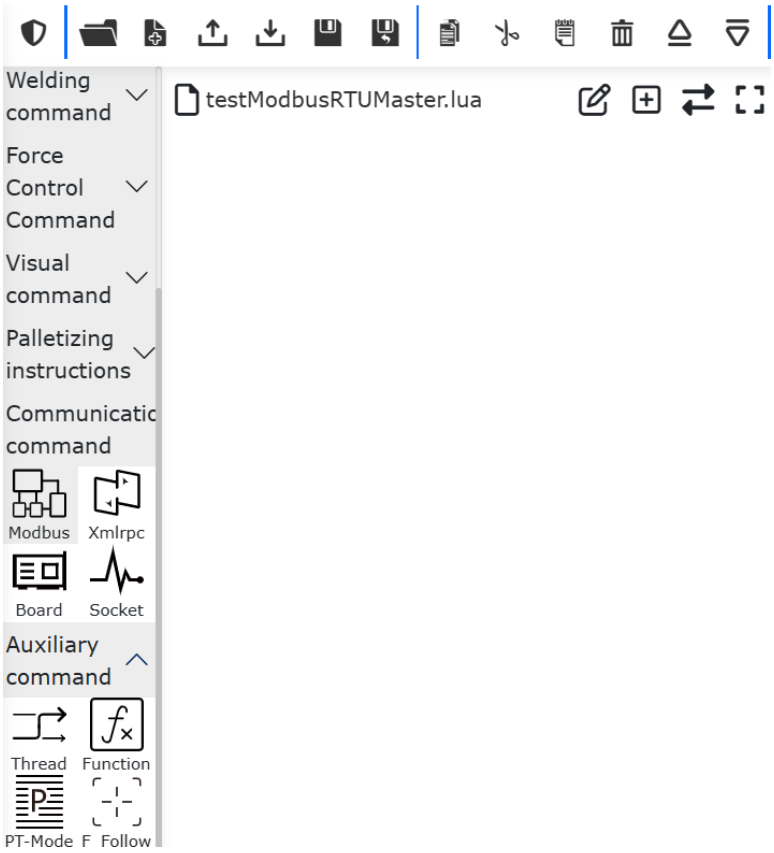


Figure 9.30-14 Selecting Modbus

Click “Modbus_RTU”, select “Master (Client)”, to open the ModbusRTU Master instruction addition page.

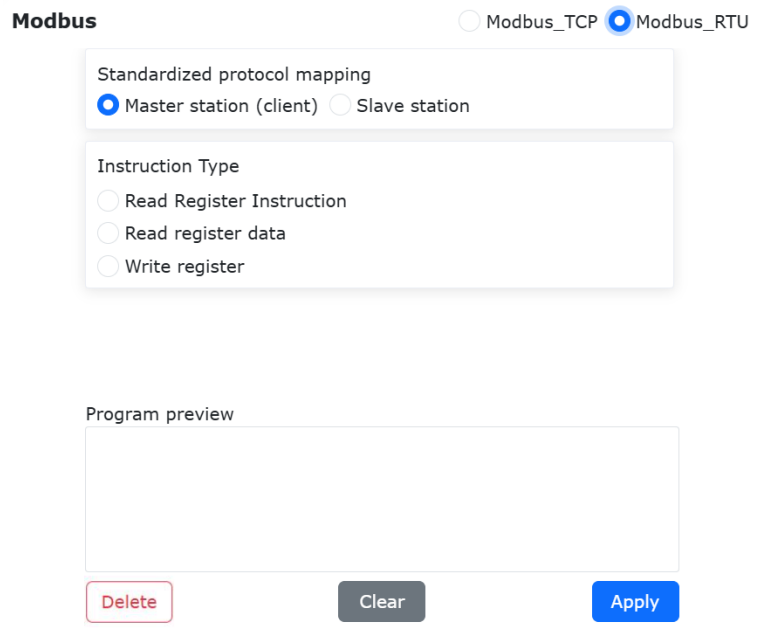


Figure 9.30-15 Selecting Modbus_RTU

1.3.9.30.2.7 Write Single Coil

Select “Write Register”, function code 0x05-Single Coil, register/coil address 0x1000, register value/coil quantity 1, byte array {1}, click “Add” button. Finally, scroll to the bottom of the page and click “Apply” button (Figure 2-16).

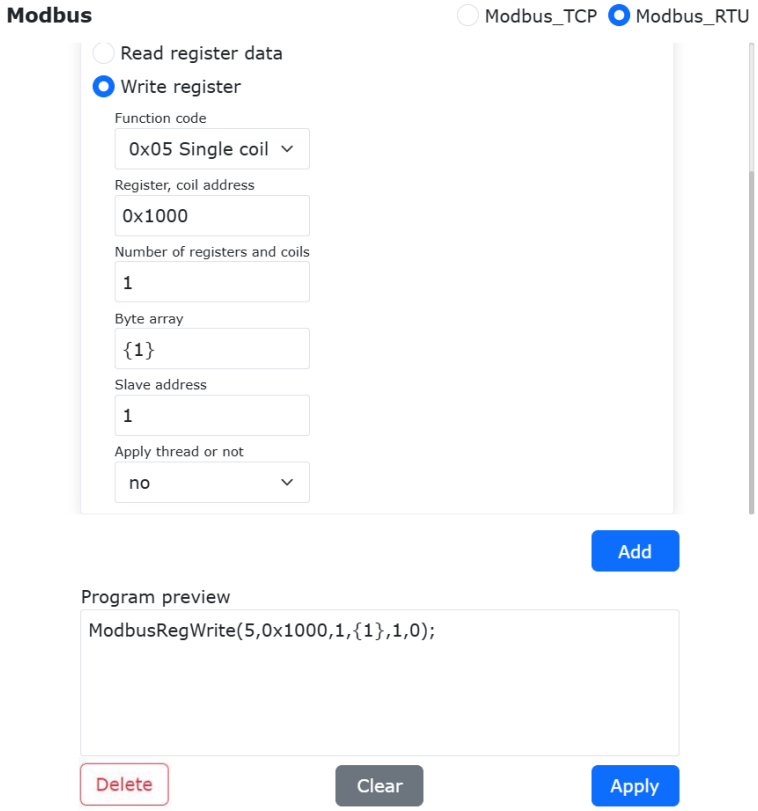


Figure 9.30-16 Writing a Single Coil

Now the robot program “testModbusRTUSlave.lua” has added a robot Modbus Master instruction to write a single digital output. Switch the robot to automatic mode and click the start button. The robot will write the value 1 to the coil register address 0x1000.

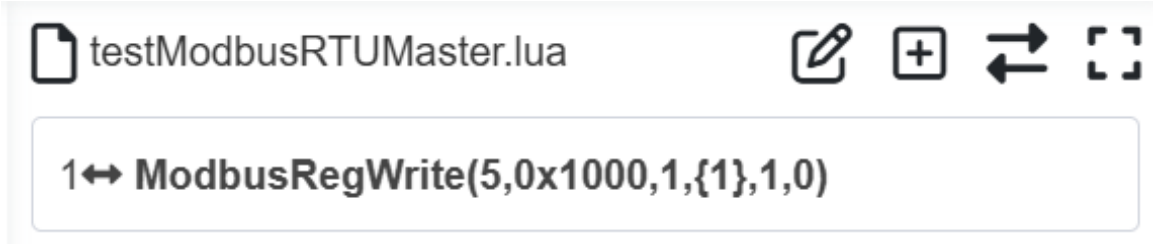


Figure 9.30-17 LUA Program for Writing a Single Coil

1.3.9.30.2.8 Write Multiple Coils

Select “Write Register”, function code 0x0F-Multiple Coils, register/coil address 0x1010, register value/coil quantity 3, byte array {1,0,1}, click “Add” button. Finally, scroll to the bottom of the page and click “Apply” button (Figure 2-18).

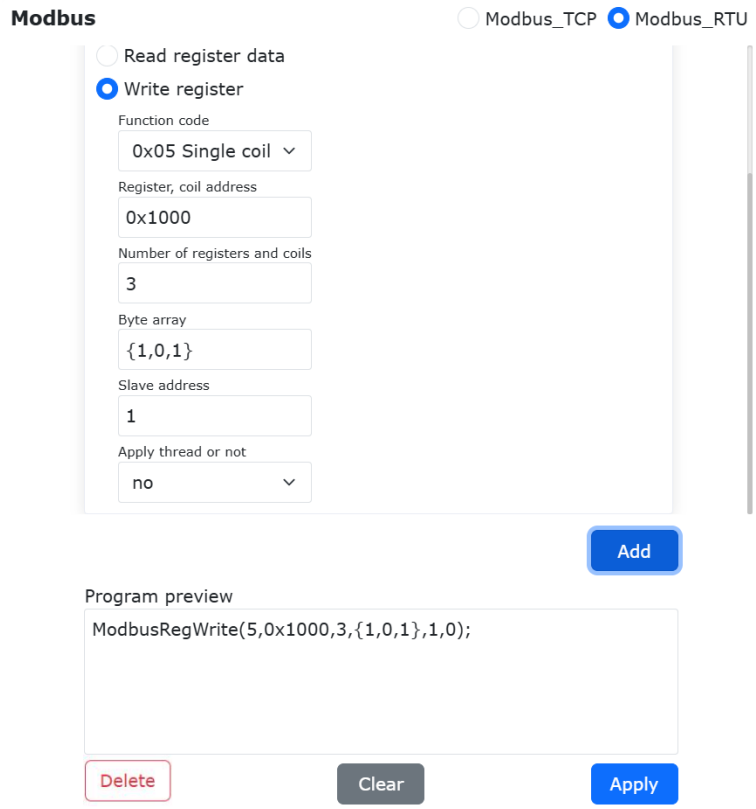


Figure 9.30-18 Writing Multiple Coils

Now the robot program “testModbusRTUSlave.lua” has added a robot Modbus Master instruction to write multiple digital outputs. Switch the robot to automatic mode and click the start button. The robot will write the values 1, 0, 1 to the coil registers starting at address 0x1000.

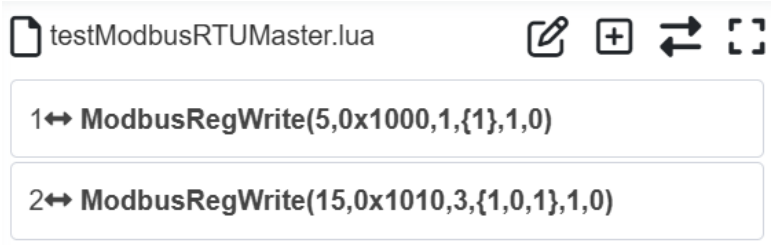


Figure 9.30-19 LUA Program for Writing Multiple Coils

1.3.9.30.2.9 Read Coils and Discrete Inputs

Select “Read Register Instruction”, function code 0x01-Coils (select 0x02-Discrete Inputs if reading discrete inputs), register/coil address 0x2000, register/coil quantity 3, click “Add” button. Then select “Read Register Data”, register/coil/discrete input quantity 3, click “Add” button. Finally, scroll to the bottom of the page and click “Apply” button (Figure 2-20).

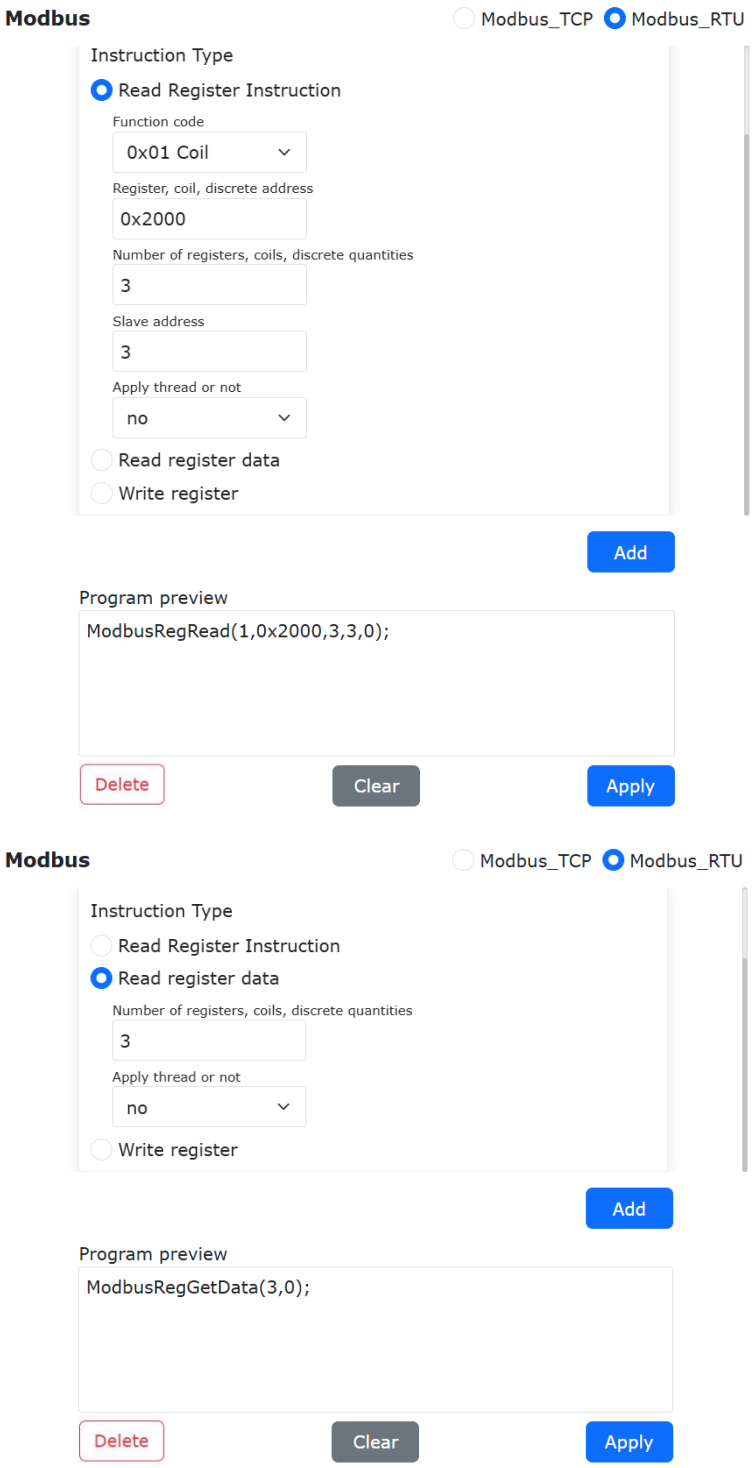


Figure 9.30-20 Reading Coils

Now the robot program "testModbusRTUSlave.lua" has added two robot Modbus Master instructions to read coils.

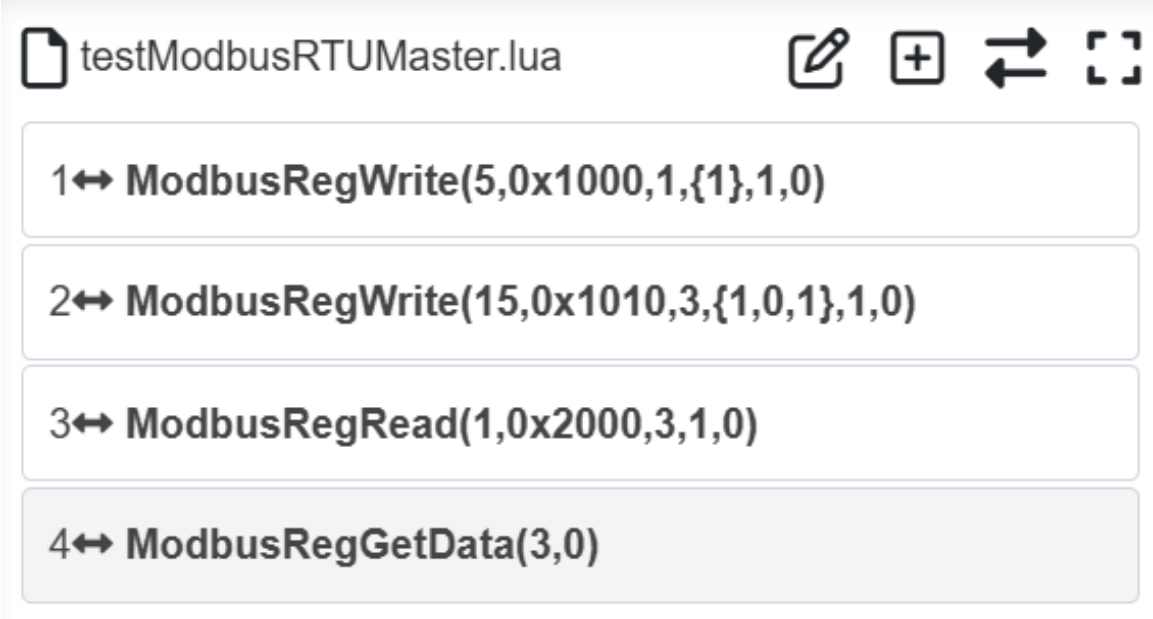


Figure 9.30-21 Program for Reading a Single Coil

Usually, after reading Modbus registers, the read values are stored in variables. Therefore, you need to define variables to store the read values. Click the "Switch Mode" button to switch the robot LUA program to editable mode. Before the "ModbusRegRead" instruction, add return value variables "value1", "value2", "value3". After executing the program, the read values will be stored in "value1", "value2", and "value3".

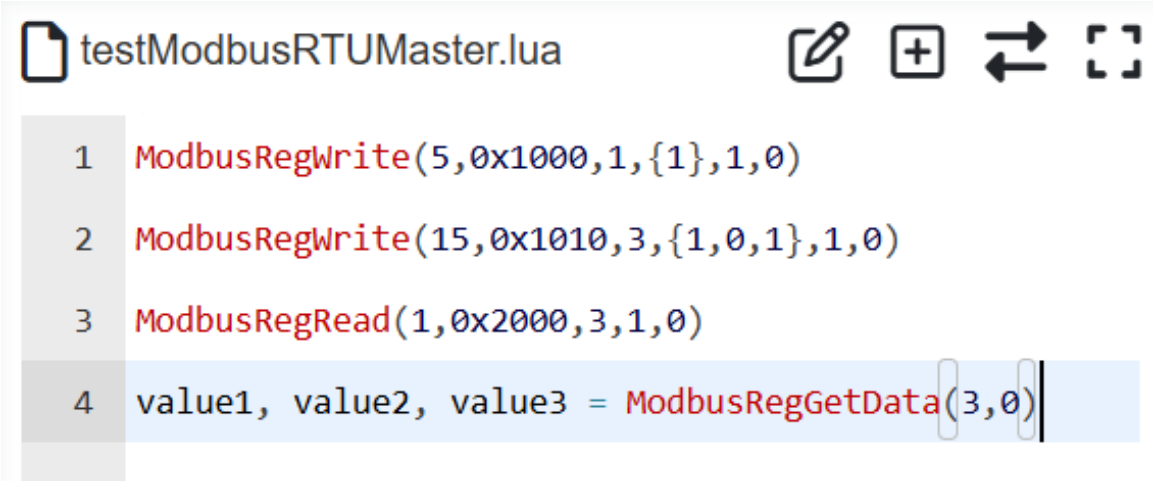


Figure 9.30-22 Storing Multiple Coil Values in Variables

Coil and discrete input type register values are only 0 or 1. In the robot program, you can perform different operations by judging different register values.

1.3.9.30.2.10 Read Holding Registers and Input Registers

Select “Read Register Instruction”, function code 0x03-Holding Registers (select 0x04-Input Registers if reading input registers), register/coil address 0x4000, register/coil quantity 5, click “Add” button. Then select “Read Register Data”, register/coil/discrete input quantity 5, click “Add” button. Finally, scroll to the bottom of the page and click “Apply” button (Figure 2-23).

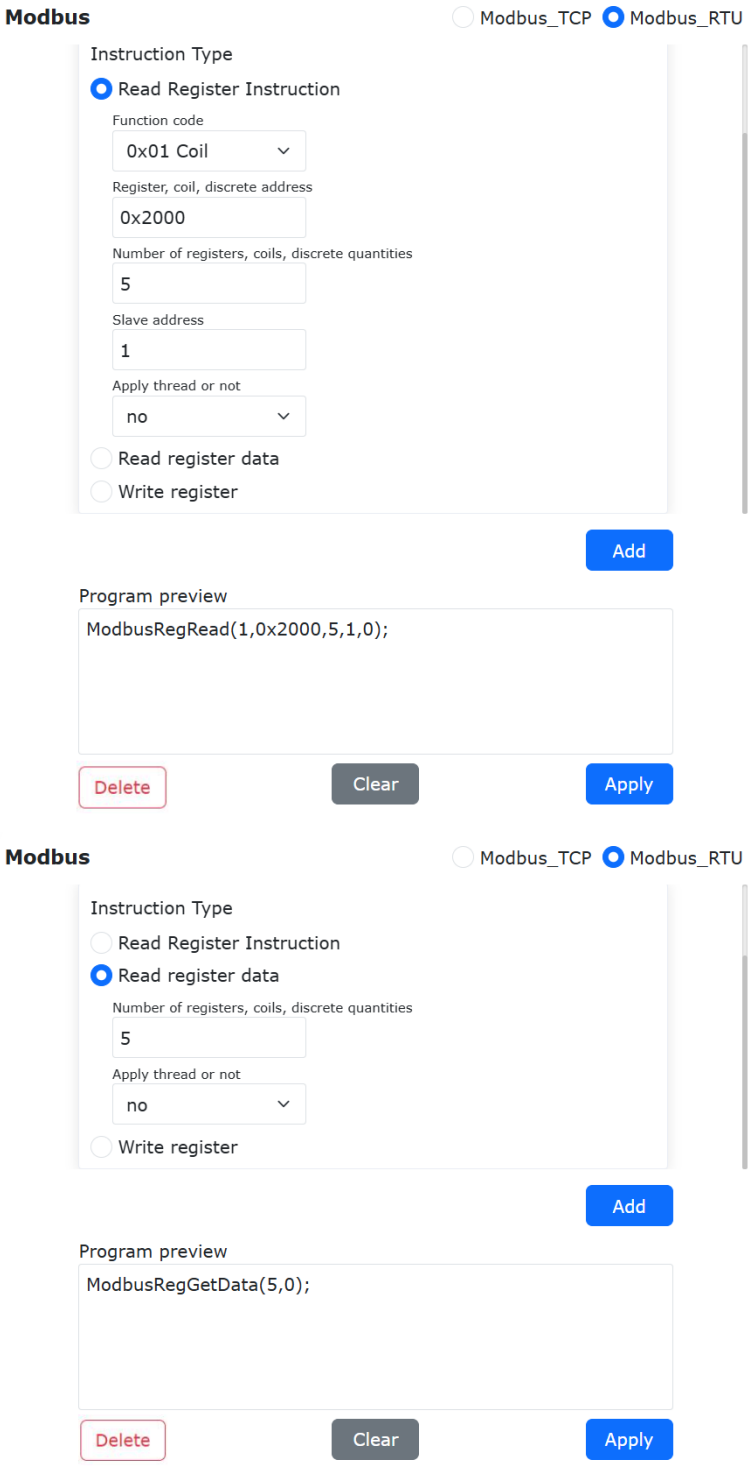


Figure 9.30-23 Reading Holding Registers

Now the robot program “testModbusRTUSlave.lua” has added two robot Modbus Master instructions to read coils.

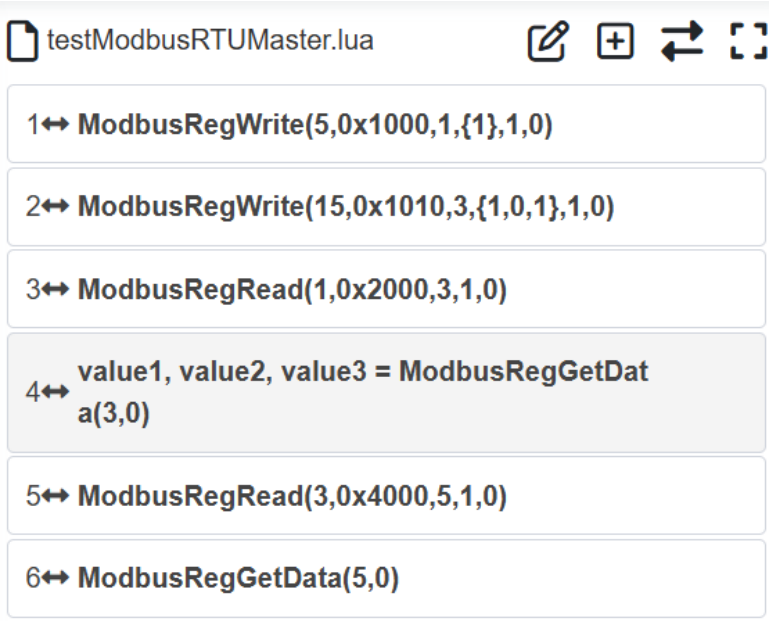


Figure 9.30-24 Program for Reading a Single Coil

Usually, after reading Modbus registers, the read values are stored in variables. Therefore, you need to define variables to store the read values. Click the “Switch Mode” button to switch the robot LUA program to editable mode. Before the “ModbusRegRead” instruction, add return value variables “value1”, “value2”, “value3”, “value4”, “value5”. After executing the program, the read values will be stored in these variables.

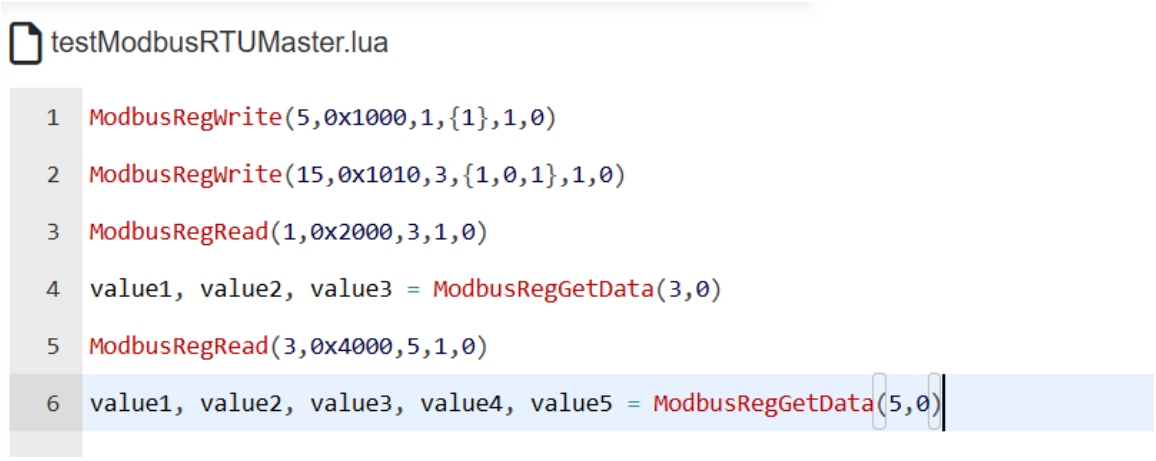


Figure 9.30-25 Storing Multiple Holding Register Values in Variables

1.3.9.30.3 Robot ModbusRTU Slave Operation Instructions

The robot ModbusRTU Slave provides four types of registers: general-purpose digital inputs (coils), general-purpose digital outputs (discrete inputs), general-purpose analog inputs (holding registers), and general-purpose analog outputs (input registers). The general-purpose digital inputs and analog inputs are mainly used for the robot to read data from external ModbusRTU Master devices to control robot operations. The general-purpose digital outputs and analog outputs are mainly used for the robot to send data signals to external ModbusRTU Master devices, which read the relevant register values to control their device operations.

In addition to the above general-purpose inputs and outputs, the robot also provides some "functional digital inputs (coils)" for external master devices to control robot operations such as starting programs and stopping programs. Some input registers are dedicated to displaying the current robot status information, including the robot's current Cartesian position and operation status (see Appendix 1: ModbusRTU Slave Address Mapping Table for specific definitions). The robot ModbusRTU Slave only supports connection with one master. The process of using the robot ModbusRTU Slave mainly includes: Parameter configuration; Communication test; Program writing.

1.3.9.30.3.1 ModbusRTU Slave Communication Parameter Configuration

Open WebApp, click "Teach Simulation" and "Program Teaching" in sequence, and create a new user program "testModbusRTUSlave.lua".

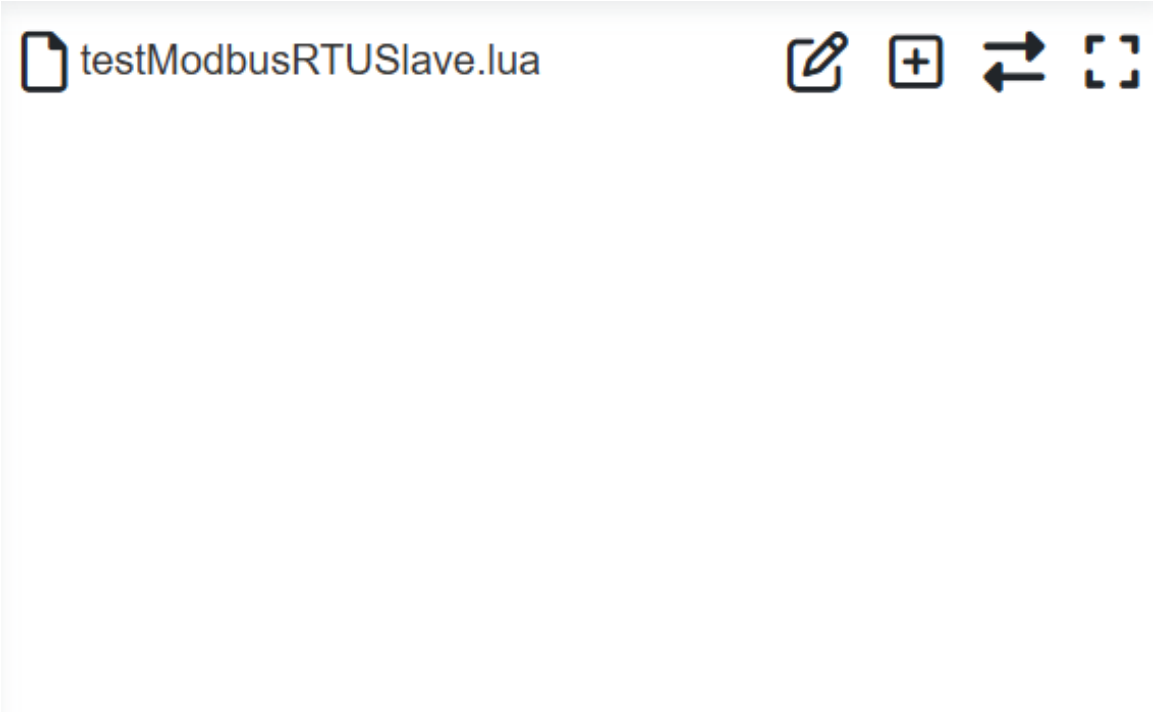


Figure 9.30-26 Creating a ModbusRTU Slave User Program

Click the "ModbusRTU Settings" button to open the ModbusRTU configuration page.

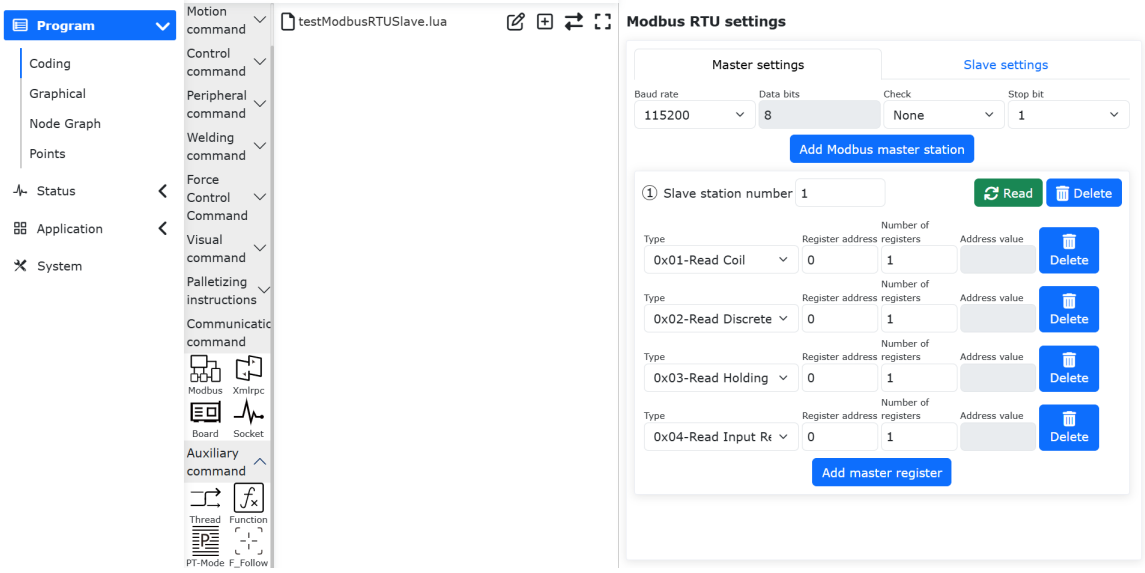


Figure 9.30-27 Opening ModbusRTU Settings

Click “Slave Settings” and enter the robot slave’s baud rate, data bits, parity, stop bits, and slave number. “Baud Rate”, “Data Bits”, “Parity”, and “Stop Bits” are the parameter configurations for the robot as a ModbusRTU Slave. “Slave Number” is the slave device number in the instructions sent by the external master.

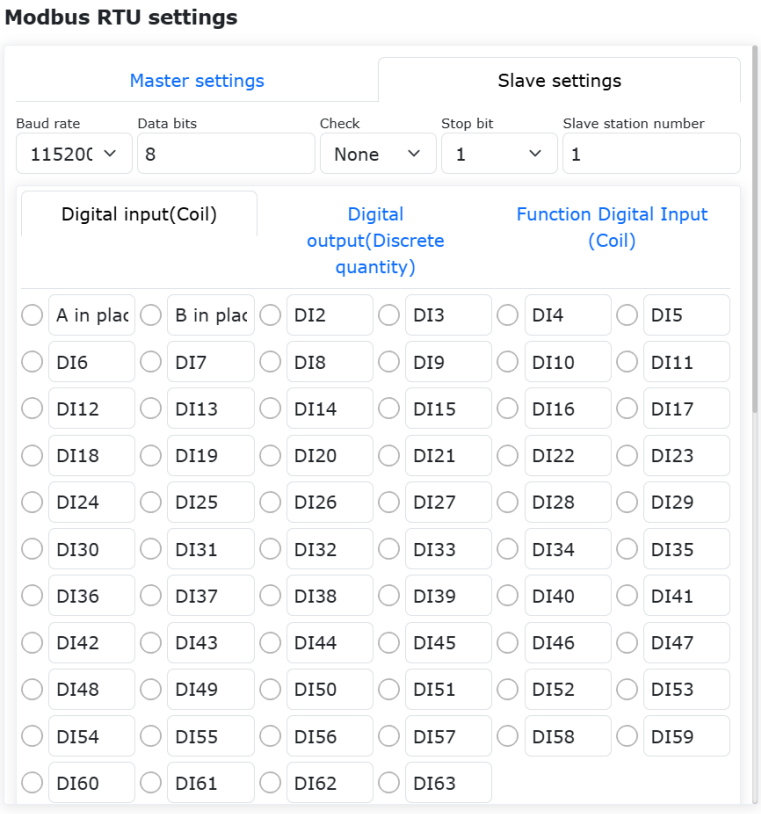


Figure 9.30-28 ModbusRTU Slave Settings

1.3.9.30.3.2 ModbusRTU Slave Communication Test

1.3.9.30.3.3 General-Purpose Digital Inputs (Coils)

The robot ModbusRTU Slave provides 64 coil registers with addresses 0x4000~0x403F (see Appendix 1: ModbusRTU Slave Address Mapping Table for specific definitions). The general-purpose registers of the robot ModbusRTU Slave can be aliased. Change the name of the robot slave coil register DI0 to “A Ready” and DI1 to “B Ready”. According to the address mapping table, the Modbus coil addresses for “A Ready” and “B Ready” are 0x4000 and 0x4001, respectively. On the external ModbusRTU Master device, set the robot slave coil register addresses 0x4000 and 0x4001 to 1. The indicator lights for these two registers on the robot ModbusRTU Slave monitoring page will light up.

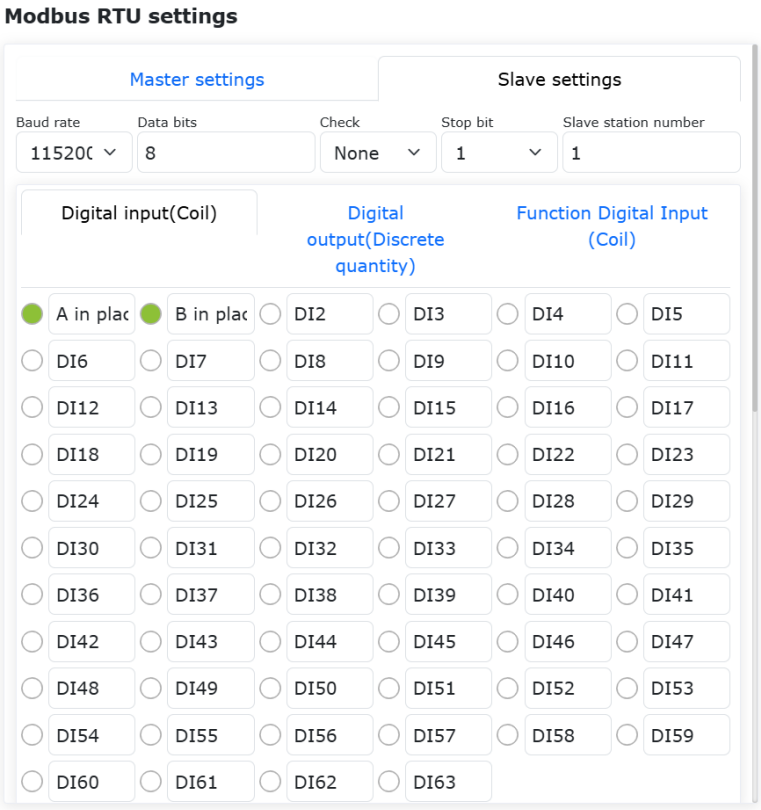


Figure 9.30-29 ModbusRTU Slave Coil Status Monitoring (Command Frame Screenshot)

1.3.9.30.3.4 General-Purpose Digital Outputs (Discrete Inputs)

The robot ModbusRTU Slave provides 64 discrete input registers with addresses 0x3000-0x303F (see Appendix 1: ModbusRTU Slave Address Mapping Table for specific definitions). Similarly, the discrete input registers of the robot ModbusRTU Slave can also be aliased. Click “General-Purpose Digital Outputs (Discrete Inputs)” to change the name of the robot slave discrete input register DO0 to “A Start” and DO1 to “B Start”. According to the address mapping table, the Modbus discrete input addresses for “A Start” and “B Start” are 0x3000 and 0x3001, respectively. Click the indicator light corresponding to “A Start”. The light will turn on, and the value of the corresponding register address 0x3000 will change to 1. The external ModbusRTU Master device can read this register value.

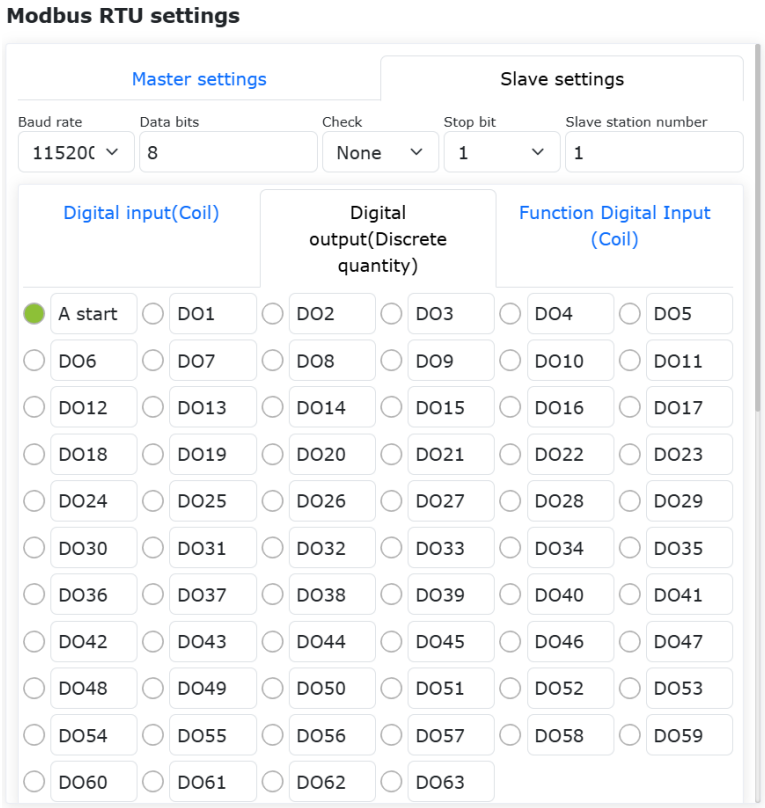


Figure 9.30-30 ModbusRTU Slave Discrete Input Control

1.3.9.30.3.5 Analog Inputs (Holding Registers)

The robot provides three types of holding registers: unsigned, signed, and floating-point, totaling 32. The addresses for AI0~AI32 are 0x2000-0x202F (see Appendix 1: ModbusRTU Slave Address Mapping Table for specific definitions). The data range for signed registers is -32768~32767, and floating-point registers are displayed in big-endian format. Change the names of AI0 and AI1 to “Voltage” and “Current”, respectively. From the ModbusRTU Slave address mapping table, the addresses of these two registers are 0x2000 and 0x2001, respectively. Therefore, when the connected master device modifies the values of holding register addresses 0x2000 and 0x2001, the “Voltage” and “Current” register address values on the robot ModbusRTU Slave monitoring page will update accordingly. The robot’s analog inputs are mainly used for the robot to obtain numerical signals from external master devices.

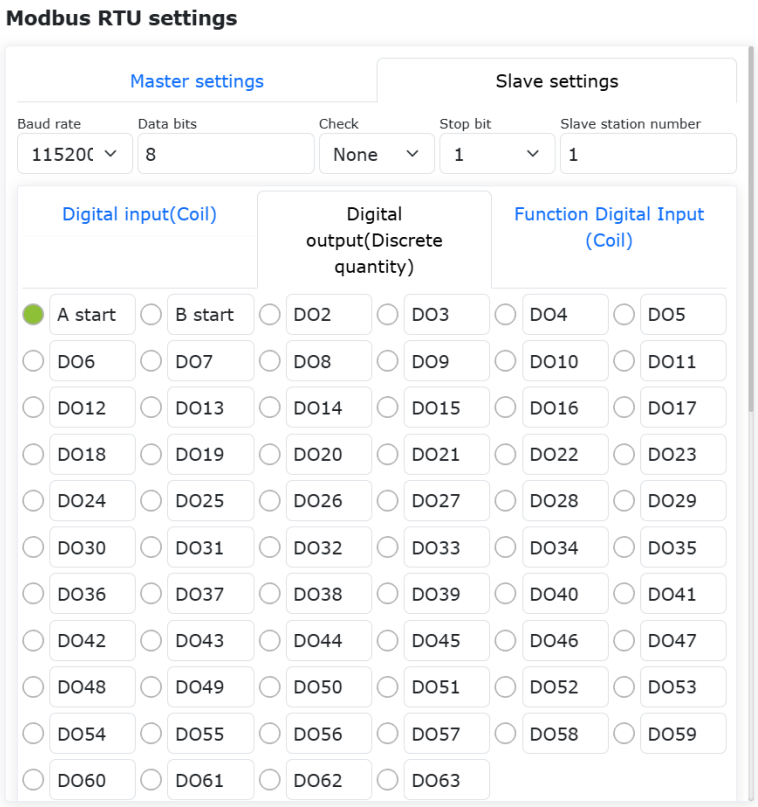


Figure 9.30-31 ModbusRTU Slave Analog Input Monitoring (Command Frame Screenshot)

1.3.9.30.3.6 Analog Outputs (Input Registers)

The robot provides three types of input registers: unsigned, signed, and floating-point, totaling 64. The addresses for AO0~AO63 are 0x1000-0x100F, 0x104D-0x106C (see Appendix 1: ModbusRTU Slave Address Mapping Table for specific definitions). The data range for signed registers is -32768~32767, and floating-point registers are displayed in big-endian format. Change the names of AO0 and AO1 to “Target Position A” and “Target Position B”, respectively, with input register values of 2000 and 1500. From the ModbusRTU Slave address mapping table, the addresses of these two registers are 0x1000 and 0x1001, respectively. Therefore, when the connected master device reads the input register addresses 0x1000 and 0x1001, it will obtain the set values. The robot slave analog outputs are mainly used for the robot to send numerical signals to external master devices.

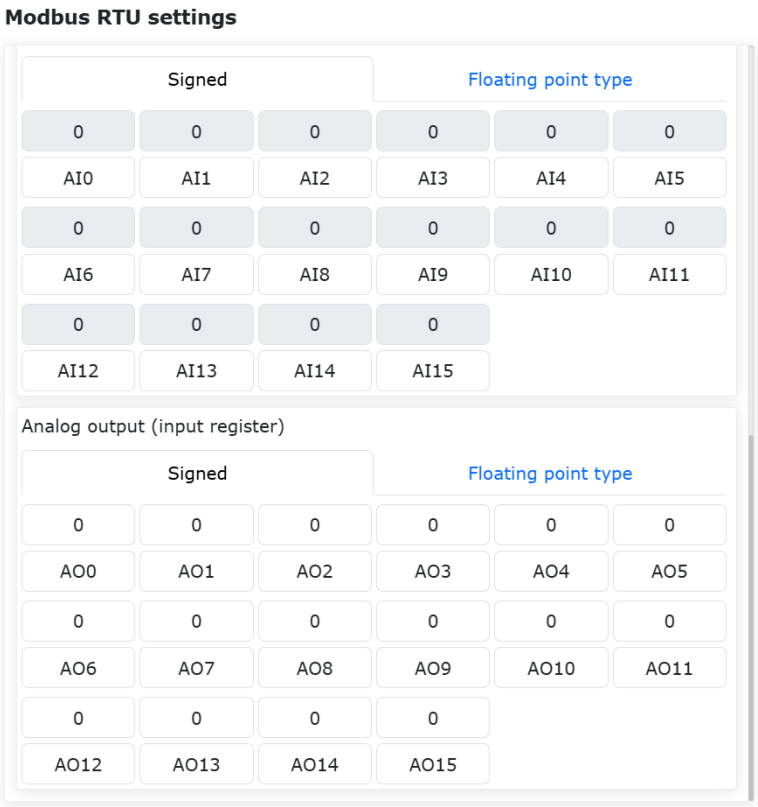


Figure 9.30-32 Modbus Slave Modifying Analog Inputs

1.3.9.30.3.7 Writing a ModbusRTU Slave Program

Click “All” and then “Communication Instructions” to open the communication instruction addition page.

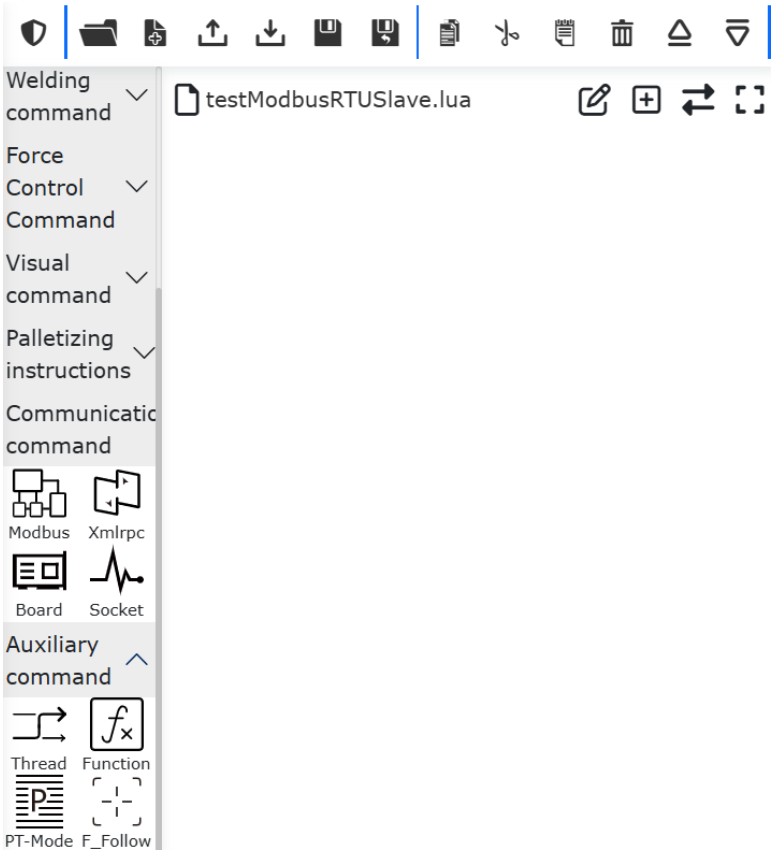


Figure 9.30-33 Opening the Communication Instruction Addition Page

Click “Modbus”.

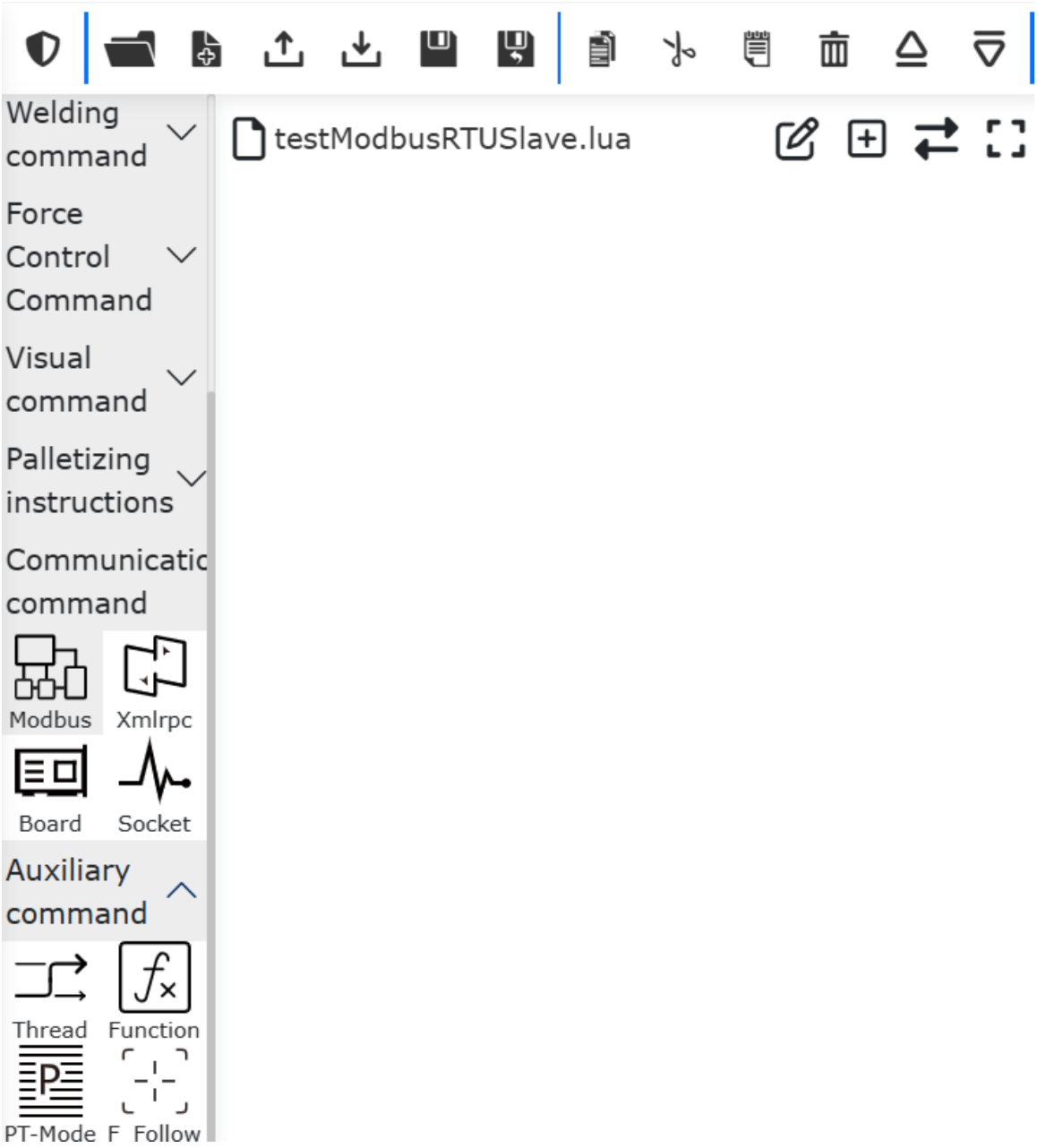


Figure 9.30-34 Selecting Modbus

Click “Modbus_RTU”, select “Slave”, to open the ModbusRTU Slave instruction addition page (Figure 60).

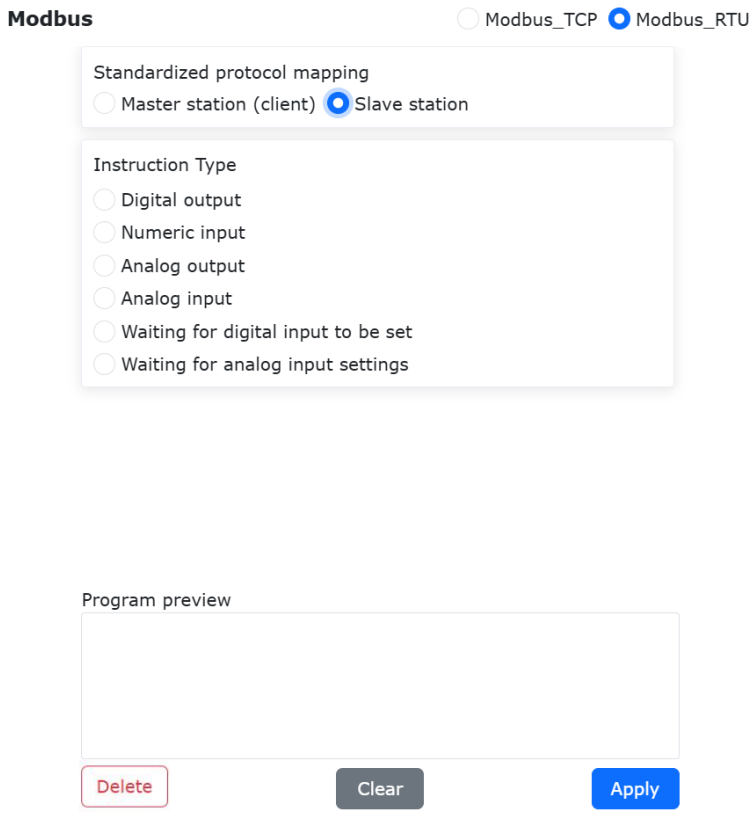


Figure 9.30-35 Selecting Modbus_RTU, Slave

1.3.9.30.3.8 Write Single Digital Output DO (Discrete Input)

Select DO name as “A Start”, register quantity as 1, register value as 0, click “Write Single Digital Output”. Finally, scroll to the bottom of the page and click “Apply” button.

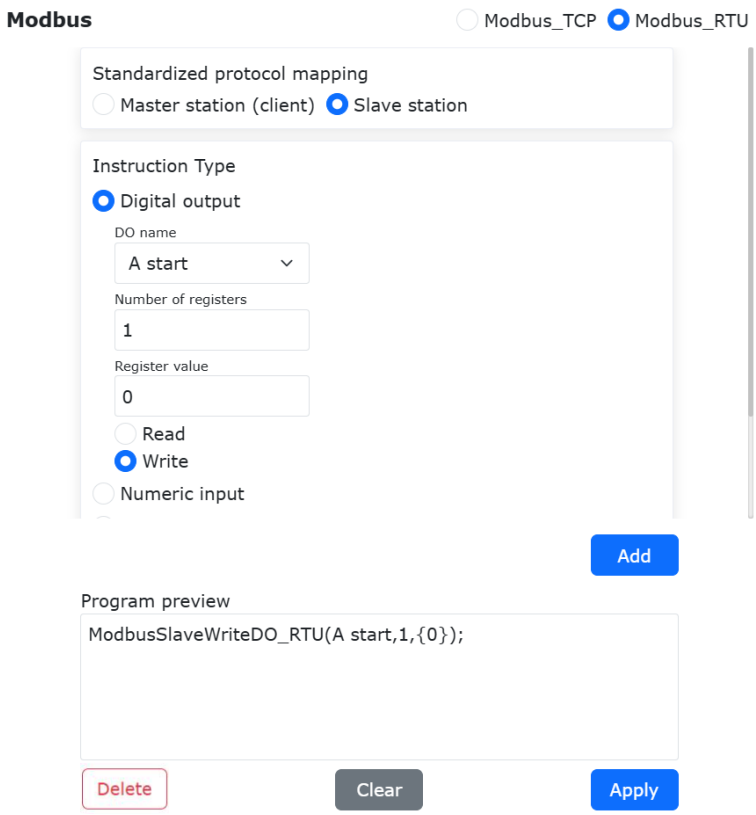


Figure 9.30-36 Adding a Write Single Digital Output Instruction, Applying the Write Single Digital Output Instruction

Now the robot program "testModbusRTUSlave.lua" has added a robot Modbus Slave instruction to write a single digital output. Switch the robot to automatic mode and click the start button. The robot will write the value 0 to the digital output named "A Start".

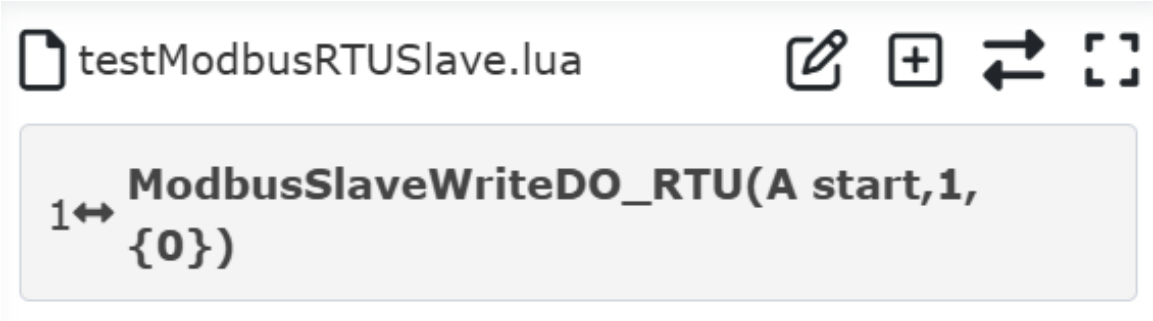


Figure 9.30-37 LUA Program for Writing a Single Digital Output

1.3.9.30.3.9 Write Multiple Digital Outputs DO (Discrete Inputs)

Open the ModbusRTU Slave instruction addition page, find “Digital Output Settings”, select DO name as “A Start”, register quantity as 5, register value as 1,0,1,0,1. The number of register values must match the set register quantity, and multiple register values should be separated by English commas. Click “Write Digital Output”. Finally, scroll to the bottom of the page and click “Apply” button.

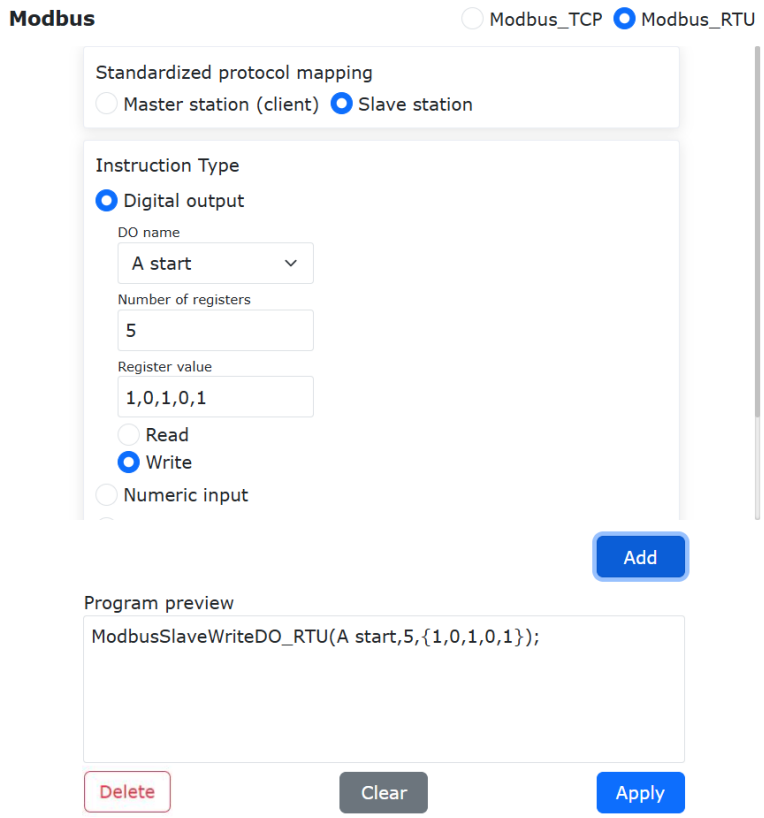


Figure 9.30-38 Configuring Write Multiple Digital Outputs, Applying Write Multiple Digital Outputs

Now the robot program “testModbusRTUSlave.lua” has added a robot Modbus Slave instruction to write multiple digital outputs. Switch the robot to automatic mode and click the start button. The robot will write the values 1, 0, 1, 0, 1 to the discrete input registers starting from “A Start” and the next four.



Figure 9.30-39 LUA Program for Writing Multiple Digital Outputs

1.3.9.30.3.10 Read Single Digital Output DO (Discrete Input)

Open the ModbusRTU Master instruction addition page, find “Digital Output Settings”, DO name as “A Start”, register quantity as 1, register value does not need to be filled, click “Read Digital Output”. Finally, scroll to the bottom of the page and click “Apply” button.

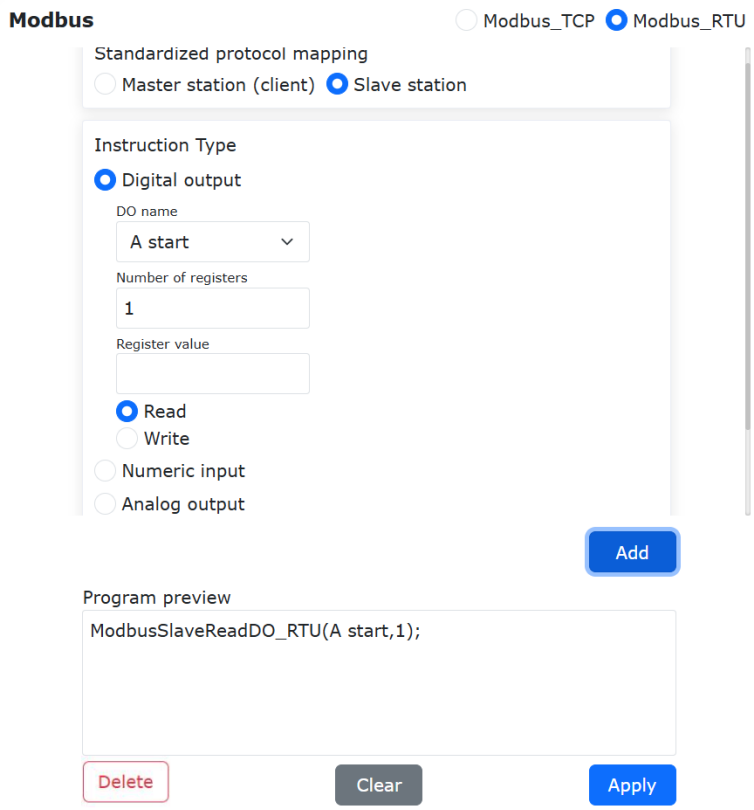


Figure 9.30-40 Configuring Read Single Digital Output, Applying Read Single Digital Output

Now the robot program “testModbusRTUSlave.lua” has added a robot Modbus Slave instruction to read a single digital output.



Figure 9.30-41 Program for Reading a Single Digital Output

Usually, after reading Modbus registers, the read values are stored in variables. Therefore, you need to define a variable to store the read value. Click the “Switch Mode” button to switch the robot LUA program to editable mode. Before the “ModbusSlaveReadDO_RTU” instruction, add a return value variable “AStartValue”. After executing the program, the read value will be stored in “AStartValue”.

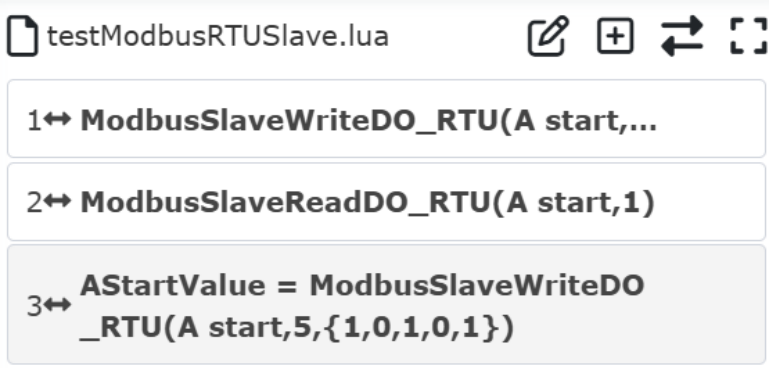


Figure 9.30-42 Storing a Single Digital Output Read in a Variable

Coil-type register values are only 0 or 1. In the robot program, you can perform different operations by judging different register values.

1.3.9.30.3.11 Read Multiple Digital Outputs DO (Discrete Inputs)

Open the ModbusRTU Master instruction addition page, find "Digital Output Settings", select DO name as "A Start", register quantity as 2, register value does not need to be filled, click "Read Digital Output". Finally, scroll to the bottom of the page and click "Apply" button.

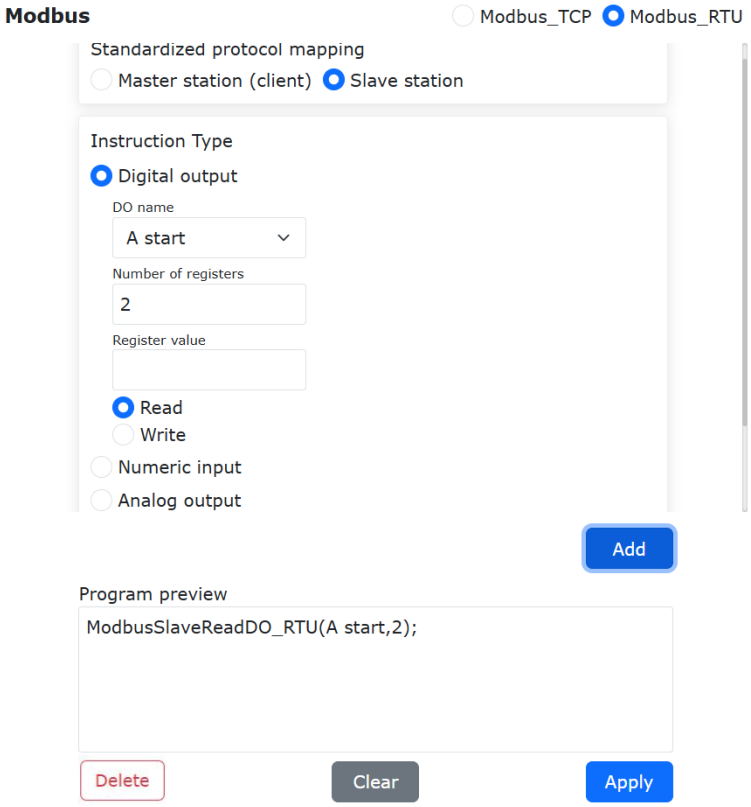


Figure 9.30-43 Configuring Read Multiple Digital Outputs, Applying Read Multiple Digital Outputs

Now the robot program "testModbusRTUSlave.lua" has added a robot Modbus Slave instruction to read multiple digital outputs.

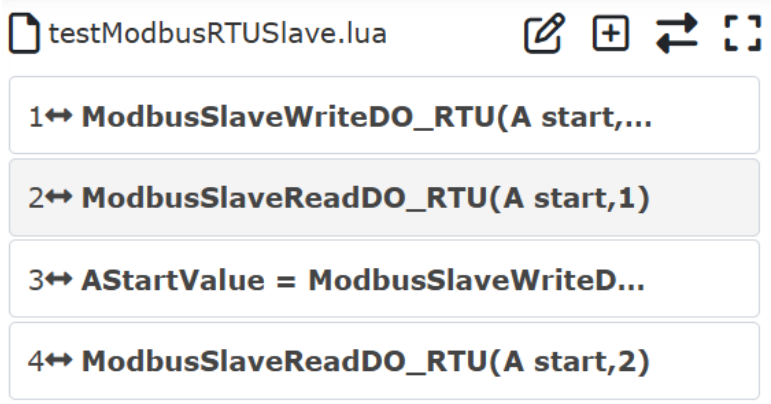


Figure 9.30-44 Program for Reading Multiple Digital Outputs

Click the "Switch Mode" button to switch the robot LUA program to editable mode. Since the read quantity is 2, you need to add two return value variables "value1,value2" before the "ModbusSlaveReadDO_RTU" instruction. After executing the program, the two digital output register values will be stored in these two variables. Similarly, you can judge the values of "value1" and "value2" to make the robot perform different actions.

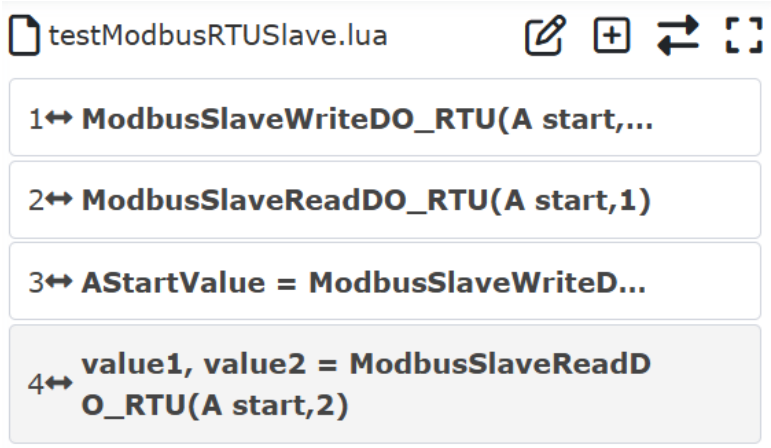


Figure 9.30-45 Storing Multiple Digital Output Reads in Variables

1.3.9.30.3.12 Read Digital Input DI (Coils)

Open the ModbusRTU Slave instruction addition page, find "Digital Input Settings", select DI name as "A Ready", register quantity as 2, click "Read Digital Input". Finally, scroll to the bottom of the page and click "Apply" button.

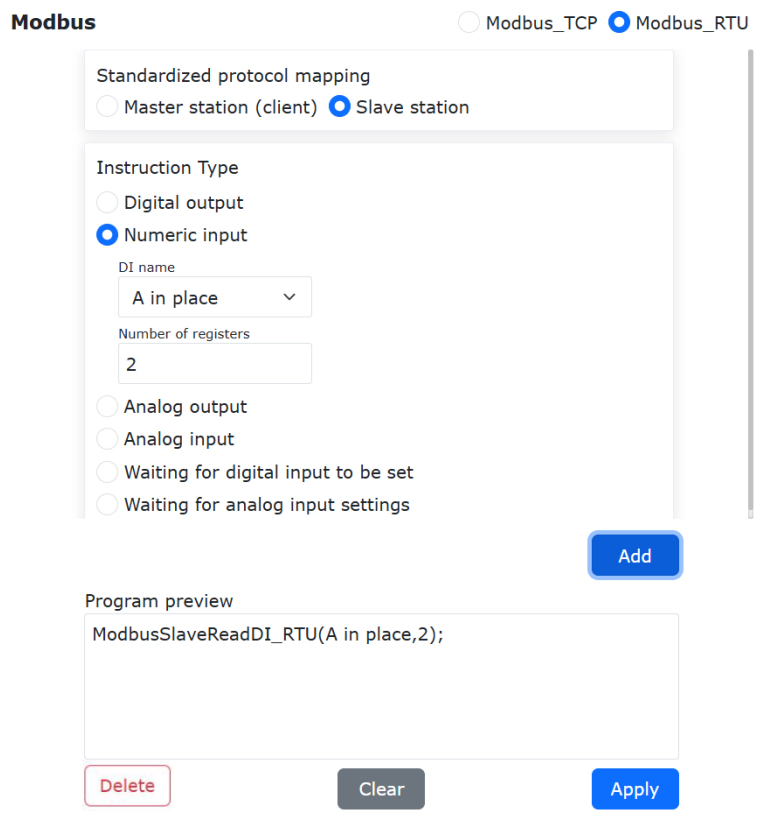


Figure 9.30-46 Configuring Read Digital Input, Applying Read Digital Input

Now the robot program "testModbusRTUSlave.lua" has added a robot Modbus Slave instruction to read digital inputs.

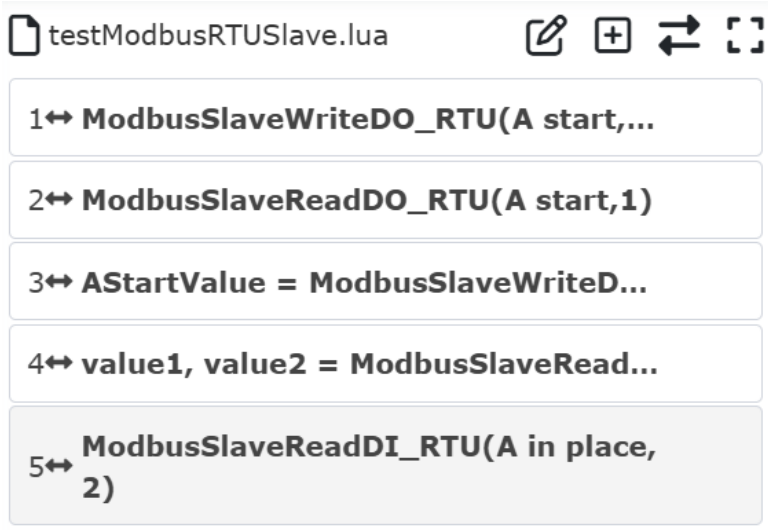


Figure 9.30-47 Program Instruction for Reading Digital Inputs

Click the "Switch Mode" button to switch the robot LUA program to editable mode. Before the "ModbusSlaveReadDI_RTU" instruction, add return value variables "AState,BState". After executing the program, the two digital input values will be stored in variables "AState" and "BState", respectively. You can control the robot to perform different operations by judging the variable values.



Figure 9.30-48 Program for Reading Digital Inputs

1.3.9.30.3.13 Read/Write Operations for Analog Output AO (Input Registers) and Analog Input AI (Holding Registers)

The read/write operations for analog outputs (input registers) and analog inputs (holding registers) are basically the same as those for digital outputs (discrete inputs) and digital inputs (coils). The difference is that the data range of the latter is limited to 0 or 1, while the former has a larger data range. Therefore, specific operations can refer to the writing of digital output and digital input programs. Here, only the program examples for reading analog inputs (Figure 3-24) and reading/writing analog outputs (Figure 3-25) are shown.

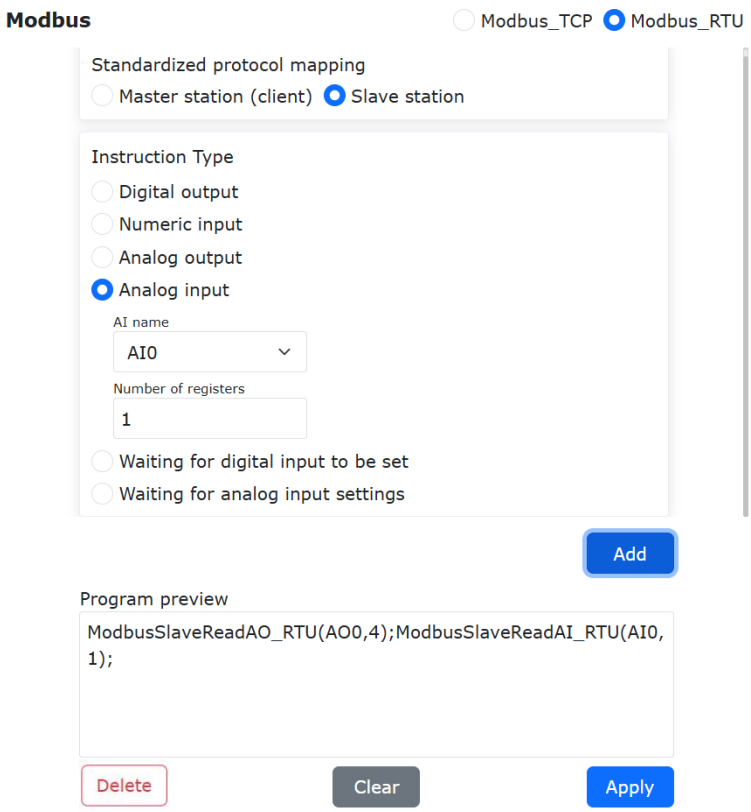


Figure 9.30-49 Reading Analog Inputs

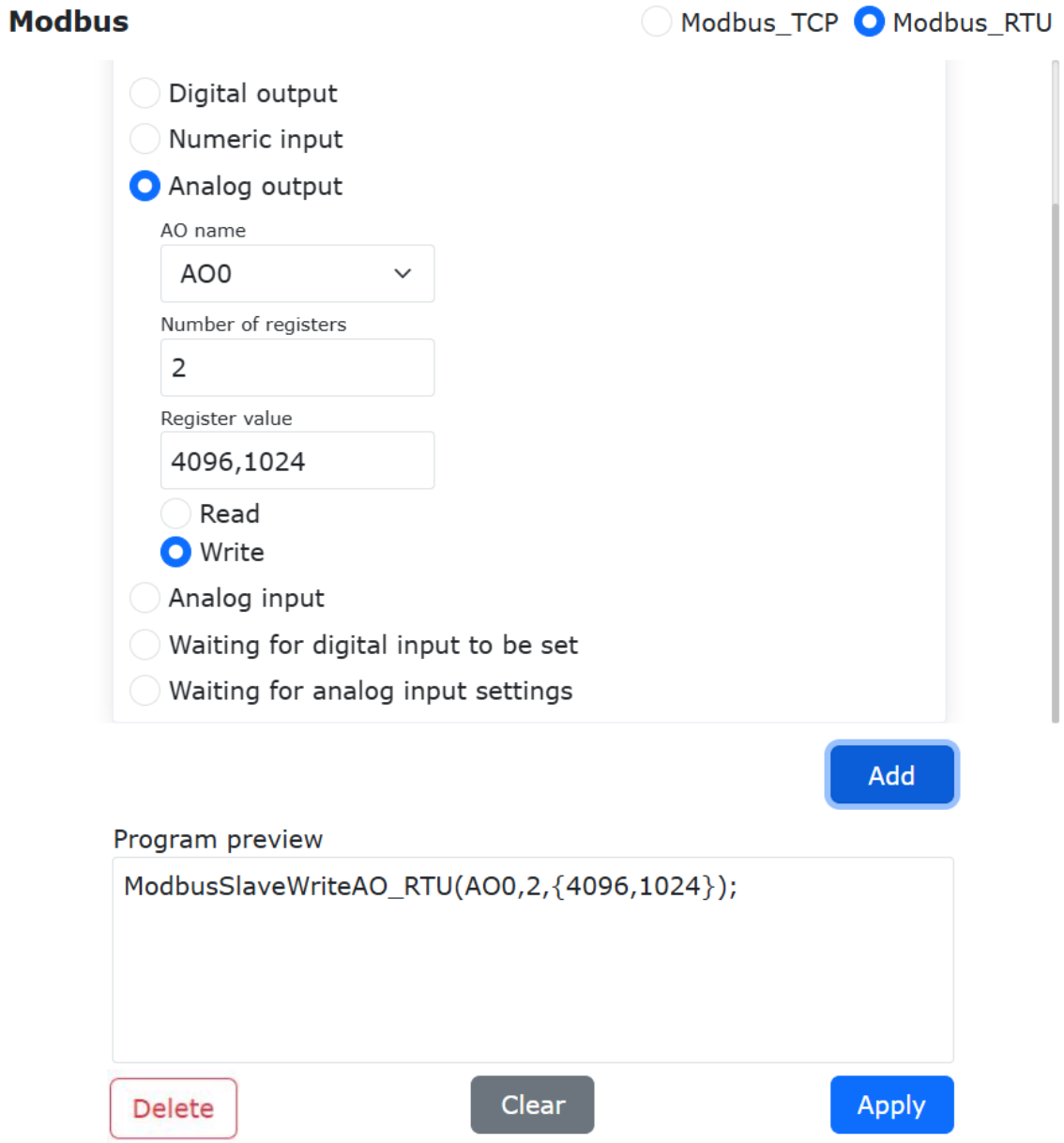


Figure 9.30-50 Reading/Writing Analog Outputs

1.3.9.30.3.14 Wait for Digital Input

Open the ModbusRTU Slave instruction addition page, find “Wait for Digital Input Settings”, select DI name as the configured “A Ready” register, wait state as “True”, timeout as 5000ms. Click “Add” button, then click “Apply” button.

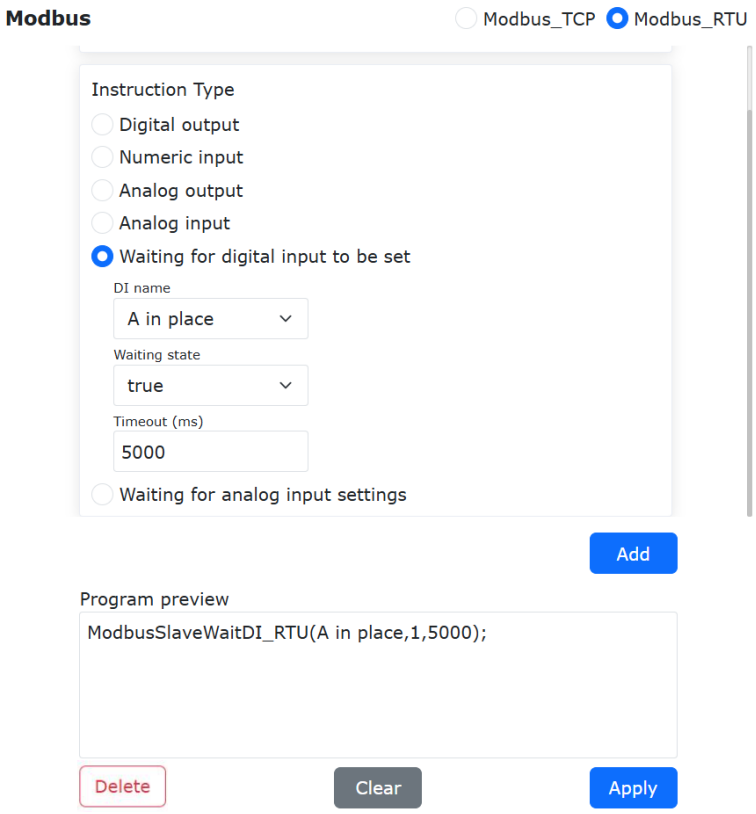


Figure 9.30-51 Adding a Wait for Digital Input Instruction

Now the robot program “testModbusRTUSlave.lua” has added a robot Modbus Slave instruction to wait for a digital input. After starting the program, the robot will wait for the “A Ready” coil register value of the slave to become true (value 1). Since the timeout is set to 5s, if the “A Ready” signal is still 0 after 5s, the robot program will report a timeout error and automatically stop running.

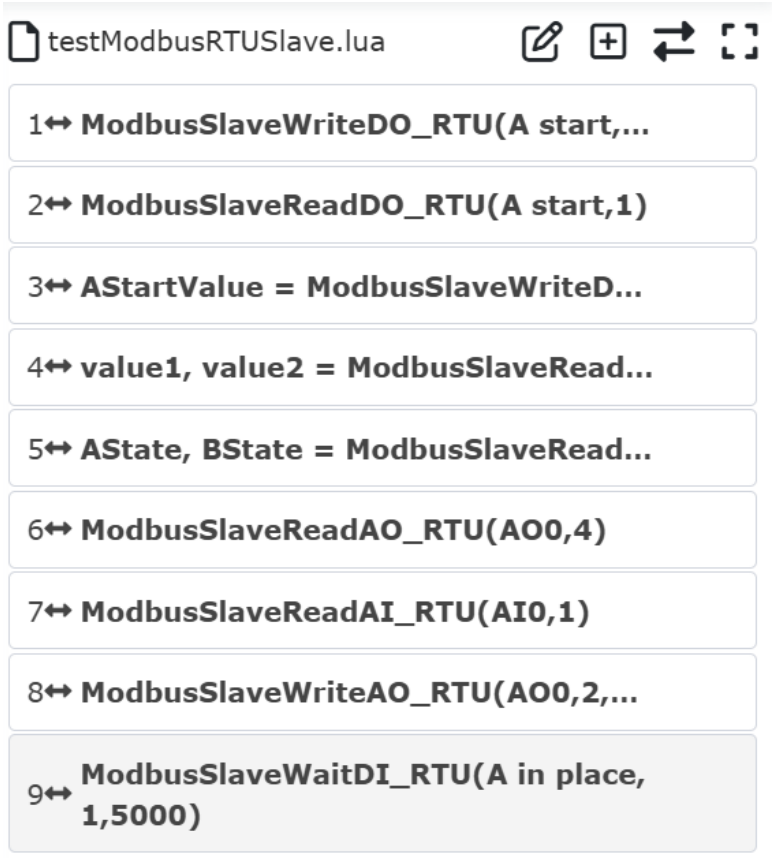


Figure 9.30-52 Program for Waiting for Digital Input

1.3.9.30.3.15 Wait for Analog Input

Open the ModbusRTU Slave instruction addition page, find “Wait for Analog Input Settings”, select AI name as the configured “Current” register, wait state as “>”, register value as 255, timeout as 5000ms. Click “Add” button, then click “Apply” button.

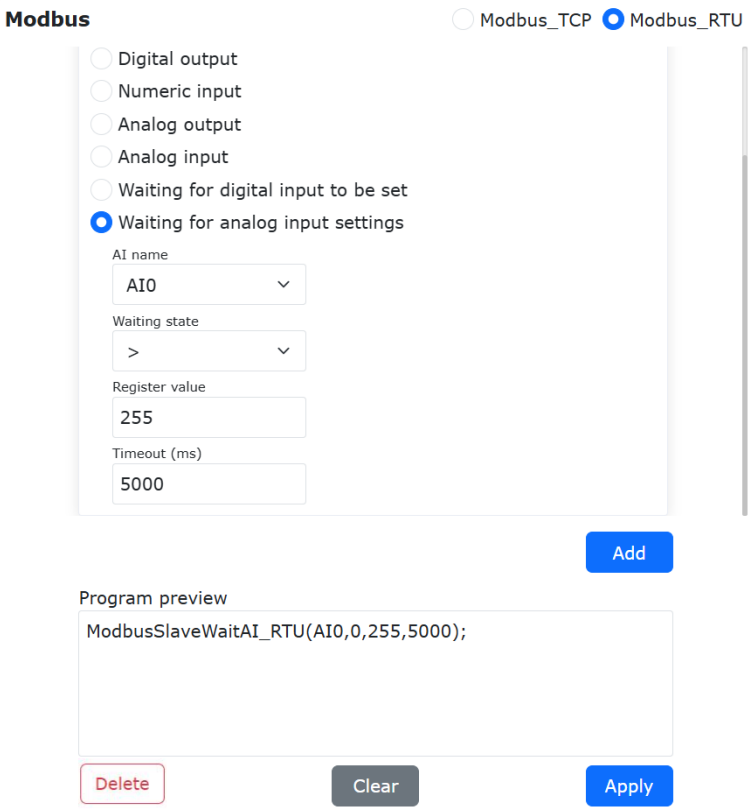


Figure 9.30-53 Adding a Wait for Analog Input Instruction

Now the robot program “testModbusRTUSlave.lua” has added a robot Modbus Slave instruction to wait for an analog input value. After starting the program, the robot will wait for the “Current” register value of the slave to be greater than 255. Since the timeout is set to 5s, if the “Current” signal is still not greater than 255 after 5s, the robot program will report a timeout error and automatically stop running.



Figure 9.30-54 Program for Waiting for Analog Input Register

Open the ModbusRTU Slave command addition page, find the “Wait for Analog Input Setting” (i.e., wait for AI input register setting). Select the AI name as the configured “Liquid Level” register, set the wait condition to “=”, the register value to 255, and the timeout to 5000ms. Click the “Add” button, and finally click the “Apply” button.

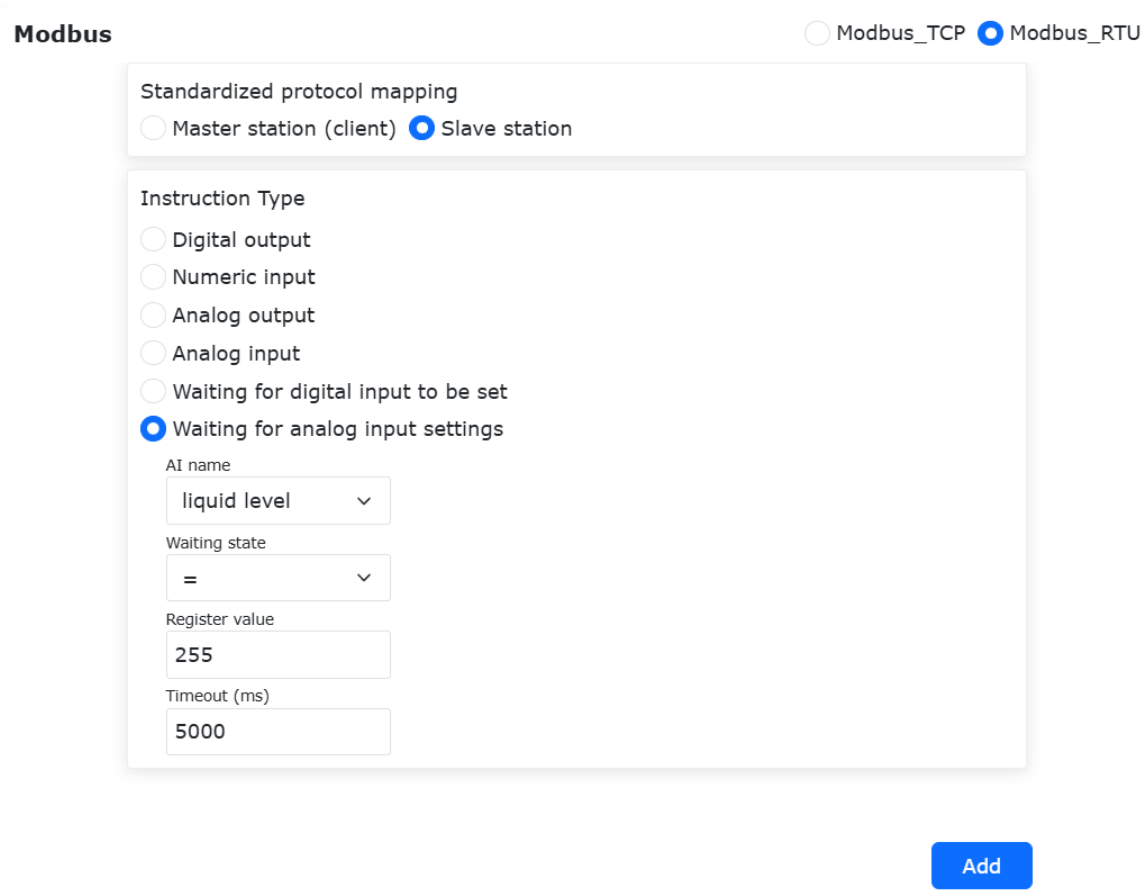


Figure 9.30-54-2 Add Wait for Analog Input

At this point, a Modbus RTU Slave command to wait for the AI input register value has been added to the robot program "test.lua". After starting the program, the robot will continuously wait for the "Liquid Level" register value to equal 255. Since the timeout is set to 5s, if the "Liquid Level" signal still does not equal 255 after the robot waits for 5 seconds, the robot program will report a timeout error and automatically stop running.

1.3.9.30.3.16 Robot Status Feedback and Control via ModbusRTU Slave

The input register addresses 0x1010-0x104C of the collaborative robot ModbusRTU Slave are used to feedback real-time robot status (see Appendix 1: ModbusRTU Slave Address Mapping Table for specific address definitions). You only need to read the corresponding register values with the master device to obtain the corresponding real-time robot status data.

The coil register addresses 0x4040-0x405C of the collaborative robot ModbusRTU Slave are used for the master device to control the robot (see Appendix 1: ModbusRTU Slave Address Mapping Table for specific address definitions). Take coil address 0x4054 as an example. This address function is "Start Program". When the robot is in automatic mode, the master device changes the value of address 0x4054 from 0 to 1, and the robot automatically starts running the currently configured program. Another example is coil address 0x4040, which is used to control the output of control box DO0. When the external master changes the coil address 0x4040 from 0 to 1, the control box DO0 automatically outputs effectively. Similarly, when the external master changes the coil address 0x4040 from 1 to 0, the control box DO0 output becomes invalid. On the ModbusRTU Slave settings page, click "Functional Digital Inputs (Coils)" to monitor all current functional digital inputs.

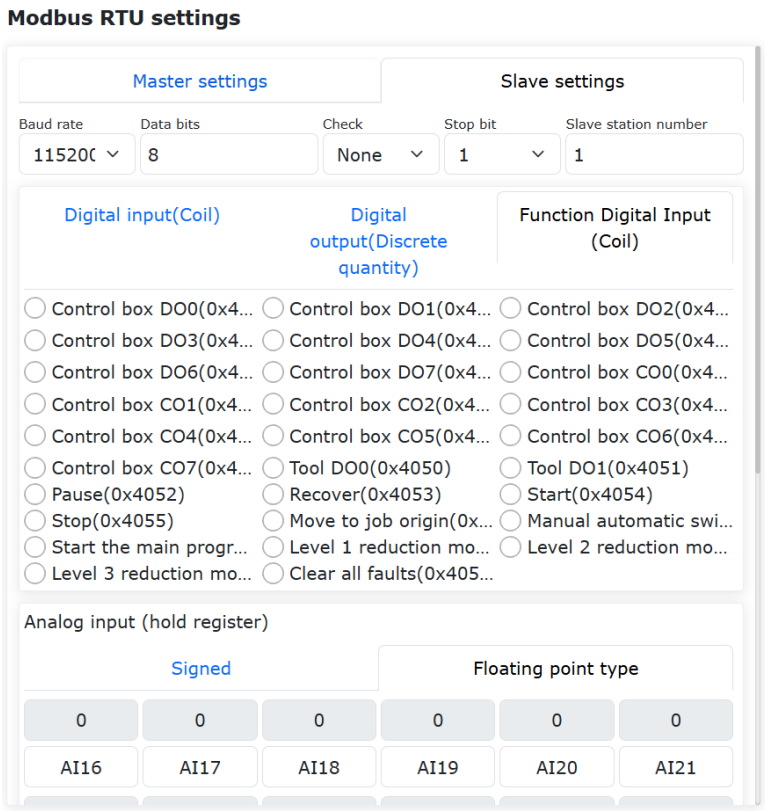


Figure 9.30-55 Digital input of the robot slave function

Attachment 1: Modbus Slave Address Mapping Table

Third-party Controller Address	Type	Name	Data Type	Function Code	R/W
0x3000	General Digital Output (Discrete)	DO0	BOOL	0x02	Read Only
0x3001	General Digital Output (Discrete)	DO1	BOOL	0x02	Read Only
0x3002	General Digital Output (Discrete)	DO2	BOOL	0x02	Read Only
0x3003	General Digital Output (Discrete)	DO3	BOOL	0x02	Read Only
…	General Digital Output (Discrete)	…	BOOL	0x02	Read Only
0x303F	General Digital Output (Discrete)	DO127	BOOL	0x02	Read Only
0x4000	General Digital Input (Coil)	DI0	BOOL	0x01, 0x05, 0x0F	R/W
0x4001	General Digital Input (Coil)	DI1	BOOL	0x01, 0x05, 0x0F	R/W
0x4002	General Digital Input (Coil)	DI2	BOOL	0x01, 0x05, 0x0F	R/W
0x4003	General Digital Input (Coil)	DI3	BOOL	0x01, 0x05, 0x0F	R/W

continues on next page

Table 7 – continued from previous page

…	General Digital Input (Coil)	…	BOOL	0x01, 0x05, 0x0F	R/W
0x403F	General Digital Input (Coil)	DI64	BOOL	0x01, 0x05, 0x0F	R/W
0x4040	Robot Control	Control Box DO0	BOOL	0x01, 0x05, 0x0F	R/W
0x4041	Robot Control	Control Box DO1	BOOL	0x01, 0x05, 0x0F	R/W
0x4042	Robot Control	Control Box DO2	BOOL	0x01, 0x05, 0x0F	R/W
0x4043	Robot Control	Control Box DO3	BOOL	0x01, 0x05, 0x0F	R/W
0x4044	Robot Control	Control Box DO4	BOOL	0x01, 0x05, 0x0F	R/W
0x4045	Robot Control	Control Box DO5	BOOL	0x01, 0x05, 0x0F	R/W
0x4046	Robot Control	Control Box DO6	BOOL	0x01, 0x05, 0x0F	R/W
0x4047	Robot Control	Control Box DO7	BOOL	0x01, 0x05, 0x0F	R/W
0x4048	Robot Control	Control Box CO0	BOOL	0x01, 0x05, 0x0F	R/W
0x4049	Robot Control	Control Box CO1	BOOL	0x01, 0x05, 0x0F	R/W
0x404A	Robot Control	Control Box CO2	BOOL	0x01, 0x05, 0x0F	R/W
0x404B	Robot Control	Control Box CO3	BOOL	0x01, 0x05, 0x0F	R/W
0x404C	Robot Control	Control Box CO4	BOOL	0x01, 0x05, 0x0F	R/W
0x404D	Robot Control	Control Box CO5	BOOL	0x01, 0x05, 0x0F	R/W
0x404E	Robot Control	Control Box CO6	BOOL	0x01, 0x05, 0x0F	R/W
0x404F	Robot Control	Control Box CO7	BOOL	0x01, 0x05, 0x0F	R/W
0x4050	Robot Control	Tool DO0	BOOL	0x01, 0x05, 0x0F	R/W
0x4051	Robot Control	Tool DO1	BOOL	0x01, 0x05, 0x0F	R/W
0x4052	Robot Control	Pause	BOOL	0x01, 0x05, 0x0F	R/W
0x4053	Robot Control	Resume	BOOL	0x01, 0x05, 0x0F	R/W
0x4054	Robot Control	Start	BOOL	0x01, 0x05, 0x0F	R/W
0x4055	Robot Control	Stop	BOOL	0x01, 0x05, 0x0F	R/W
0x4056	Robot Control	Move to Home Position	BOOL	0x01, 0x05, 0x0F	R/W
0x4057	Robot Control	Manual/Auto Toggle	BOOL	0x01, 0x05, 0x0F	R/W
0x4058	Robot Control	Start Main Program	BOOL	0x01, 0x05, 0x0F	R/W
0x4059	Robot Control	Level 1 Reduction Mode	BOOL	0x01, 0x05, 0x0F	R/W
0x405A	Robot Control	Level 2 Reduction Mode	BOOL	0x01, 0x05, 0x0F	R/W
0x405B	Robot Control	Level 3 Reduction Mode (Stop)	BOOL	0x01, 0x05, 0x0F	R/W
0x405C	Robot Control	Clear All Faults	BOOL	0x01, 0x05, 0x0F	R/W
0x405D	Robot Control	Reserved	BOOL	0x01, 0x05, 0x0F	R/W
0x405E	Robot Control	Reserved	BOOL	0x01, 0x05, 0x0F	R/W
0x1000	Analog Input	AO0	INT16	0x04	Read Only
0x1001	Analog Input	AO1	INT16	0x04	Read Only
0x1002	Analog Input	AO2	INT16	0x04	Read Only
…	Analog Input	…	INT16	0x04	Read Only
0x100F	Analog Input	AO15	INT16	0x04	Read Only
0x1010	Robot Status	Enable Status (0-Disabled, 1-Enabled)	UINT16	0x04	Read Only
0x1011	Robot Status	Robot Mode (1-Manual, 0-Auto)	UINT16	0x04	Read Only

continues on next page

Table 7 – continued from previous page

0x1012	Robot Status	Robot Operation Status (1-Stop, 2-Running, 3-Paused, 4-Dragging)	UINT16	0x04	Read Only
0x1013	Robot Status	Tool Number	UINT16	0x04	Read Only
0x1014	Robot Status	Workpiece Number	UINT16	0x04	Read Only
0x1015	Robot Status	Emergency Stop Status (0-Normal, 1-Emergency Stop)	UINT16	0x04	Read Only
0x1016	Robot Status	Soft Limit Fault	UINT16	0x04	Read Only
0x1017	Robot Status	Main Fault Code	UINT16	0x04	Read Only
0x1018	Robot Status	Sub Fault Code	UINT16	0x04	Read Only
0x1019	Robot Status	Collision Detection (1-Collision, 0-No Collision)	UINT16	0x04	Read Only
0x101A	Robot Status	Motion Completion Signal	UINT16	0x04	Read Only
0x101B	Robot Status	Safety Stop Signal SI0	UINT16	0x04	Read Only
0x101C	Robot Status	Safety Stop Signal SI1	UINT16	0x04	Read Only
0x101D	Robot Status	Control Box Analog Input AI0	UINT16	0x04	Read Only
0x101E	Robot Status	Control Box Analog Input AI1	UINT16	0x04	Read Only
0x101F	Robot Status	Tool Analog Input AI0	UINT16	0x04	Read Only
0x1020	Robot Status	Control Box Analog Output AO0	UINT16	0x04	Read Only
0x1021	Robot Status	Control Box Analog Output AO1	UINT16	0x04	Read Only
0x1022	Robot Status	Tool Analog Output AO0	UINT16	0x04	Read Only
0x1023	Robot Status	Control Box Digital Input (Bit0-Bit7: DI0-DI7, Bit8-Bit15: CI0-CI7)	UINT16	0x04	Read Only
0x1024	Robot Status	Tool Digital Input (Bit0-Bit15: DI0-DI15)	UINT16	0x04	Read Only
0x1025	Robot Status	Control Box Digital Output (Bit0-Bit7: DO0-DO7, Bit8-Bit15: CO0-CO7)	UINT16	0x04	Read Only
0x1026	Robot Status	Tool Digital Output (Bit0-Bit15: DO0-DO15)	UINT16	0x04	Read Only
0x1027	Robot Status	TCP Speed	FLOAT32 (Big Endian)	0x04	Read Only

continues on next page

Table 7 – continued from previous page

0x1028	Robot Status	TCP Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x1029	Robot Status	Joint 1 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x102A	Robot Status	Joint 1 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x102B	Robot Status	Joint 2 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x102C	Robot Status	Joint 2 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x102D	Robot Status	Joint 3 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x102E	Robot Status	Joint 3 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x102F	Robot Status	Joint 4 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x1030	Robot Status	Joint 4 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x1031	Robot Status	Joint 5 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x1032	Robot Status	Joint 5 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x1033	Robot Status	Joint 6 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x1034	Robot Status	Joint 6 Position	FLOAT32 (Big Endian)	0x04	Read Only
0x1035	Robot Status	Joint 1 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x1036	Robot Status	Joint 1 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x1037	Robot Status	Joint 2 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x1038	Robot Status	Joint 2 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x1039	Robot Status	Joint 3 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x103A	Robot Status	Joint 3 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x103B	Robot Status	Joint 4 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x103C	Robot Status	Joint 4 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x103D	Robot Status	Joint 5 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x103E	Robot Status	Joint 5 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x103F	Robot Status	Joint 6 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x1040	Robot Status	Joint 6 Speed	FLOAT32 (Big Endian)	0x04	Read Only
0x1041	Robot Status	TCP Position X	FLOAT32 (Big Endian)	0x04	Read Only

continues on next page

Table 7 – continued from previous page

0x1042	Robot Status	TCP Position X	FLOAT32 (Big Endian)	0x04	Read Only
0x1043	Robot Status	TCP Position Y	FLOAT32 (Big Endian)	0x04	Read Only
0x1044	Robot Status	TCP Position Y	FLOAT32 (Big Endian)	0x04	Read Only
0x1045	Robot Status	TCP Position Z	FLOAT32 (Big Endian)	0x04	Read Only
0x1046	Robot Status	TCP Position Z	FLOAT32 (Big Endian)	0x04	Read Only
0x1047	Robot Status	TCP Position RX	FLOAT32 (Big Endian)	0x04	Read Only
0x1048	Robot Status	TCP Position RX	FLOAT32 (Big Endian)	0x04	Read Only
0x1049	Robot Status	TCP Position RY	FLOAT32 (Big Endian)	0x04	Read Only
0x104A	Robot Status	TCP Position RY	FLOAT32 (Big Endian)	0x04	Read Only
0x104B	Robot Status	TCP Position RZ	FLOAT32 (Big Endian)	0x04	Read Only
0x104C	Robot Status	TCP Position RZ	FLOAT32 (Big Endian)	0x04	Read Only
0x104D	Analog Input	AO16	FLOAT32 (Big Endian)	0x04	Read Only
0x104E	Analog Input	AO16	FLOAT32 (Big Endian)	0x04	Read Only
0x104F	Analog Input	AO17	FLOAT32 (Big Endian)	0x04	Read Only
0x1050	Analog Input	AO17	FLOAT32 (Big Endian)	0x04	Read Only
…	Analog Input	…	FLOAT32 (Big Endian)	0x04	Read Only
0x106B	Analog Input	AO31	FLOAT32 (Big Endian)	0x04	Read Only
0x106C	Analog Input	AO31	FLOAT32 (Big Endian)	0x04	Read Only
0x2000	Analog Output	AI0	INT16	0x03, 0x06, 0x10	R/W
0x2001	Analog Output	AI1	INT16	0x03, 0x06, 0x10	R/W
0x2002	Analog Output	AI2	INT16	0x03, 0x06, 0x10	R/W
…	Analog Output	…	INT16	0x03, 0x06, 0x10	R/W
0x200F	Analog Output	AI15	INT16	0x03, 0x06, 0x10	R/W
0x2010	Analog Output	AI16	FLOAT32 (Big Endian)	0x03, 0x06, 0x10	R/W
0x2011	Analog Output	AI16	FLOAT32 (Big Endian)	0x03, 0x06, 0x10	R/W
0x2012	Analog Output	AI17	FLOAT32 (Big Endian)	0x03, 0x06, 0x10	R/W
0x2013	Analog Output	AI17	FLOAT32 (Big Endian)	0x03, 0x06, 0x10	R/W
…	Analog Output	…	FLOAT32 (Big Endian)	0x03, 0x06, 0x10	R/W

continues on next page

Table 7 – continued from previous page

0x202E	Analog Output	AI31	FLOAT32 (Big Endian)	0x03, 0x06, 0x10	R/W
0x202F	Analog Output	AI31	FLOAT32 (Big Endian)	0x03, 0x06, 0x10	R/W

1.3.9.31 Protection Based on 6-Axis Force Sensor Posture Compliance Function

1.3.9.31.1 Overview

Current FR robots have no maximum adjustment angle limit in posture compliance function under constant force FT_Control. When the 6-axis force sensor receives external torque, the robot end will continuously deviate, which may cause hazards.

This enhancement adds maximum adjustment angle limit on top of the FT_Control posture compliance function, allowing custom threshold setting to make posture compliance smoother.

1.3.9.31.2 Operation Procedure

Step1: Click “Initial Setup”->”Basic”->”Coordinate System”->”Tool” to enter the tool coordinate system setup interface. Select “Coordinate System Name” and configure parameters for the end tool coordinate system.

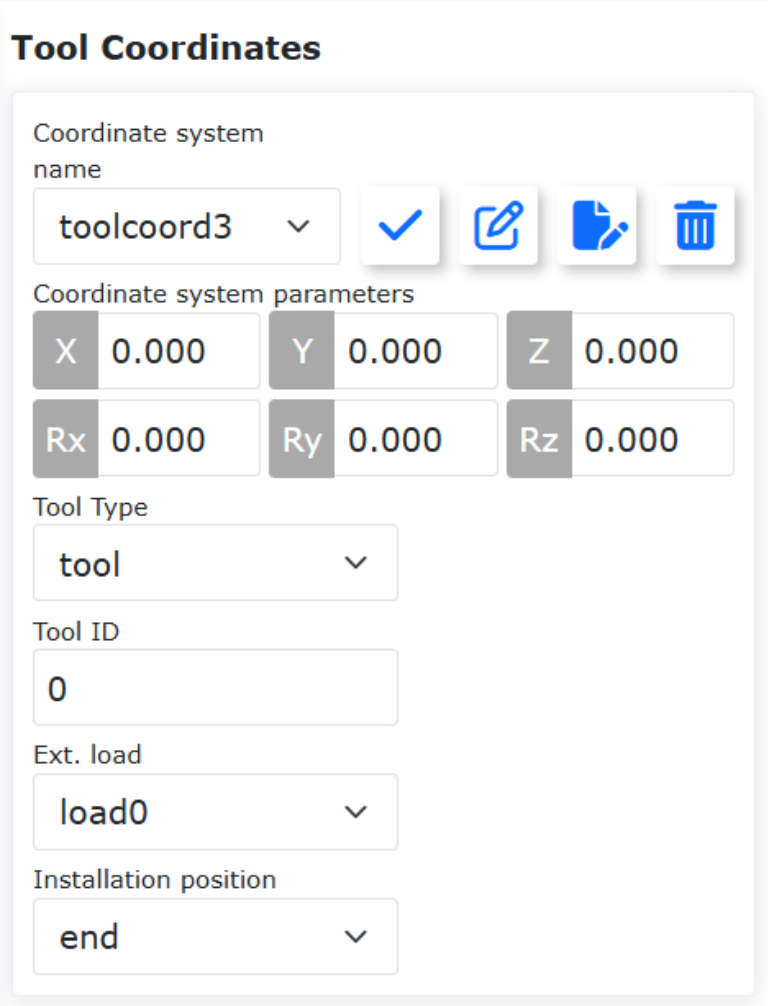


Figure 9.31-1 Tool Coordinate System Setup

Step2: Click “Teach Programming”->”Program Editing” to write constant force control Lua script. Select “Force Control Set”->”Control”, add force control motion command, set “Posture Compliance” to ON, and configure maximum adjustment angle as the threshold.

F/T

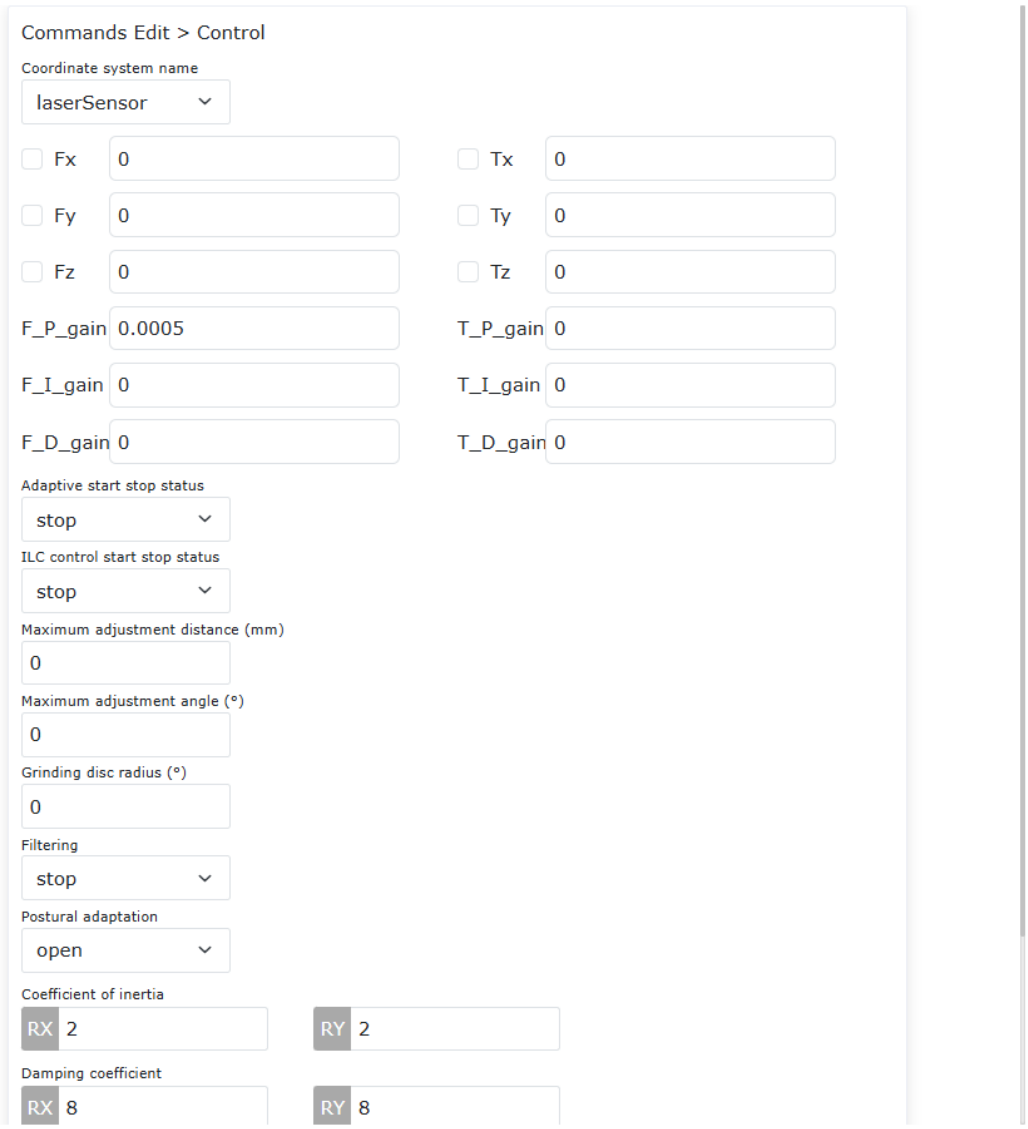


Figure 9.31-2 Force Control Motion Command

Step3: On web interface click “FT” to set 6-axis force sensor reference coordinate system. Select reference coordinate system as “Custom Coordinate System” and configure corresponding parameters. When posture compliance adjusts around tool coordinate system, set reference coordinate parameters to “0”; when adjusting around end flange coordinate system, set parameters to match the end tool coordinate system.

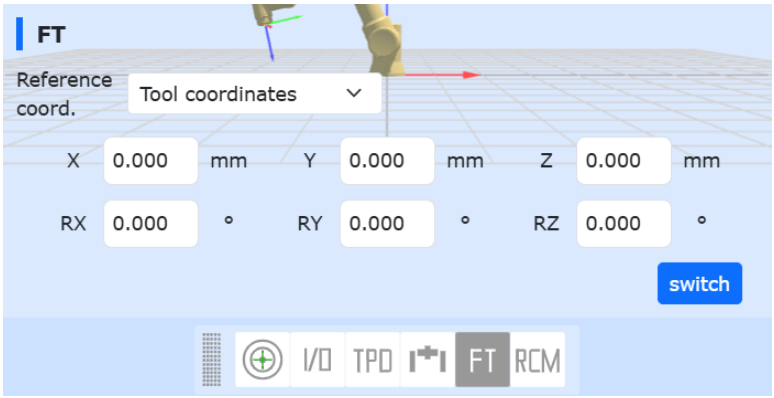


Figure 9.31-3 Setting 6-Axis Force Sensor Reference Coordinate System

Step4: Run the script to observe posture compliance effect. The adjustment angle under constant force will be limited within the custom maximum adjustment angle range.

1.3.9.32 Socket Communication Interface Function

1.3.9.32.1 Socket Configuration

When using the Socket communication interface function, after the robot is powered on, you need to first access the web page to configure the Socket protocol. The configuration is saved after power-off.

Click “Teach Program” - “Program Editing”, then click “Socket Network Debugging” in the upper right menu bar to enter the Socket configuration interface. Click “Add Socket” to configure Socket parameters. Up to four Sockets can be added.

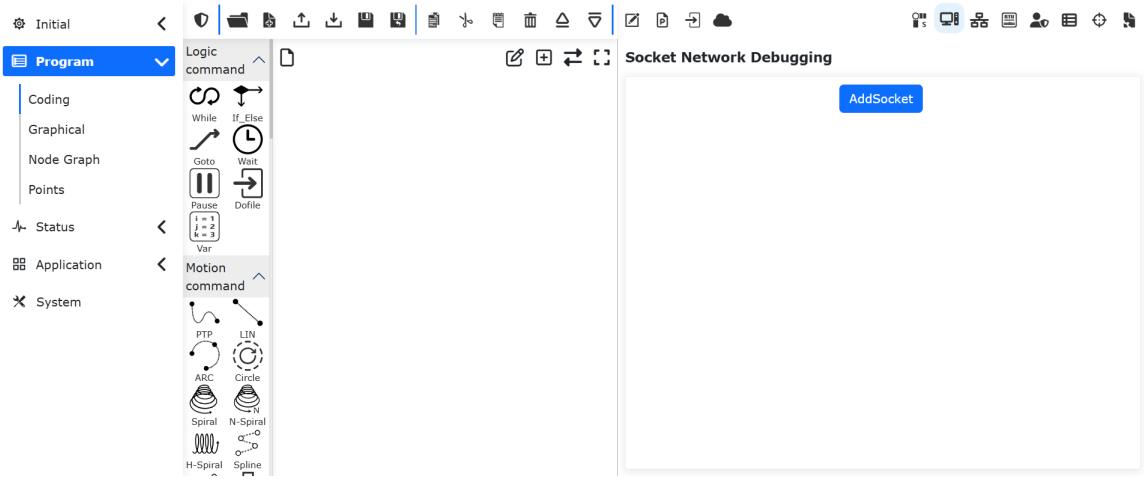


Figure 9.32-1 Socket Network Debugging Interface

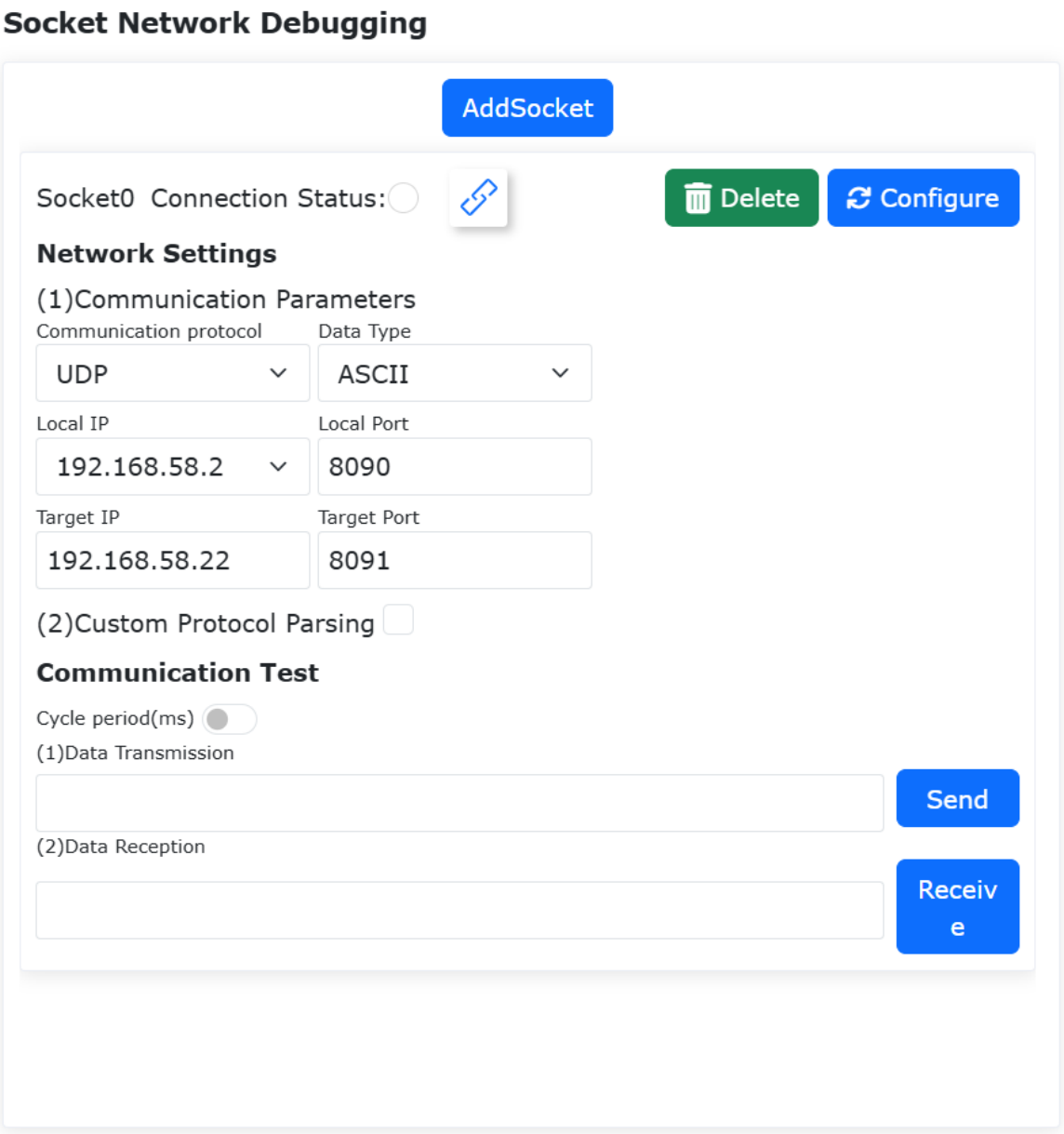


Figure 9.32-2 Socket Configuration Parameters Interface

1.3.9.32.1.1 Communication Parameter Settings

The communication protocol supports UDP, TCP server, and TCP client.

Data types support both ASCII and HEX. After configuring the data type, all data transmission and reception for that Socket connection will be processed according to the configured type.

1.3.9.32.1.2 Heartbeat Detection Mechanism

The heartbeat detection mechanism is only applicable to TCP server and TCP client.

The heartbeat detection mechanism uses the Keepalive mechanism to detect and maintain the active state of the connection, preventing accidental interruption of long-idle connections. It mainly includes the following parameters:

- Probe interval: How long to wait before sending keepalive probe packets (in seconds);
- Probe time: Interval between probe packets (in seconds);
- Probe count: Maximum number of probe packets to send.

1.3.9.32.1.3 Reconnection Mechanism

The reconnection mechanism is only applicable to TCP client.

When the reconnection mechanism is enabled and the TCP client detects a server disconnection upon startup, it will actively attempt to reconnect. If the connection cannot be established after reaching the maximum reconnection attempts, the connection will be dropped. It mainly includes the following parameters:

- Reconnection interval: Reconnection interval time (in ms, recommended to be in seconds);
- Maximum reconnection attempts: Maximum number of reconnection attempts.

1.3.9.32.1.4 Custom Protocol Parsing

When custom protocol parsing is enabled, the transmitted and received data will be encapsulated or parsed according to the protocol configuration.

Custom protocols can be automatically generated based on configuration parameters. In ASCII mode, frame header, frame count, data length, and frame tail combinations are supported. Delimiters can be used for data segmentation. In HEX mode, frame header, frame count, data length, checksum method, and frame tail combinations are supported.

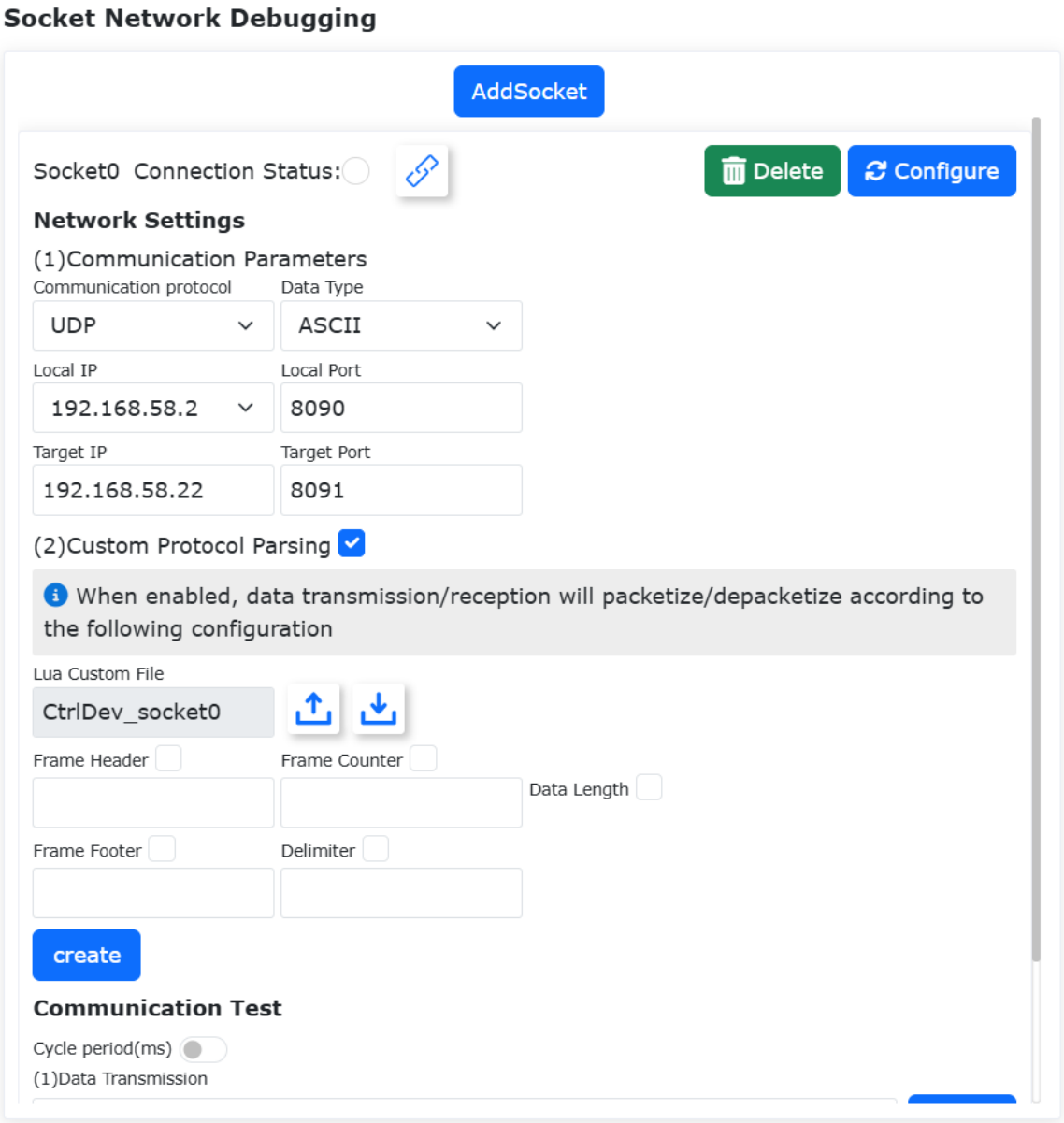


Figure 9.32-3 ASCII Mode Custom Protocol Configuration

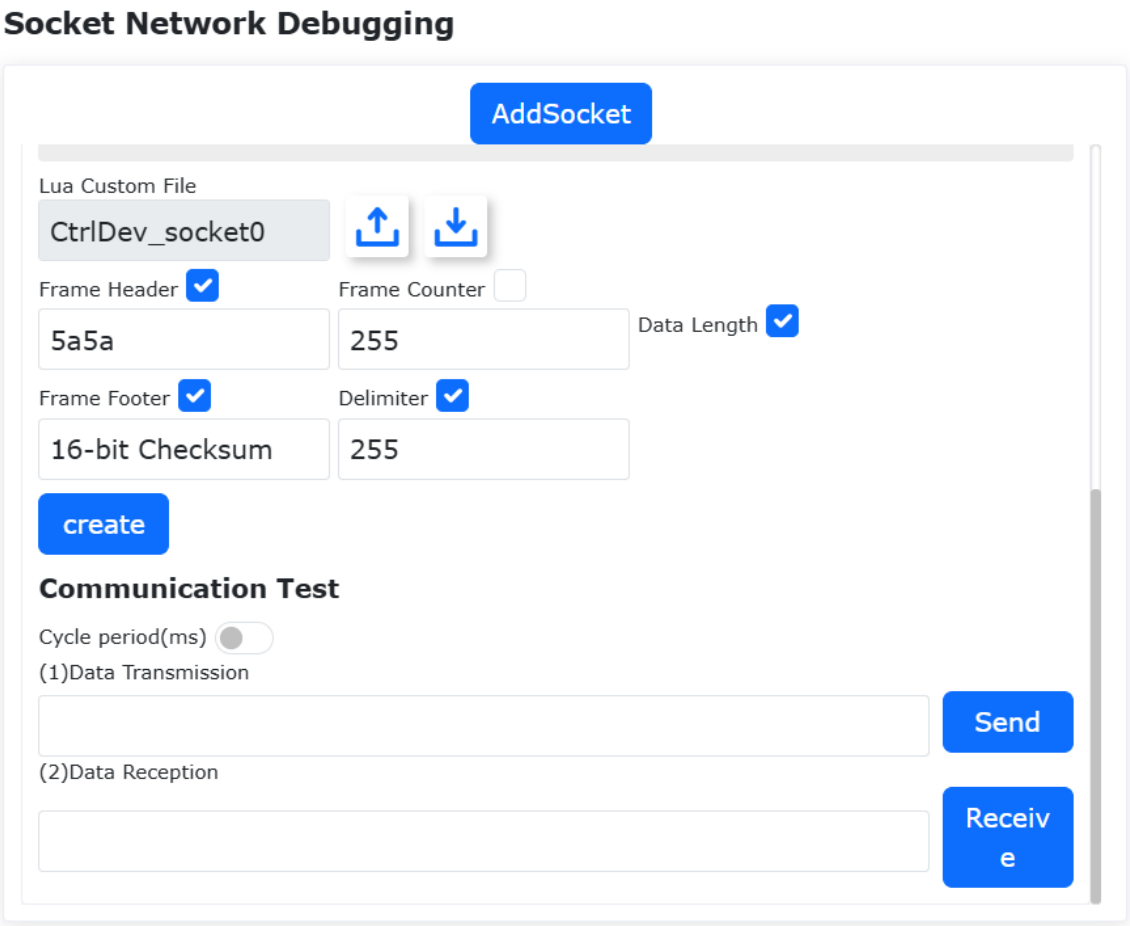


Figure 9.32-4 HEX Mode Custom Protocol Configuration

After configuring the custom protocol, click the "Generate" button to automatically generate the corresponding Lua file. The Lua file supports import and export functions. The protocol type can be flexibly configured by modifying the file code.

1.3.9.32.2 Socket Connection

1.3.9.32.2.1 Interface Connection Display

After configuring Socket information, the Socket connection can be established. The connection status includes the following three states:

- White: Connection not established.

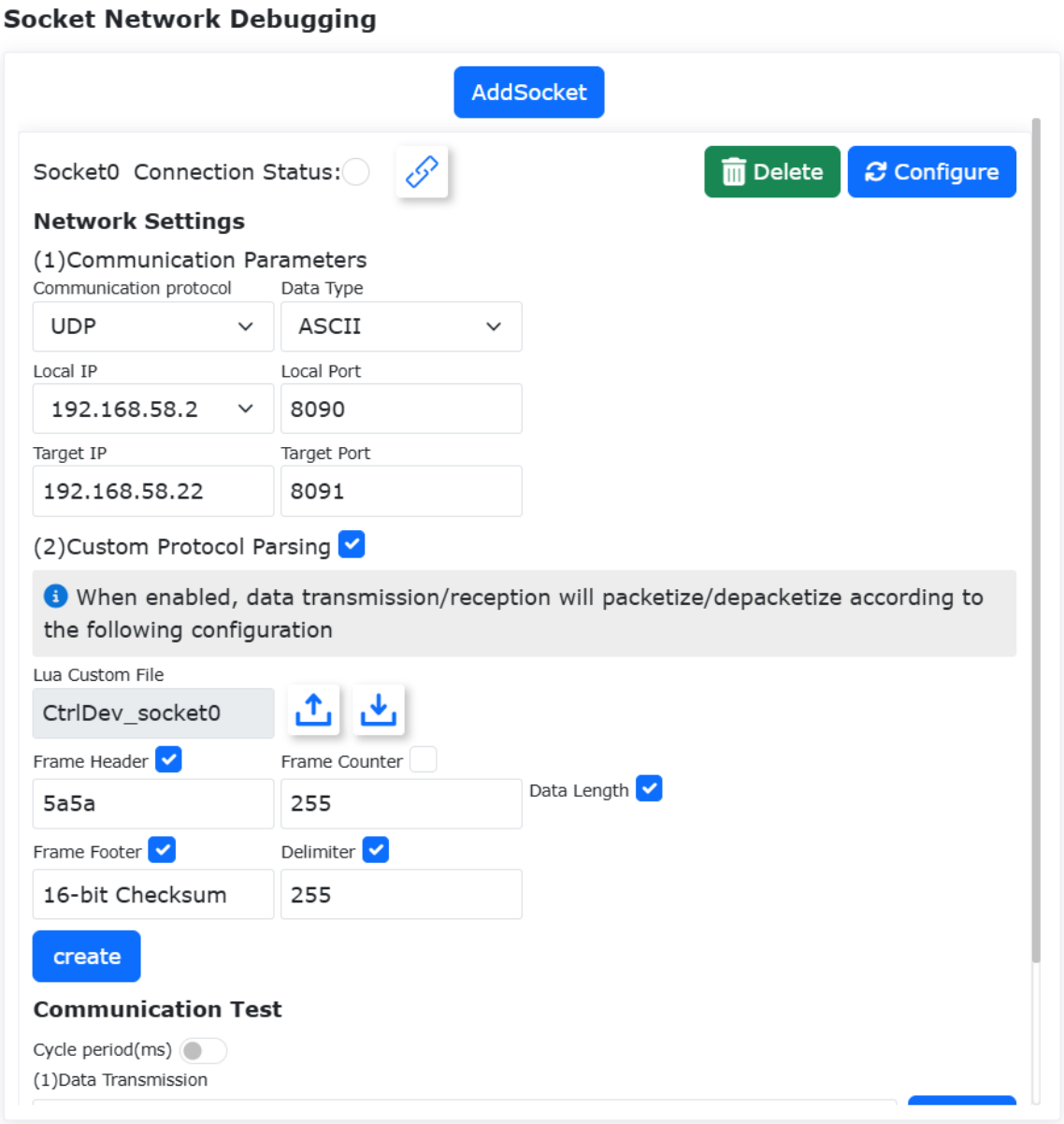


Figure 9.32-5 Disconnected State

- Yellow: TCP server waiting for connection or TCP client requesting connection.

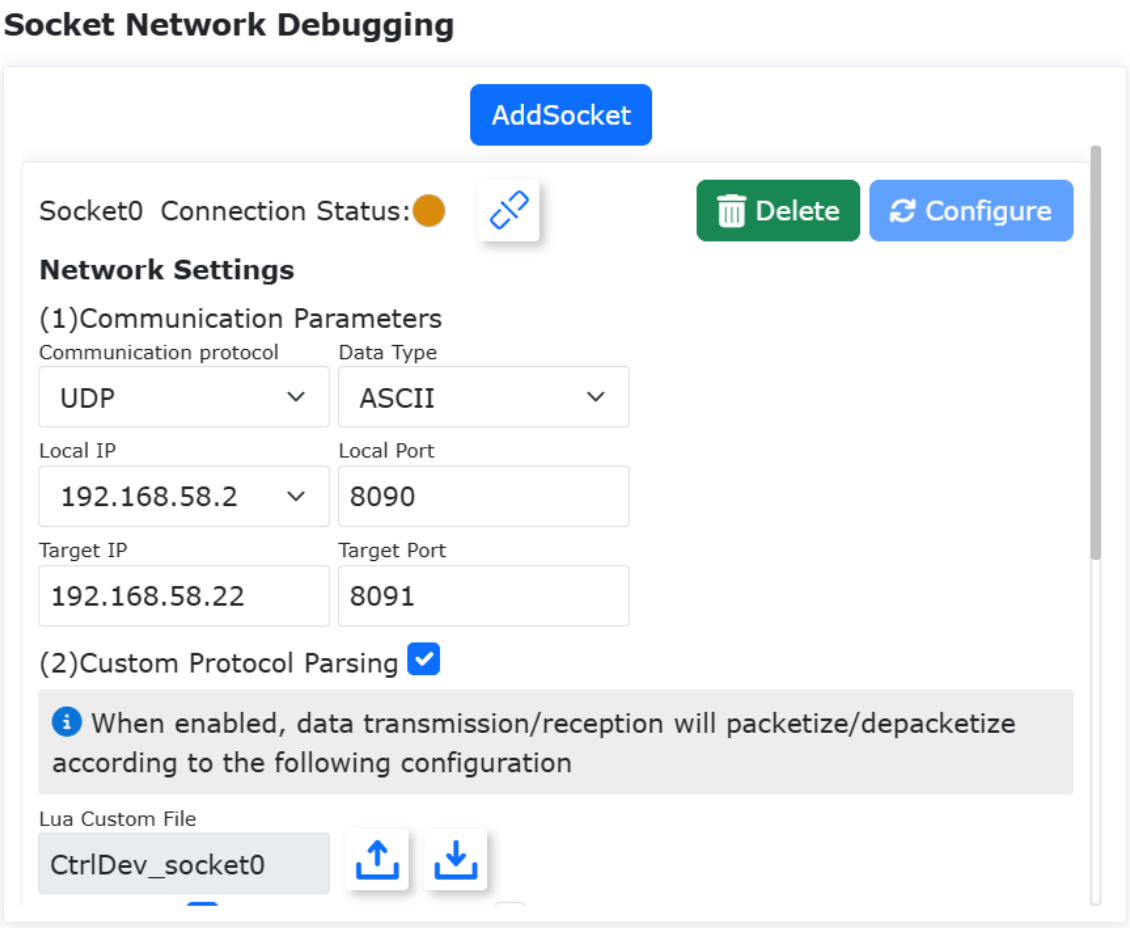


Figure 9.32-6 Waiting for Connection State

- Green: Connection successful.

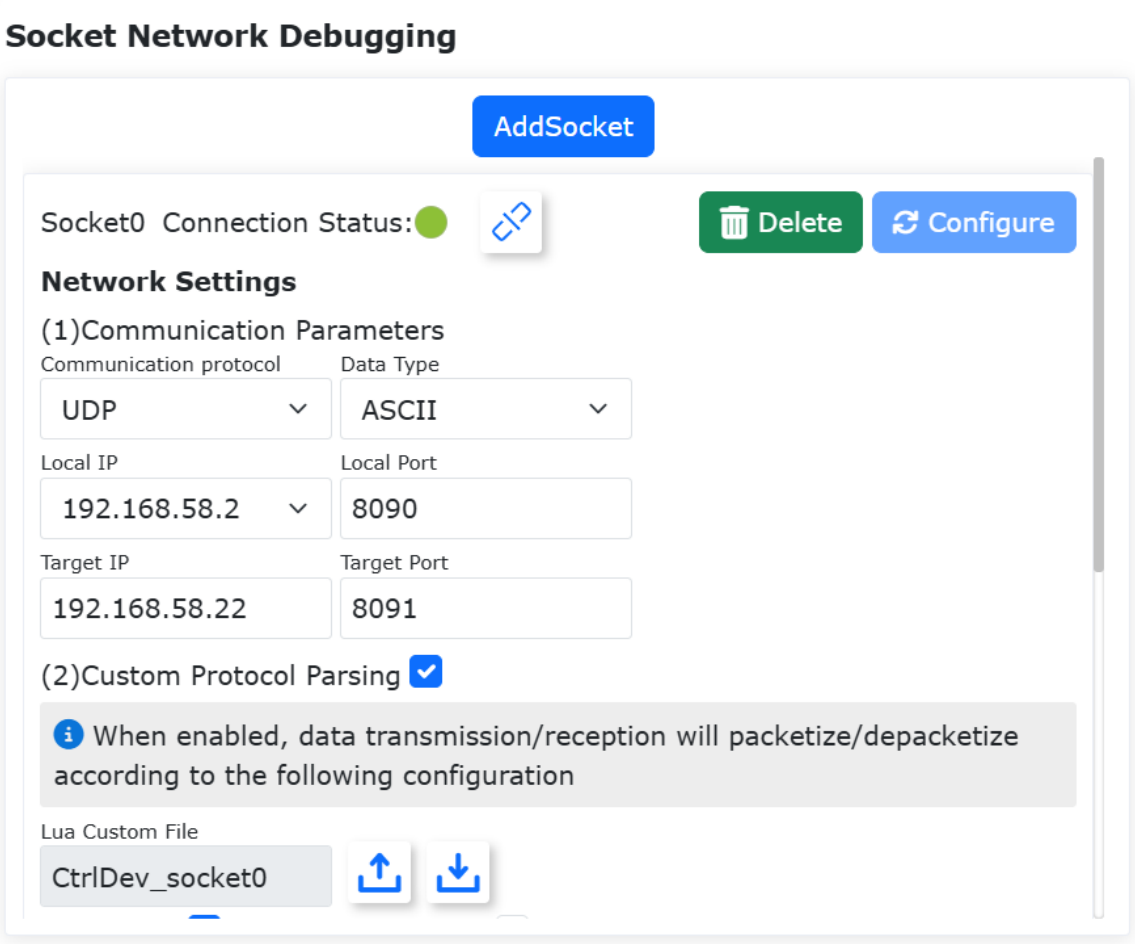


Figure 9.32-7 Connected State

1.3.9.32.2.2 Connection Command Module

Click “Teach Program” - “Program Editing” - “Communication Commands”, select “Socket” command to generate commands for opening and closing Socket connections for Lua programming. SocketID can only select already configured Socket connections.

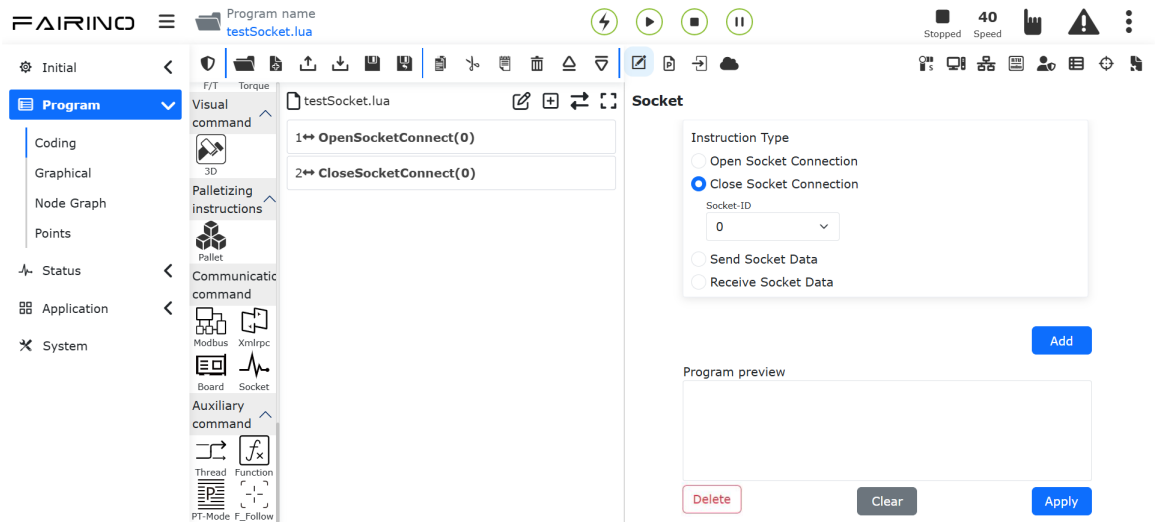


Figure 9.32-8 Socket Connection Command Module

Command details:

- Open connection command: OpenSockeConnect(id);
- Parameter id: Pre-configured socket ID, return value 0 for success;
- Close connection command: CloseSockeConnect(id);
- Parameter id: Pre-configured socket ID, return value 0 for success.

1.3.9.32.3 Socket Communication

1.3.9.32.3.1 Communication Testing

The interface provides communication testing for data transmission and reception testing, as shown in Figure 3-1 below.

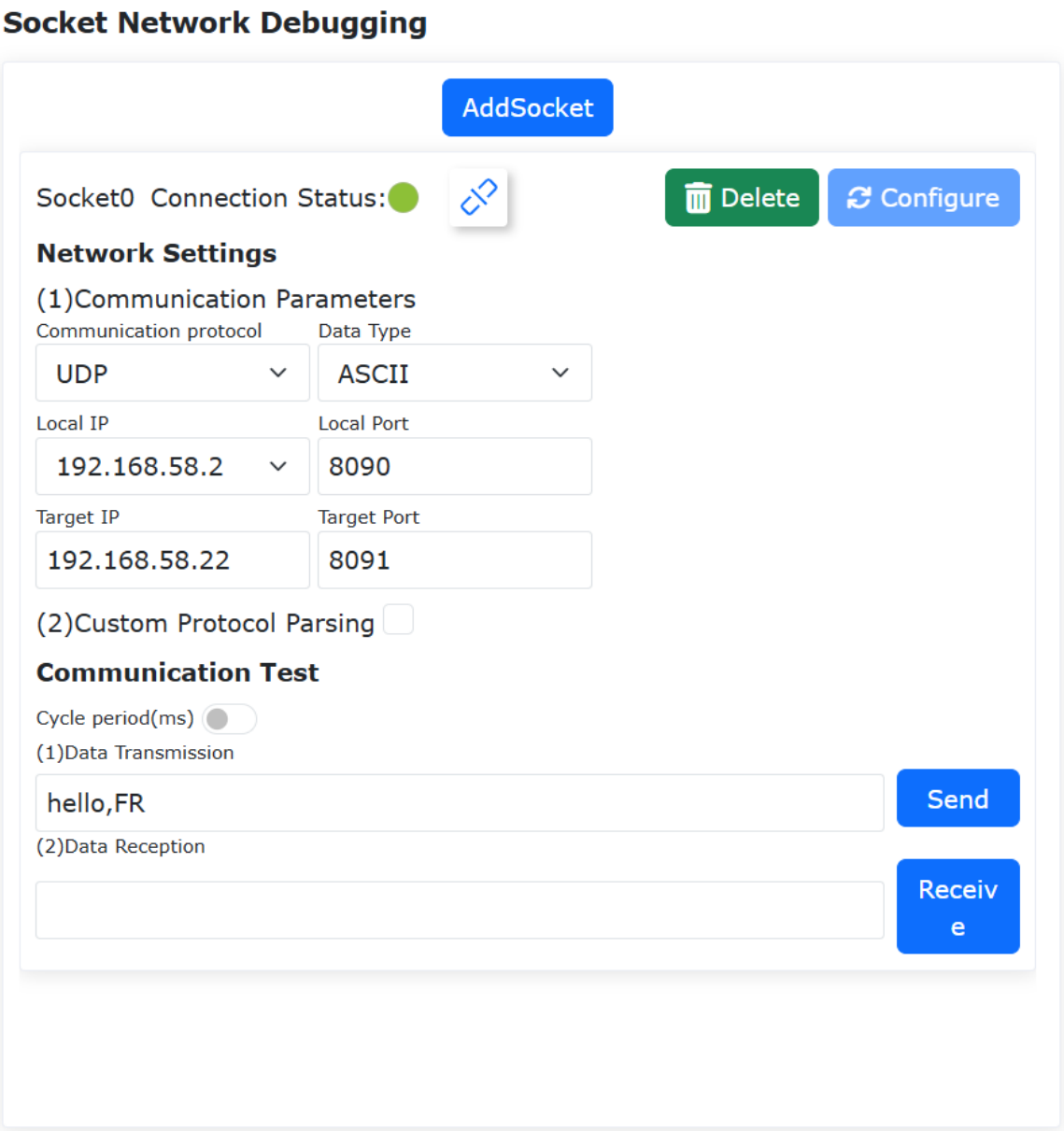


Figure 9.32-9 Communication Testing

The interface sends data in blocking mode by default, waiting for motion completion before sending data. The default data reception timeout is 5 seconds, after which an error will be reported and stopped. These parameters can be adjusted when issuing commands.

1.3.9.32.3.2 Communication Command Module

Click “Teach Program” - “Program Editing” - “Communication Commands”, select “Socket” command to generate Socket communication commands for data transmission and reception in Lua programming. SocketID can only select already configured Socket connections for sending data.

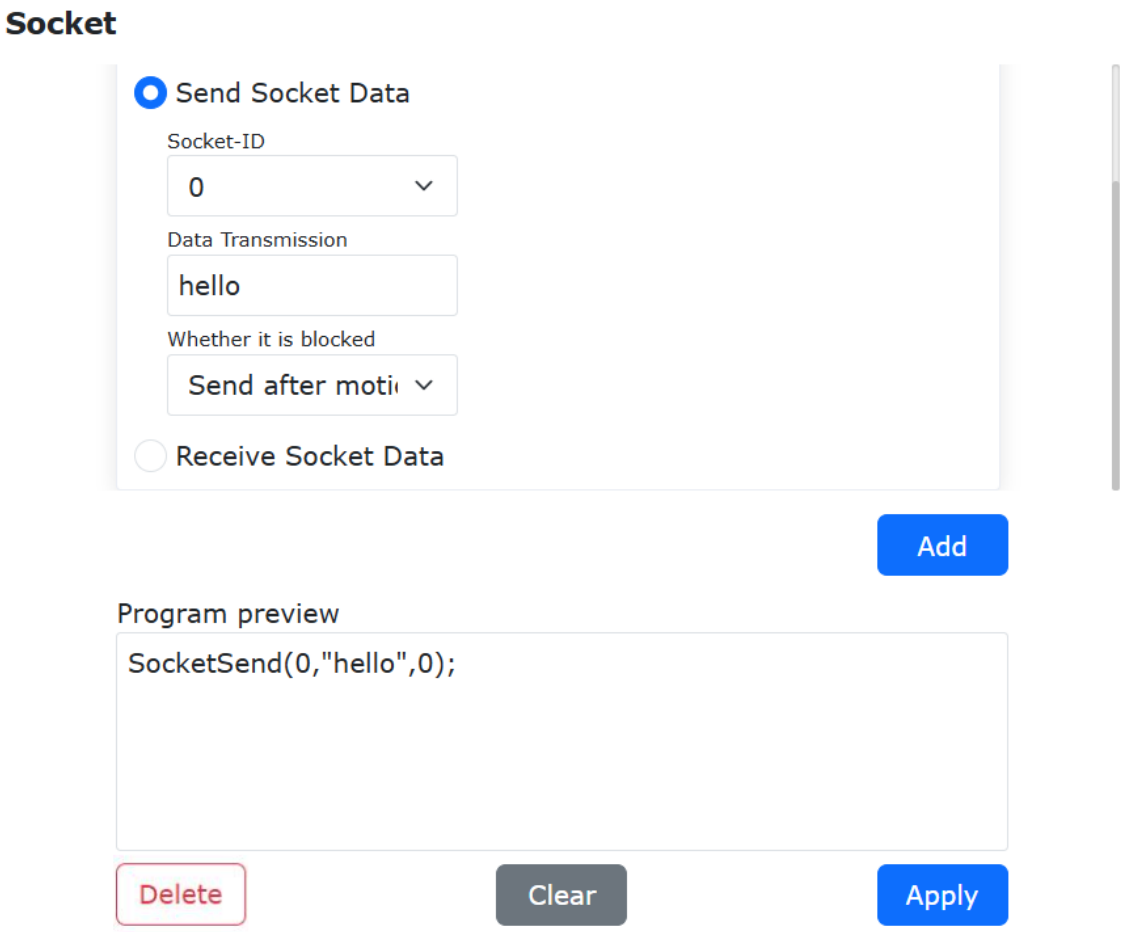


Figure 9.32-10 Sending Socket Data

Command parameters are Socket ID, data to send, and whether to wait for motion completion.

Command details:

- Send command: SocketSend(id,data,block);
- Parameters: id, connected socket ID; data: data to send (string format), content must match the configured data type, such as “hello” or “FA54DE”; block: whether to block motion (0: wait for motion completion before sending, 1: send immediately). Return value 0 for success.

Receiving data is shown below.

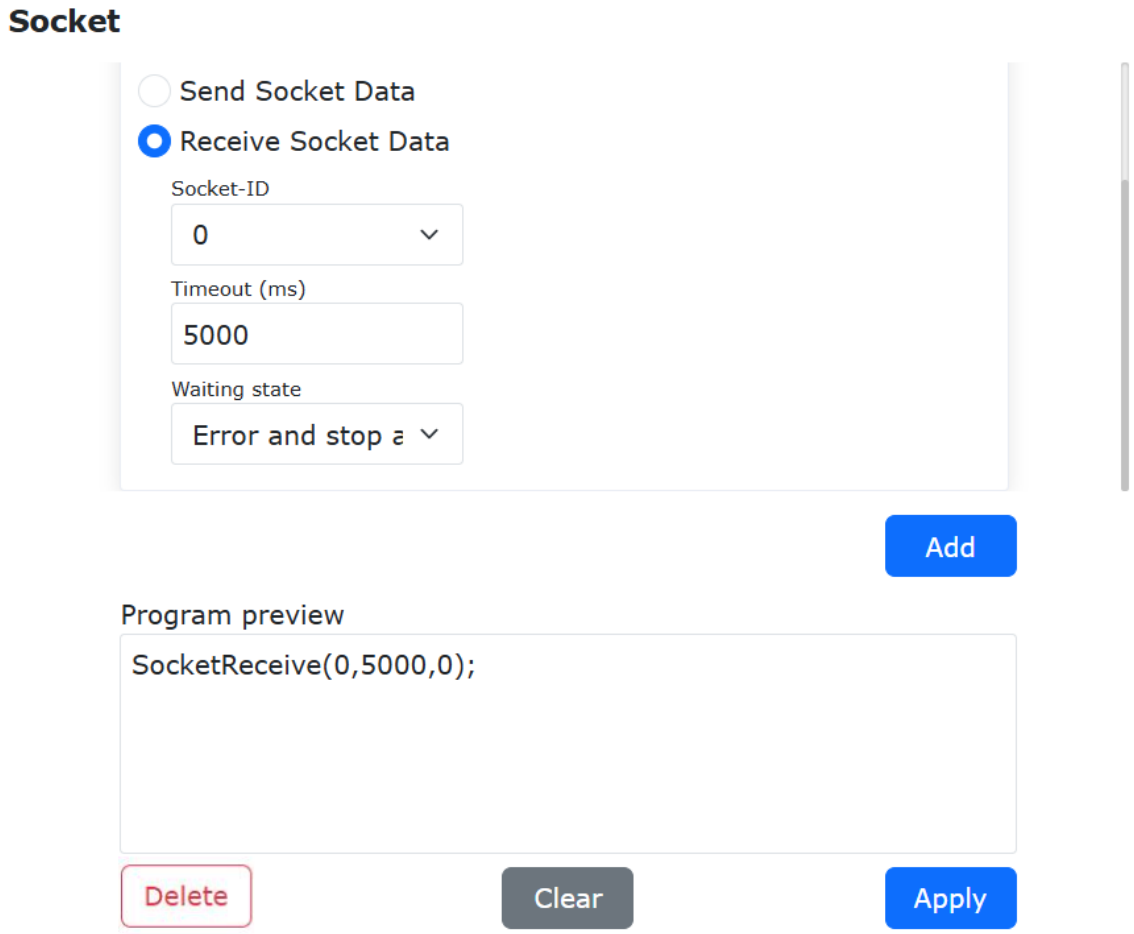


Figure 9.32-11 Receiving Socket Data

Command parameters are Socket ID, reception timeout (in milliseconds), and post-timeout strategy.

Command details:

- Receive command: SocketReceive(id,timeout,stopStartegy);
- Parameters: id, connected socket ID; timeout: reception timeout; stopStartegy: post-timeout strategy (0: report error and stop after timeout, 1: continue running after timeout);
- Return values: time: reception time, data: received data.

1.3.9.33 Impedance Control Function During Robot Motion

1.3.9.33.1 Overview

The impedance control function enables real-time detection of external forces. When the set threshold is reached, it actively complies with the external force, deviates from the motion trajectory, and returns to the motion trajectory after the external force drops below the threshold, thereby achieving better human-robot interaction. When an external force exceeding the preset force threshold is detected, this function drives the robotic arm to generate an offset in the direction of the force, achieving an active avoidance effect. After the external force is removed, the robotic arm returns to the vicinity of the original motion trajectory, thus enhancing safety during human-robot collaboration.

1.3.9.33.2 Impedance Control Function

1.3.9.33.2.1 Impedance Control Configuration and Function Start/Stop in Cartesian Space

Step1: Log in to the web interface, sequentially click “Initial Settings” → “Basic” → “Joints” → “Collision Level” to enter the robot collision level setting module, and set a reasonable collision coefficient, as shown in Figure 2-1.

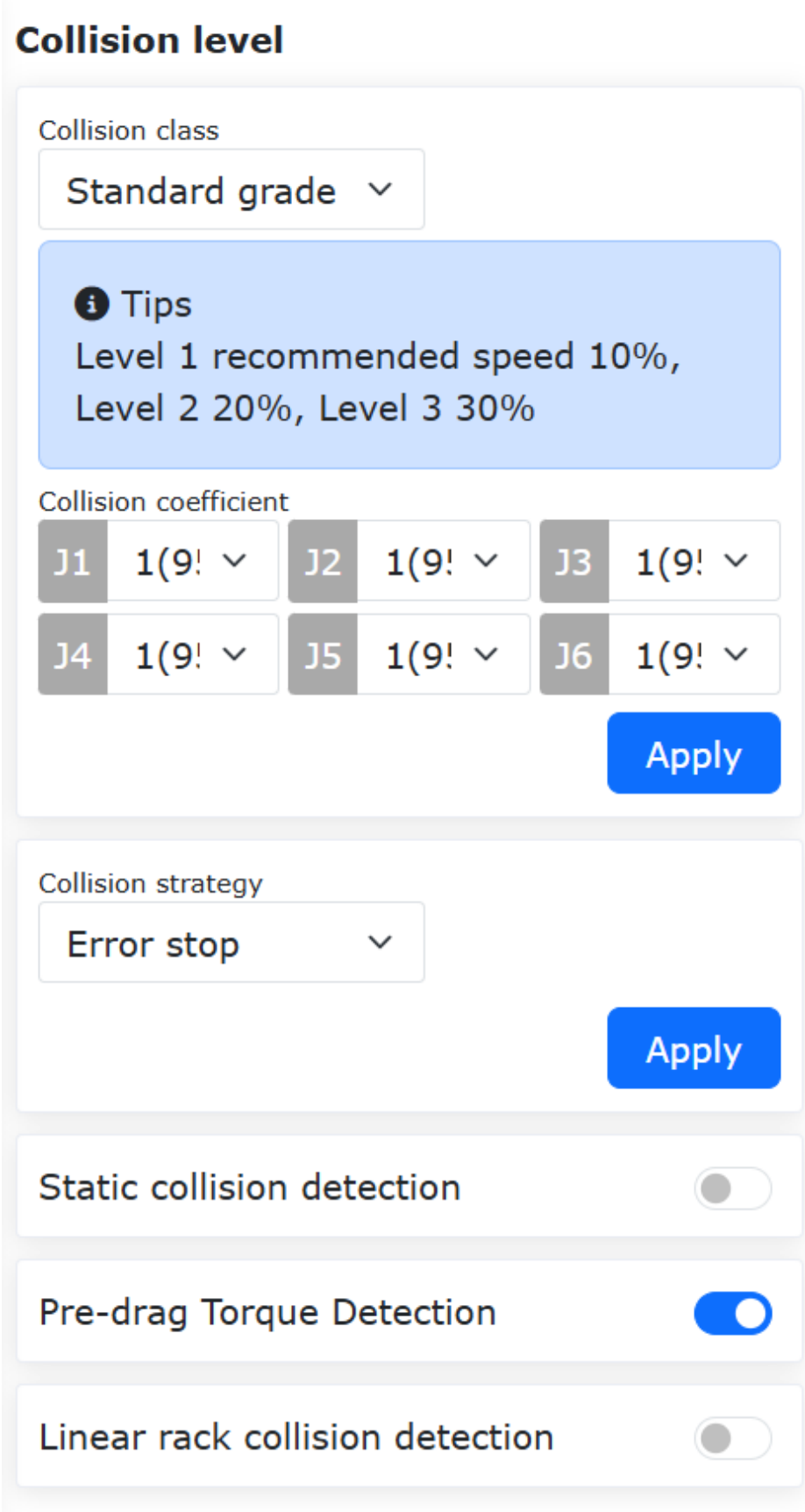

Collision level
Collision class
Standard grade
Tips
Level 1 recommended speed 10%,
Level 2 20%, Level 3 30%
Collision coefficient
J1
1(9!
J2
1(9!
J3
1(9!
J4
1(9!
J5
1(9!
J6
1(9!
Apply
Collision strategy
Error stop
Apply
Static collision detection
Pre-drag Torque Detection
Linear rack collision detection

Figure 9.33-1 Robot Collision Coefficient Setting Module

Step2: To use a force sensor to implement the impedance control function, the force sensor must be configured in the end effector peripheral configuration under "Peripherals" → "End Effector". If choosing not to use a force sensor to implement the impedance control function, this step is not required.

Step3: Sequentially click "Teach Program" → "Program Programming" → "Force Control Set", and add the "Impedance" command. The "Impedance" command enables the robot to implement impedance control on the running trajectory (Currently, only impedance control in Cartesian space is available).

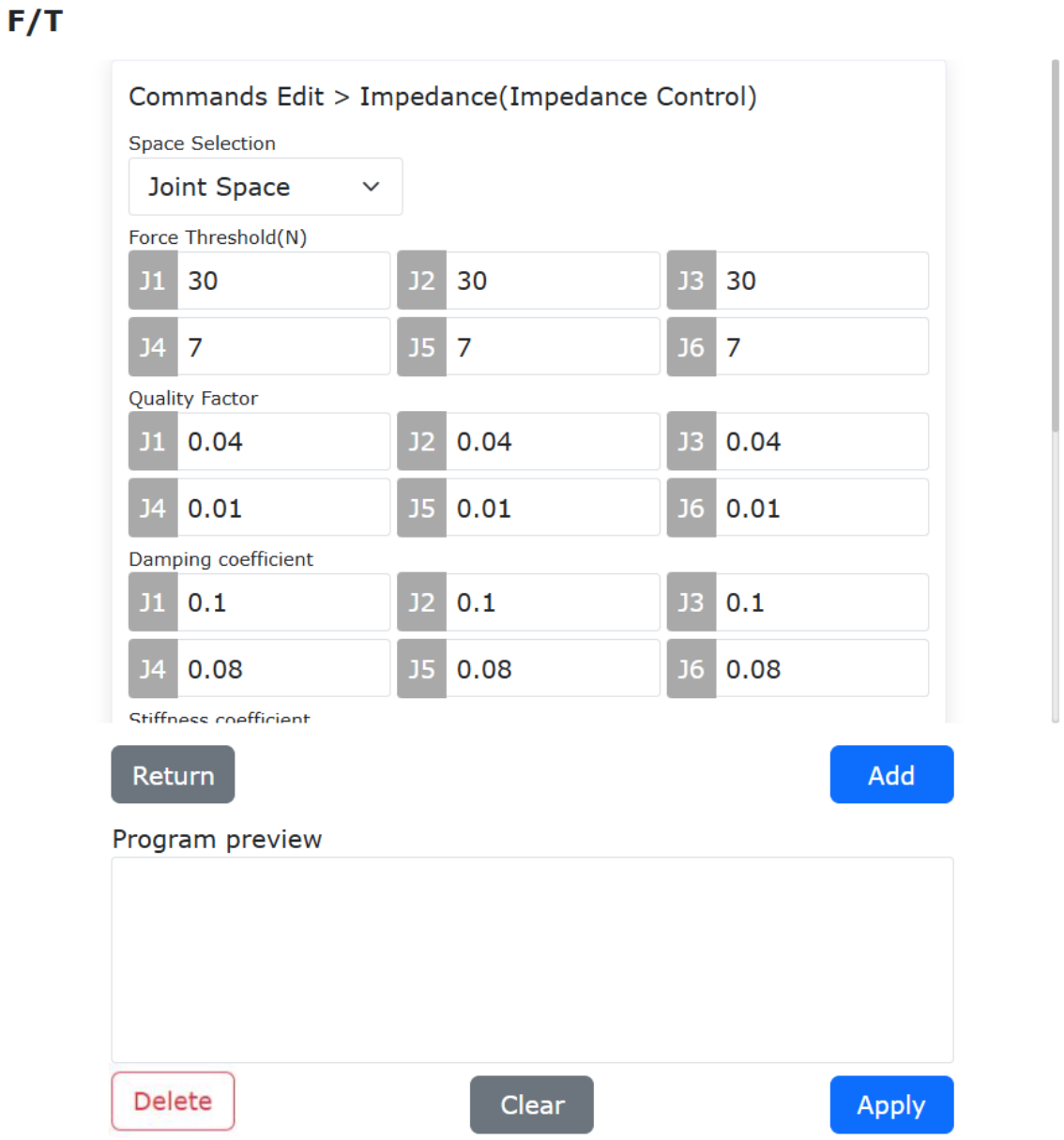


Figure 9.33-2 Adding Force Control Command

Step4: In the force control command module, select "Cartesian Space" from the space selection dropdown menu. Set appropriate values in the text boxes for force threshold, mass coefficient, damping coefficient, stiffness coefficient, maximum linear velocity, maximum linear acceleration, maximum angular velocity, and maximum angular acceleration. Click "Enable" under command type, then click "Add" to add the impedance control enable command; click "Disable"

under command type, then click “Add” to add the impedance control disable command.

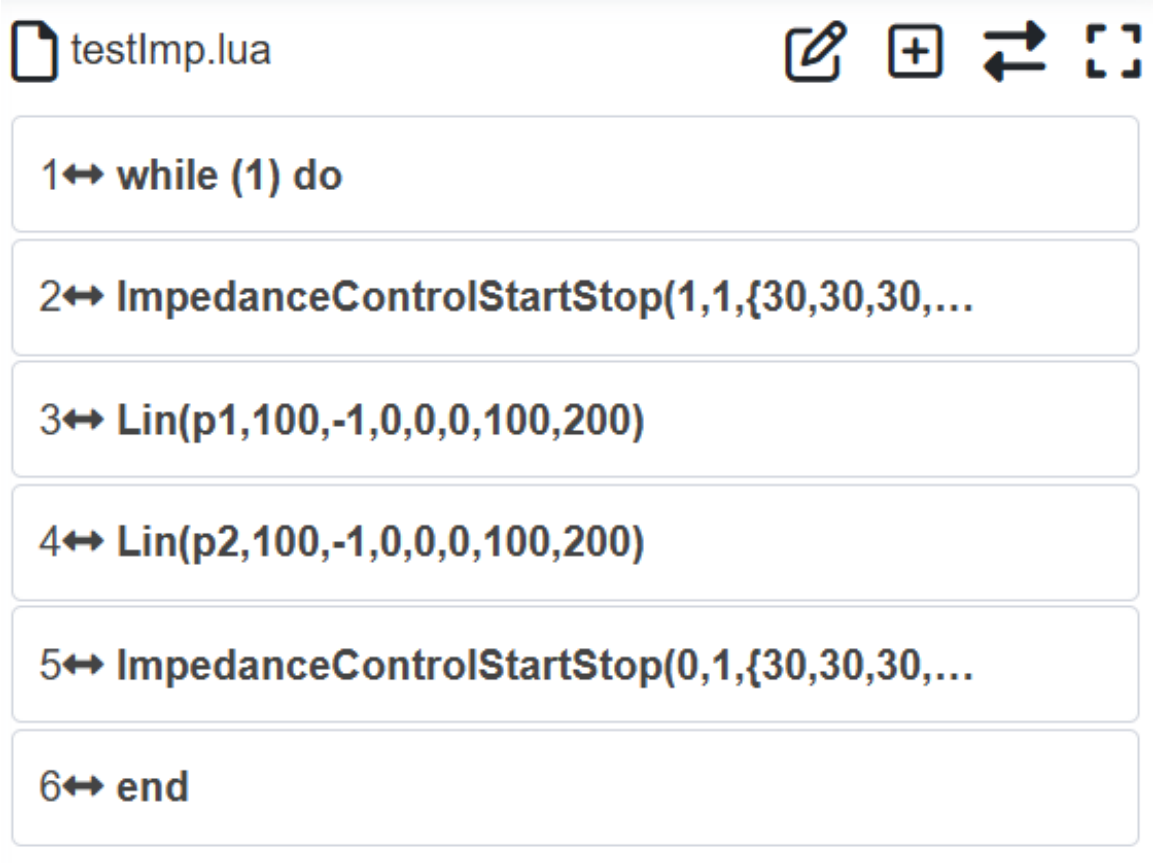


Figure 9.33-3 Impedance Control Command Example

Step5: During operation, if the robotic arm stops running and the lower left corner of the Web interface displays “500 Error: Current collision level is too low”, this is because the set force threshold is greater than the trigger threshold of the collision level. At this point, increasing the collision level or decreasing the force threshold will resolve this error.

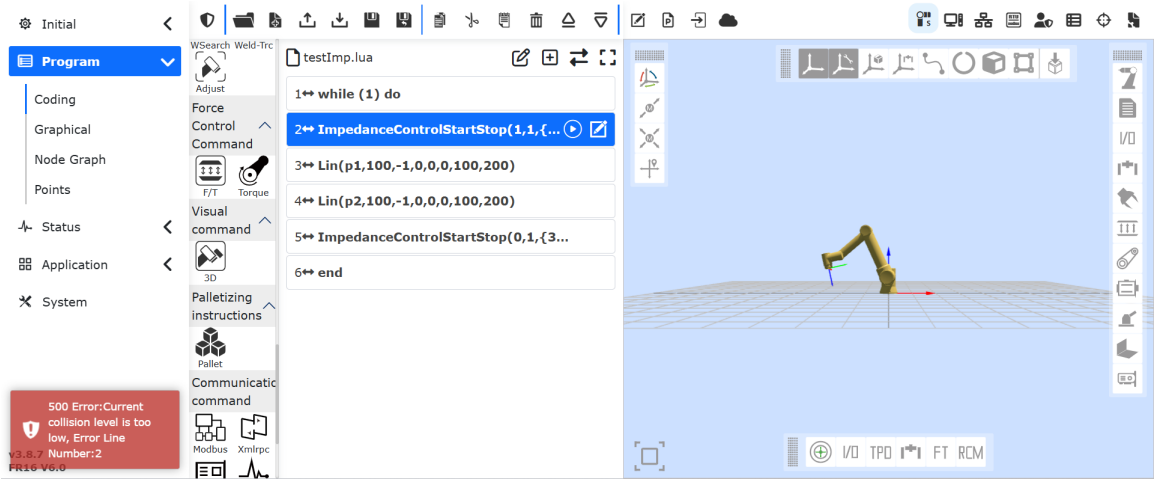


Figure 9.33-4 Collision Level Too Low Warning

Step6: During operation, if the robotic arm stops running and the lower right corner of the Web interface displays “Collision Fault”, this is because the external force on the robotic arm has exceeded the trigger threshold of the collision

level, thereby triggering a collision fault.

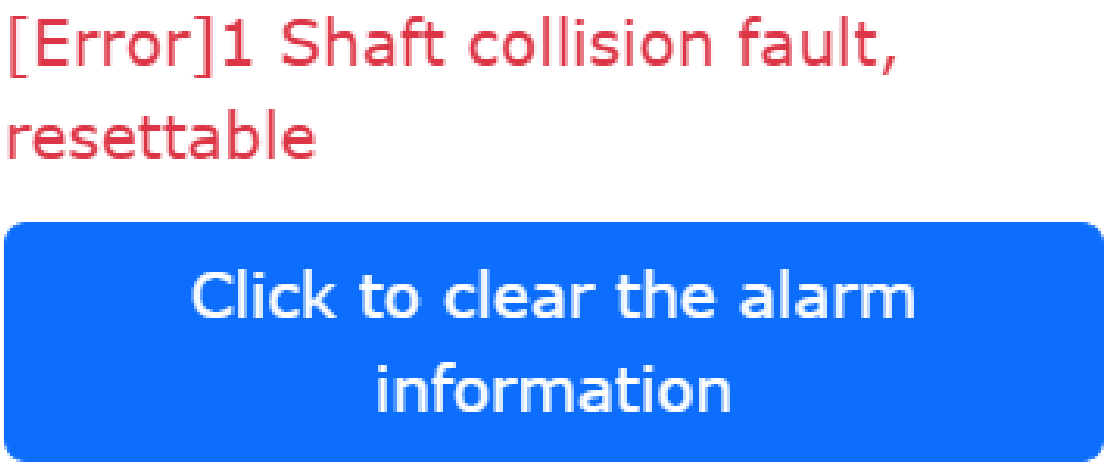


Figure 9.33-5 Collision Fault Warning

Specific Functions and Recommended Values of Parameters:

- Space Selection: Sets the operational space for impedance control. Currently, only Cartesian space impedance control is available.
- Force Threshold: The minimum trigger force for impedance control. The range for the translational direction force threshold is 30–150N, and for the rotational direction force threshold is 7–30Nm.
- Mass Coefficient: Increasing the mass coefficient will cause the offset to be slower, decreasing it will cause the robot offset to be too fast. The range for the translational direction is [0.01-1], recommended value is 0.04; the range for the rotational direction is [0.001-1], recommended value is 0.01.
- Damping Coefficient: Increasing the damping coefficient will cause the offset to be slower, decreasing it will cause the robot offset to be too fast, resulting in oscillation. The range for the translational direction is [0.1-2], recommended value is 0.1; the range for the rotational direction is [0.008-1.5], recommended value is 0.08.
- Stiffness Coefficient: Increasing the stiffness coefficient will cause the offset to be slower. Recommended value is 0.
- Maximum Linear Velocity: Limits the velocity generated by external forces in the translational direction. Recommended value is 250mm/s.
- Maximum Linear Acceleration: Limits the acceleration generated by external forces in the translational direction. Recommended value is 500mm/s^2.
- Maximum Angular Velocity: Limits the angular velocity generated by external forces in the rotational direction. Recommended value is 90°/s.
- Maximum Angular Acceleration: Limits the angular acceleration generated by external forces in the rotational direction. Recommended value is 180°/s^2.

1.3.9.34 Custom Weaving Welding Function

1.3.9.34.1 Overview

The custom weaving welding function allows the execution of weaving using a pattern designed by the user.

Description of the Custom Weaving Welding Function:

- (1) In the weaving parameters interface, select any one of the weaving types: “Custom Weave 0”, “Custom Weave 1”, or “Custom Weave 2”. A maximum of 3 custom weaving patterns can be set.
- (2) The number of custom weaving endpoints can be set to a maximum of 10 and a minimum of 2. The X, Y, Z data of the last endpoint is fixed at 0 and cannot be modified, though the dwell time for all endpoints can be set.
- (3) The X, Y, and Z values of the custom weaving endpoints must be within the range of -10mm to 10mm, and the weaving frequency must not exceed 10 Hz.
- (4) Currently, linear, circular, and full-circle trajectories support custom weaving welding, but the weaving gradient function is not yet supported.
- (5) Note: When the weaving dwell time is set to “Include”, the total weaving dwell time must not exceed half of the weaving cycle time.

1.3.9.34.2 Operational Procedure for Custom Weaving Welding Function

The operational procedure for the custom weaving welding function is as follows:

Step 1: First, record the start and end teach points of the linear trajectory. Then, click “Teach Program” -> “Program Editing”, select “Point to Point” to move the robot end to the linear start point “custWeaveP1”. Finally, select “Linear” to move the robot to the linear end point “custWeaveP2”.

Step 2: Select the “Weaving” button, click the weaving process editing button to enter the weaving parameters setting interface. Select “Custom Weave N” (N=0, 1, 2) for the “Weaving Type”.

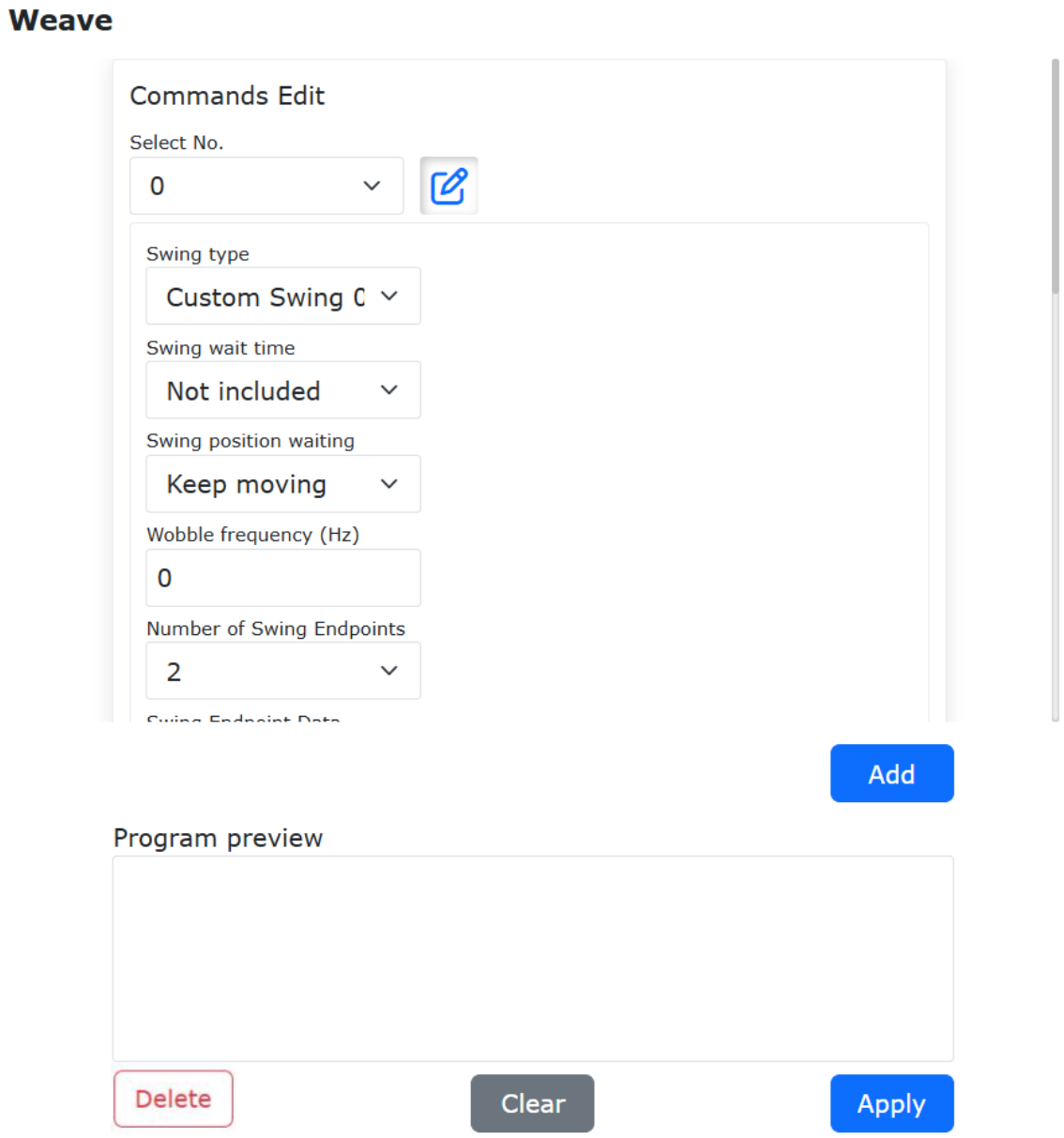


Chart 9.34-1 Weaving Parameters Setting Interface

Step 3: After selecting the "Weaving Type", scroll down in the weaving parameters setting interface. In this interface, select the number of custom weaving endpoints, set the position and dwell time of each point in the weaving coordinate system, and finally click the "Configure" button.

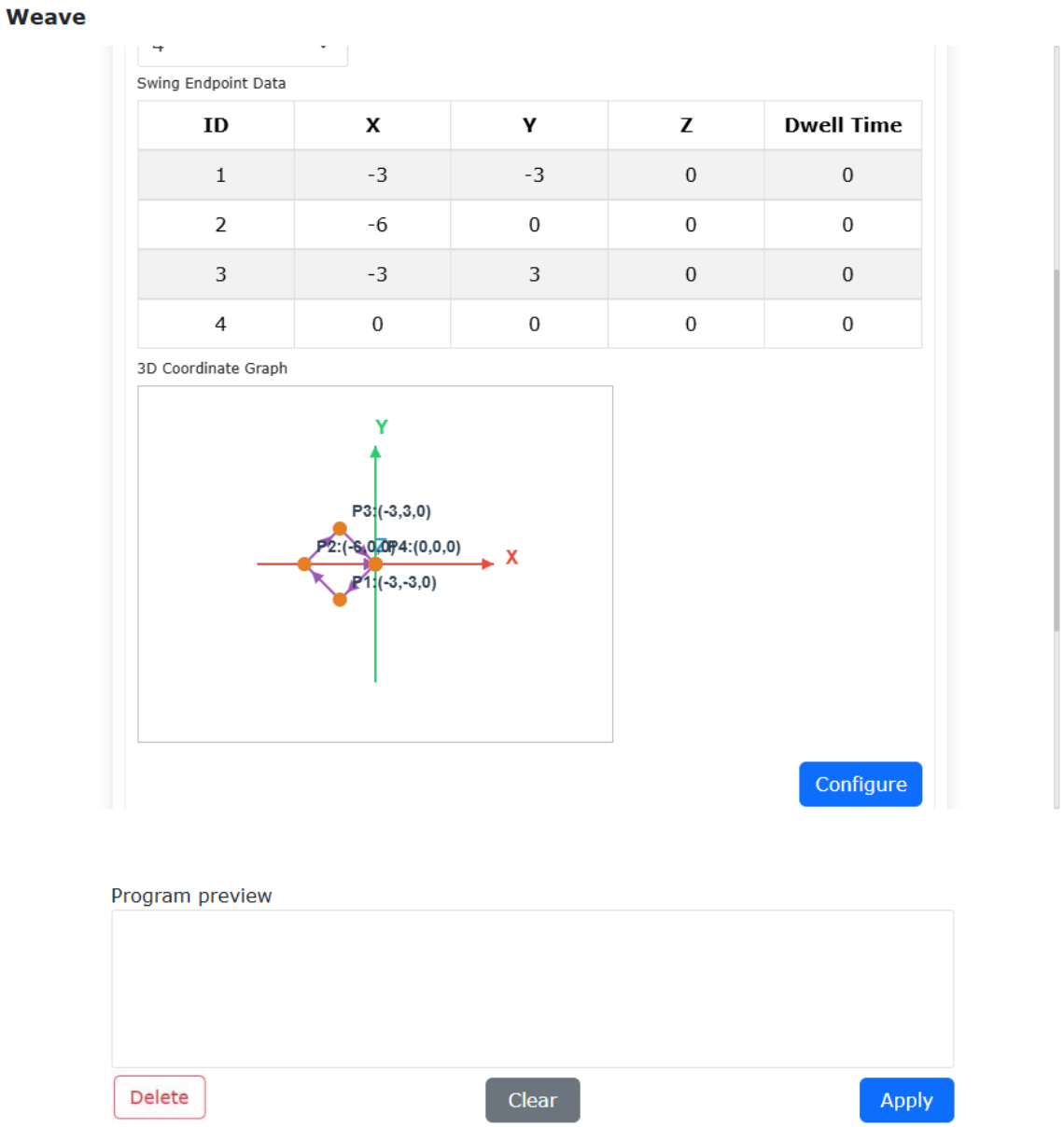


Chart 9.34-2 Custom Weaving Setting Interface

Step 4: In the weaving interface, sequentially select “Start Weaving” and “Stop Weaving” in the “Command Type” dropdown, click the “Add” button, and finally click the “Apply” button.

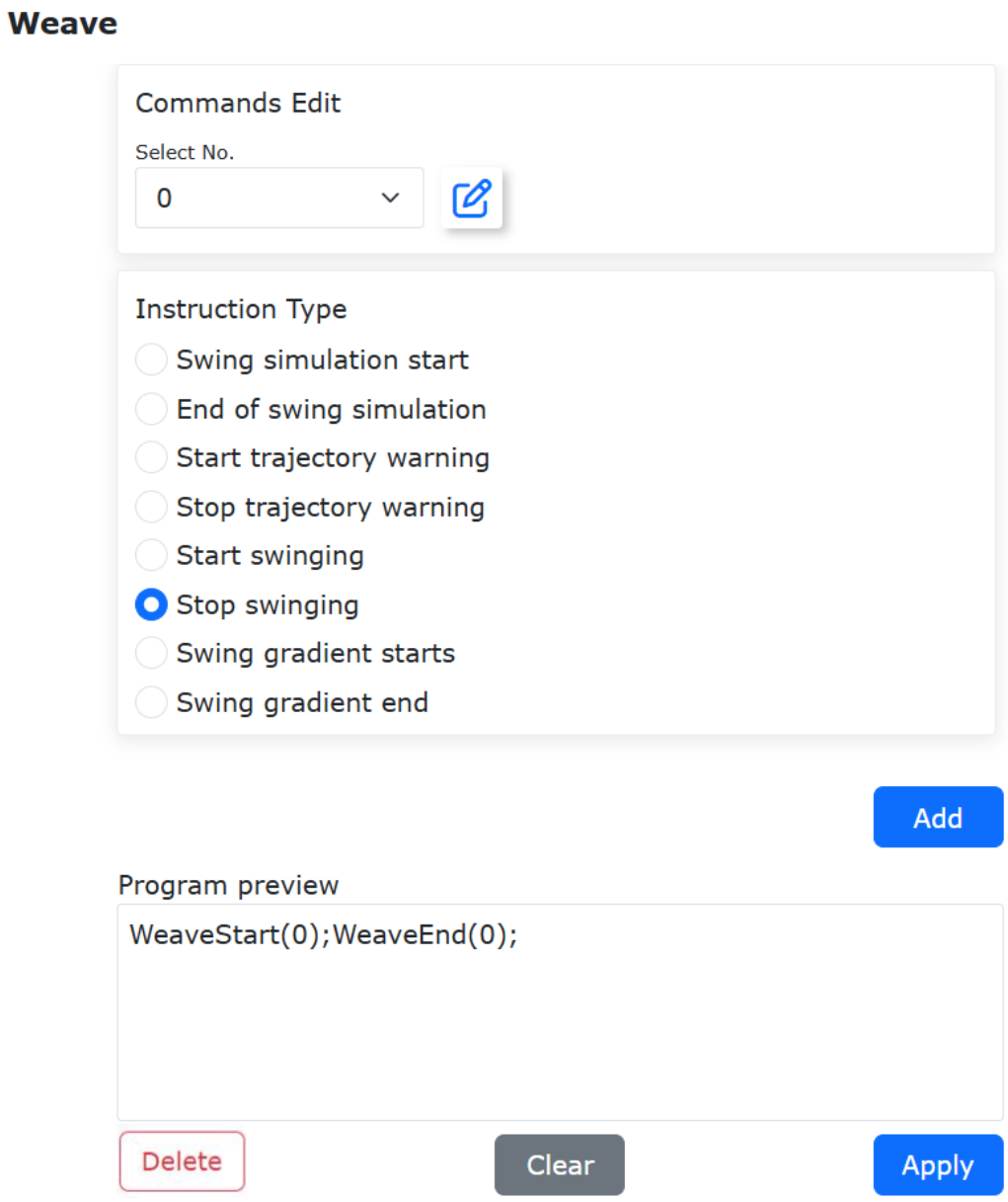


Chart 9.34-3 Weaving Command Setting Interface

Step 5: In the program editing interface, select the weaving start command, click the "Move Up" button at the top of the interface, and finally save the program. Switch the robot to automatic mode and click the "Start" button. The robot will then begin custom weaving along the linear trajectory.

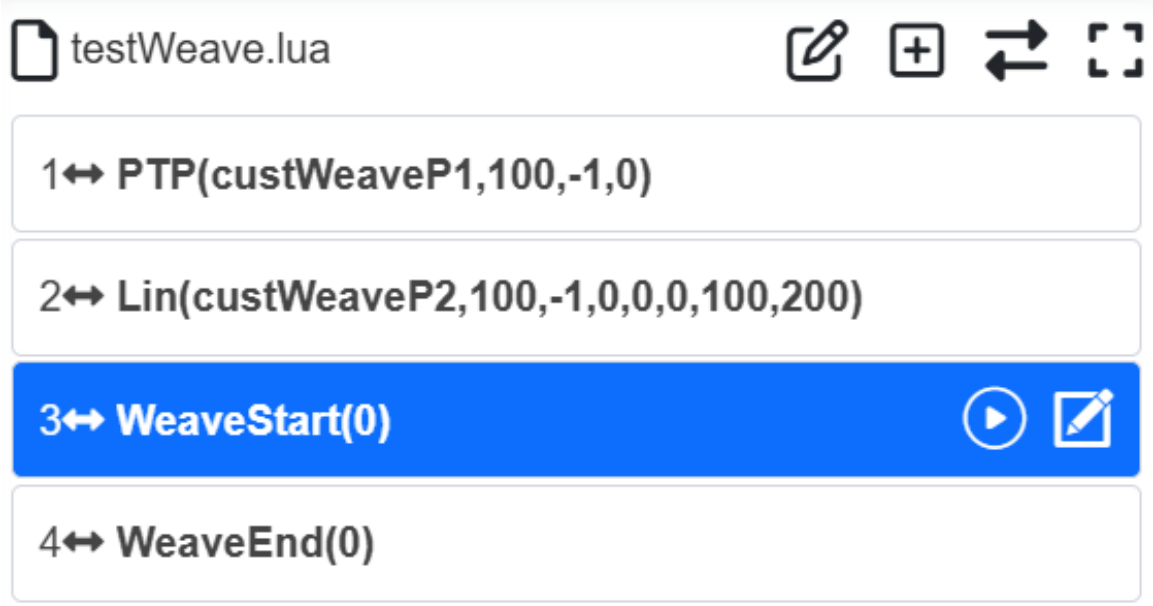


Chart 9.34-4 Original LUA Command Interface

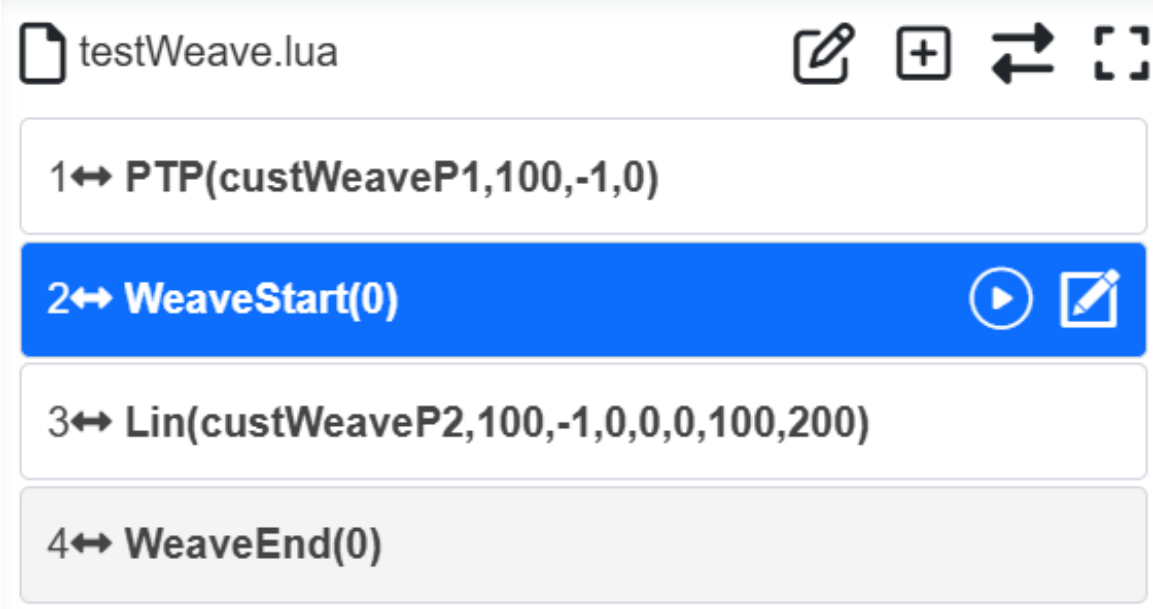


Chart 9.34-5 Modified LUA Command Interface

Step 6: The steps for setting up custom weaving for circular and full-circle trajectories are the same as Step 1 to Step 5 above.

1.3.9.35 Teach point configuration

Click “Points Config.” to enter the teaching point configuration interface.

Before using the button box or other IO signals to record the teaching point function, the user first configures the teaching point name prefix, the upper limit of the number and the teaching method. The name prefix supports two modes: custom prefix and current program name as the prefix. For example, customize the name prefix “P”, number upper limit “3”, teaching method “robot teaching”, record the current end (tool) points of the robot in sequence: P1, P2, P3, and record again will overwrite the previous record points.

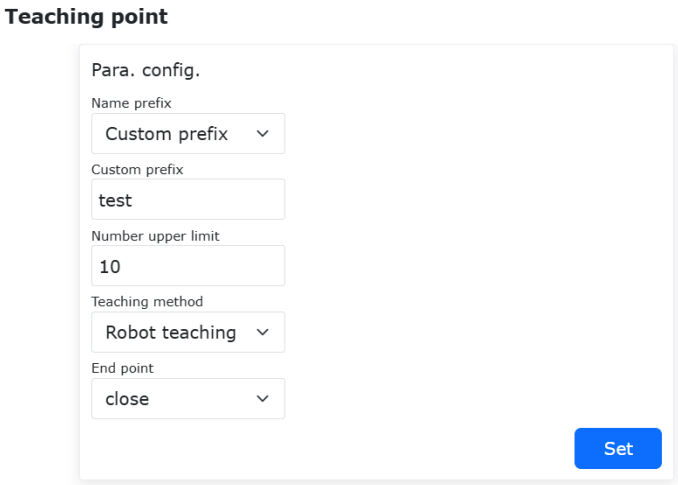


Chart 9.35-1 Teach point configuration

1.3.9.35.1 End-point dot automatic overwrite update Lua program function

1.3.9.35.1.1 End-point dot configuration

1. Enable the end-point dot function and click on settings. You can use the switch to select the Lua programs that need to be updated for specific positions.

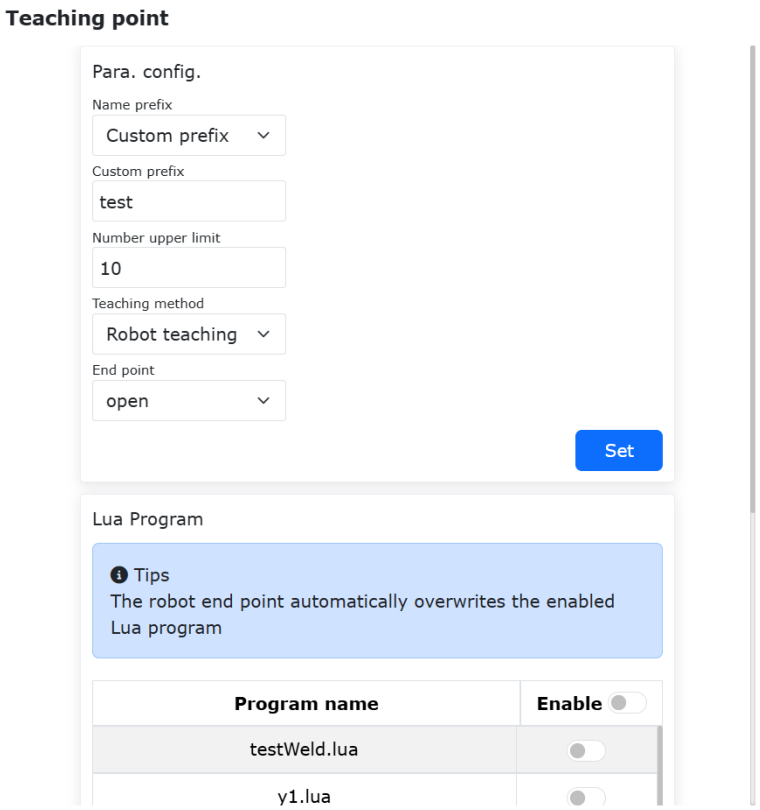


Chart 9.35-2 Enable end point recording

2. The configuration is complete, with the end-tip dot name prefixed as “test”, the numbering limit set to 10, and all Lua programs selected for enabling updates. Close the webApp, and the function remains active.

1.3.9.35.1.2 End-button dot automatic update Lua program

1. Click the robot end-point dot button.

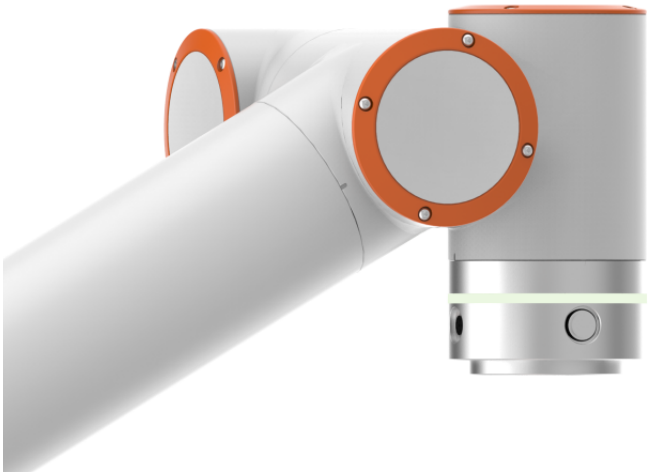


Chart 9.35-3 end-point dot button

2. At this point, the end-tip LED flashes: with the purple light flashing (start) -> blue light on (dot recording and updating in Lua) -> green light on (dot recording completed), and the position information corresponding to the selected Lua program's name is synchronized and updated.

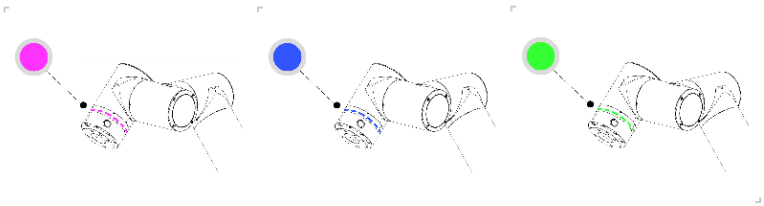


Chart 9.35-3 End-tip dot recording and updating Lua program LED changes

3. When the dot recording fails, the end-tip LED flashes: Purple light flashing (start) -> Red light flashing (dot recording failed) -> Green light on (returning to normal).

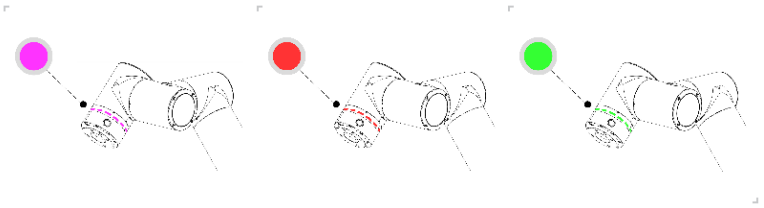


Chart 9.35-4 LED changes when end-tip dot recording fails

1.3.9.35.1.3 Function usage example

1. Click on Auxiliary Applications - Tool Applications - Teaching Point Configuration, customize the prefix to "test", set the number limit to 5, select Robot Teaching as the teaching method, enable the end-point dot function, and click on settings.
2. Activate the Lua program "program1" that requires position updates.

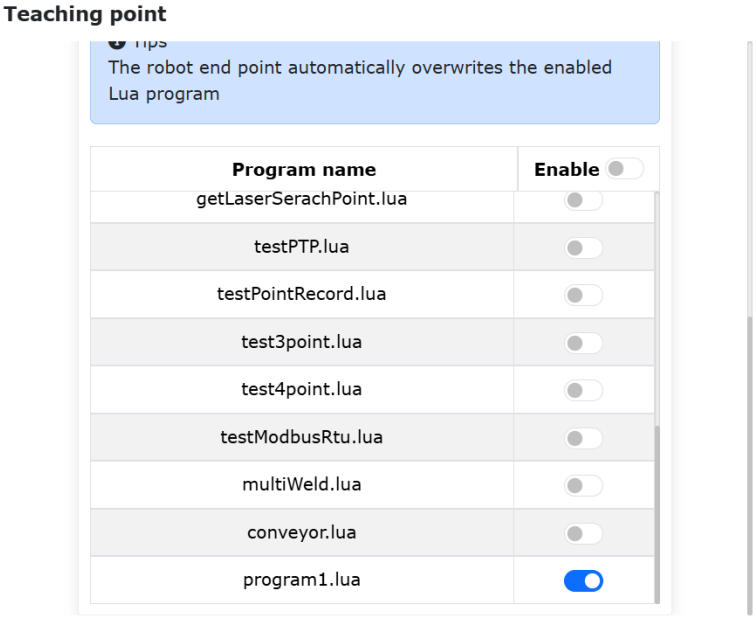


Chart 9.35-5 Teaching Point Configuration

3. As shown below, it depicts the “program1” program and its current running trajectory.

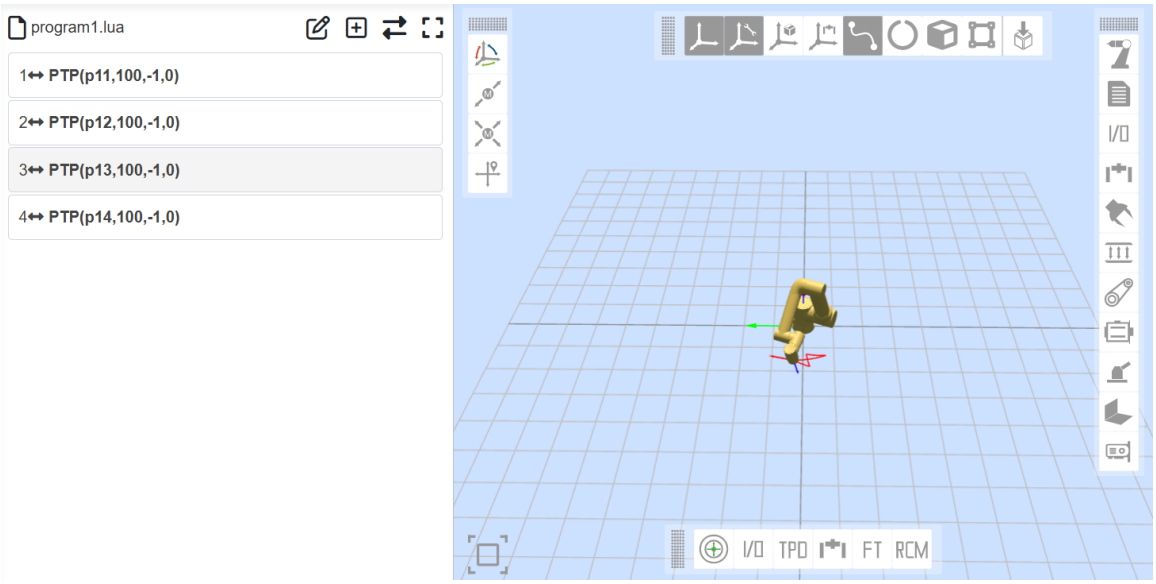


Chart 9.35-6 Program1 program and its current running trajectory

4. Switch the page to manual mode, move the robot to a new position, click the end-tip dot button, and wait for the end-tip LED to complete its flashing sequence: Purple light flashing (start) -> Blue light on (dot recording and updating in Lua) -> Green light on (dot recording completed), at which point the recorded position is labeled as test1.
5. Repeat step 4 to record positions “test2,” “test3,” “test4,” and “test5,” completing the recording of 5 points. At this stage, the positions for the “program1” program have been synchronized and updated.
6. Re-run the “program1” program. The motion trajectory will have been updated, and the updated motion trajectory is shown below.

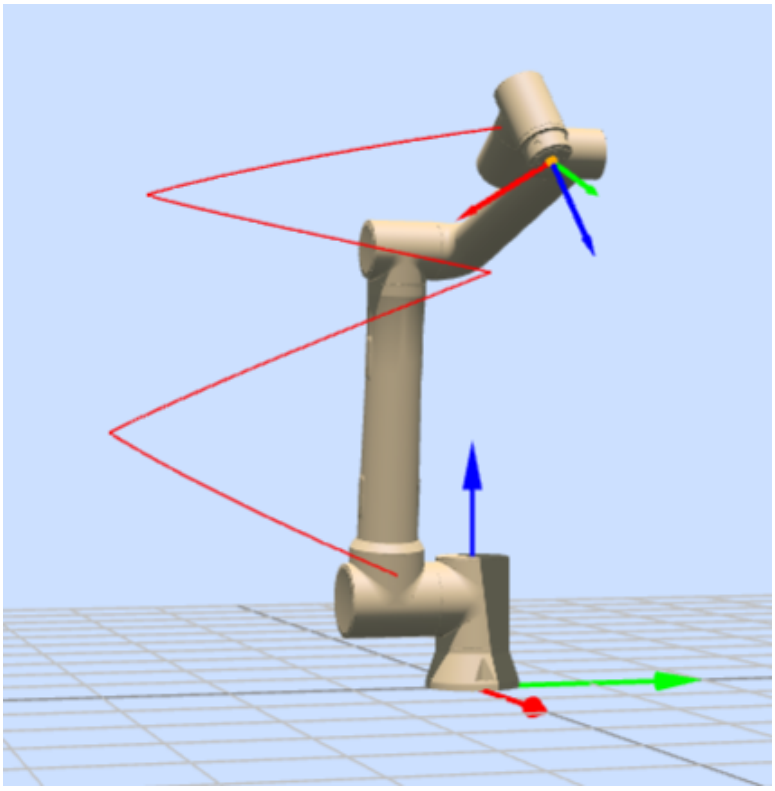


Chart 9.35-7 Updated running trajectory

1.3.9.36 Main program configuration

Click “Main program” to enter the main program configuration function interface.

The configuration of the main program can be used in conjunction with the DI configuration of the main program startup. The configured main program needs to be trial run first to ensure safety. After configuring the corresponding DI in the robot settings to start the main program signal function, the user can control the DI signal to run the main program.

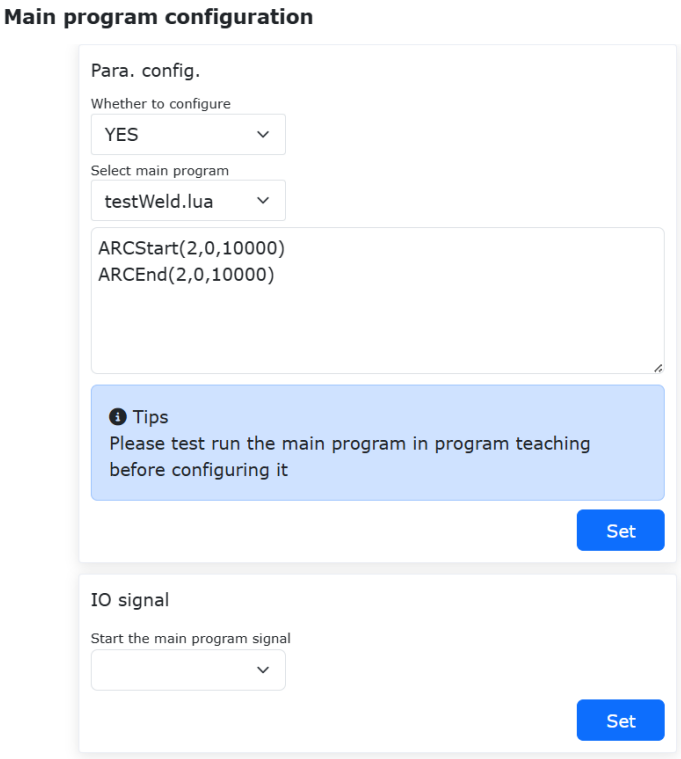


Chart 9.36-1 Main program configuration

1.3.9.37 Robot Extended Axis Intersecting Line Welding

1.3.9.37.1 System Configuration

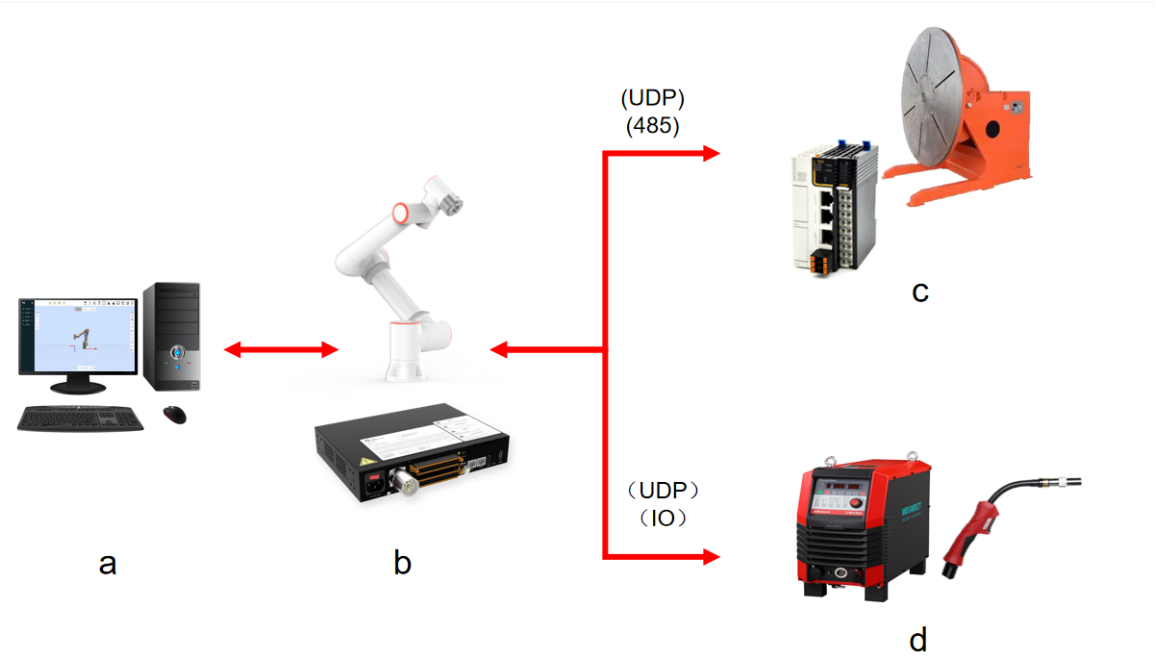


Figure 9.37-1 Robot Extended Axis Intersecting Line Welding System Configuration

In the system, (a) is the computer, (b) is the robot and its control box, (c) is the positioner and drive equipment, (d) is the welder and supporting equipment.

1.3.9.37.2 Extended Axis Communication Configuration

The communication methods between the robot and the extended axis include using UDP or RS485.

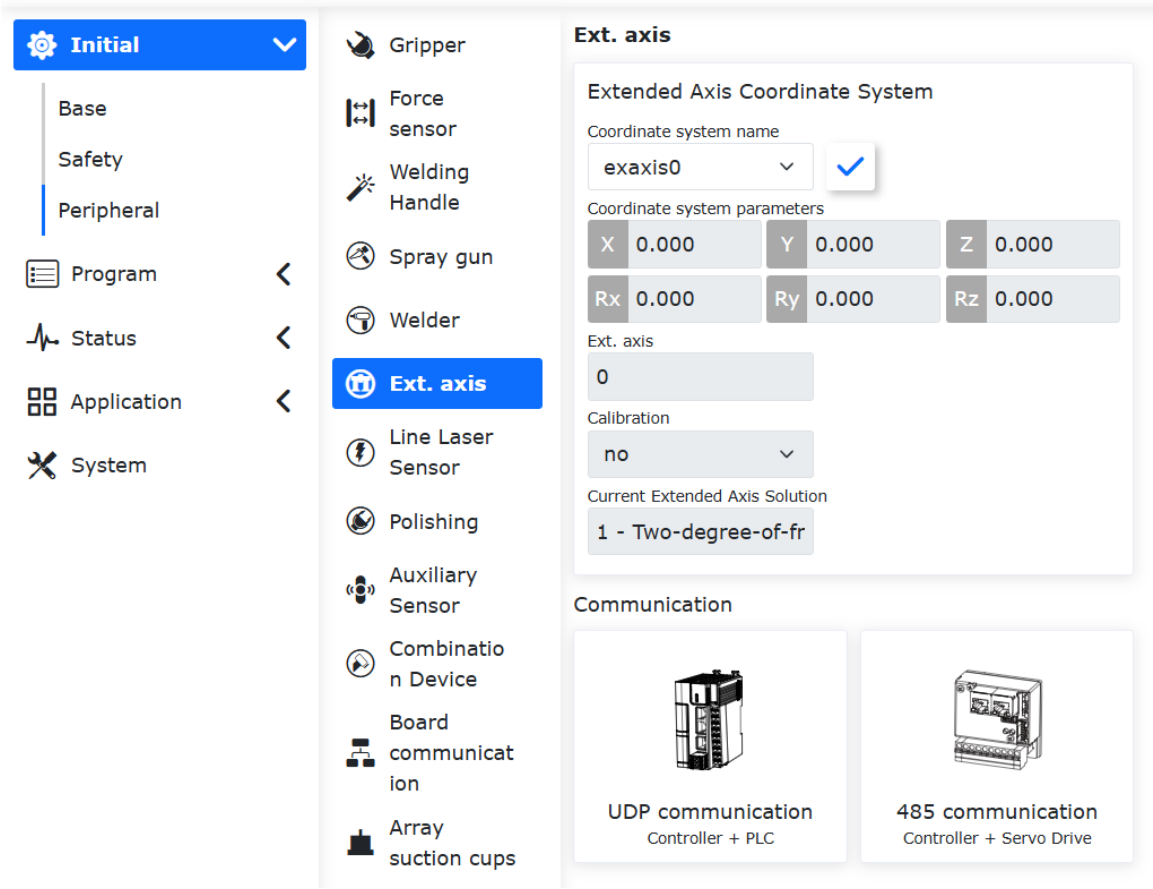


Figure 9.37-2 Extended Axis Configuration Page

Click “Initial Setup”, “Peripherals”, “Extended Axis” on the robot operation interface to enter the extended axis configuration page. Taking using a PLC connected to the robot via UDP communication as an example, click the “UDP Communication” icon to enter the extended axis configuration page for UDP communication.

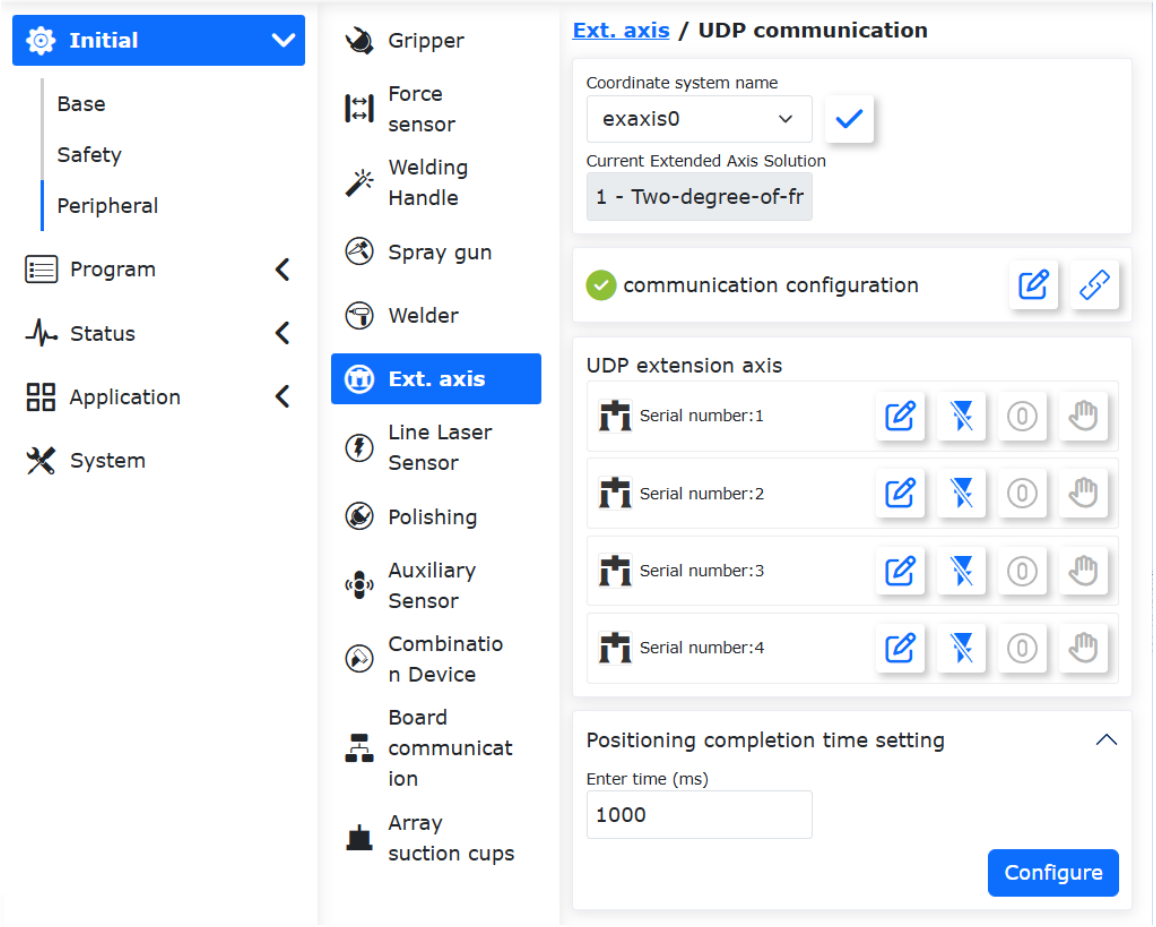


Figure 9.37-3 UDP Communication Configuration Interface

On the extended axis configuration page for UDP communication, you can select the corresponding extended axis number, connect and configure UDP communication parameters (address, port, cycle, packet loss detection, etc.), and the extended axis positioning completion time.

Extended axis configuration content is not the focus of this function introduction. For detailed configuration, refer to the corresponding section of the user manual.

1.3.9.37.3 Welder Connection Configuration

Configure the welder through the following configuration page:

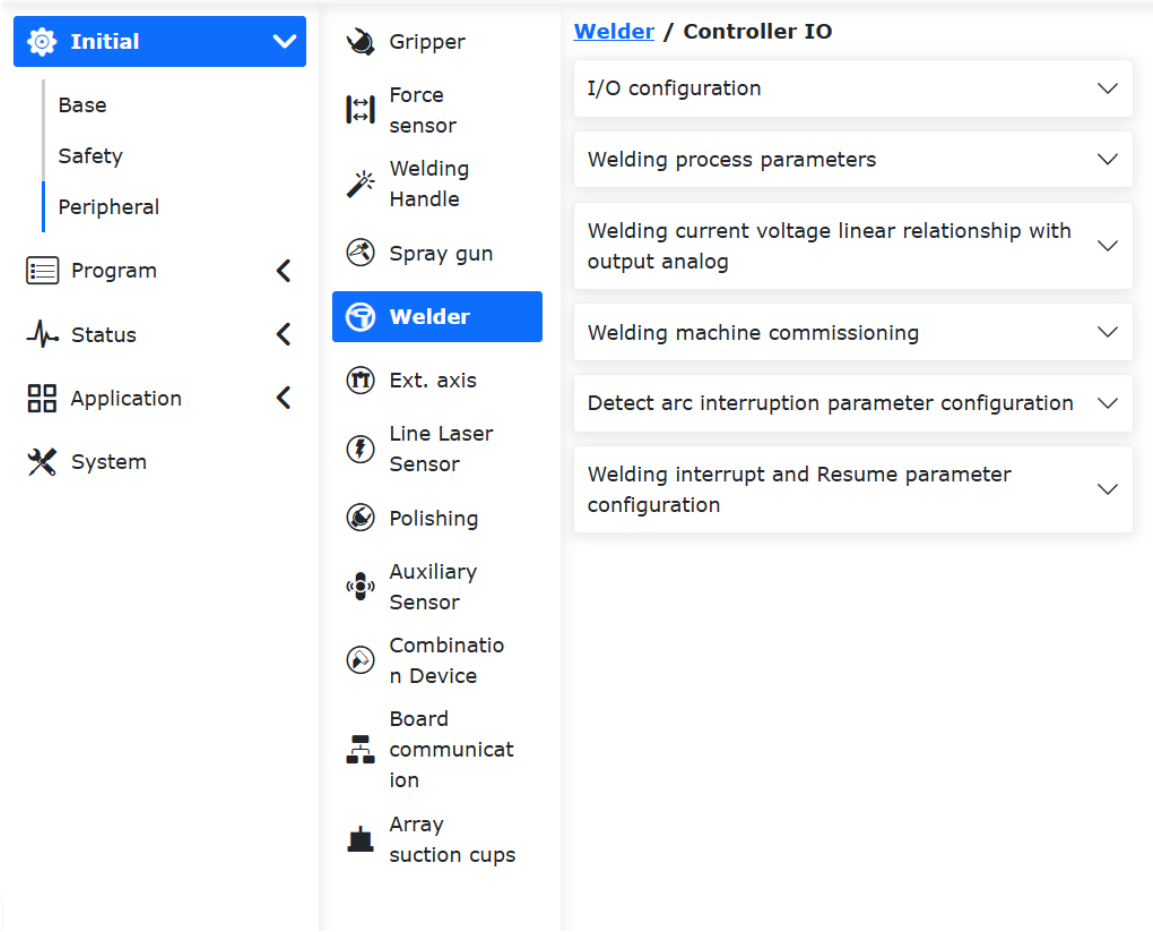


Figure 9.37-4 Welder Configuration Page

Welder communication can use IO communication or RS485 communication. Click “Initial Setup”, “Peripherals”, “Welder” to enter the configuration and connection interface, where you can configure modules such as “Control Type”, “I/O Configuration”, “Welding Process Parameters”, “Welder Debugging”, etc.

Welder configuration content is not the focus of this function introduction. For detailed configuration, refer to the corresponding section of the user manual.

1.3.9.37.4 Tool Coordinate System Calibration

After installing the welding torch on the robot end, calibrate the welding torch:

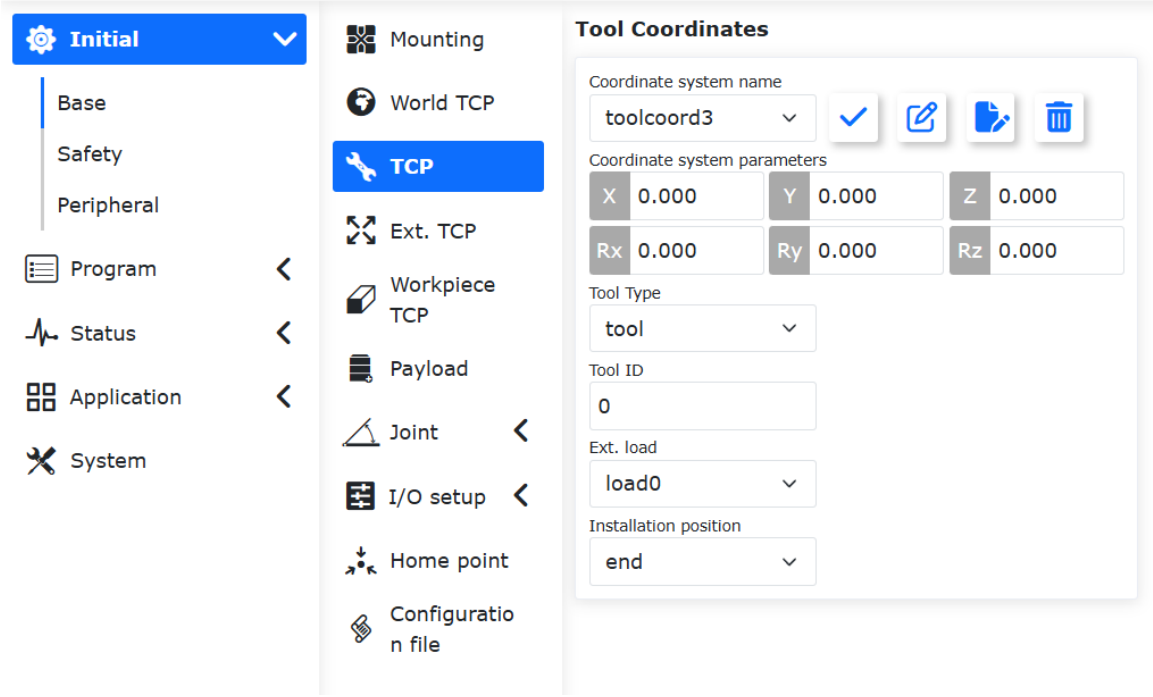


Figure 9.37-5 Tool Coordinate System Configuration Page

Click “Initial Setup”, “Basic”, “Tool Coordinates” to enter the tool coordinate system settings page.

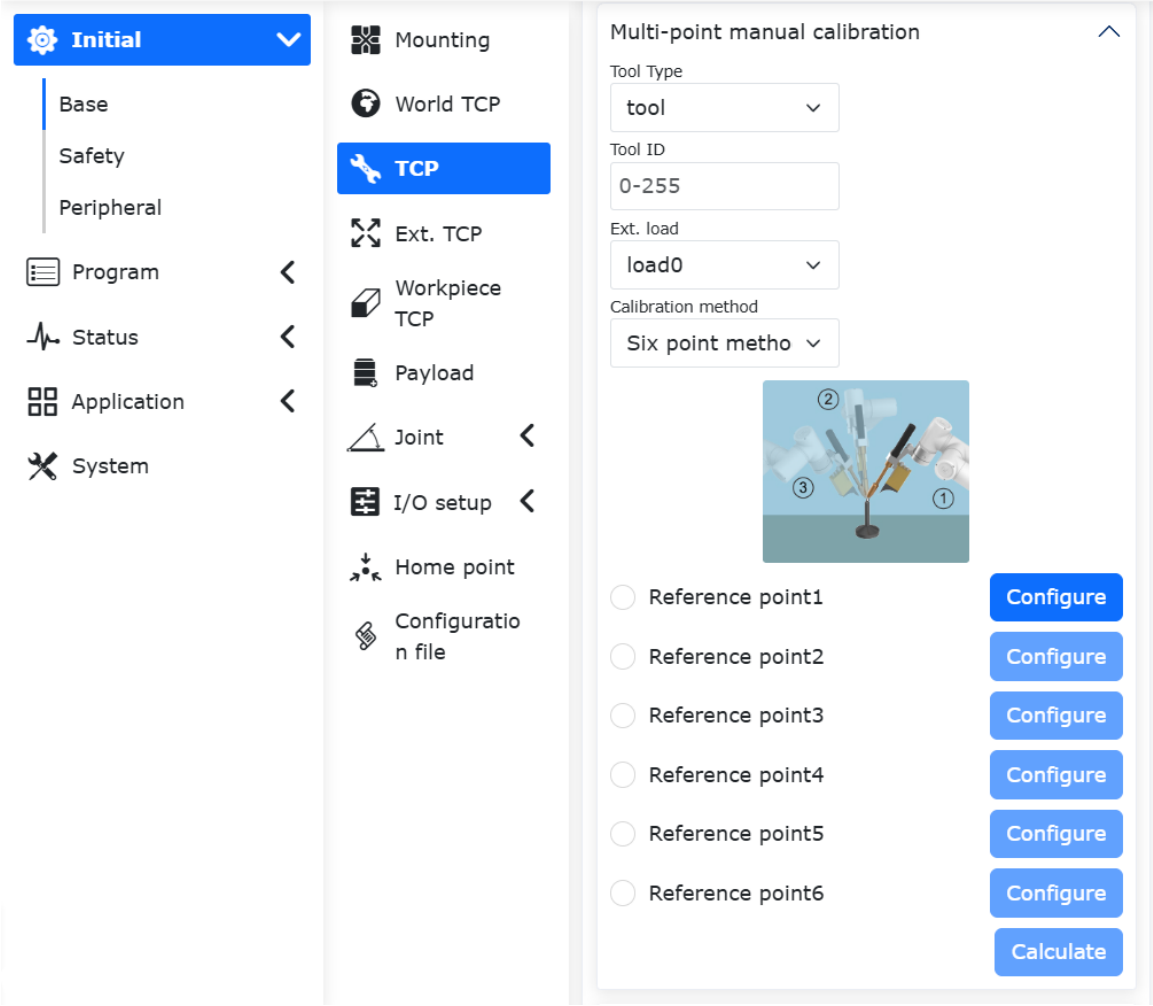


Figure 9.37-6 Select the 6-Point Method to Calibrate the Welding Torch

Select an empty coordinate system, choose the tool type as “Tool”, select the 6-point method to calibrate the welding torch tool. It is recommended to calibrate the posture of the tool coordinate system as shown in Figure 4-3 below.

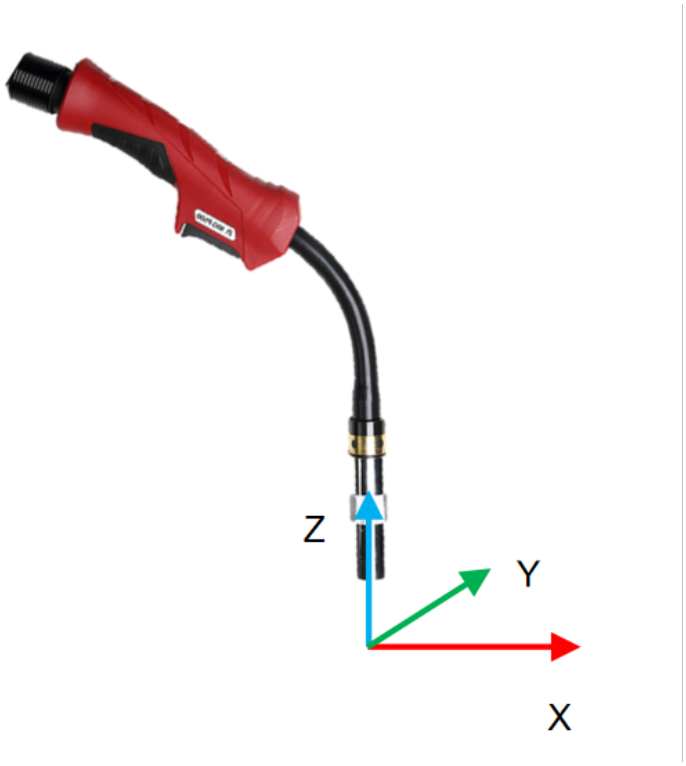


Figure 9.37-7 Welding Torch Coordinate System Posture Diagram

Tool coordinate system calibration content is not the focus of this function introduction. For detailed calibration methods, refer to the corresponding section of the user manual.

1.3.9.37.5 Intersecting Line Welding Function

There are two forms of trajectory motion for intersecting line welding: one is using a two-degree-of-freedom L-type positioner for intersecting line motion, and the other is performing intersecting line motion directly without using a positioner.

1.3.9.37.5.1 Extended Axis Coordinate System Calibration

When using the extended axis coordinate system to achieve synchronized motion between the positioner and the robot, the extended axis coordinate system needs to be calibrated.

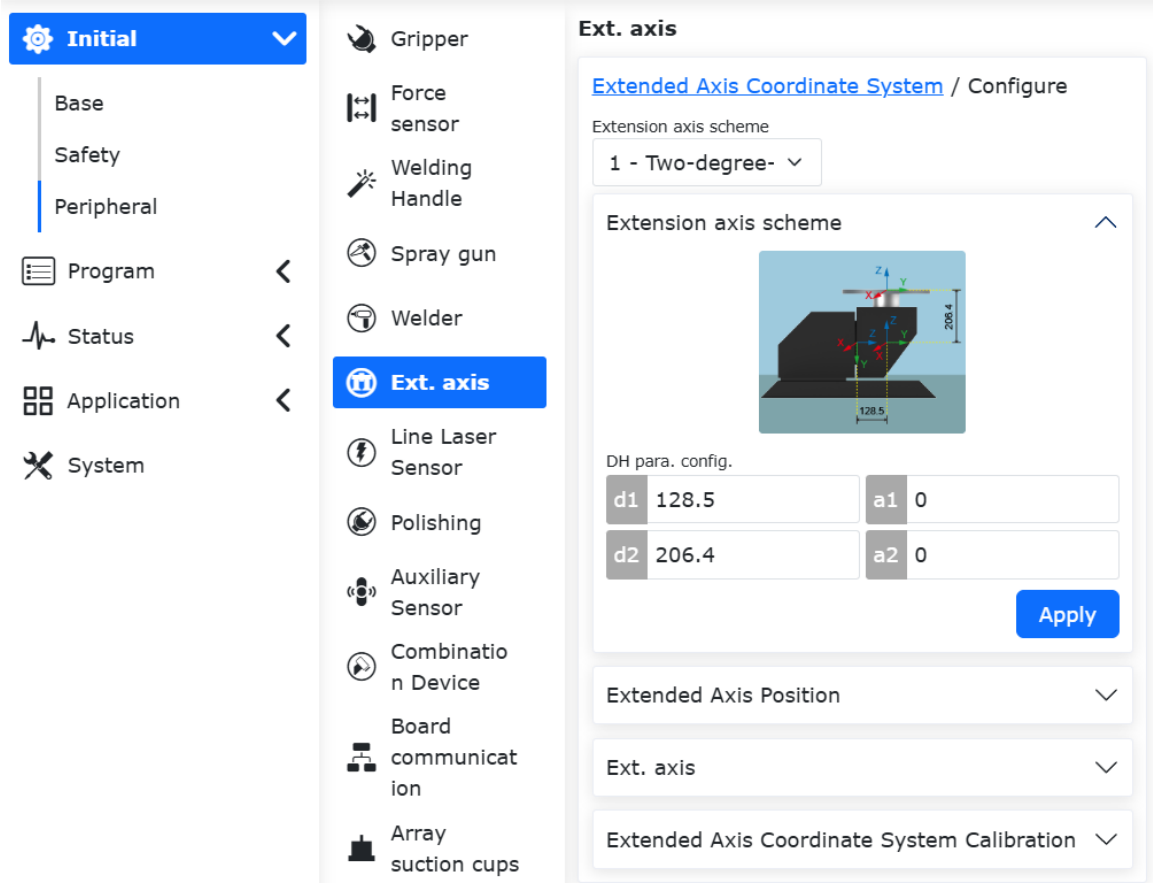


Figure 9.37-8 Extended Axis Coordinate System Settings Page

Click “Initial Setup”, “Peripherals”, “Extended Axis” to enter the extended axis coordinate system settings interface. Select the extended axis number that needs to be set, click the edit button, select “1-Two-degree-of-freedom L-type Positioner” and save.

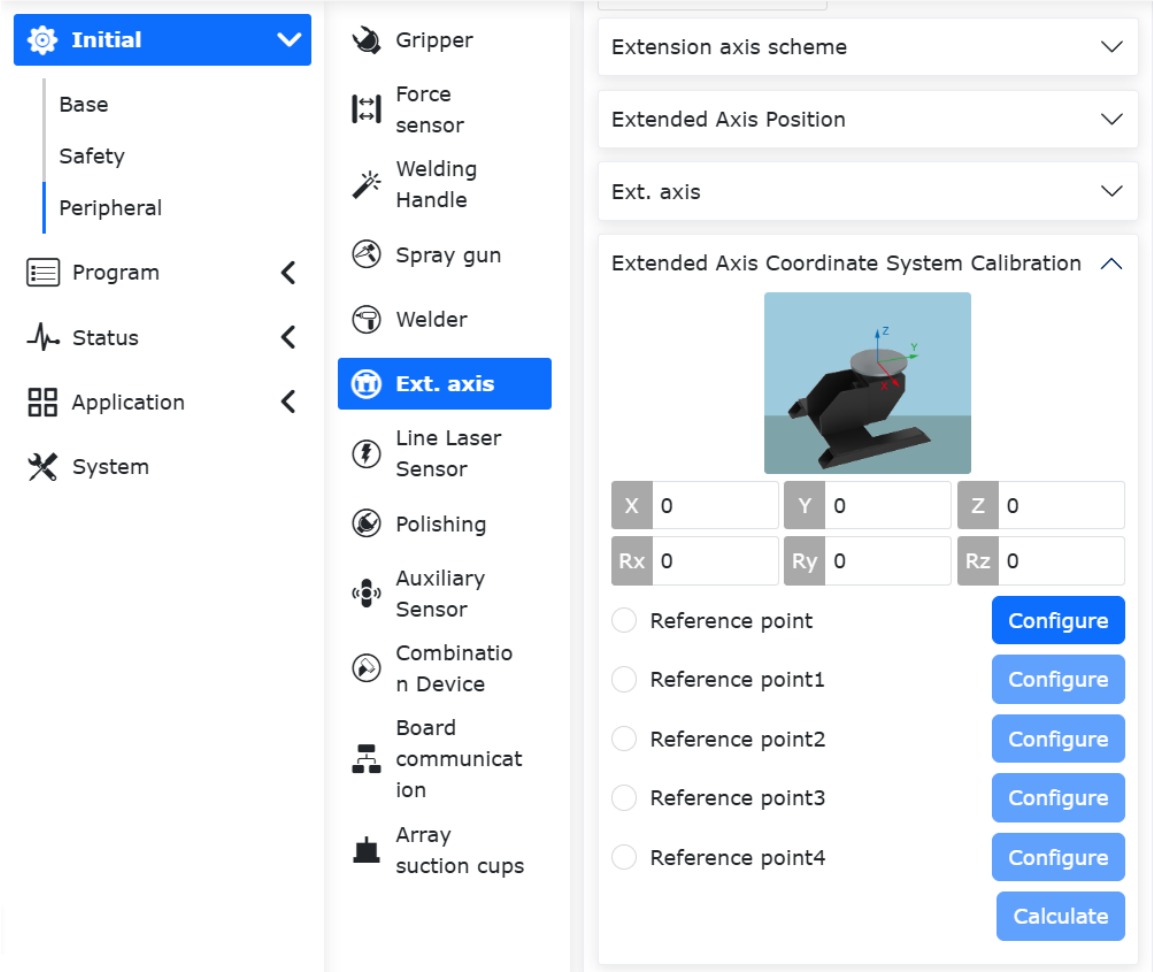


Figure 9.37-9 Extended Axis Calibration Page

When calibrating the extended axis, pay attention to select “Robot Position Relative to Extended Axis” as “Outside Extended Axis”. For the positioner case, select the 4-point method for calibration.

Extended axis calibration content is not the focus of this function introduction. For detailed calibration methods, refer to the corresponding section of the user manual.

1.3.9.37.5.2 Intersecting Line Trajectory Welding

Based on the taught points recorded on the cross-sections of the main pipe and the branch pipe, a workpiece coordinate system as shown in the figure below can be established. The origin of the coordinate system is located at the intersection point of the main pipe and branch pipe axes, the X-axis is parallel to the main pipe axis pointing towards the section where the taught points are recorded, and the Z-axis is parallel to the branch pipe axis pointing towards the plane where the taught points are recorded.

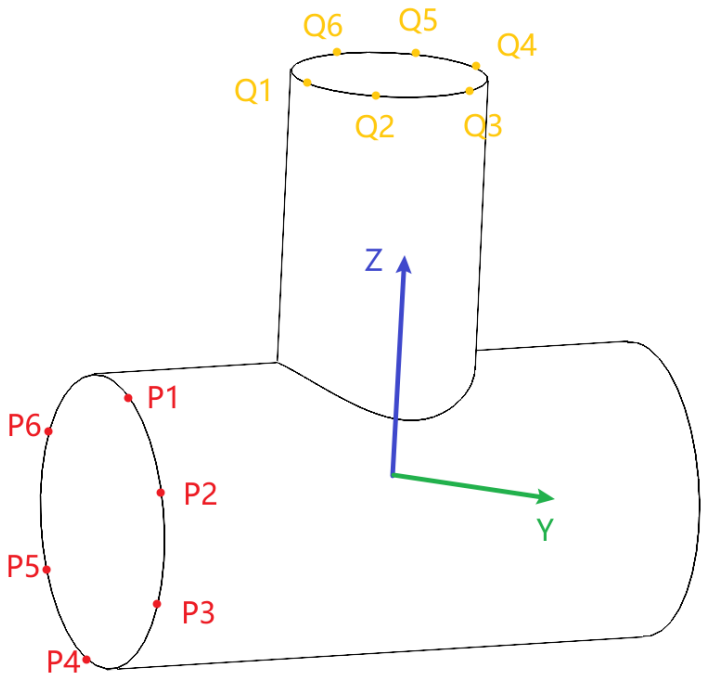


Figure 9.37-10 Intersecting Line Trajectory Workpiece Coordinate System

1.3.9.37.5.3 Method Without Using a Positioner

Step 1: Record 6 taught points on the cross-section of the main pipe and the cross-section of the branch pipe respectively.

Step 2: Click "Teach Program", "Program Programming", find "Intersecting Line" in the "Motion Instructions", and enter the intersecting line trajectory settings page.

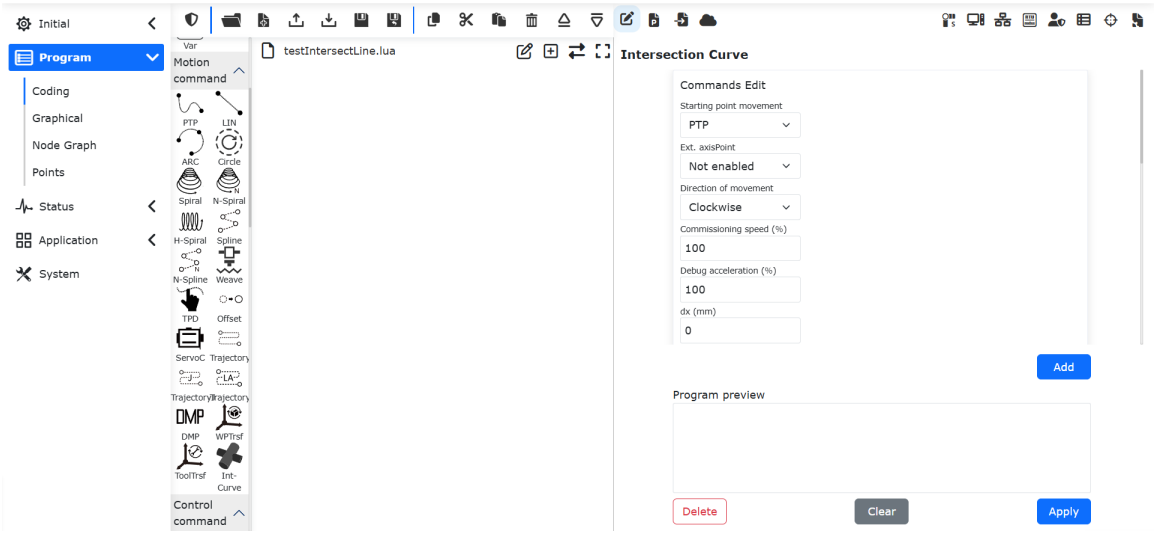


Figure 9.37-11 Intersecting Line Trajectory Settings Page

Step 3: On the intersecting line trajectory settings page, select "Disable" for "Extended Axis Points", complete the settings for starting point motion, motion direction, speed and acceleration, and offset value. The counterclockwise

motion direction is the direction of the four fingers when the right hand grips the Z-axis of the workpiece coordinate system.

Step 4: In the "Intersecting Line Point Data" section of the intersecting line trajectory settings page, select the recorded taught points. After setting, click the "Add", "Apply" buttons.

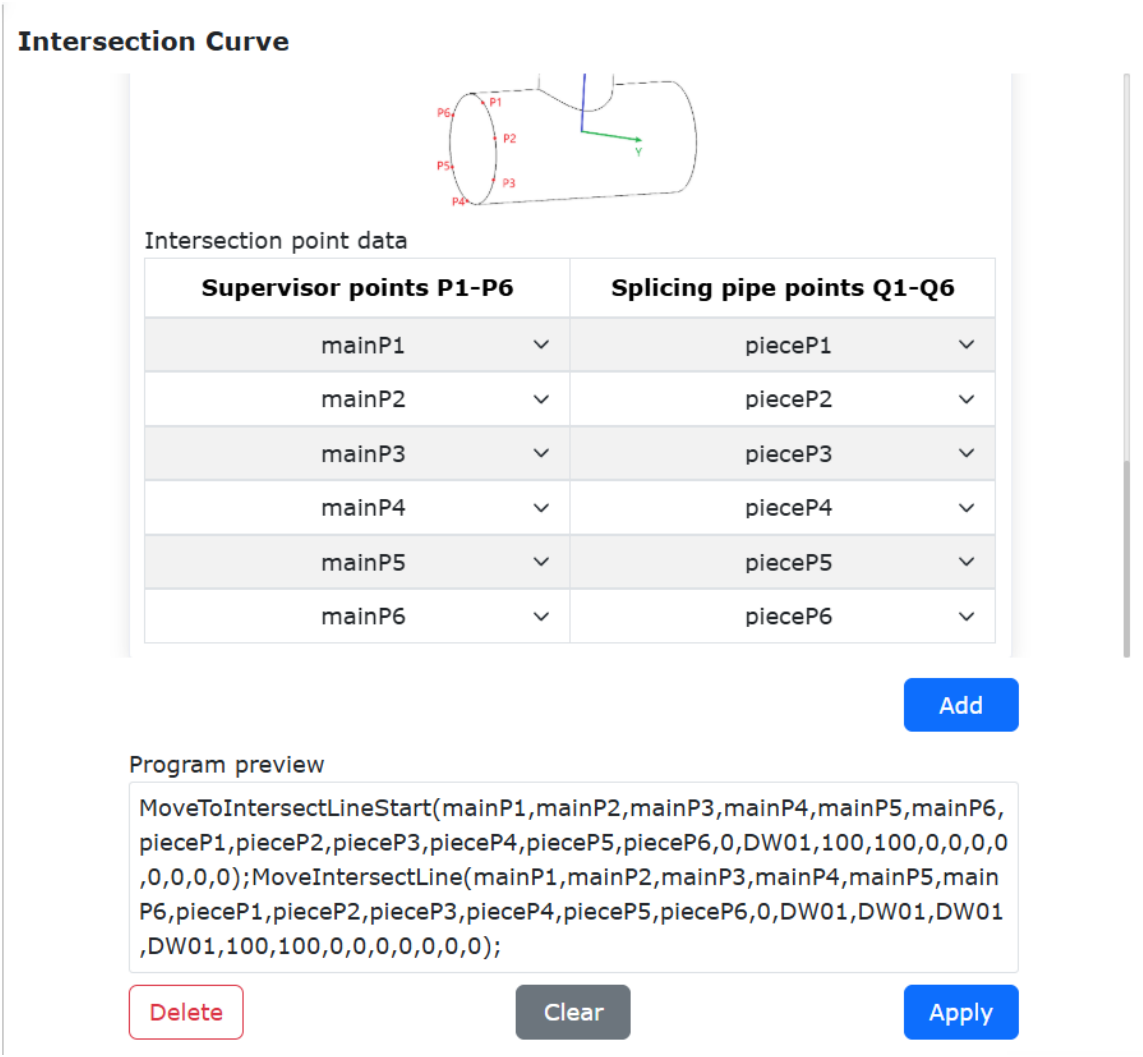


Figure 9.37-12 Intersecting Line Trajectory Instruction Settings

Step 5: Click the "Welding" button under "Welding Instructions" to enter the welding settings page, select "Arc Start" and "Arc End" instructions, click the "Add", "Apply" buttons. After successful addition, move the Arc Start LUA instruction up one line.

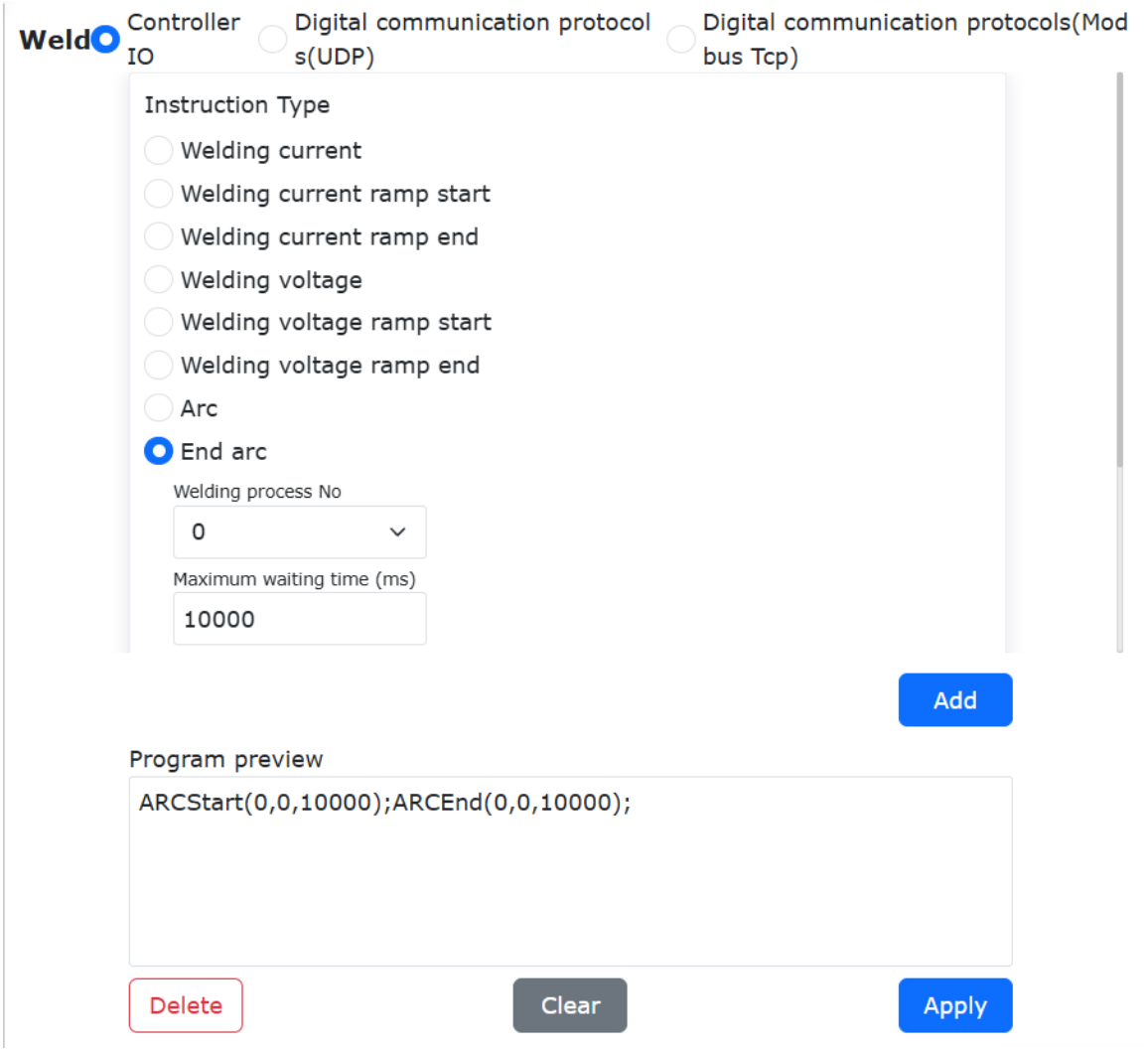


Figure 9.37-13 Welding Instruction Settings

The following is a typical LUA program for intersecting line welding without a positioner:



Figure 9.37-14 No Positioner - Intersecting Line Welding Example Program

1.3.9.37.5.4 Method Using a Two-Degree-of-Freedom L-Type Positioner

Step 1: Record 6 taught points on the cross-section of the main pipe and the cross-section of the branch pipe respectively, and rotate axes 1 and 2 of the positioner to record 4 positioner taught points.

Step 2: Click “Teach Program”, “Program Programming”, find “Intersecting Line” in the “Motion Instructions”, and enter the intersecting line trajectory settings page.

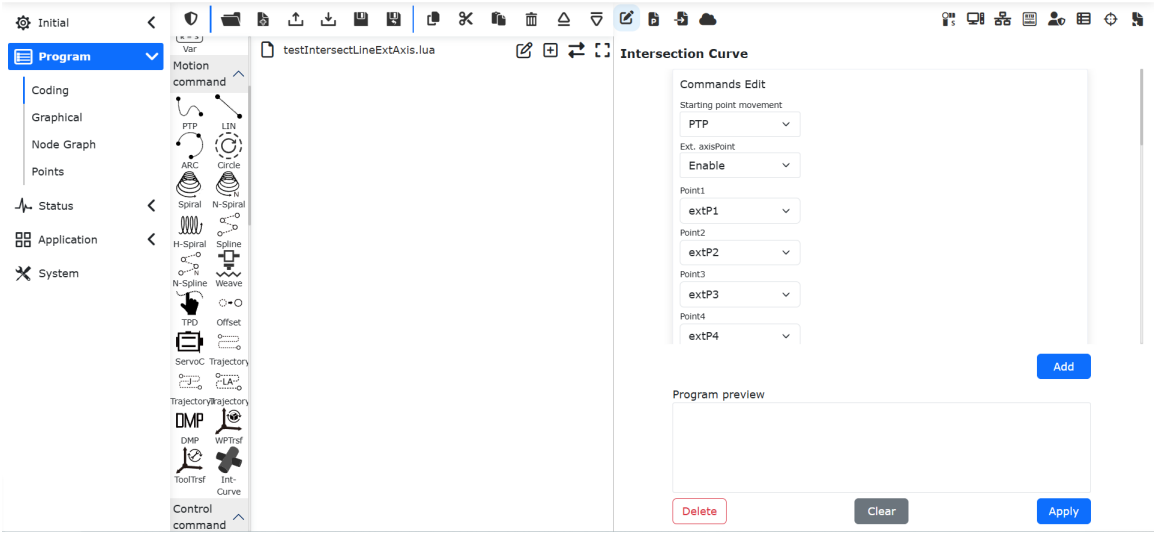


Figure 9.37-15 Intersecting Line Trajectory Settings Page

Step 3: On the intersecting line trajectory settings page, select “Enable” for “Extended Axis Points”, select the recorded positioner taught points, complete the settings for starting point motion, motion direction, speed and acceleration, and offset value.

Step 4: In the “Intersecting Line Point Data” section of the intersecting line trajectory settings page, select the recorded taught points. After setting, click the “Add”, “Apply” buttons.

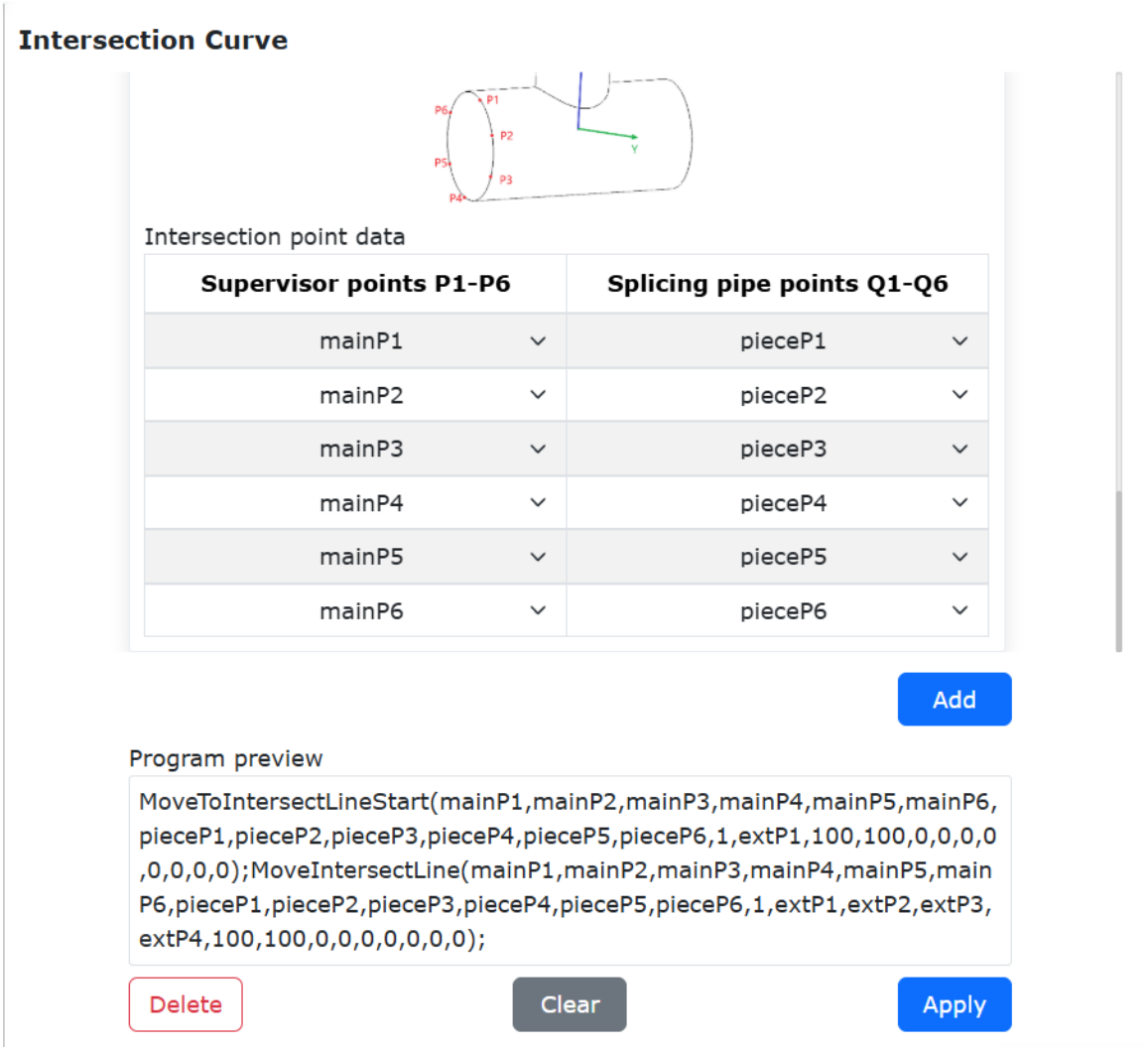


Figure 9.37-16 Intersecting Line Trajectory Instruction Settings

Step 5: Click the “Welding” button under “Welding Instructions” to enter the welding settings page, select “Arc Start” and “Arc End” instructions, click the “Add”, “Apply” buttons. After successful addition, move the Arc Start LUA instruction up one line.

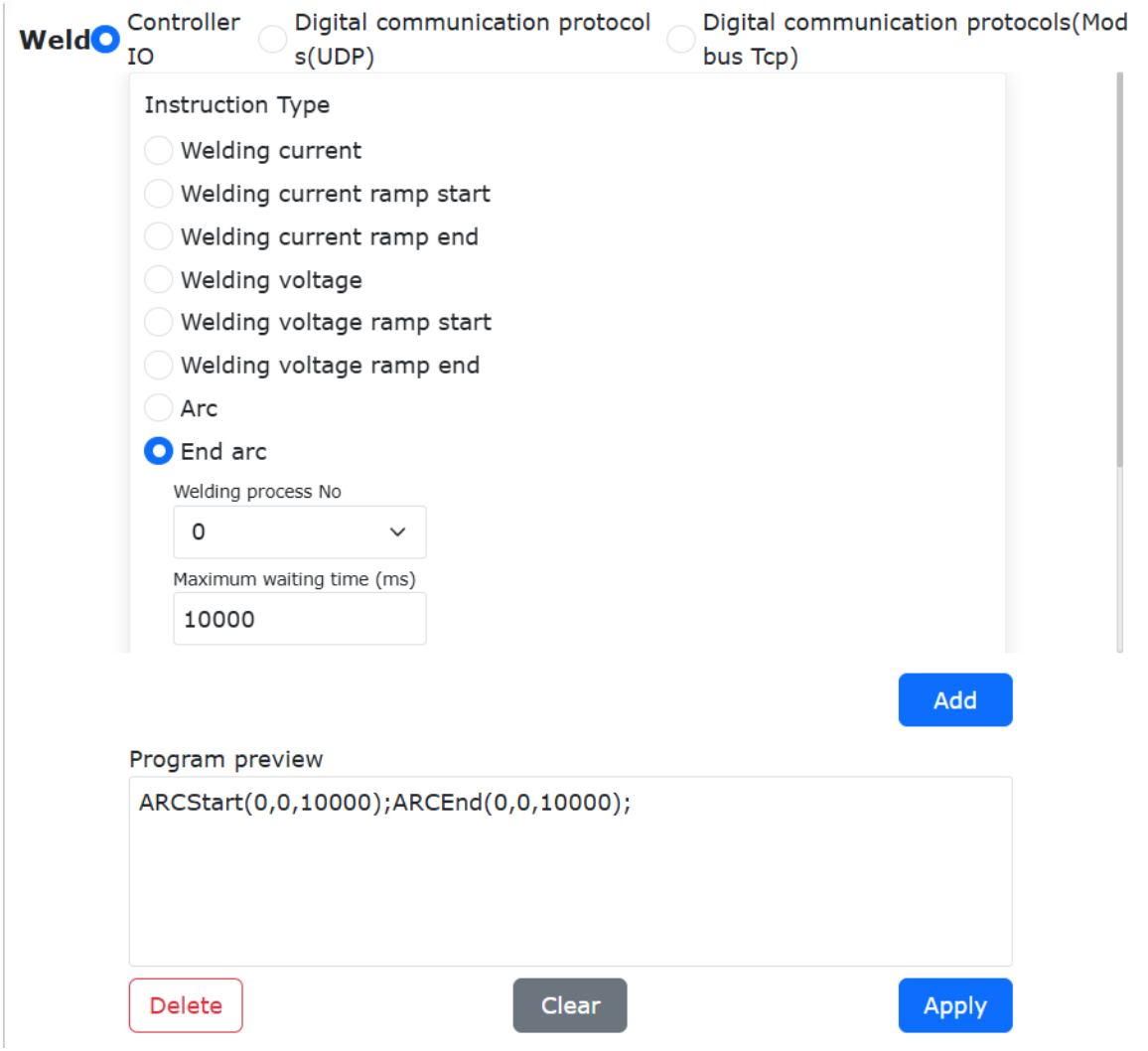


Figure 9.37-17 Welding Instruction Settings

The following is a typical LUA program for positioner-based intersecting line welding:

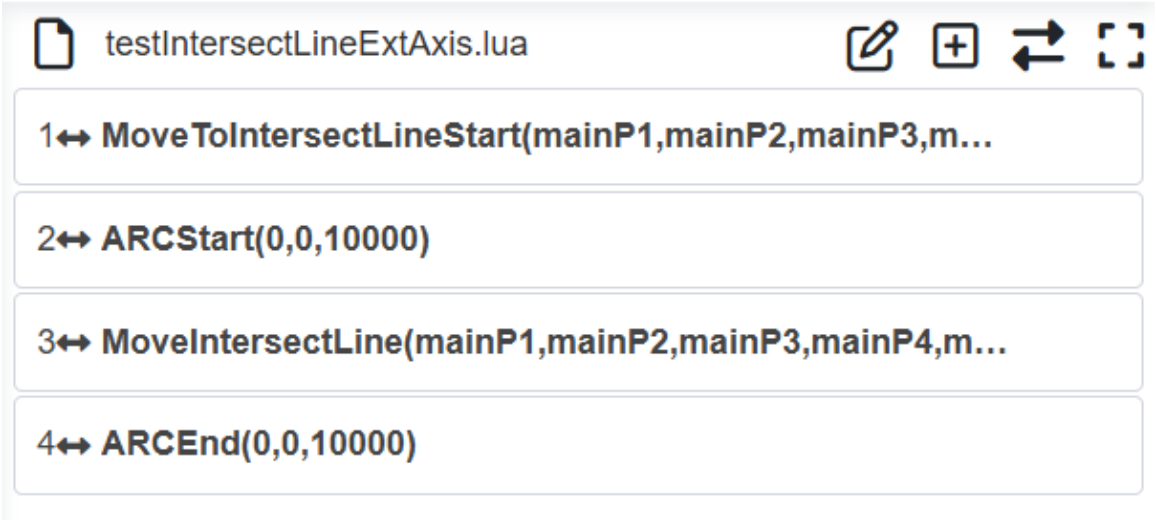


Figure 9.37-18 Positioner - Intersecting Line Welding Example Program

1.3.10 Graphical programming

1.3.10.1 Introduction

Since the teaching pendant is generally not connected to external peripherals such as keyboard and mouse, when accessing the robot WebAPP on the teaching pendant side, users can edit the robot teaching program through the graphical programming function. Functional standardization functions are implemented using the Blockly library, which can be integrated into the WebAPP system, and custom code blocks can be implemented as needed. After dragging and dropping programming, it is converted into a LUA program and sent and run through the existing instruction protocol.

By using graphical programming, it can be simple and easy to understand, easy to operate, and Chinese language operation.

The page is divided into three areas: "operation bar", "toolbox toolbar" and "workspace code editing area". The overall layout design is as follows.

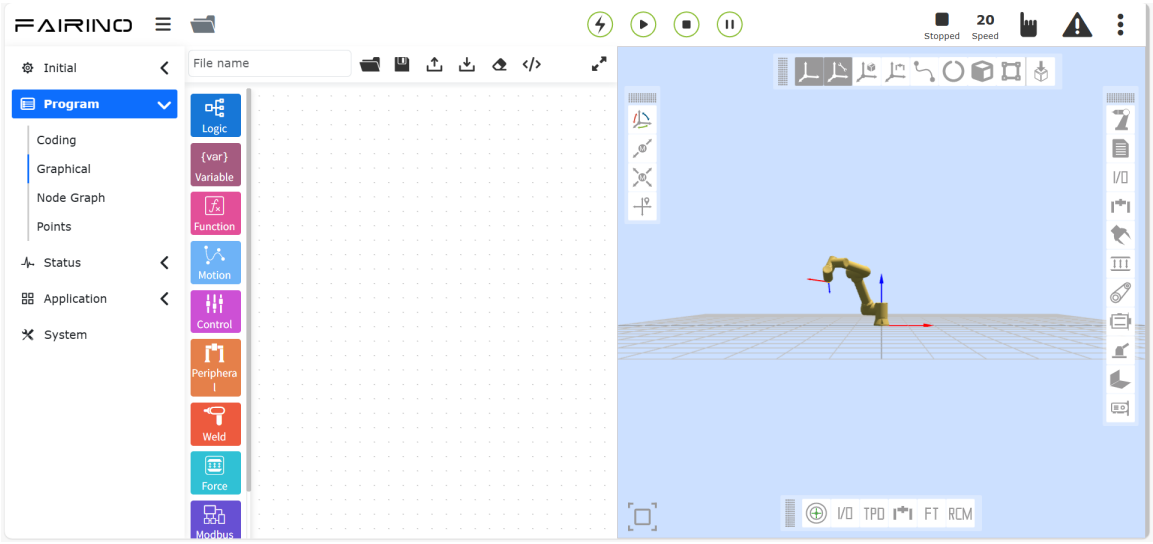


Figure 10.1 Graphical programming interface

Operation bar

1) **Load**: responsible for reloading the workspace;
2) **Import**: responsible for importing related graphical programming programs;
3) **Export**: responsible for exporting the saved graphical programming programs in the workspace. The "Save" button function is to save the code block as the corresponding teaching program after editing is completed;
4) **Save**: responsible for saving the edited graphical code block;
5) **Clear**: responsible for quickly clearing the code editing area;
6) **Code**: responsible for translating code blocks into Lua code.

Toolbox

1) Code blocks containing all instructions and logic codes can be dragged to the workspace to create code blocks and edit them;
2) The Toolbox toolbar section will be further classified according to the instruction type;

3) Logic instructions: if-else, while, etc;

4) Basic motion instructions: PTP, LIN, ARC, etc. Instructions are classified according to application scenarios: gluing, welding, conveyor belts, etc. You can easily find the required code blocks during use.

Workspace: Graphical code blocks can be edited and displayed in the code editing area.

1.3.10.2 Logic Graphical Programming Commands

Logic Graphical Programming Commands include loops, numbers and other logic commands.

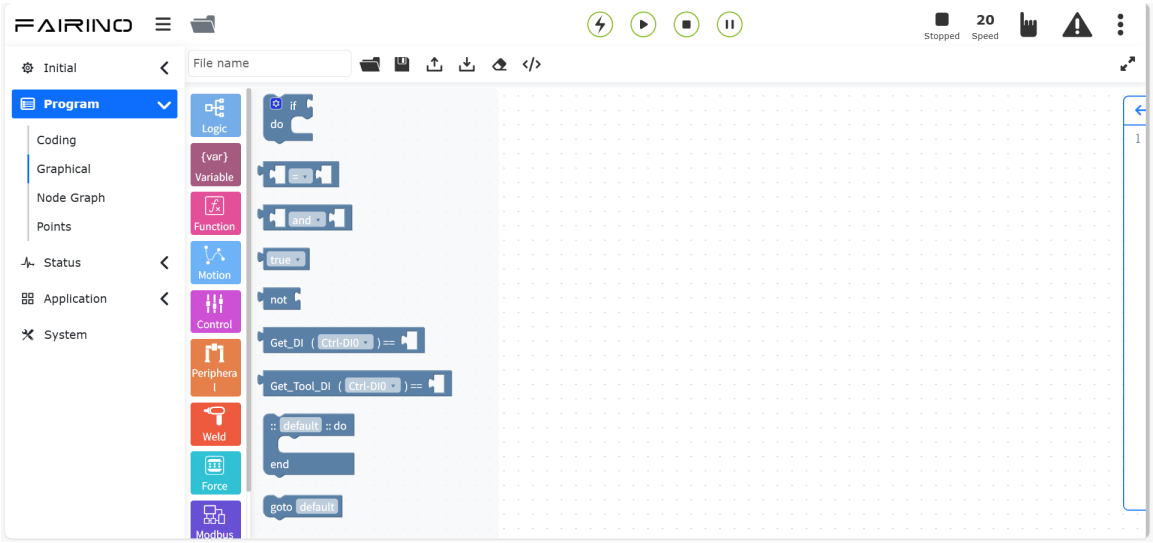


Figure 10.2 Logic Graphical Programming

1.3.10.2.1 If/Else Judgment Instruction

Drag the “If/Else Judgment Instruction” code block to enter the graphical editing interface workspace. (This instruction requires a certain programming foundation. If you need help, please contact us)

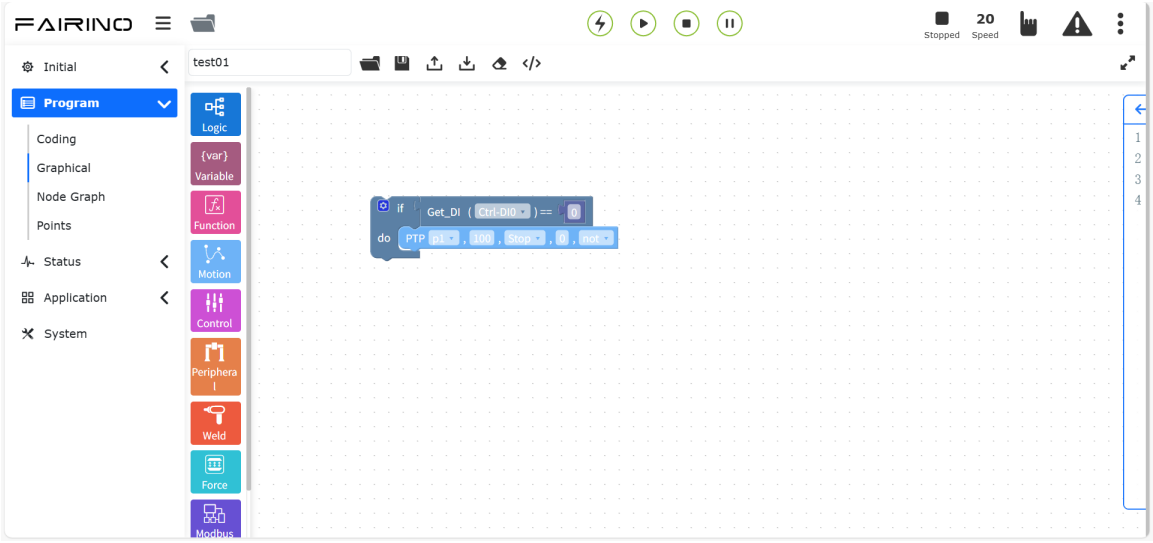


Figure 10.2-1 If/Else Judgment Instruction Code Block

1.3.10.2.2 While Instruction

Drag the “While Instruction” code block to enter the graphical editing interface workspace. (This instruction requires a certain programming foundation. If you need help, please contact us)

Add the input waiting condition after While, add the motion instruction code block inside while, and click Save. (For ease of operation, you can enter any do content and edit other instructions in the program to insert instead)

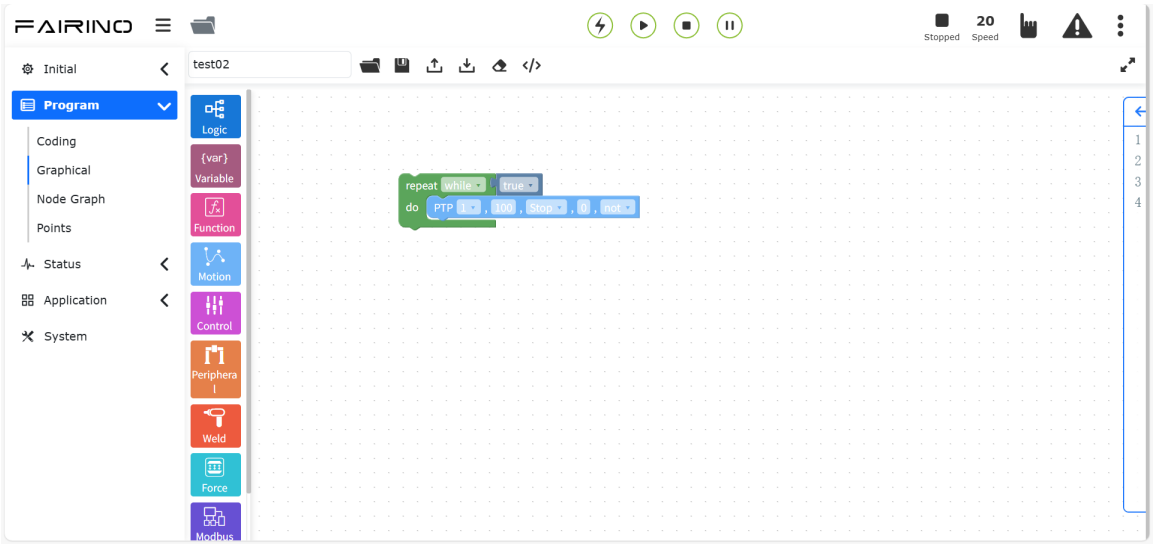


Chart 10.2-2 While instruction code block

1.3.10.2.3 Jump instruction

Drag the “Jump instruction” code block to enter the graphical editing interface workspace. (This instruction requires a certain programming foundation. If you need help, please contact us)

- Jump name: Enter the jump name to determine the jump position

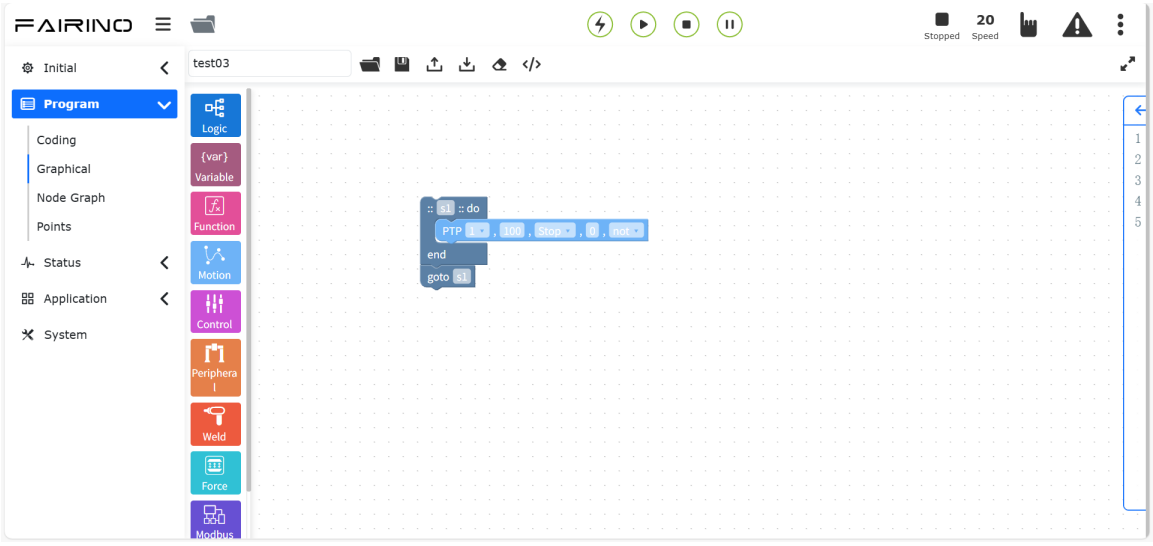


Figure 10.2-3 Jump instruction code block

Important: Jump name cannot start with a number

1.3.10.3 Variable class graphical programming command

Variable class graphical programming commands include create variable commands.

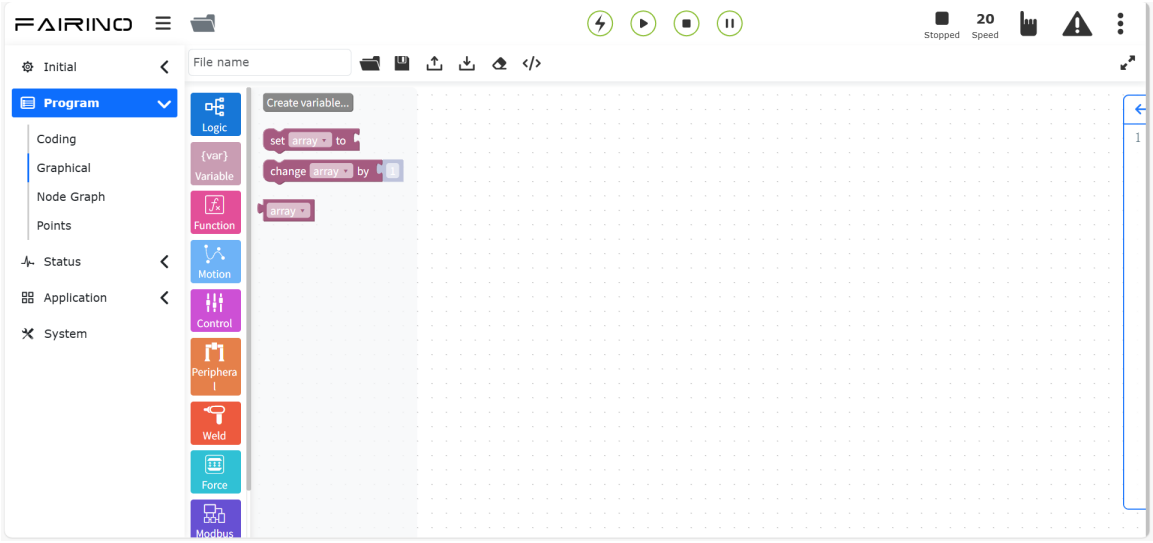


Figure 10.3 Variable class graphical programming

1.3.10.3.1 Variable instruction

Click the “Create” button to enter the variable name to be defined.

Drag the “Variable instruction” code block to enter the graphical editing interface workspace.

“Variable” instruction node, parameters:

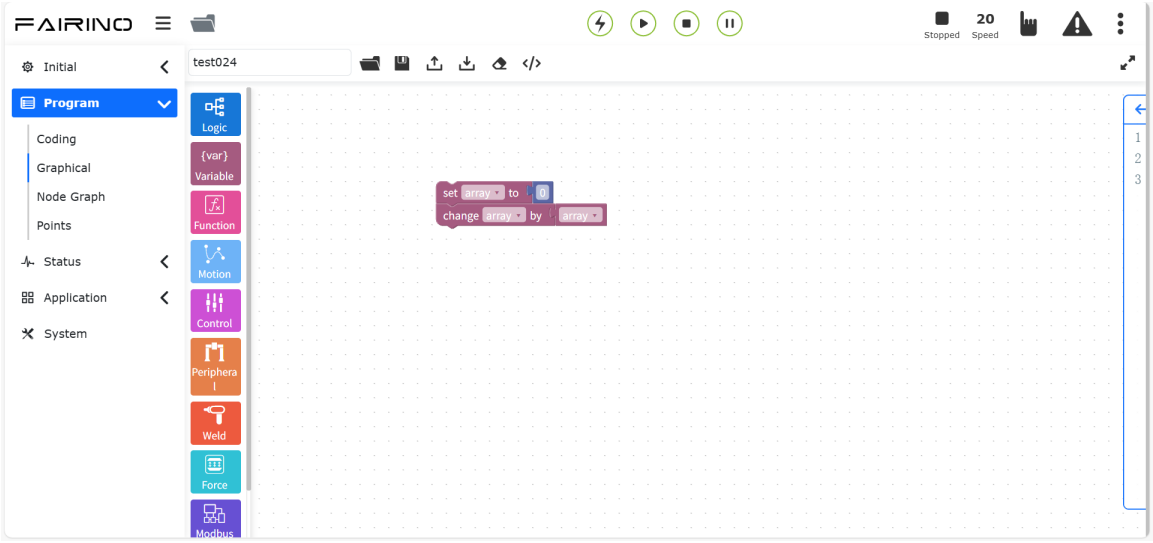


Figure 10.3-1 Variable instruction code block

1.3.10.4 Function class graphical programming command

Function class graphical programming command includes create function command.

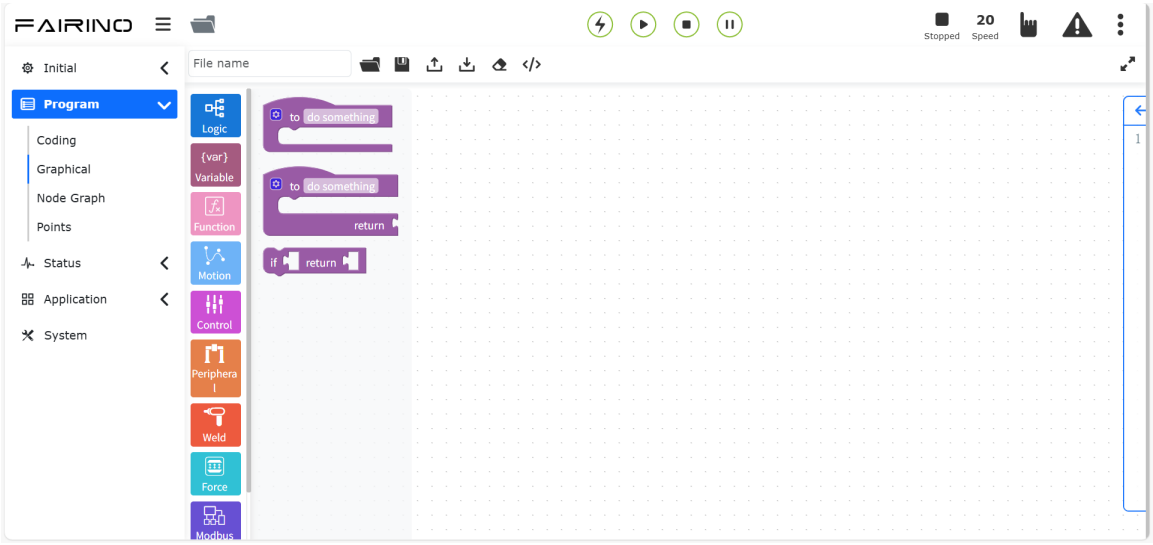


Figure 10.4 Function class graphical programming

1.3.10.4.1 Function method instruction

Drag the “Function method instruction” code block to enter the graphical editing interface workspace.

“Function method” instruction node, parameters:

- Function name: the name of the function to be run.

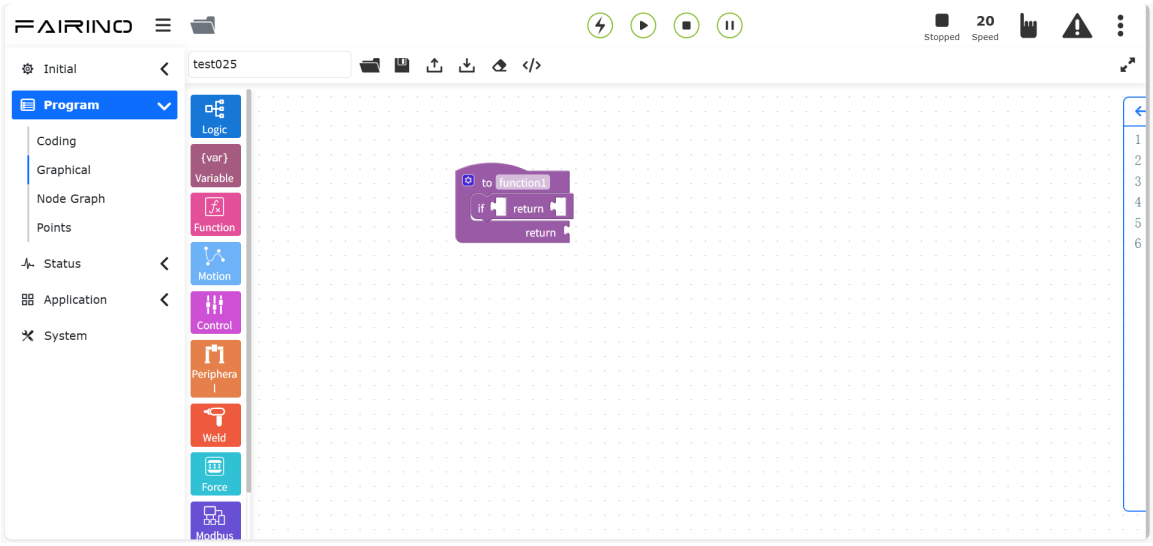


Figure 10.4-1 Function method instruction code block

1.3.10.5 Motion graphic programming commands

Motion graphic programming commands include PTP, Lin, ARC and other motion commands.

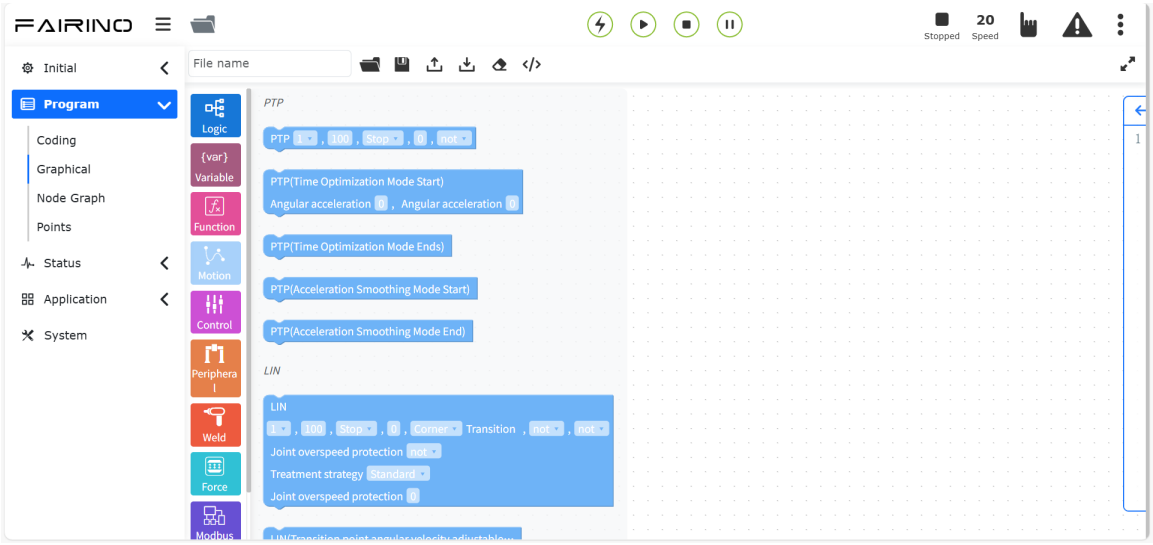


Figure 10.5 Motion graphic programming

1.3.10.5.1 Point-to-point instruction

Drag the “point-to-point instruction” code block to enter the graphical editing interface workspace.

You can select the point you want to reach. The smooth transition time setting can make the movement from this point to the next point continuous. For the offset setting, you can choose to offset based on the base coordinate system and offset based on the tool coordinate system, and pop up the x, y, z, rx, ry, rz offset settings. The specific PTP path is the optimal path automatically planned by the motion controller.

“Point to point” instruction node, parameters:

- Point name: teaching point
- Debug speed (%): 0 ~ 100
- Stop: false/true
- Smooth transition (ms): Smooth transition time 0 ~ 500
- Whether to offset No/base coordinate offset/tool coordinate offset When No is selected, the dx~drz parameter values are not effective
- dx~drz: offset

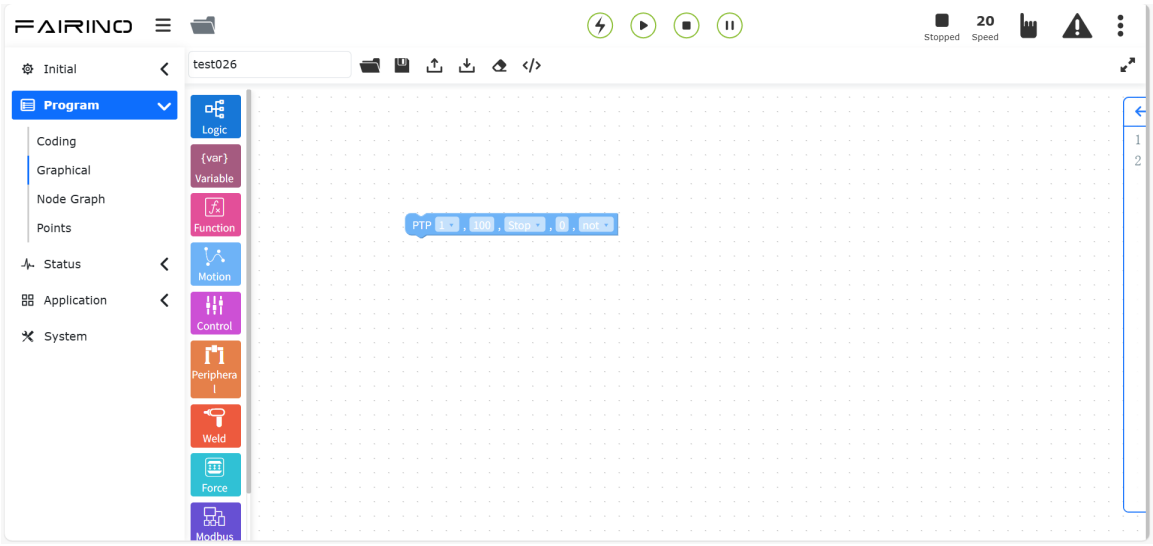


Chart 10.5-1 Point-to-point instruction code block

1.3.10.5.2 Straight line instruction

Drag the “Straight line instruction” code block to enter the graphical editing interface workspace.

This instruction function is similar to the “point to point” instruction, but the path to the point reached by this instruction is a straight line.

“Straight line” instruction node, parameters:

- Point name: teaching point
- Debug speed (%): 0 ~ 100
- Stop: false/true, when true is selected, the smooth transition parameter value is not effective
- Smooth transition (mm): smooth transition radius 0 ~ 1000
- Whether to seek: false/true
- Seek point variable: REF0~99/RES0~99, when false is selected for whether to seek, the parameter is not effective;
- Whether to offset: No
- Joint overspeed protection: No/Yes
- Processing strategy: Standard/Stop with error when overspeeding/Adaptive deceleration
- Allowed deceleration threshold: 0~100

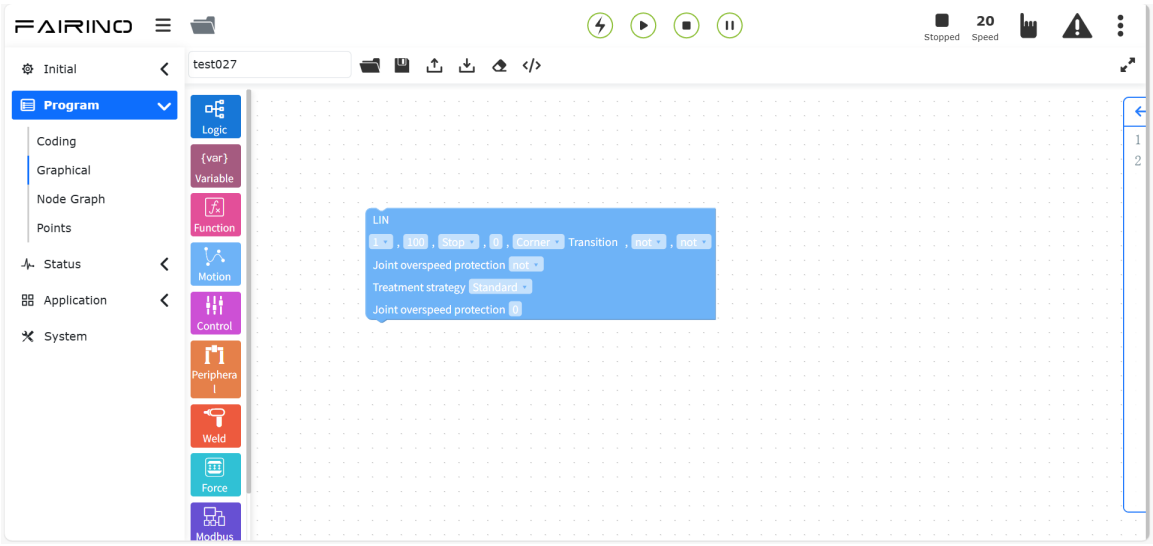


Chart 10.5-2 Straight line instruction code block

1.3.10.5.3 Straight line (adjustable angular velocity at transition point) instruction

Drag the “Straight line (adjustable angular velocity at transition point) instruction” code block to enter the graphical editing interface workspace.

The function of this instruction is similar to the “point to point” instruction, but this instruction includes adjustable angular velocity at transition point.

“Straight line (adjustable angular velocity at transition point)” command node, parameters:

- Maximum angular velocity: 0~300

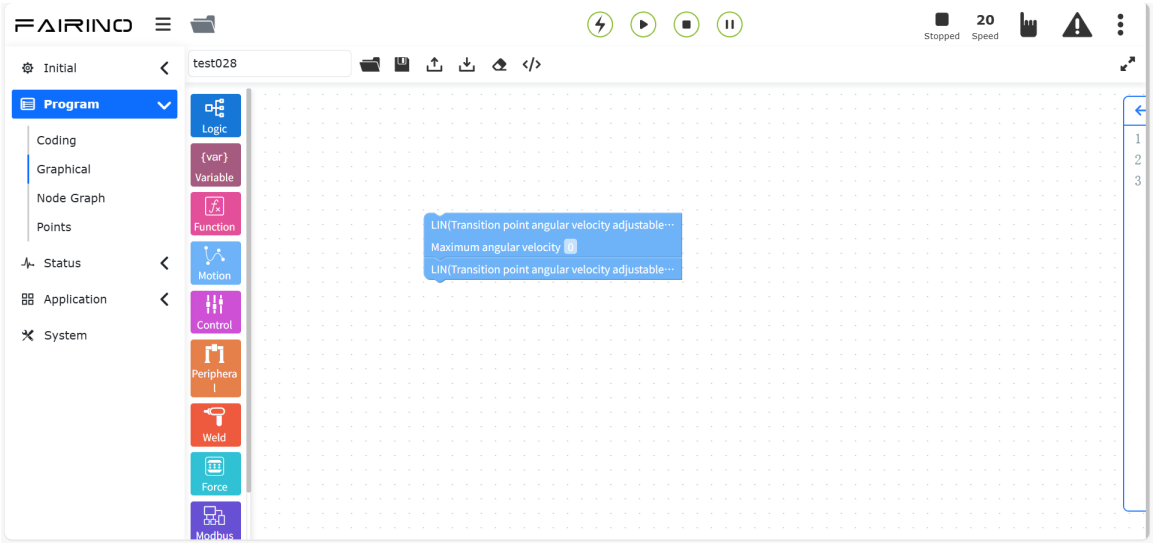


Chart 10.5-3 Straight line (adjustable angular velocity at transition point) command code block

1.3.10.5.4 Straight line (seamPos) command

Drag the “Straight line (seamPos) command” code block to enter the graphical editing interface workspace.

This command function is used to use laser sensors in welding scenarios.

“Straight line (seamPos)” command node, parameters:

- Point name: teaching point
- Debug speed (%): 0 ~ 100
- Stop: false/true, when true is selected, the smooth transition parameter value is not effective
- Smooth transition (mm): smooth transition radius 0 ~ 1000
- Weld cache data selection: execution planning data/execution record data;
- Plate type: corrugated plate/corrugated plate/fence plate/oil drum/corrugated shell steel;
- Whether to offset: No/base coordinate offset/tool coordinate offset/laser raw data offset When no is selected, the dx~drz parameter value is not effective
- dx~drz: offset

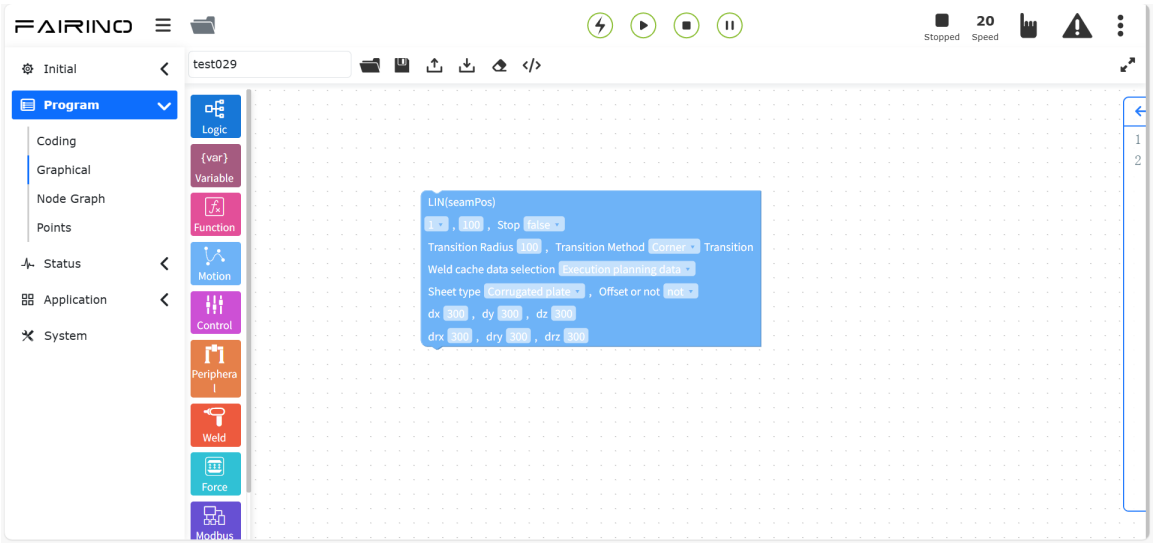


Chart 10.5-4 Straight line (seamPos) command code block

1.3.10.5.5 Arc command

Drag the “Arc command” code block to enter the graphical editing interface workspace.

Arc motion contains two points, the first point is the transition point in the middle of the arc, and the second point is the end point. Both the transition point and the end point can be set to offset or not. You can choose to offset based on the base coordinate system and offset based on the tool coordinate system, set the x, y, z, rx, ry, rz offset, and the end point can set the smooth transition radius to achieve a continuous motion effect

“Arc” instruction node, parameters:

- Arc middle point: teaching point
- Whether to offset: No/base coordinate offset/tool coordinate offset When no is selected, the dx~drz parameter value is not effective

- dx~drz: offset
- Arc end point: teaching point
- Whether to offset: No/base coordinate offset/tool coordinate offset When no is selected, the dx~drz parameter value is not effective
- dx~drz: offset
- Debug speed (%): 0 ~ 100
- Stop: false/true, when true is selected, the smooth transition parameter value is not effective
- Smooth transition (mm): smooth transition radius 0 ~ 1000

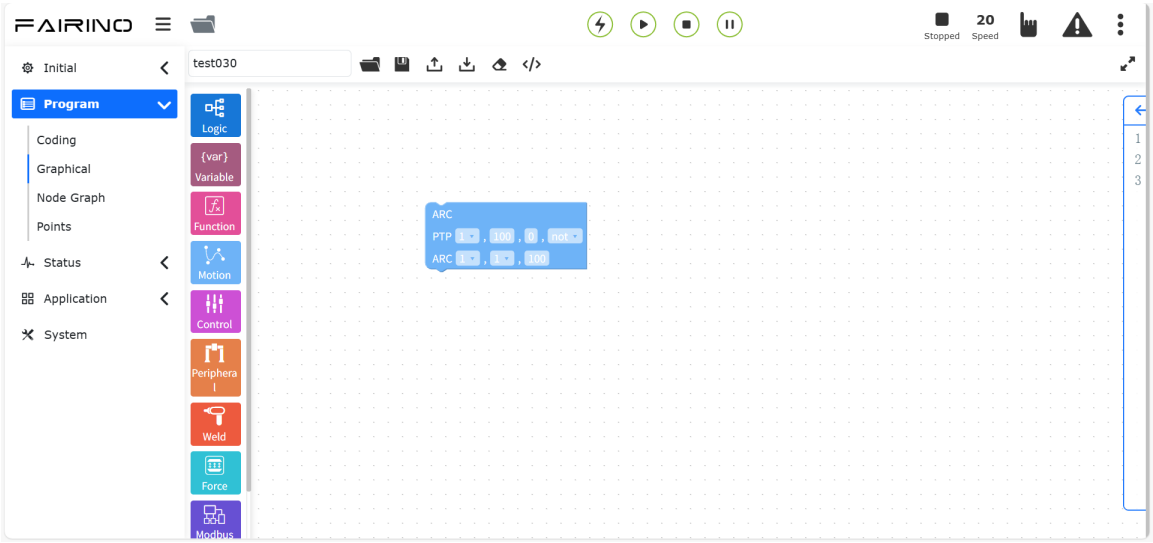


Figure 10.5-5 Arc instruction code block

1.3.10.5.6 Full circle command

Drag the “full circle command” code block to enter the graphical editing interface workspace.

Click the “full circle” command node to enter the node diagram editing interface.

Full circle motion includes two points, the first point is the full circle middle transition point 1, the second point is the full circle middle transition point 2, transition point 2 can be set whether to offset, the offset is effective for both transition point 1 and transition point 2.

“Full circle” command node, parameters:

- Full circle middle point 1: teaching point
- Full circle middle point 2: teaching point
- Debug speed (%): 0 ~ 100
- Whether to offset: No/base coordinate offset/tool coordinate offset When No is selected, the dx~drz parameter value is not effective
- dx~drz: offset

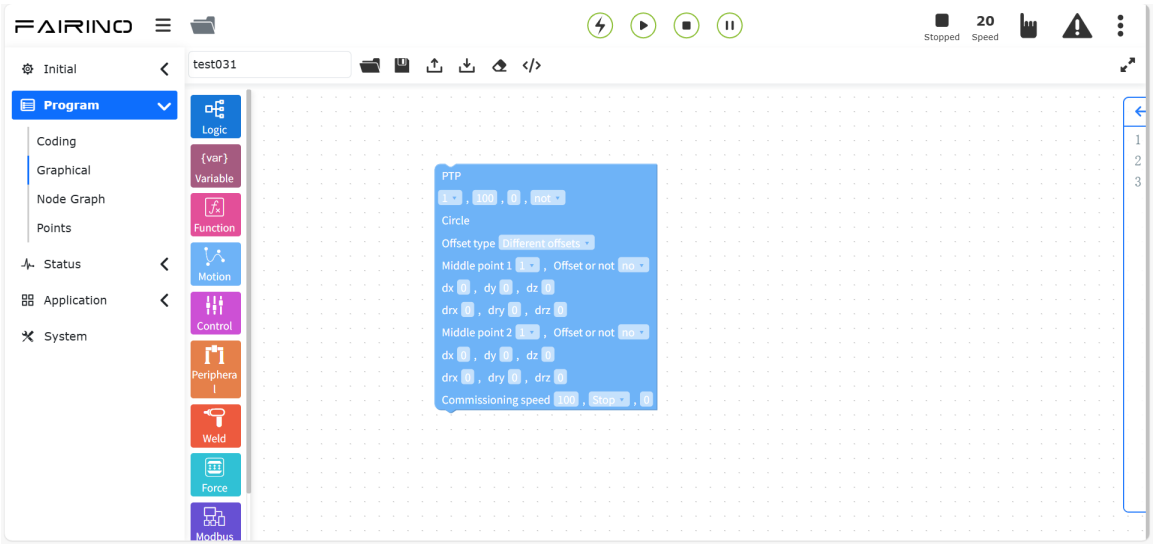


Chart 10.5-6 Full circle instruction code block

1.3.10.5.7 Spiral instruction

Drag the “Spiral instruction” code block to enter the graphical editing interface workspace.

The spiral motion includes three points, which form a circle. In the third point setting page, there are several parameter settings, including the number of spiral turns, attitude correction angle, radius increment and axis direction increment. The number of spiral turns is the number of spiral motion turns. The attitude correction angle corrects the attitude at the end of the spiral and the attitude of the first point of the spiral. The radius increment is the increment of each circle radius, and the axis direction increment is the increment of the spiral axis direction. Set whether to offset. The offset is effective for the entire spiral trajectory.

“Spiral” command node, parameters:

- Spiral middle point 1: teaching point
- Spiral middle point 2: teaching point
- Spiral middle point 3: teaching point
- Debug speed (%): 0 ~ 100
- Offset: No/base coordinate offset/tool coordinate offset When No is selected, the dx~drz parameter values are not effective
- dx~drz: offset
- Spiral turns: 0 ~ 100
- Posture angle correction rx (°): -1000 ~ 1000
- Posture angle correction ry (°): -1000 ~ 1000
- Posture angle correction rz (°): -1000 ~ 1000
- Radius increment (mm): -100 ~ 100
- Rotation axis direction increment (mm): -100 ~ 100

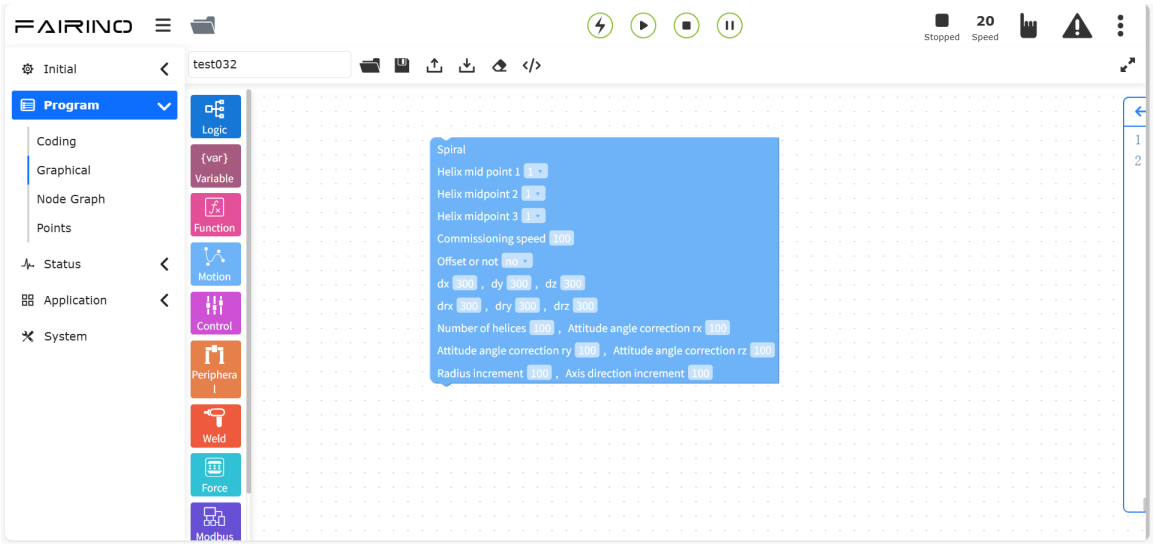


Figure 10.5-7 Spiral instruction code block

1.3.10.5.8 New spiral instruction

Drag the “New Spiral Instruction” code block to enter the graphical editing interface workspace.

Click the “New Spiral” instruction node to enter the node graph editing interface.

The new spiral motion is an optimized version of the spiral motion. This instruction only requires one point plus the configuration of various parameters to realize the spiral motion. The robot takes the current position as the starting point. The user sets the debugging speed, whether to offset, the number of spiral turns, spiral inclination, initial radius, radius increment, axis direction increment and rotation direction. The number of spiral turns refers to the number of spiral turns of the spiral line. The spiral inclination refers to the angle between the tool Z axis and the horizontal direction. The posture correction angle corrects the posture at the end of the spiral line and the posture of the first point of the spiral line. The initial radius refers to the radius of the first circle. The radius increment refers to the increment of each circle radius. The axis direction increment refers to the increment of the spiral axis direction. The rotation direction is clockwise and counterclockwise.

“New spiral” command node, parameters:

- Spiral start point: teaching point
- Debug speed (%): 0 ~ 100
- Offset: No/base coordinate offset/tool coordinate offset When No is selected, the dx~drz parameter values are not effective
- dx~drz: offset
- Spiral turns: 0 ~ 100
- Spiral inclination (°): -100 ~ 100
- Initial radius: 0 ~ 100
- Radius increment (mm): -100 ~ 100
- Rotation axis direction increment (mm): -100 ~ 100
- Rotation direction: clockwise/counterclockwise

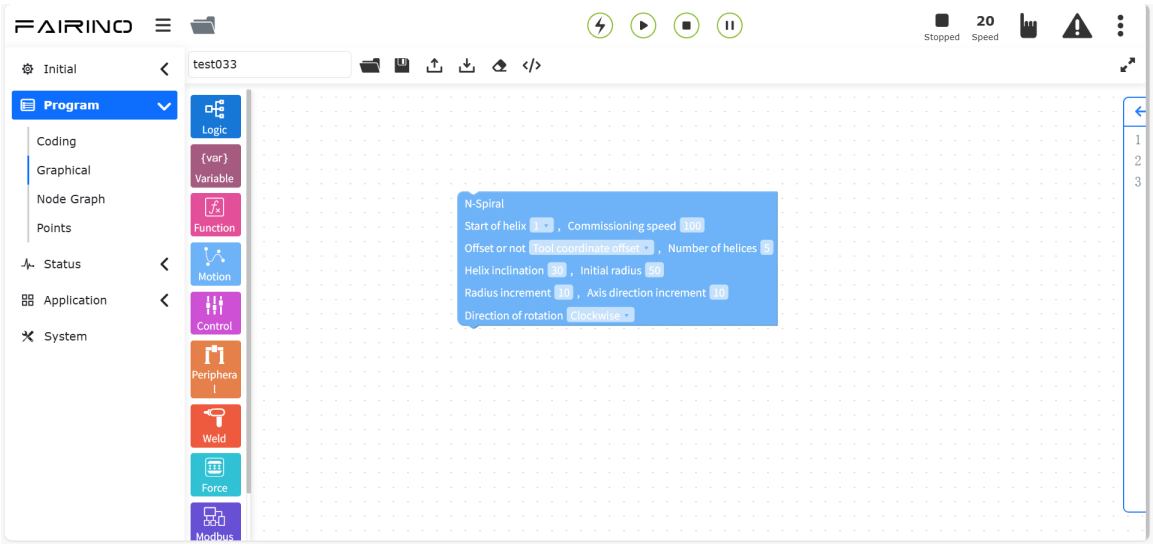


Chart 10.5-8 New spiral instruction code block

1.3.10.5.9 Horizontal spiral instruction

Drag the “Horizontal spiral instruction” code block to enter the graphical editing interface workspace.

The “H-Spiral” instruction is a horizontal space spiral motion. This instruction is set after the single-segment motion (straight line) instruction.

“Horizontal spiral” instruction node, parameters:

- Spiral radius: 0~100mm
- Spiral angular velocity: 0~2rev/s
- Rotation direction: spiral clockwise/counterclockwise
- Spiral inclination: 0~40°

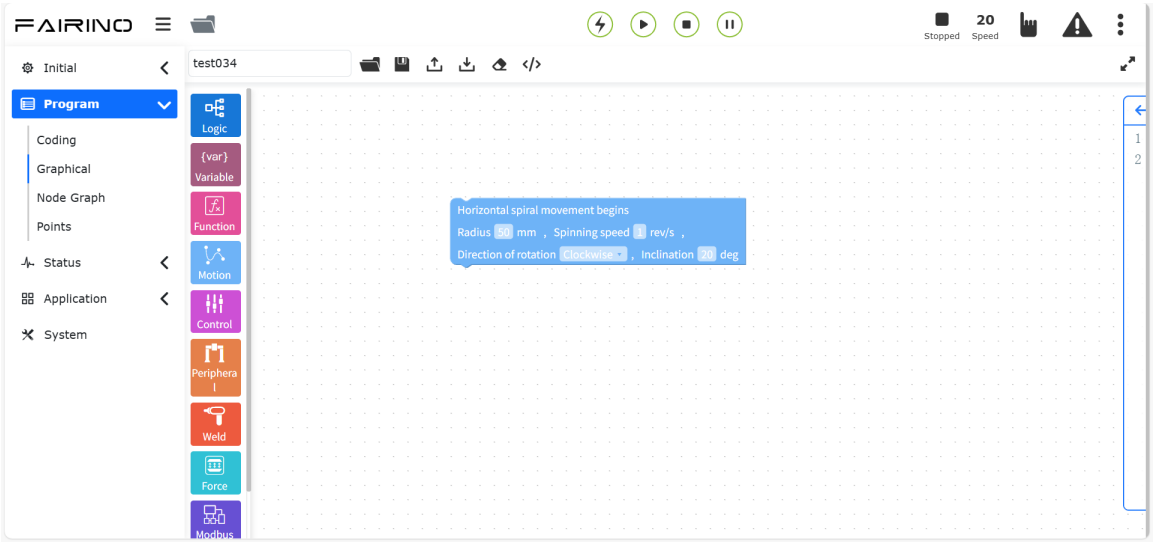


Figure 10.5-9 Horizontal spiral instruction code block

1.3.10.5.10 Spline instruction

Drag the “Spline instruction” code block to enter the graphical editing interface workspace.

This instruction is divided into three parts: spline group start, spline segment and spline group end. The spline group start is the start mark of the spline motion. The spline segment currently only contains the SPL segment in the node diagram. The spline group end is the end mark of the spline motion.

“Spline-SPTP” instruction node, parameters:

- Point name: teaching point
- Debug speed (%): 0 ~ 100

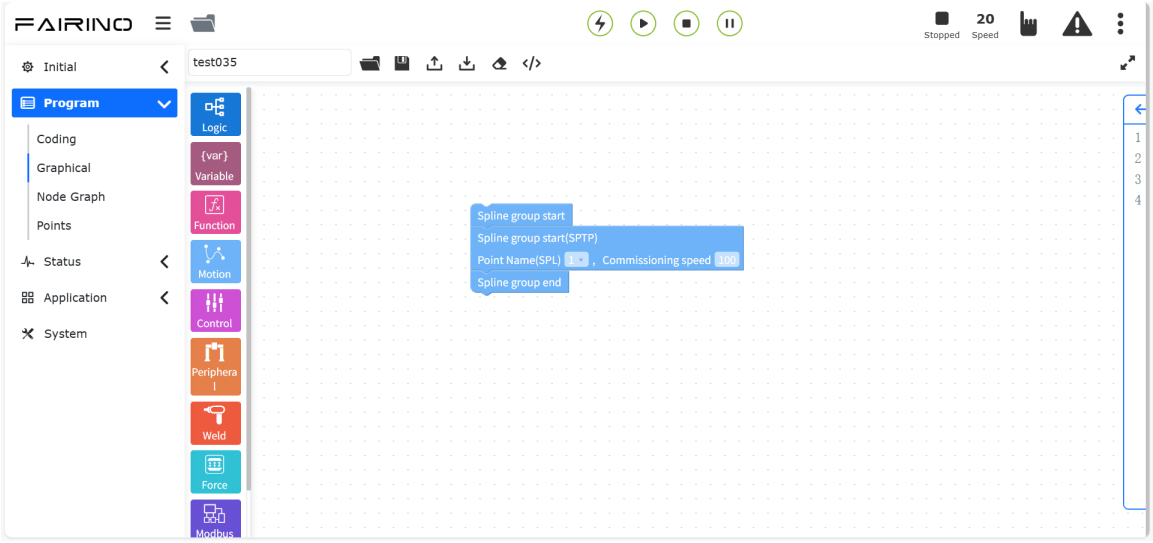


Figure 10.5-10 Spline instruction code block

1.3.10.5.11 New spline instruction

Drag the “New spline instruction” code block to enter the graphical editing interface workspace.

This instruction is a spline instruction algorithm optimization instruction, which will replace the existing spline instruction in the future. This instruction is divided into three parts: multi-point trajectory start, multi-point trajectory segment and multi-point trajectory end. Multi-point trajectory start is the starting mark of multi-point trajectory movement. Multi-point trajectory segment is to set each trajectory point. Click the icon to enter the point adding interface. Multi-point trajectory end is the end mark of multi-point trajectory movement. Here you can set the control mode and debugging speed. The control mode is divided into given control point and given path point.

“New Spline” instruction node, parameters:

- Control mode: teach point
- Global average connection time: integer type, greater than 10, default value is 2000ms

“New Spline-SPL” instruction node, parameters:

- Point name: teach point
- Debug speed (%): 0 ~ 100
- Smooth transition radius: 0 ~ 1000
- Is it the last point: No/Yes

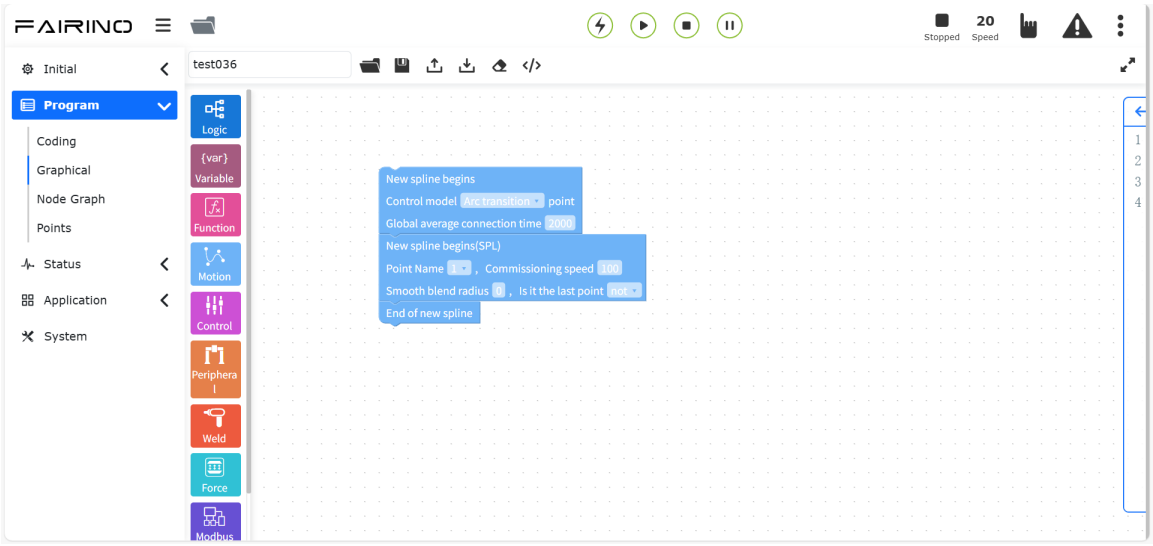


Chart 10.5-11 New Spline Instruction Code Block

1.3.10.5.12 Swing Instruction

Drag the “Swing Instruction” code block to enter the graphical editing interface workspace.

“Swing” instruction node, parameters:

- Number: 0~7

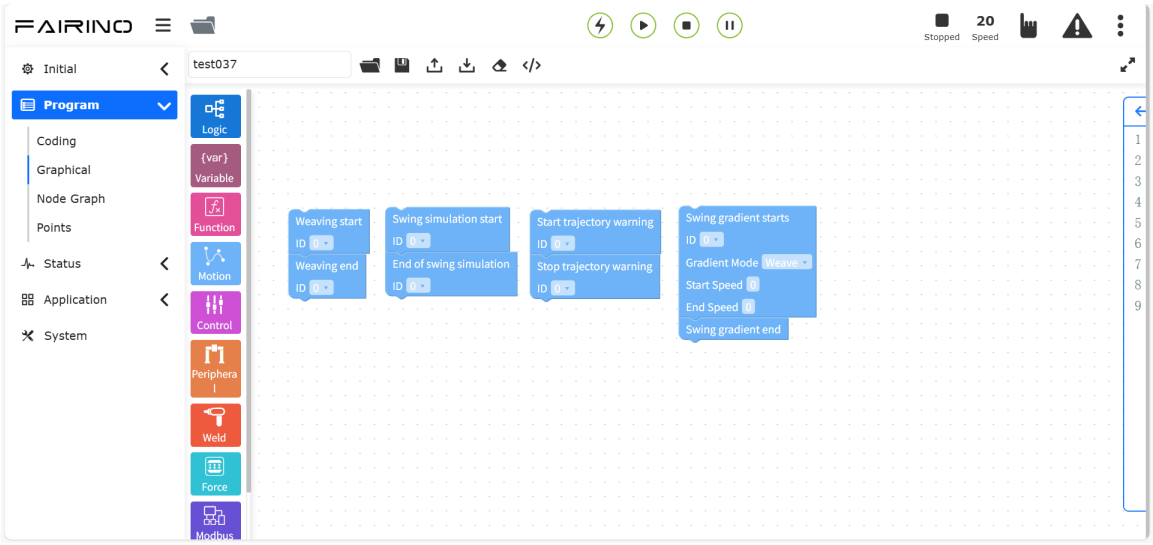


Figure 10.5-12 Swing instruction code block

1.3.10.5.13 Point offset instruction

Drag the “point offset instruction” code block to enter the graphical editing interface workspace.

This instruction is an overall offset instruction. Enter each offset. The intermediate motion instructions will be offset based on the base coordinates (or workpiece coordinates).

“Point offset” command node, parameters:

- x: offset, -300~300
- y: offset, -300~300
- z: offset, -300~300
- rx: offset, -300~300
- ry: offset, -300~300
- rz: offset, -300~300

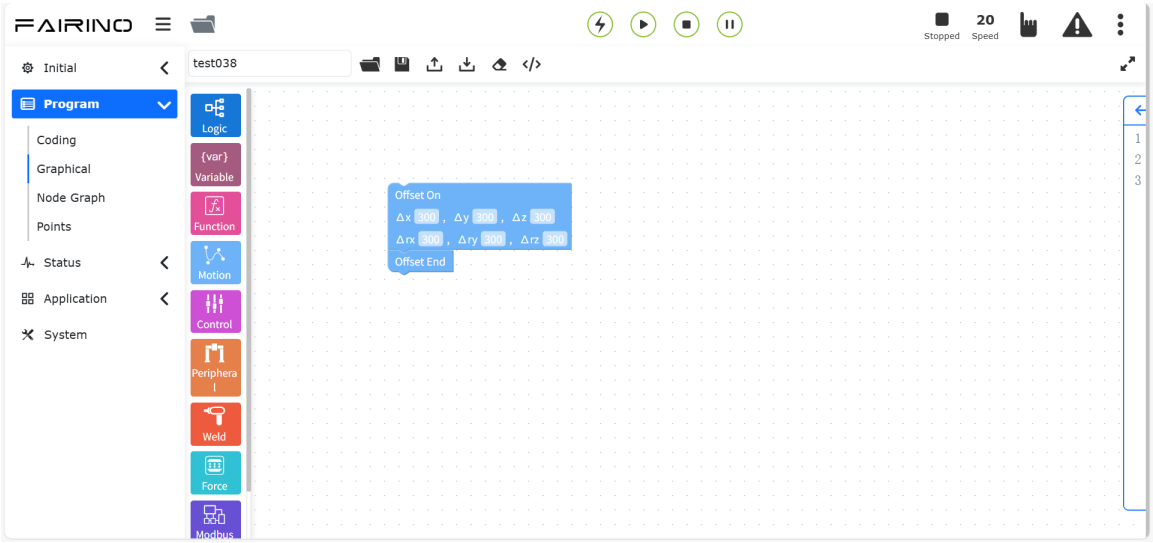


Figure 10.5-13 Point offset command code block

1.3.10.5.14 Servo command

Drag the “Servo command” code block to enter the graphical editing interface workspace.

Servo control (Cartesian space motion) instructions, which can control the robot motion through absolute posture control or based on the current posture offset.

“Servo” command node, parameters:

- Movement mode: absolute position/base coordinate offset/tool coordinate offset
- x: offset, -300~300
- y: offset, -300~300
- z: offset, -300~300
- rx: offset, -300~300
- ry: offset, -300~300

- rz: offset, -300~300
- Proportional coefficient x: 0~1
- Proportional coefficient y: 0~1
- Proportional coefficient z: 0~1
- Proportional coefficient rx: 0~1
- Proportional coefficient ry: 0~1
- Proportional coefficient rz: 0~1
- Acceleration (%): 0~100
- Speed (%): 0~100
- Command cycle (s): 0.001~0.016
- Filter time (s): 0~1
- Scale up: 0~100

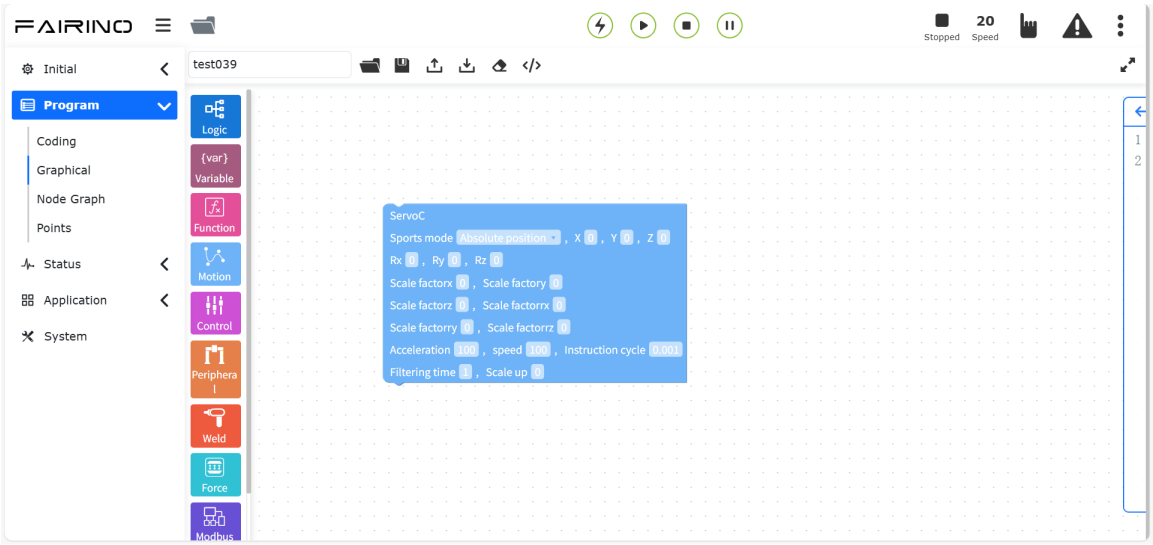


Chart 10.5-14 Servo command code block

1.3.10.5.15 Trajectory command

Drag the “Trajectory command” code block to enter the graphical editing interface workspace.

In this command, the user first needs to have a recorded trajectory.

“Trajectory” command node, parameters:

- Select trajectory file: recorded trajectory
- Debug speed (%): 0 ~ 100, default value is 25

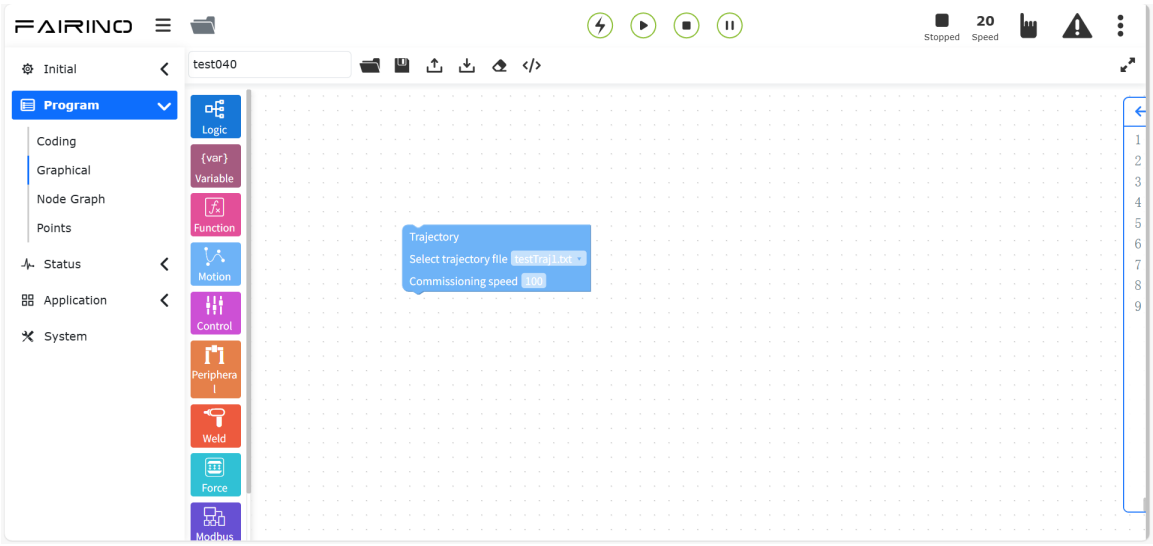


Chart 10.5-15 Trajectory command code block

1.3.10.5.16 TrajectoryJ command

Drag the "TrajectoryJ command" code block to enter the graphical editing interface workspace.

In this command, the user first needs to have a recorded trajectory, and the trajectory file can be pre-imported in the teaching program interface. The trajectory command and trajectoryJ command are suitable for the general interface of the camera directly giving the trajectory. When there is a discrete trajectory point file in a fixed format, it can be imported into the system to make the robot move according to the trajectory of the imported file.

"Track J" command node, parameters:

- Select track file: recorded track
- Debug speed (%): 0 ~ 100, default value is 25
- Track mode: path point/control point

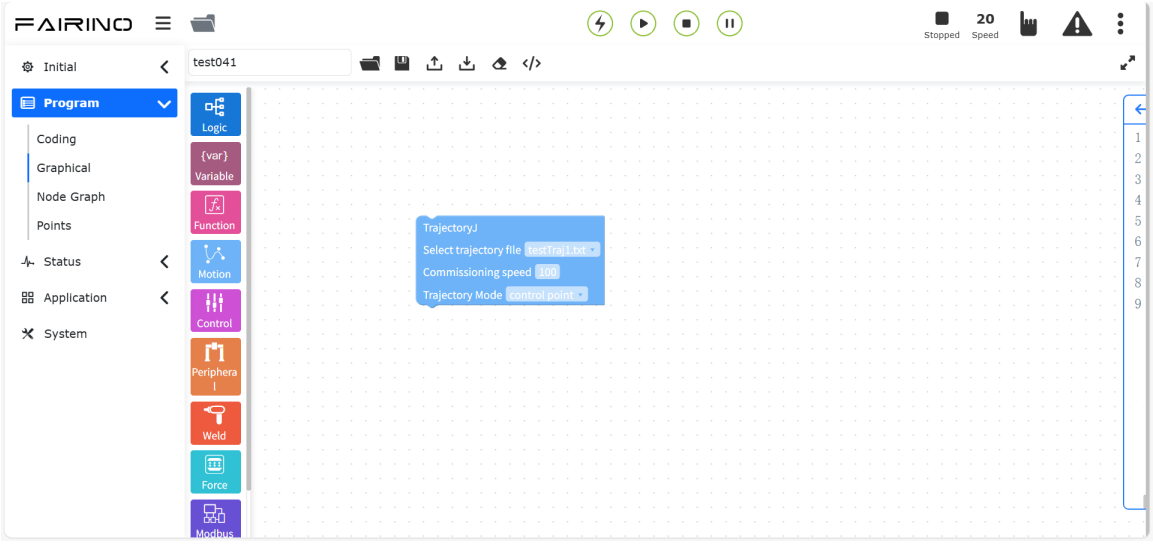


Chart 10.5-16 Track J command code block

1.3.10.5.17 Track reproduction command

Drag the “Track reproduction command” code block to enter the graphical editing interface workspace.

In this command, the user first needs to have a recorded track.

When programming, first use the point-to-point command to reach the corresponding track starting point, then select the track in the track reproduction command, select the smooth track, and set the debugging speed. The track loading command is mainly used to pre-read the track file and extract it into a track command for better application in conveyor belt tracking scenarios.

“Trajectory Reproduction” command node, parameters:

- Trajectory name: recorded trajectory
- Smooth trajectory: No/Yes
- Debug speed (%): 0 ~ 100, default value is 25

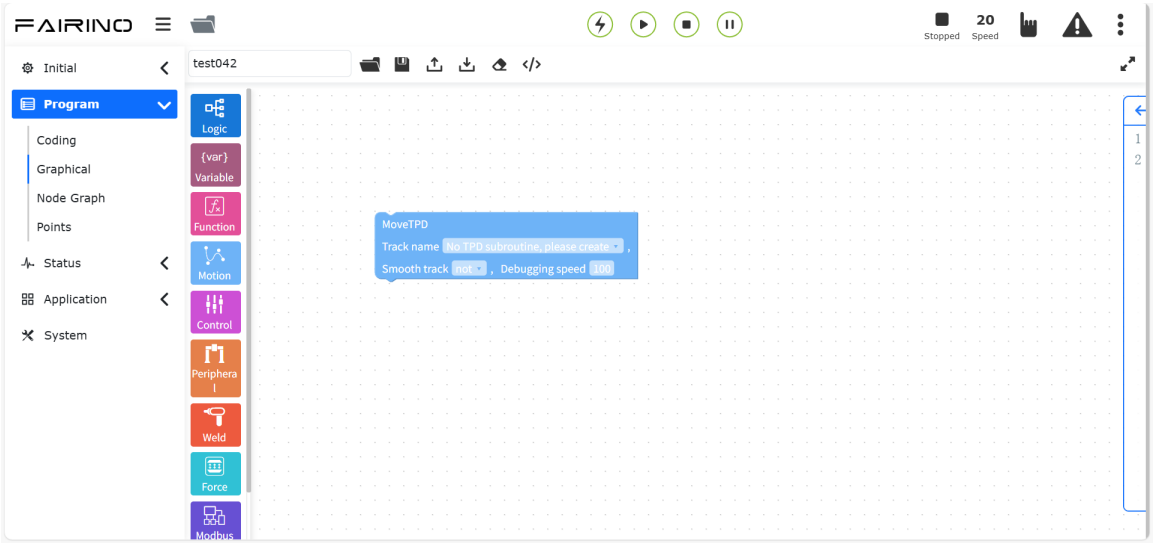


Chart 10.5-17 Trajectory Reproduction Command Code Block

1.3.10.5.18 DMP Command

Drag the “DMP Command” code block to enter the graphical editing interface workspace.

DMP is a method of trajectory imitation learning, which requires planning the reference trajectory in advance. In the command editing interface, select the teaching point as the new starting point, click “Add” and “Apply” to save the command. The specific path of DMP is a new trajectory that imitates the reference trajectory with a new starting point.

“DMP” command node, parameters:

- Point name: teaching point
- Debug speed (%): 0 ~ 100, default value is 100

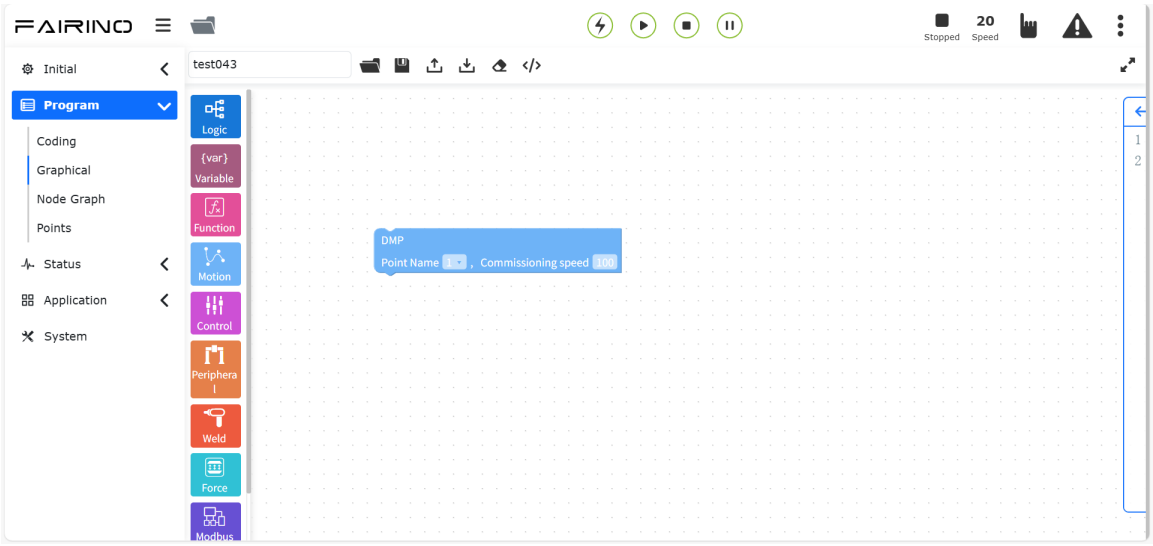


Chart 10.5-18 DMP command code block

1.3.10.5.19 Tool conversion instruction

Drag the “Tool conversion instruction” code block to enter the graphical editing interface workspace.

Select the tool coordinate system to be automatically converted, click “Add” and “Apply” to save the instruction. The tool coordinate system are automatically converted.

“Tool conversion” instruction node, parameters:

- Tool coordinate system: Tool coordinate system table

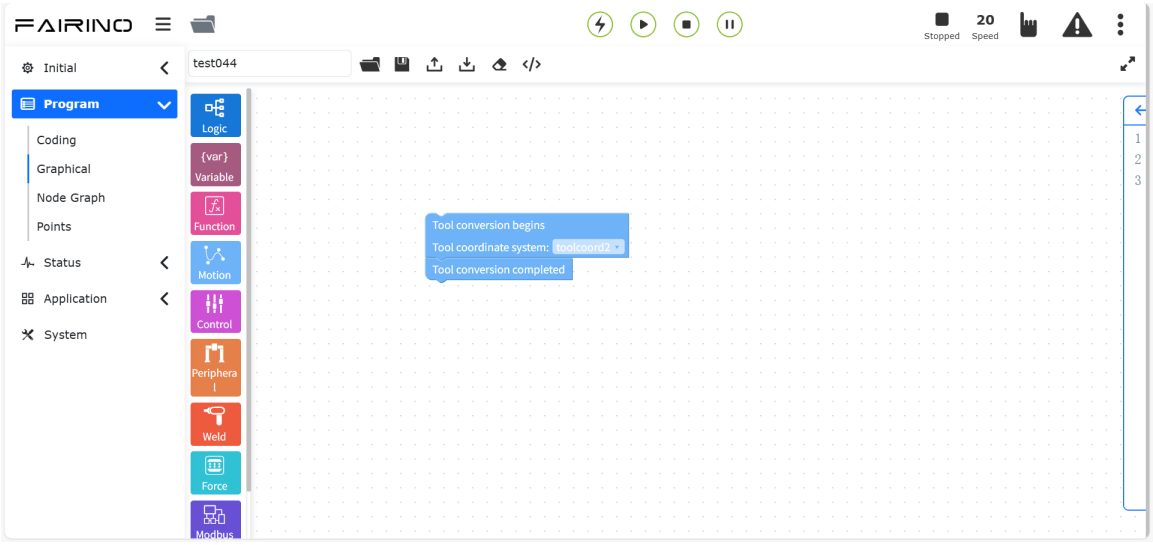


Figure 10.5-19 Tool conversion instruction code block

1.3.10.5.20 Workpiece conversion instruction

Drag the “Workpiece conversion instruction” code block to enter the graphical editing interface workspace.

Select the workpiece coordinate system to be automatically converted, the points in the workpiece coordinate system are automatically converted.

“Workpiece conversion” instruction node, parameters:

- Workpiece coordinate system: Workpiece coordinate system table

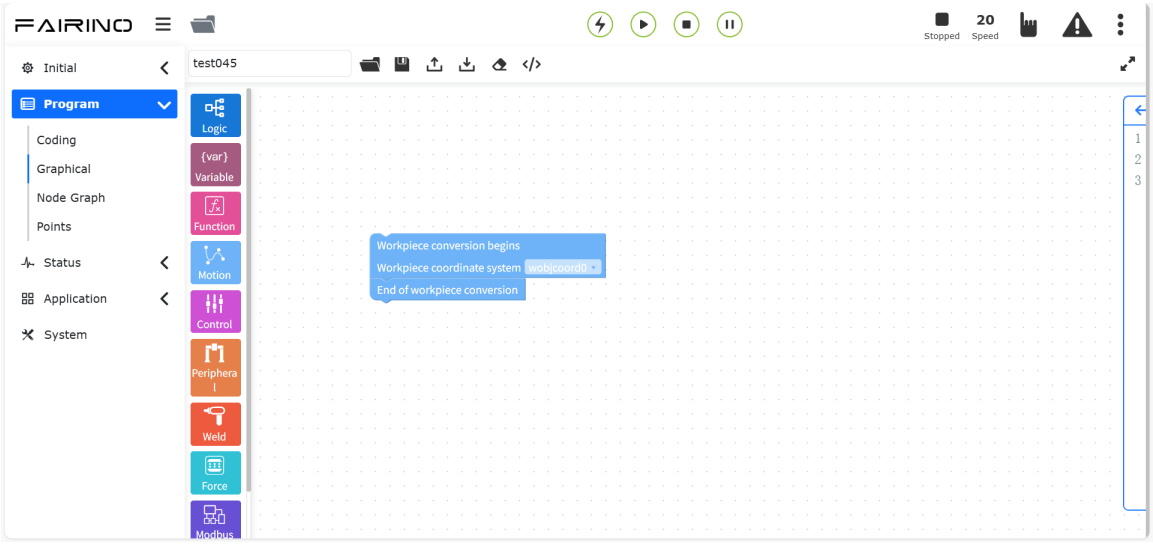


Chart 10.5-20 Workpiece conversion instruction code block

1.3.10.6 Control class graphical programming commands

Control class graphical programming commands include Wait, IO and other control commands.

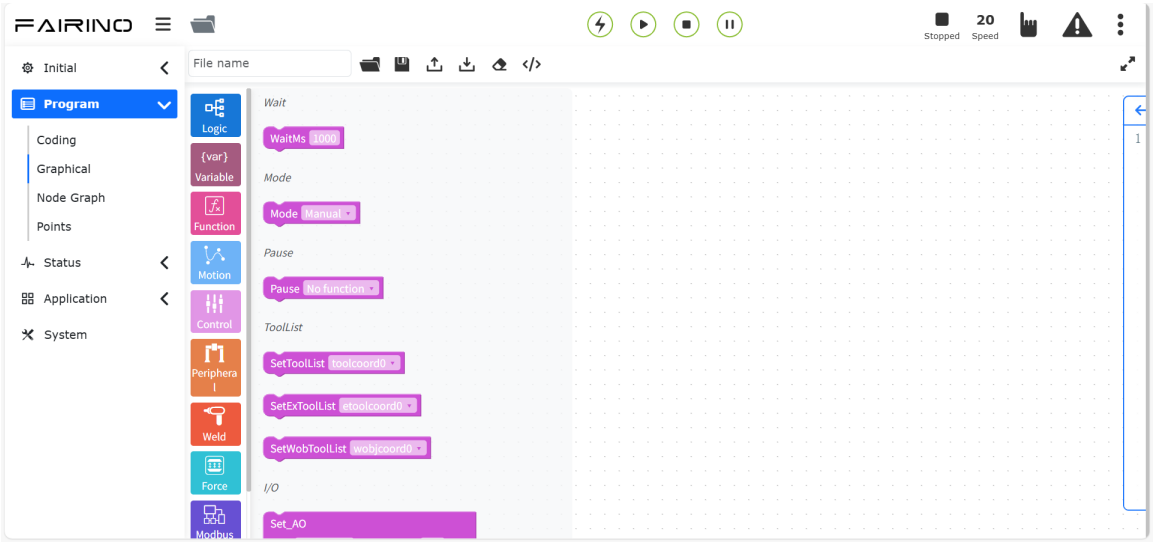


Figure 10.6 Control Class Graphical Programming Commands

1.3.10.6.1 Wait Command

Drag the “Wait Command” code block to enter the graphical editing interface workspace.

This command is a delay command, which is divided into four parts: “WaitMs”, “WaitDI”, “WaitMultiDI” and “WaitAI”.

1. “Wait” command node, parameters:

- Wait time (ms): The delay waiting time unit is milliseconds, enter the number of milliseconds to wait

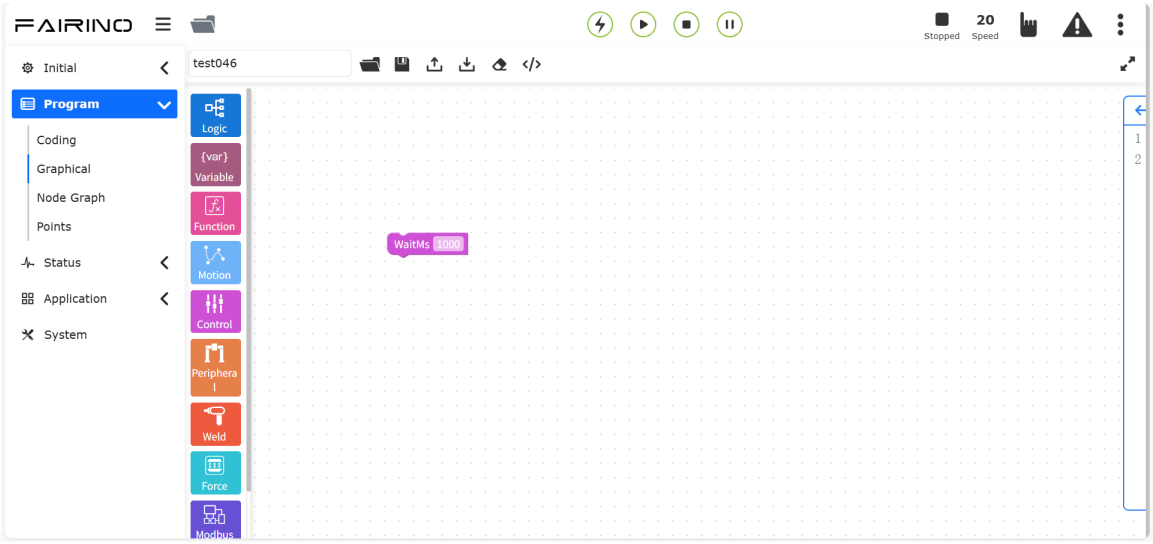


Figure 10.6-1 Wait command code block

2. “Wait DI” command node, parameters:

- DI port number: Ctrl-DI0 ~ Ctrl-CI7(WaitDI,[0~15]), End-DI0 ~ End-DI1(WaitToolDI,[0~1])
- Status: false/true
- Maximum time (ms): 0 ~ 10000
- Wait timeout processing: stop error/continue execution/keep waiting

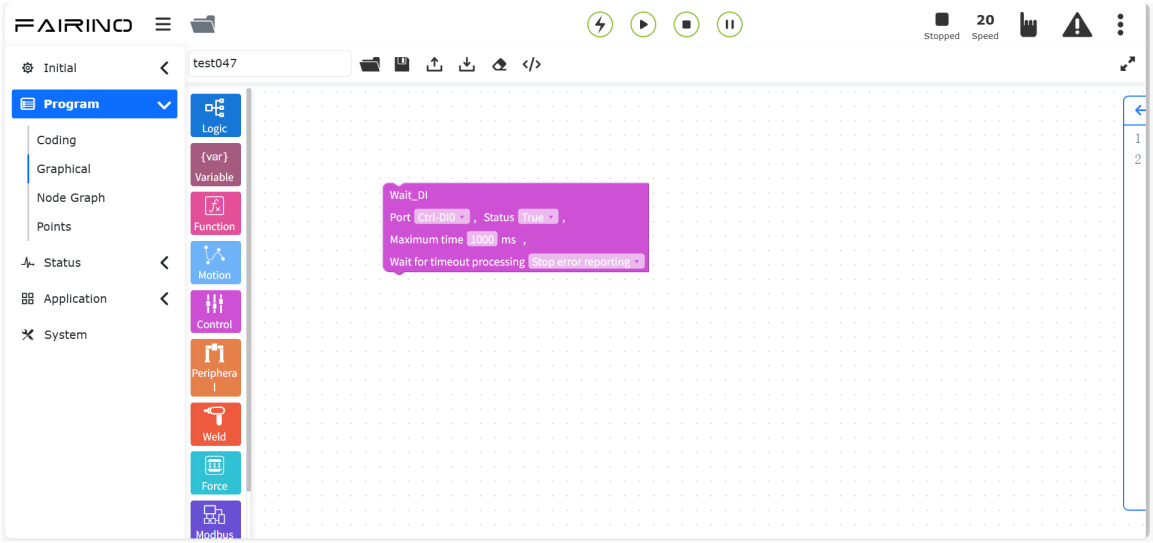


Figure 10.6-2 Wait for DI instruction code block

3. “Wait for multiple DI” instruction node, parameters:

- Condition: and/or
- Condition selection: select the port number of the bit state to be turned on, separated by commas, such as DI0, DI1
- True value corresponding port: select the port number of the true value, separated by commas, such as DI0, DI1
- Maximum time (ms): 0 ~ 10000, maximum waiting time
- Waiting timeout processing: stop error/continue execution/wait forever

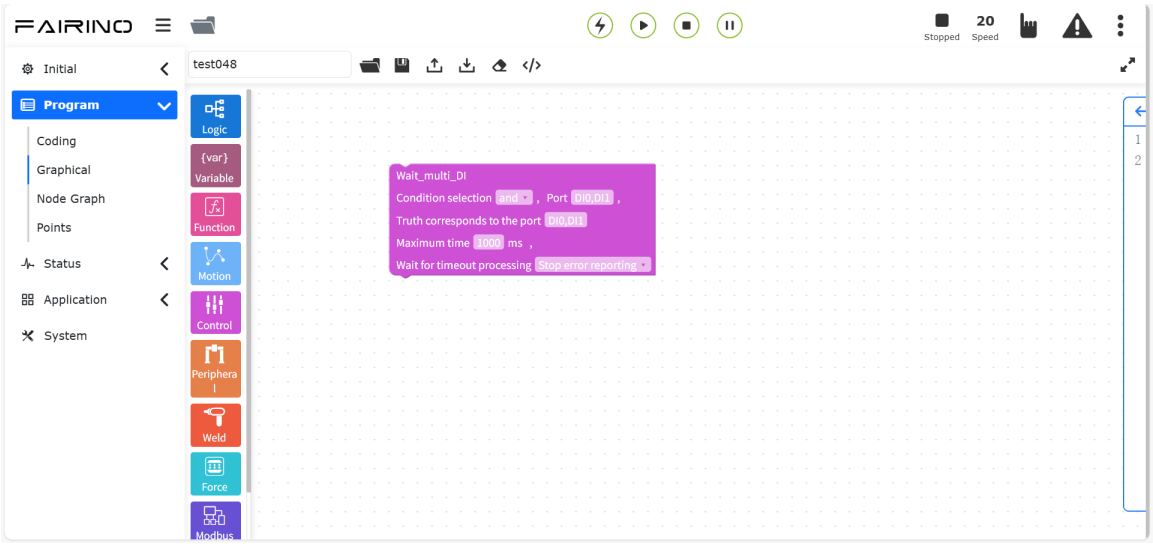


Figure 10.6-3 Wait for multiple DI instruction code block

4. “Wait for AI” instruction node, parameters:

- Condition: and/or
- AI port number: Ctrl-AI0 ~ Ctrl-AI1(WaitAI,[0~1]), End-AI0(WaitToolAI,[0])
- Condition: greater than/less than
- Value (%): 1 ~ 100
- Maximum time (ms): 0 ~ 10000
- Waiting timeout processing: stop reporting errors/continue execution/keep waiting. When waiting for timeout processing, the maximum time defaults to 0

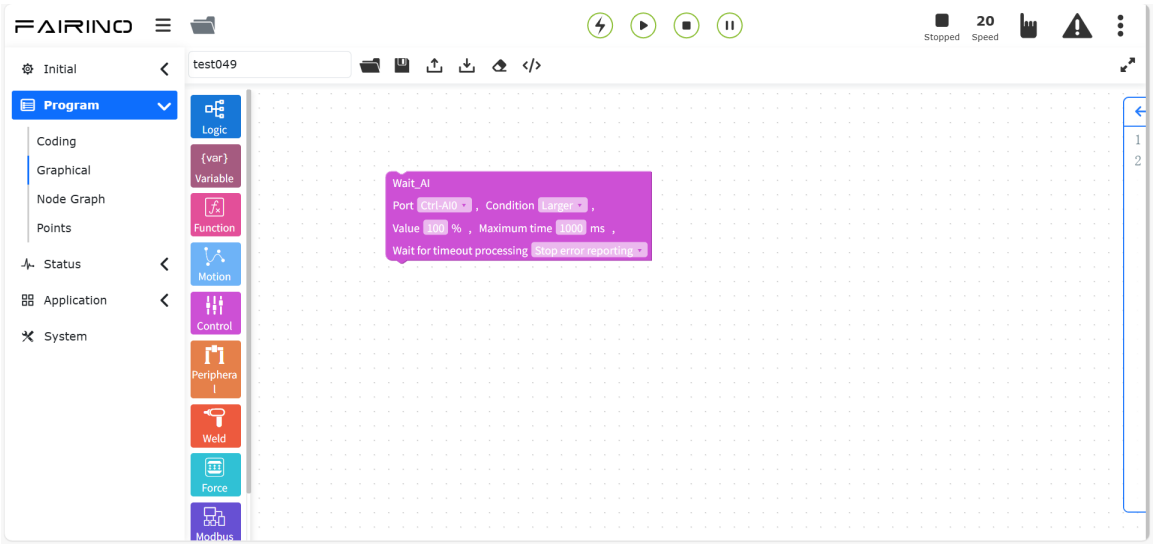


Figure 10.6-4 Waiting for AI command code block

1.3.10.6.2 Mode switching command

Drag the “mode switching command” code block to enter the graphical editing interface workspace.

This command can switch the robot to manual mode. It is usually added at the end of a program so that the user can automatically switch the robot to manual mode and drag the robot after the program ends.

“Mode switch” instruction node, parameters:

- Mode switch: manual mode

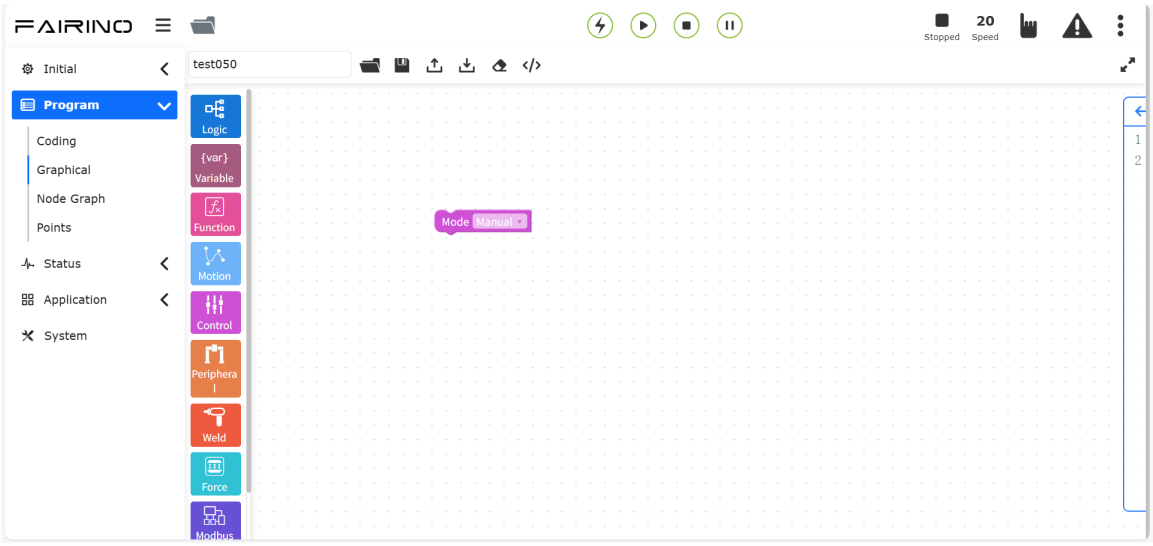


Figure 10.6-5 Mode switch instruction code block

1.3.10.6.3 Pause instruction

Drag the “Pause instruction” code block to enter the graphical editing interface workspace.

This instruction is a pause instruction. Insert this instruction in the program. When the program executes this instruction, the robot will be in a paused state. If you want to continue running, click the “Pause/Resume” button in the control area.

“Pause” command node, parameters:

- Pause type: no function, cylinder not in place, etc.

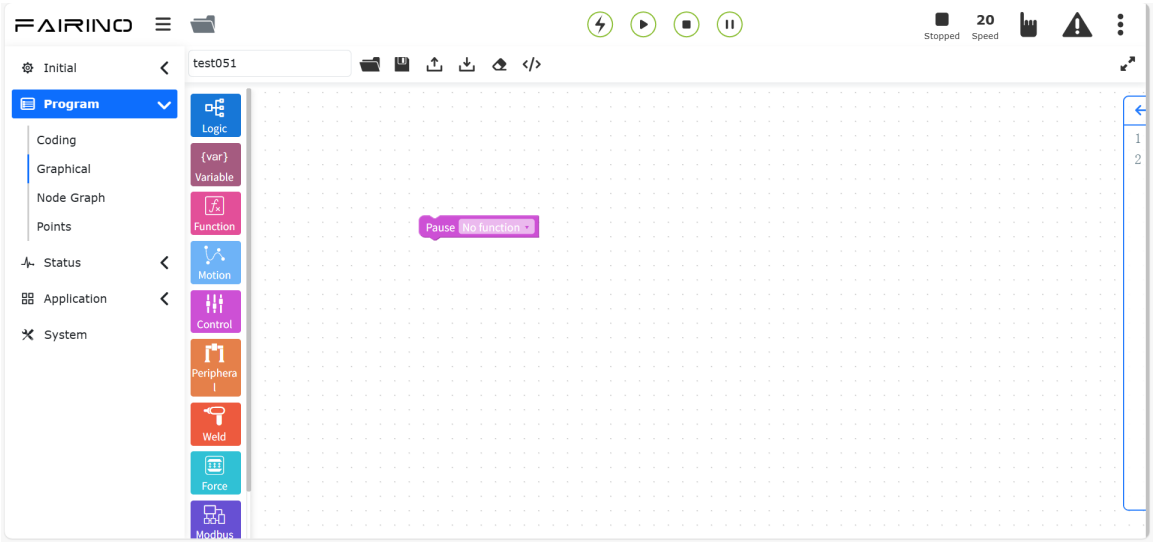


Figure 10.6-6 Pause command code block

1.3.10.6.4 Coordinate system command

Drag the “Set tool coordinate system”/”Set workpiece coordinate system” code block to enter the graphical editing interface workspace.

1. “Set tool coordinate system” command node, parameters:

- Tool coordinate system name: toolcoord1 ~ toolcoord19(SetToolList,[0~19]), etoolcoord0 ~ etoolcoord14(SetExToolList, [0~14])

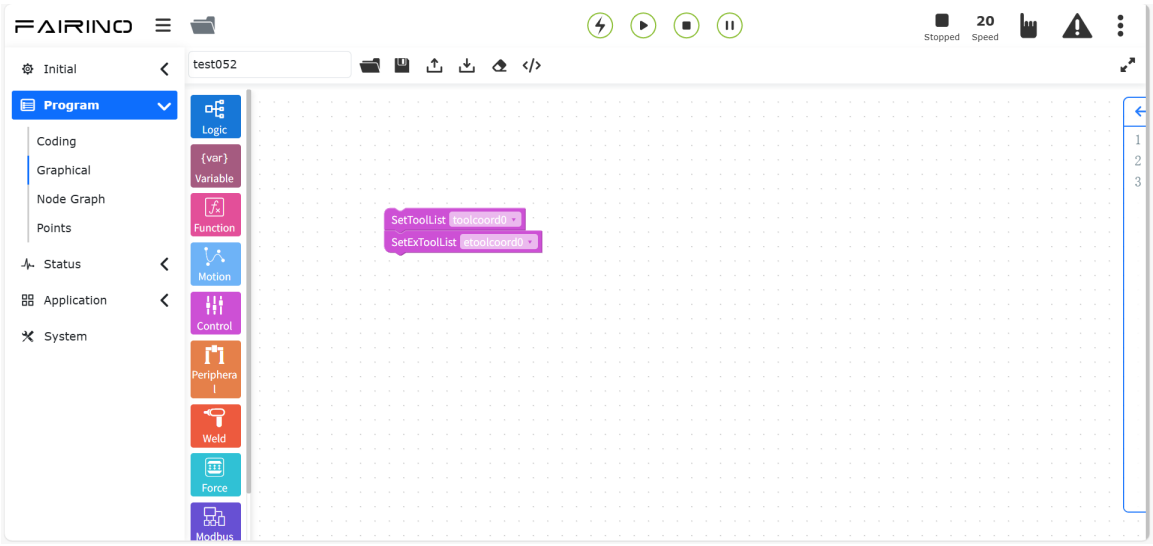


Figure 10.6-7 Set tool coordinate system command code block

2. “Set workpiece coordinate system” command node, parameters:

- Workpiece coordinate system name: wobjcoord1 ~ wobjcoord14

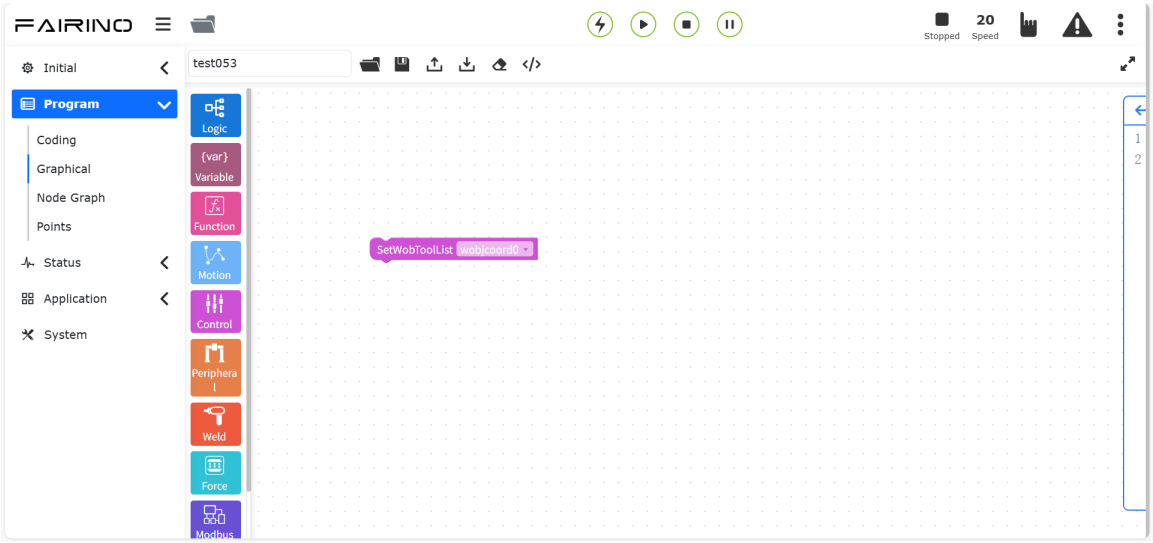


Figure 10.6-8 Set workpiece coordinate system instruction code block

1.3.10.6.5 Simulation AI instruction

Drag the “Set AO”/”Get AI” code block to enter the graphical editing interface workspace.

In this instruction, there are two functions: setting analog output (SetAO/SPLCSetAO) and getting analog input (GetAI/SPLCGetAI).

1. “Set AO” instruction node, parameters:

- Port: Ctrl-AO0 ~ Ctrl-AO1 (blocking: SetAO, non-blocking: SPLCSetAO, [0~1]), End-AO0 (blocking: SetToolAO, non-blocking: SPLCSetToolAO, [0])
- Value (%): 0 ~ 100

- Whether to block: blocking/non-blocking
- Whether to apply thread: no/yes

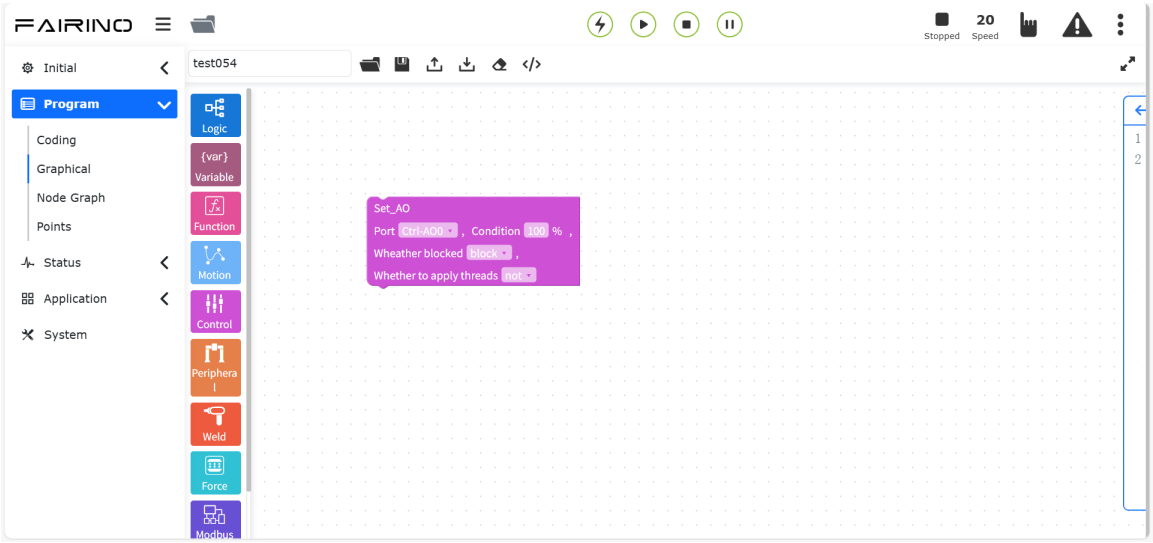


Figure 10.6-9 Set AO instruction code block

2. “Get AI” instruction node, parameters:

- Port: Ctrl-AI0 ~ Ctrl-DI1 (blocking: GetAI, non-blocking: SPLCGetAI, [0~1]), End-AI0 (blocking: GetToolAI, non-blocking: SPLCGetToolAI, [0])
- Condition: greater than/less than
- Value (%): 0 ~ 100
- Maximum time (ms): 0 ~ 10000
- Blocking: Blocking/Non-blocking
- Apply thread: No/Yes

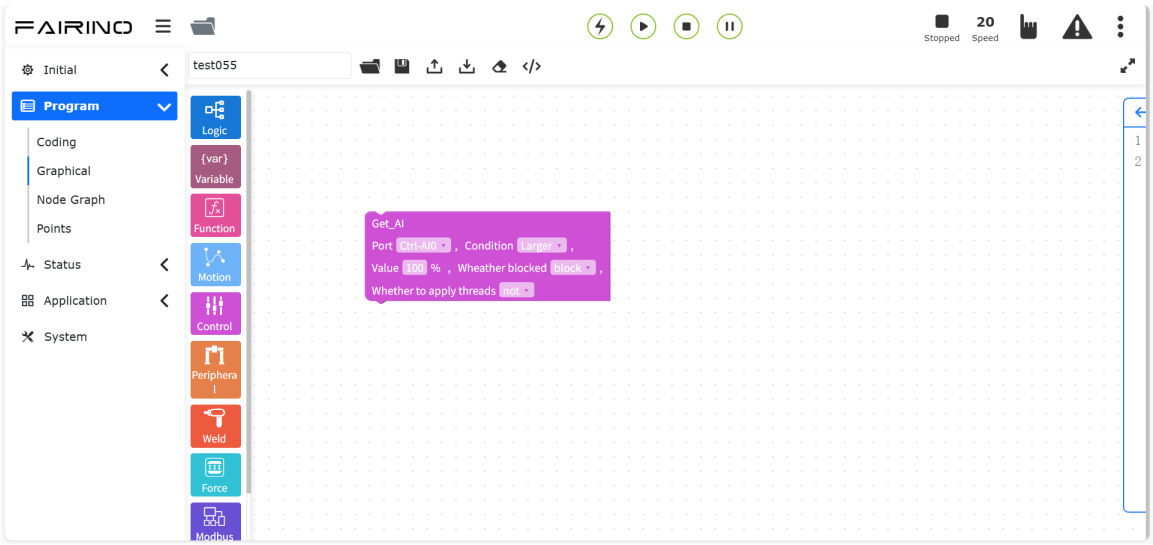


Figure 10.6-10 Get AI instruction code block

1.3.10.6.6 Digital IO Instructions

Drag the "Set DO"/"Get DI" code block to enter the graphical editing interface workspace.

The instructions are IO instructions, which are divided into two parts: Set IO (SetDO/SPLCSetDO) and Get IO (GetDI/SPLCGetDI).

1. "Set DO" instruction node, parameters:

- Port: Ctrl-DO0 ~ Ctrl-CO7 (blocking: SetDO, non-blocking: SPLCSetDO, [0~15]), End-DO0 ~ End-DO1 (blocking: SetToolDO, non-blocking: SPLCSetToolDO, [0~1])
- Status: false/true
- Blocking: blocking/non-blocking
- Smooth trajectory: Break/Serious
- Apply thread: No/Yes

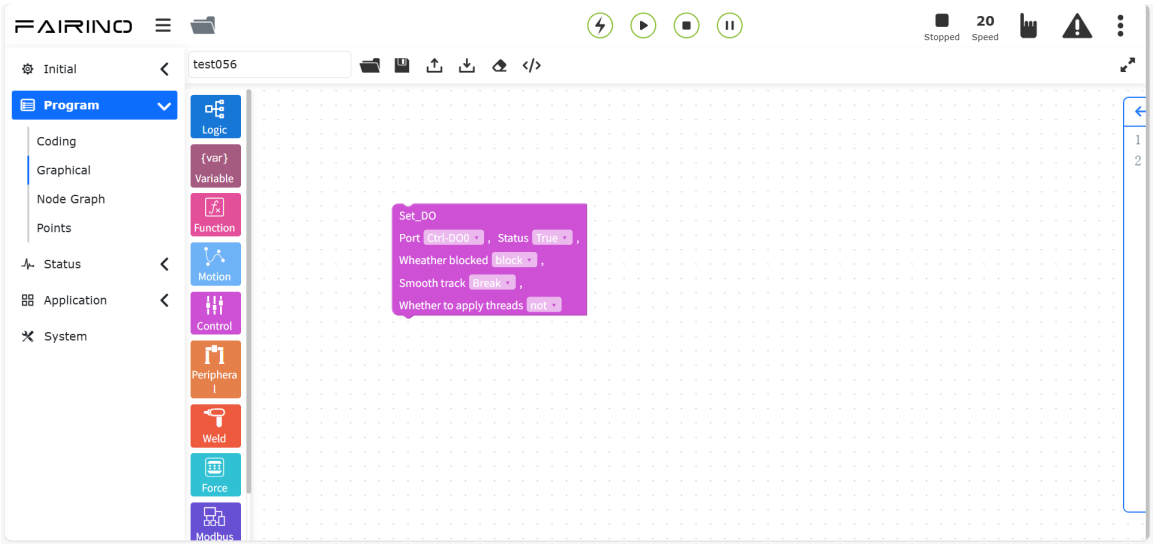


Figure 10.6-11 Set DO instruction code block

2. "Get DI" instruction node, parameters:

- Port: Ctrl-DI0 ~ Ctrl-CI7 (blocking: GetDI, non-blocking: SPLCGetDI, [0~15]), End-DI0 ~ End-DI1(Blocking: GetToolDI, Non-blocking: SPLCGetToolDI, [0~1])
- Blocking: Blocking/Non-blocking
- Status: false/true
- Maximum waiting time (ms): 0 ~ 10000
- Apply thread: No/Yes

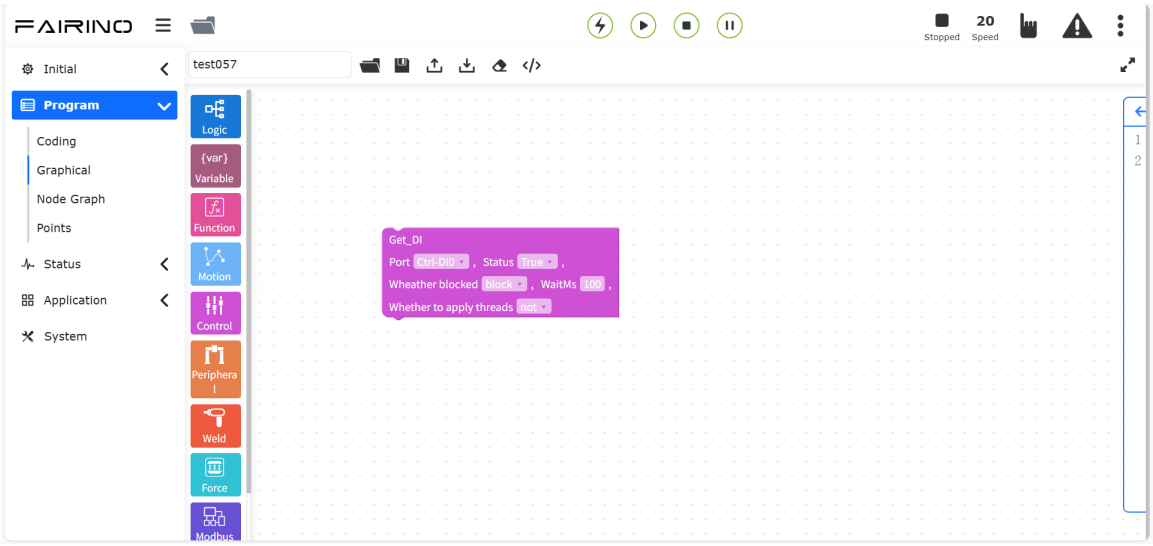


Chart 10.6-12 Get DI instruction code block

1.3.10.6.7 Motion DO instruction

Drag the “Motion DO instruction” code block to enter the graphical editing interface work area.

This instruction implements the function of continuously outputting DO signals according to the set interval during linear motion.

1. “Motion DO continuous output” instruction node, parameters:

- Port: Ctrl-DO0 ~ Ctrl-DO0(MoveDOStart,[0~15]), End-DO1(MoveDOStart,[0~1])
- Setting interval (mm): 0 ~ 500
- Output pulse duty cycle (%): 0 ~ 99

2. “Motion DO single output” instruction node, parameters:

- Port: Ctrl-DO0 ~ Ctrl-DO0(MoveDOOnceStart,[0~15]), End-DO1(MoveDOOnceStart,[0~1])
- Output mode: uniform speed output/free configuration
- Set time (ms): 0 ~ 1000 (the uniform speed output mode defaults to -1)
- Reset time (ms): 0 ~ 1000 (the uniform speed output mode defaults to -1)

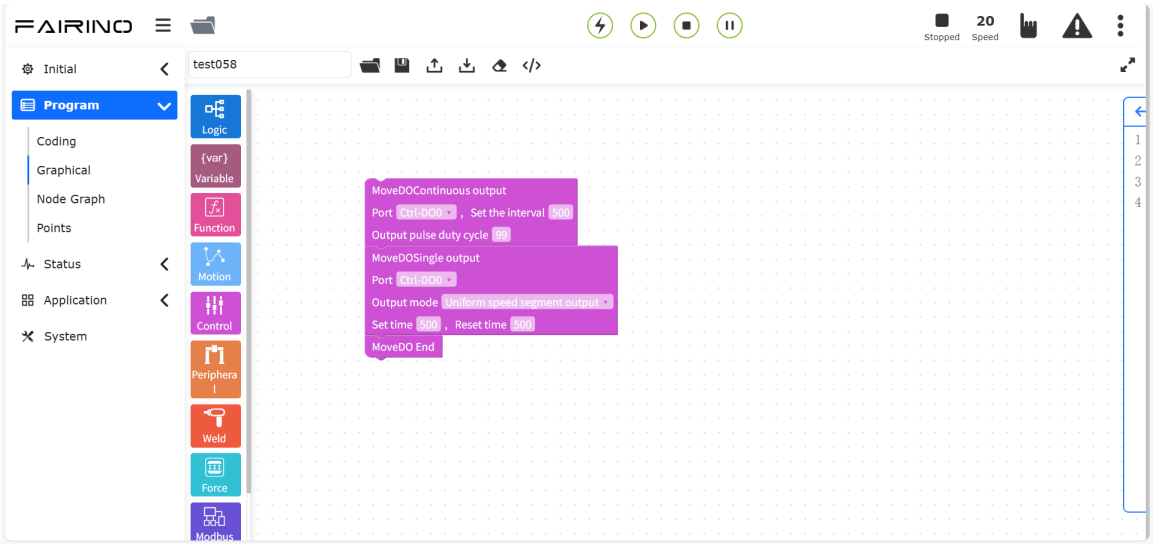


Figure 10.6-13 “Motion DO single/continuous output” instruction code block

1.3.10.6.8 Motion AO instruction

Drag the “Motion AO instruction” code block to enter the graphical editing interface workspace.

When this instruction is used with the motion instruction, it can output the AO signal in proportion to the real-time TCP speed during the motion process.

“Move AO” command node, parameters:

- Control box AO number: Ctrl-AO0 ~ Ctrl-AO1 (MoveAOStart, [0~1]), End-AO0 (MoveToolAOStart, 0)
- Maximum TCP speed: 0 ~ 100
- Maximum TCP speed AO percentage: 0 ~ 100
- Dead zone compensation value AO percentage: 0 ~ 100

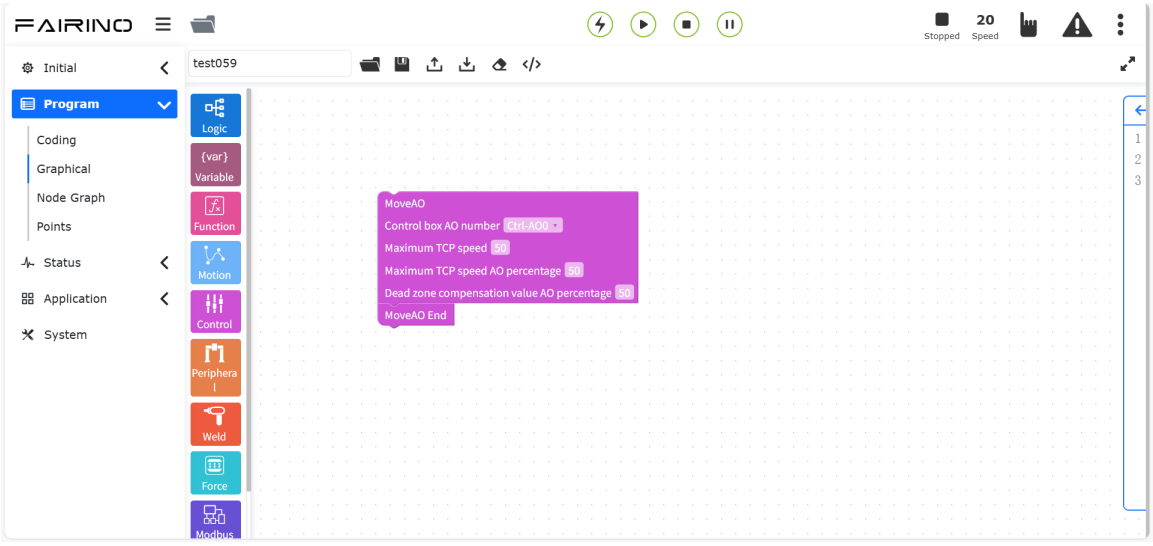


Figure 10.6-14 “Move AO” command code block

1.3.10.6.9 Collision level command

Drag the “Collision level command” code block to enter the graphical editing interface workspace.

This command sets the collision level. Through this command, the collision level of each axis can be adjusted in real time during program operation, and the application scenario can be deployed more flexibly.

“Collision level” command node, parameters:

- Standard level: standard level/custom percentage
- joint1-joint6(N): 0 ~ 100, collision threshold, array type

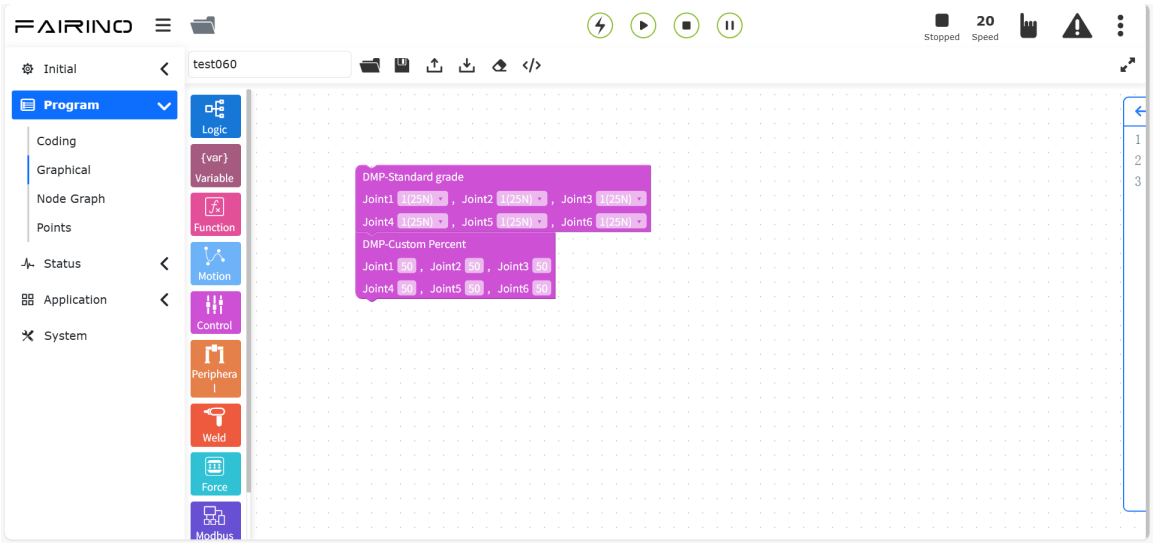


Figure 10.6-15 Collision level command code block

1.3.10.6.10 Acceleration command

Drag the “acceleration command” code block to enter the graphical editing interface workspace.

The “acceleration” command is to realize the function of setting the robot acceleration separately. By adjusting the motion command acceleration scaling factor, the acceleration and deceleration time can be increased or decreased, and the robot action beat time can be adjusted.

“Acceleration” command node, parameters:

- Acceleration percentage (%): 0 ~ 100

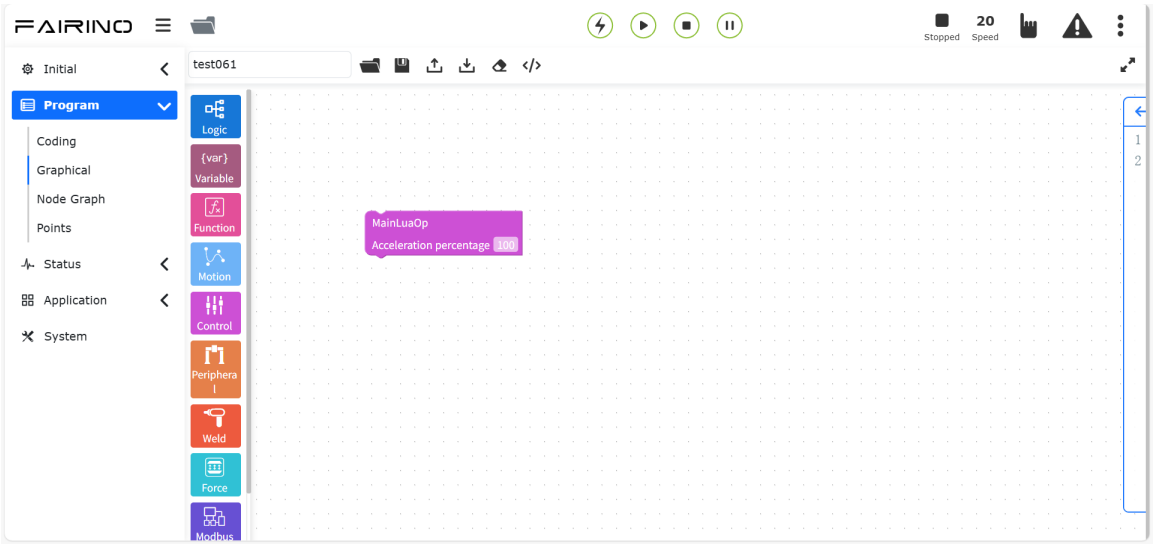


Figure 10.6-16 Acceleration command code block

1.3.10.7 Peripheral graphical programming commands

Peripheral graphical programming commands include peripheral commands such as grippers, spray guns, and extended axes.

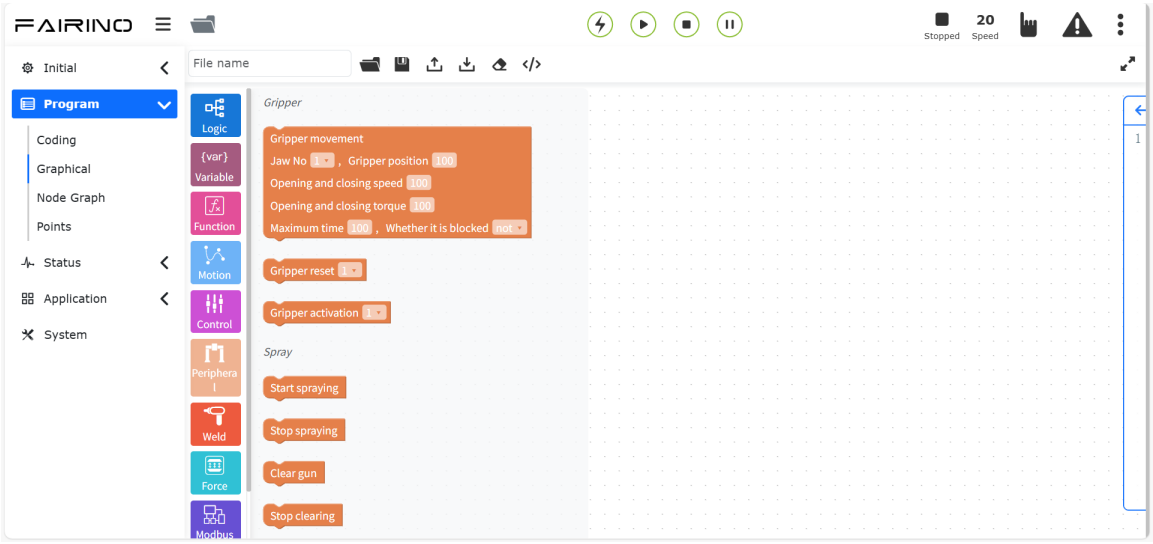


Figure 10.7 Peripheral graphical programming commands

1.3.10.7.1 Gripper commands

Drag the “gripper movement”, “gripper activation” and “gripper reset” code blocks to enter the graphical editing interface workspace.

In the command, the number of the configured and activated gripper is displayed, the settings of the gripper opening and closing, opening and closing speed, and opening and closing torque are displayed in percentages, and the blocking function option is displayed. If blocking is selected, the gripper movement needs to wait for the previous motion instruction to be executed before execution. If non-blocking is selected, the gripper movement is parallel to the previous motion instruction.

1. “Gripper Movement” node, parameters:

- Gripper number: the number of the activated gripper
- Gripper position: 0~100
- Opening and closing speed: 0~100
- Opening and closing torque: 0~100
- Maximum time (ms): 0~30000
- Blocking: false/true

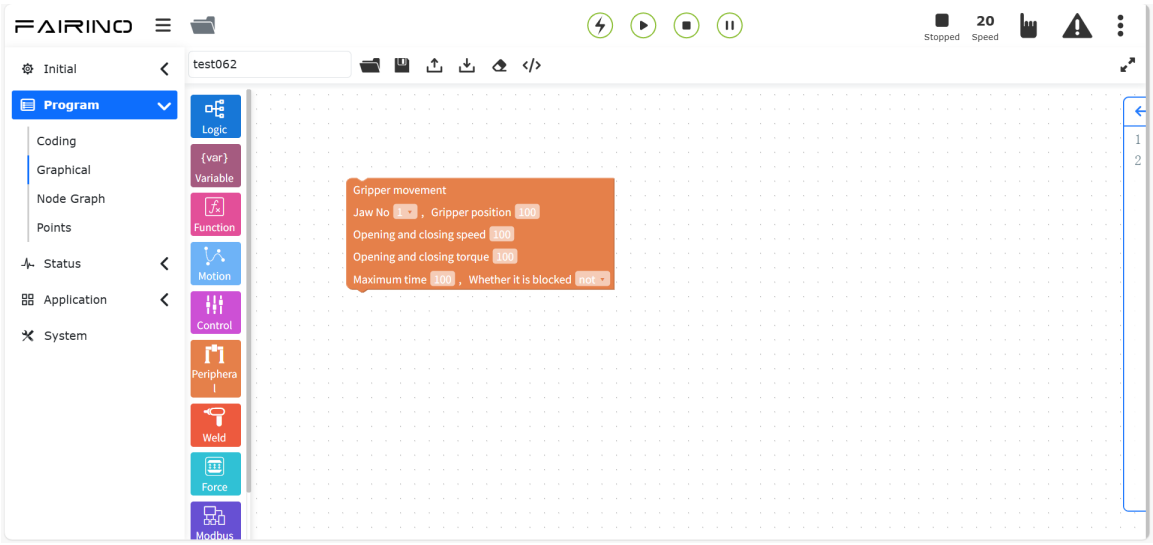


Figure 10.7-1 Gripper Movement Instruction Code Block

Gripper reset instruction, display the configured gripper number, and you can add the gripper reset instruction to the program.

2. “Gripper reset” node, parameters:

- Gripper number: the number of the activated gripper

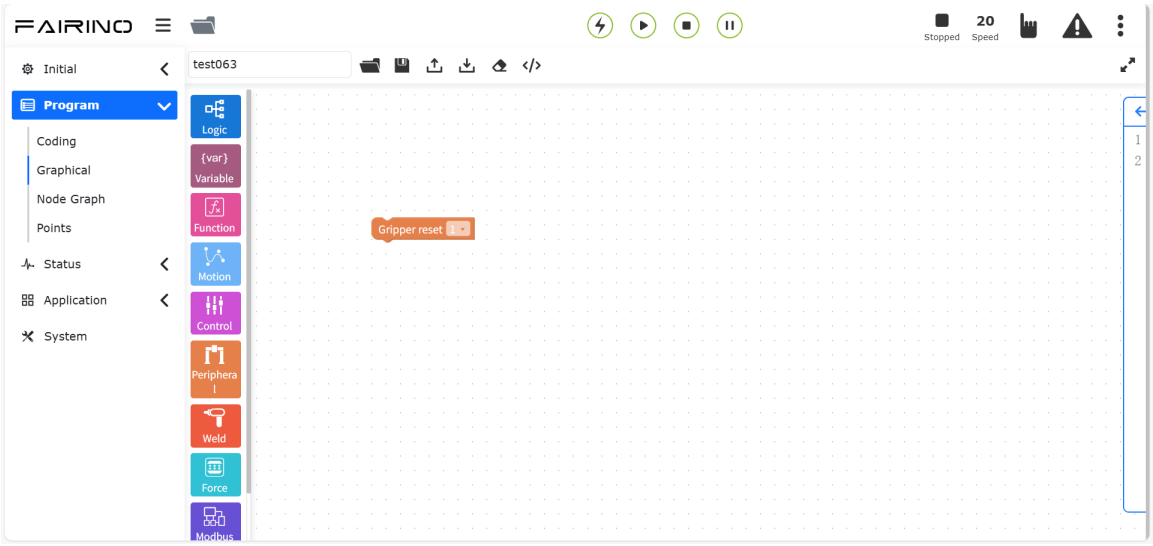


Figure 10.7-2 Gripper reset instruction code block

Gripper activation instruction, display the configured gripper number, you can add the gripper activation instruction to the program.

3. “Gripper activation” node, parameters:

- Gripper number: the number of the activated gripper

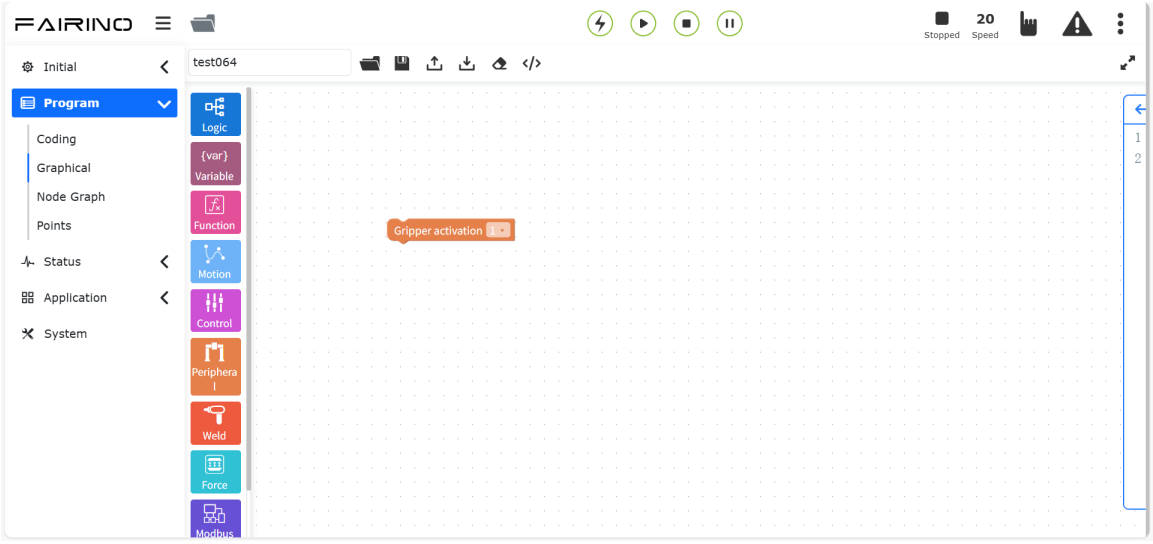


Figure 10.7-3 Gripper activation instruction code block

1.3.10.7.2 Spray gun instruction

Drag the “Spray gun instruction” code block to enter the graphical editing interface workspace.

This command is related to spraying, controlling the spray gun to “start spraying”, “stop spraying”, “start cleaning gun” and “stop cleaning gun”. When editing the relevant nodes of this program, you need to confirm that the spray gun peripherals have been configured, otherwise it cannot be saved. See the robot peripherals section for details.

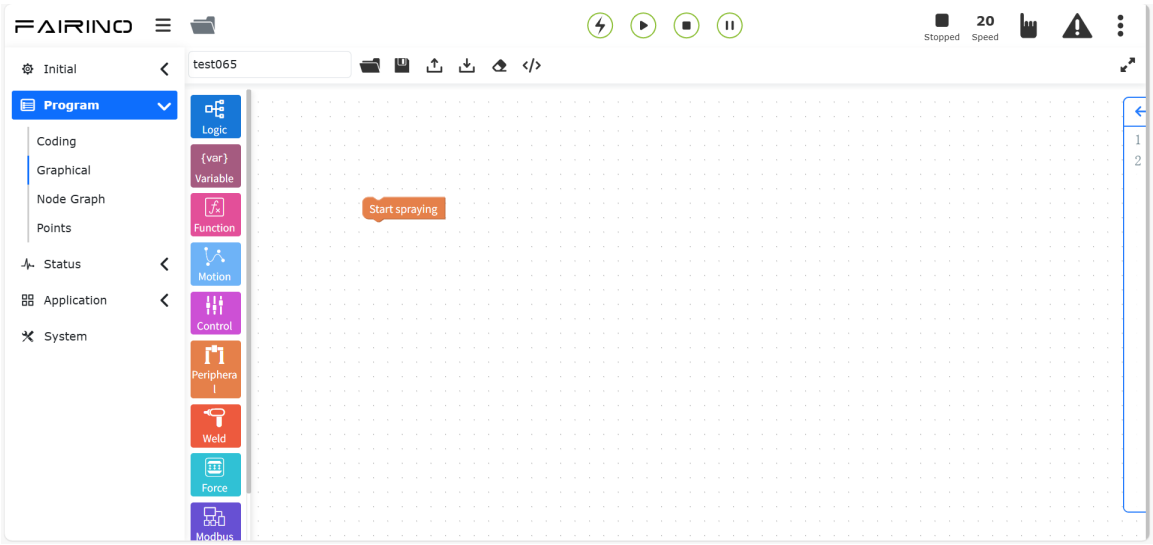


Figure 10.7-4 Start spraying instruction code block

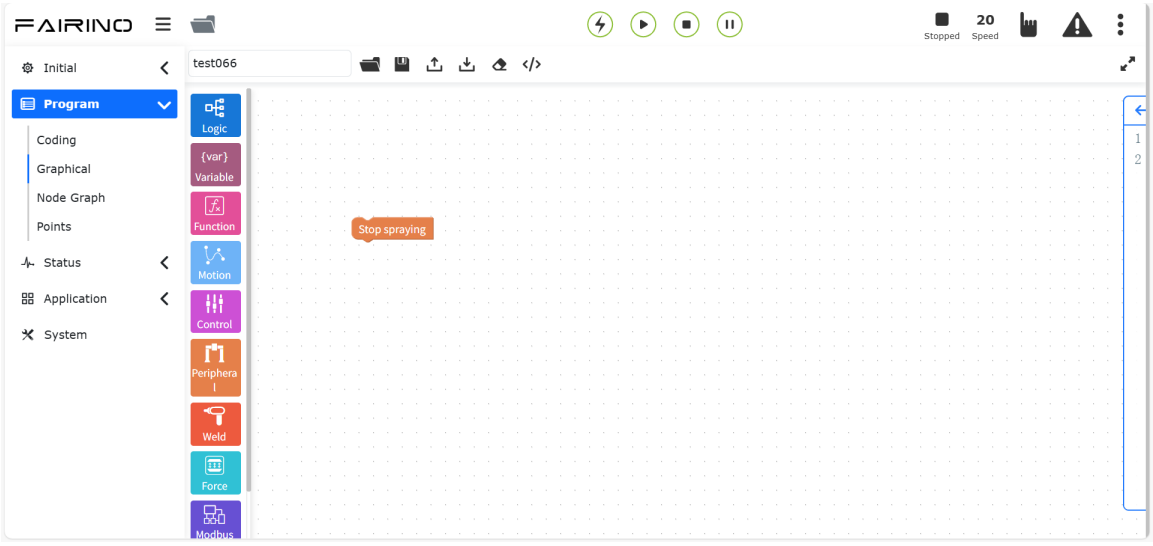


Figure 10.7-5 Stop spraying instruction code block

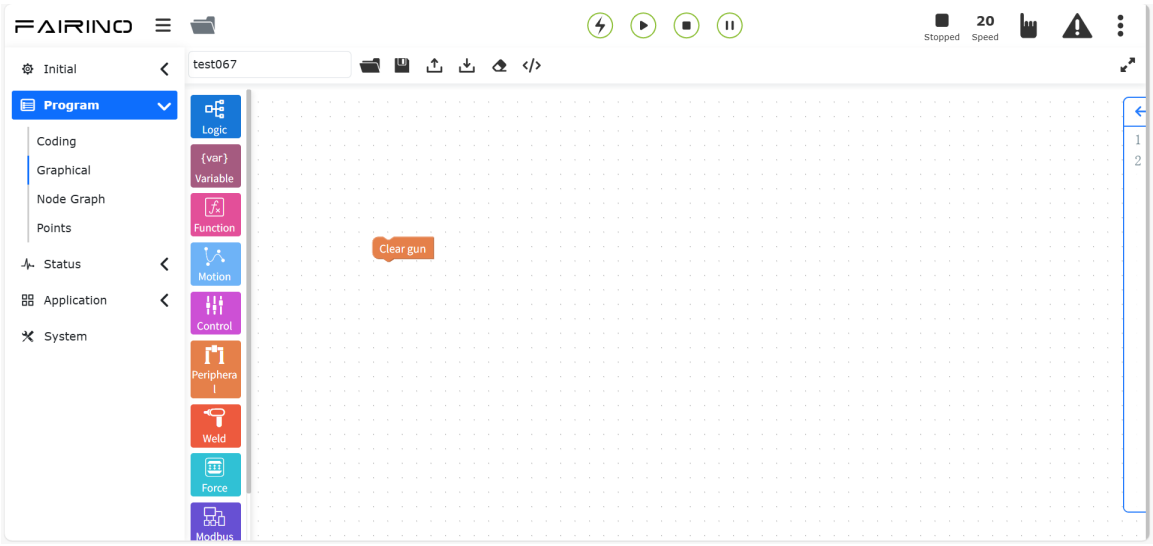


Figure 10.7-6 Start gun cleaning instruction code block

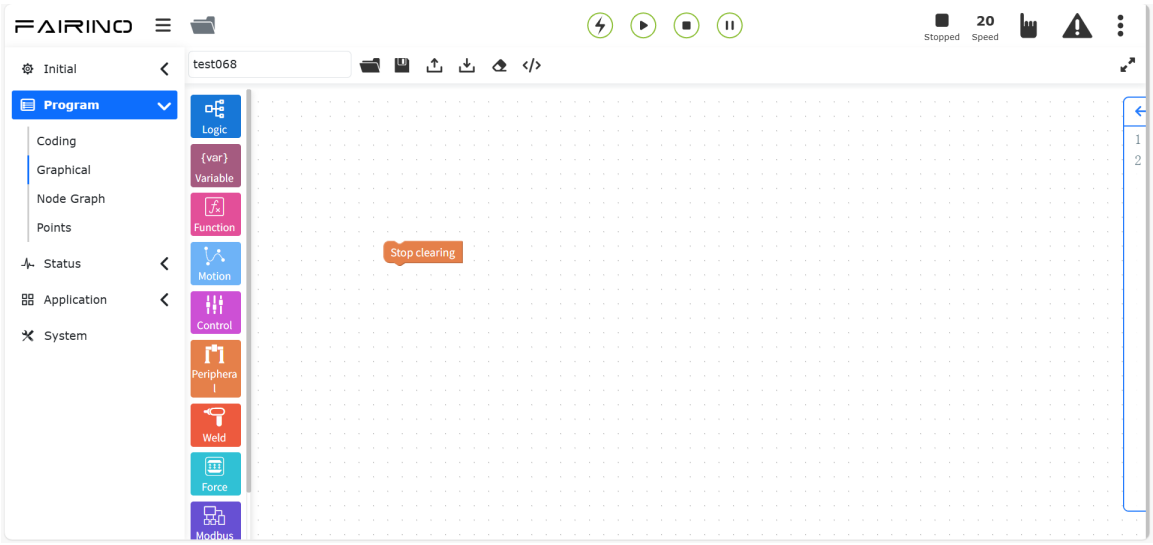


Figure 10.7-7 Stop gun cleaning instruction code block

1.3.10.7.3 Extended axis instruction (controller+PLC)

Drag the “Extended axis instruction” code block to enter the graphical editing interface workspace.

This instruction is used in combination with the PTP instruction for scenarios using external axes. It can decompose the movement of a point in the X-axis direction in space into external axis movement. Select the external axis number, select synchronous movement mode, and select the point to be reached.

It is divided into UDP communication loading/configuration, asynchronous movement, synchronous PTP/LIN movement, synchronous ARC movement, return to zero instruction and enable instruction.

“UDP communication configuration” command node, enter the IP address, port number and communication cycle;

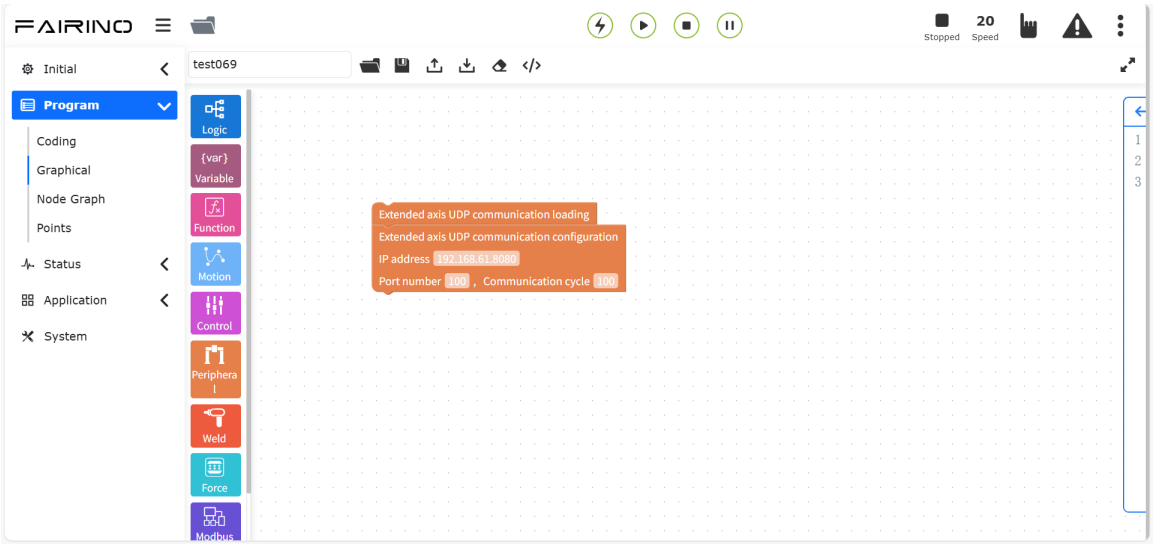


Figure 10.7-8 UDP communication configuration command code block

“Asynchronous motion” command node, parameters:

- Point name: teaching point
- Debug speed (%): 0~100

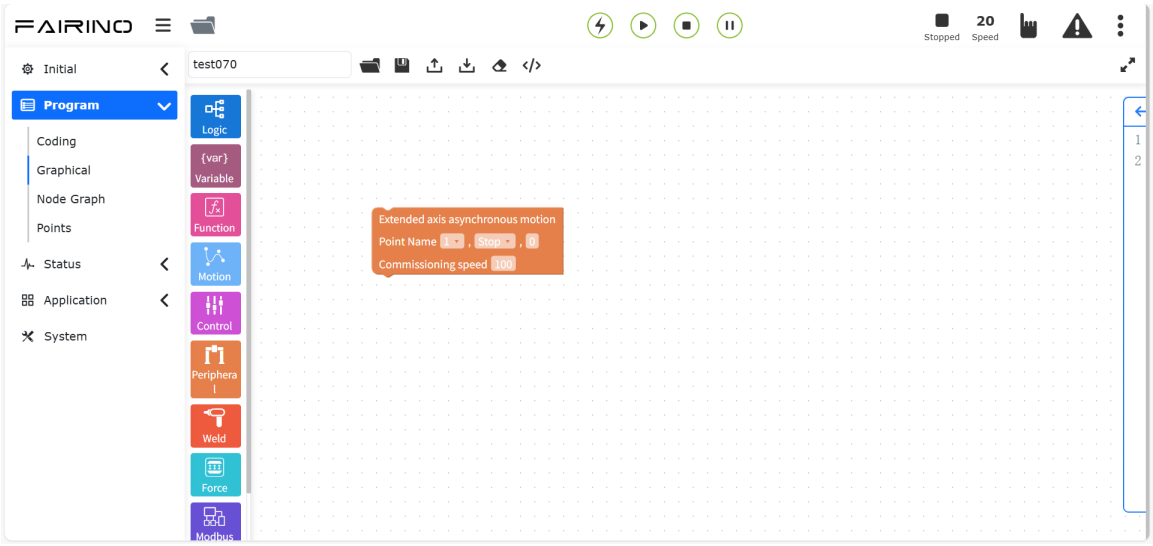


Figure 10.7-9 Asynchronous motion command code block

“Synchronous PTP/LIN motion” command node, parameters:

- Motion selection: PTP/LIN
- Point name: teaching point
- Debug speed (%): 0~100

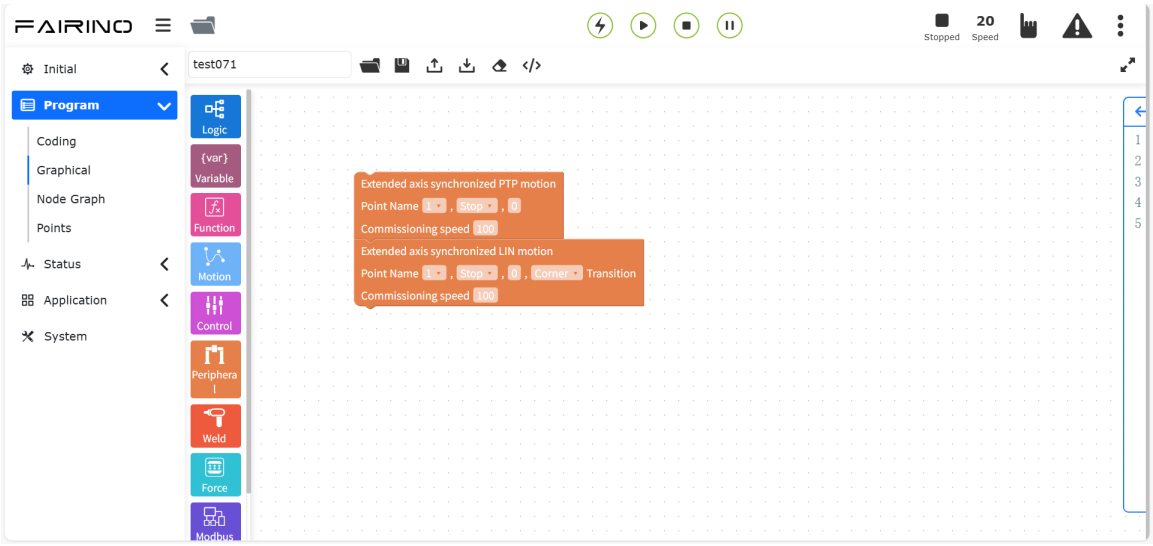


Figure 10.7-10 “Synchronous PTP/LIN motion” instruction code block

“Synchronous ARC motion” instruction node, default motion mode is ARC, parameters:

Point name: teaching point

Debug speed (%): 0~100

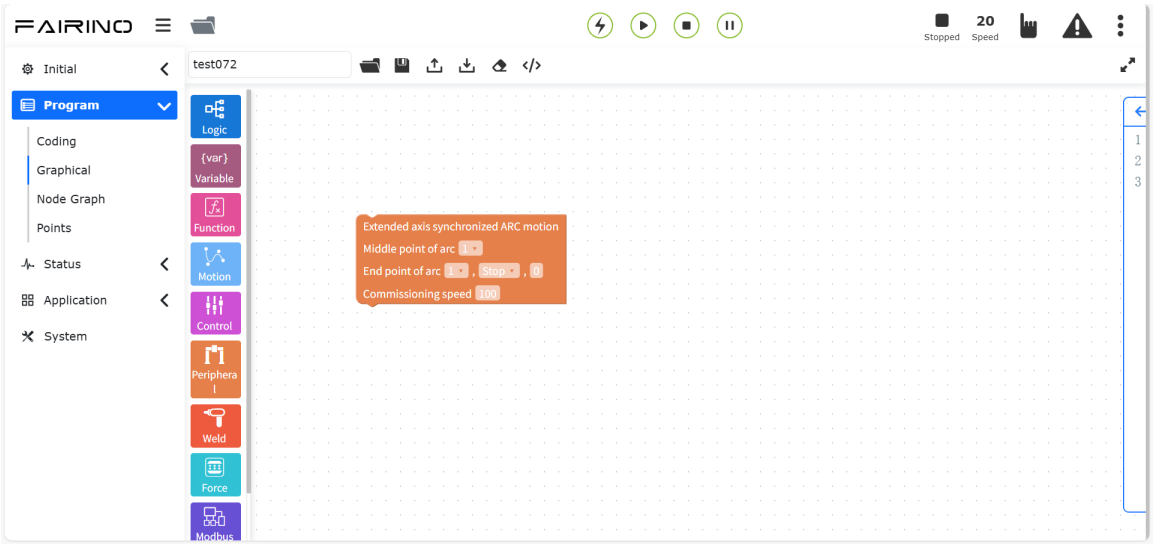


Chart 10.7-11 “Synchronous ARC motion” instruction code block

“Extended axis return to zero” instruction node, parameters:

- Extended axis number: 1~4
- Return to zero mode: current position return to zero/negative limit return to zero/positive limit return to zero
- Home speed: 0~2000, default position 5
- Zero point clamp speed: 0~2000, default is 1

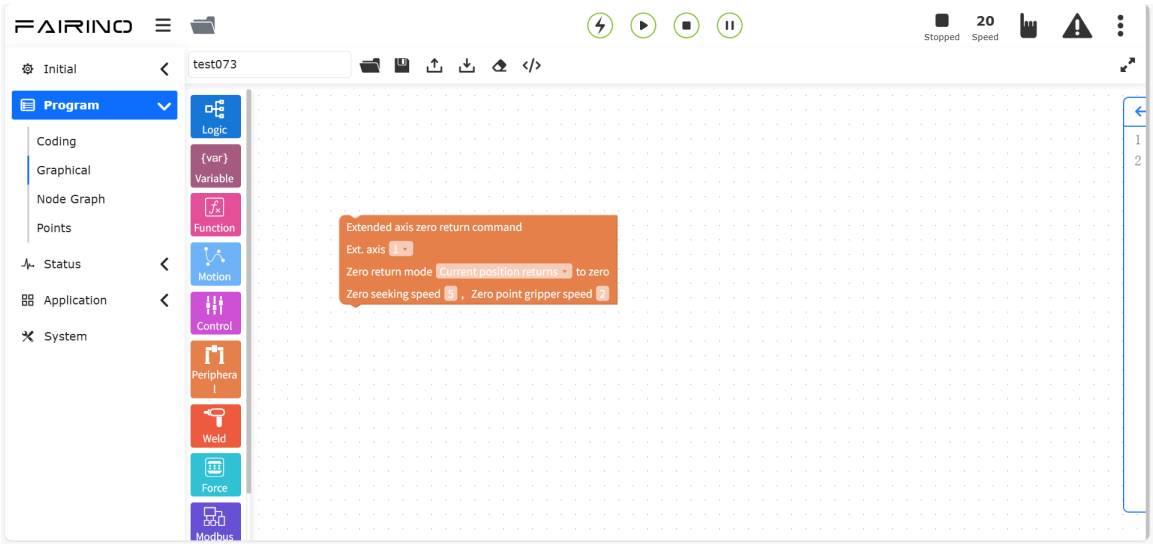


Chart 10.7-12 Extended axis return to zero instruction code block

“Extended axis enable” instruction node,, parameters:

- Extended axis number: 1~4

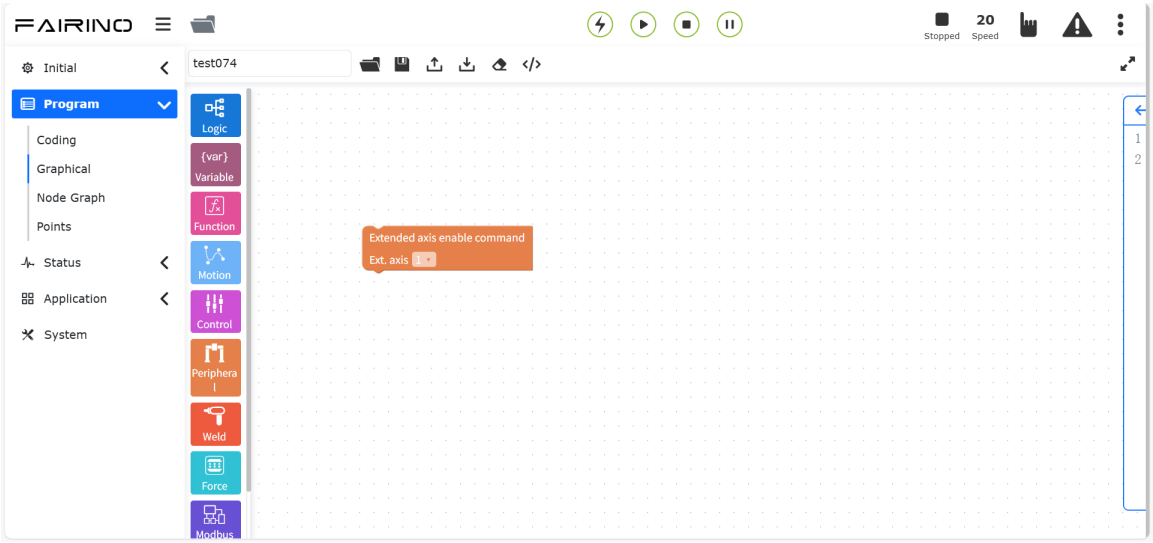


Figure 10.7-13 Extended axis enable instruction code block

1.3.10.7.4 Extended axis instruction (controller+servo drive)

Drag the “Extended axis instruction” code block to enter the graphical editing interface workspace.

This instruction can configure the extended axis parameters. Set different parameters according to different control modes. The configured extended axis can set its zero point.

It is divided into servo ID, control mode, servo enable and servo return to zero; the control mode is further divided into position mode and speed mode. These two nodes need to be used in conjunction with the control mode, otherwise they cannot take effect if added separately.

“Servo ID” command node, parameters:

- Servo ID: 1~15

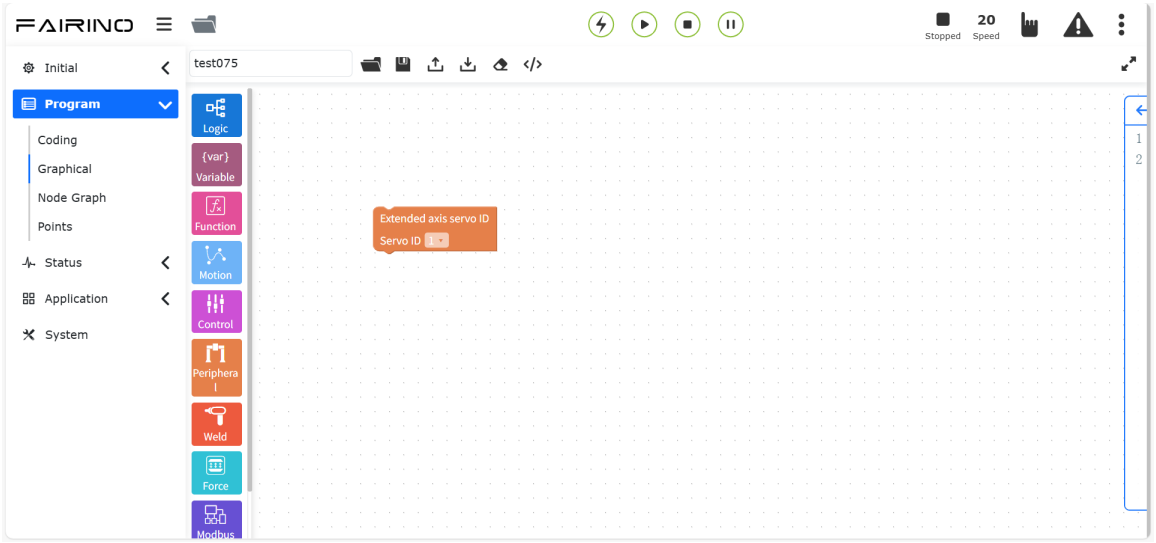


Figure 10.7-14 Servo ID command code block

“Control mode” command node, parameters:

- Servo ID: 1~15
- Control mode: Position mode/Speed mode

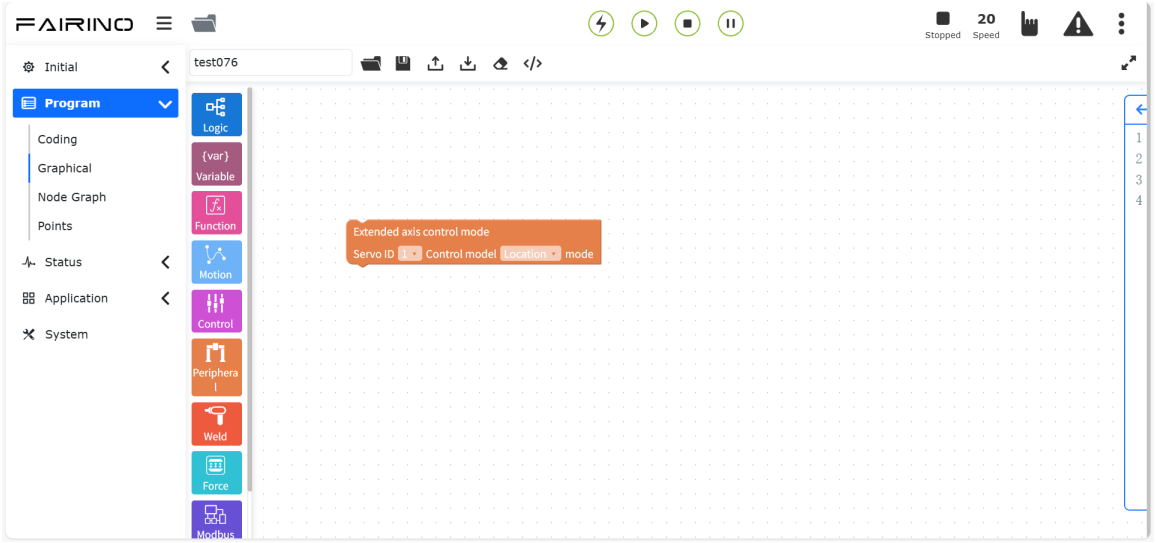


Figure 10.7-15 Control mode command code block

“Servo enable” command node, parameters:

- Servo ID: 1~15
- Servo enable: Servo enable/disable

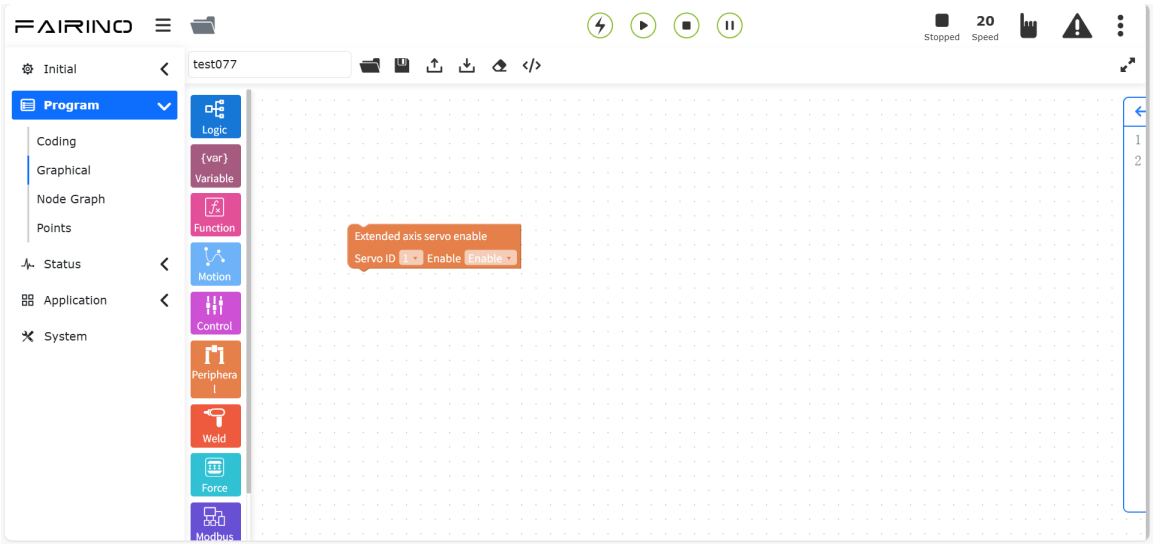


Figure 10.7-16 Servo enable command code block

“Servo return to zero” command node, parameters:

- Servo ID: 1~15
- Return to zero mode: current position return to zero/negative limit return to zero/positive limit return to zero
- Homing speed: 0~2000, default position 5
- Zero point clamp speed: 0~2000, default is 1
- Acceleration percentage: 1~100, default is 100

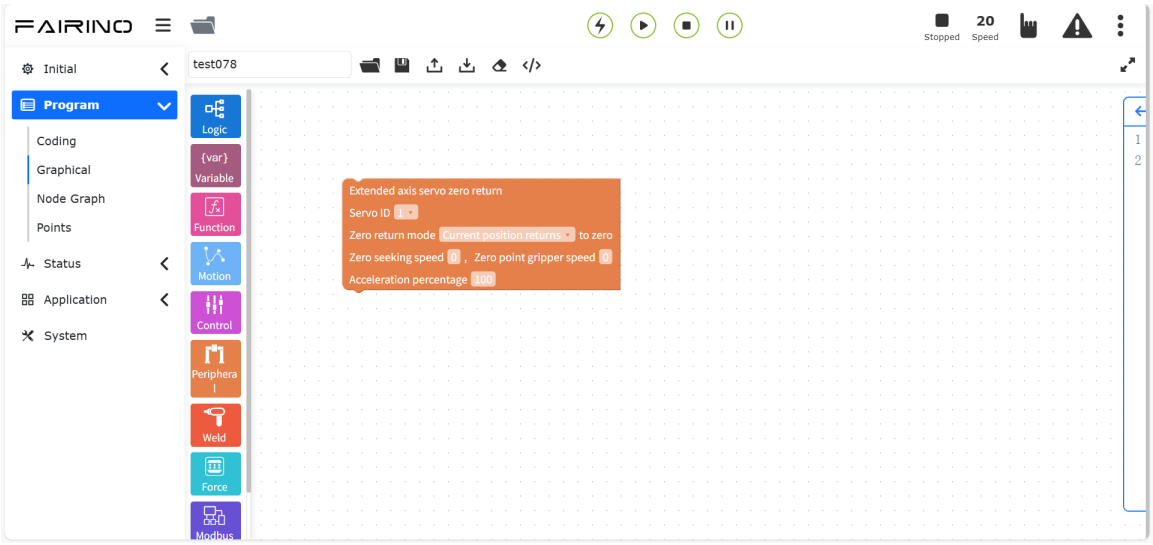


Chart 10.7-17 Servo return to zero command code block

“Position mode” command node, parameters:

- Servo ID: 1~15
- Target position: unlimited
- Homing speed: unlimited
- Acceleration percentage: 1~100, default is 100

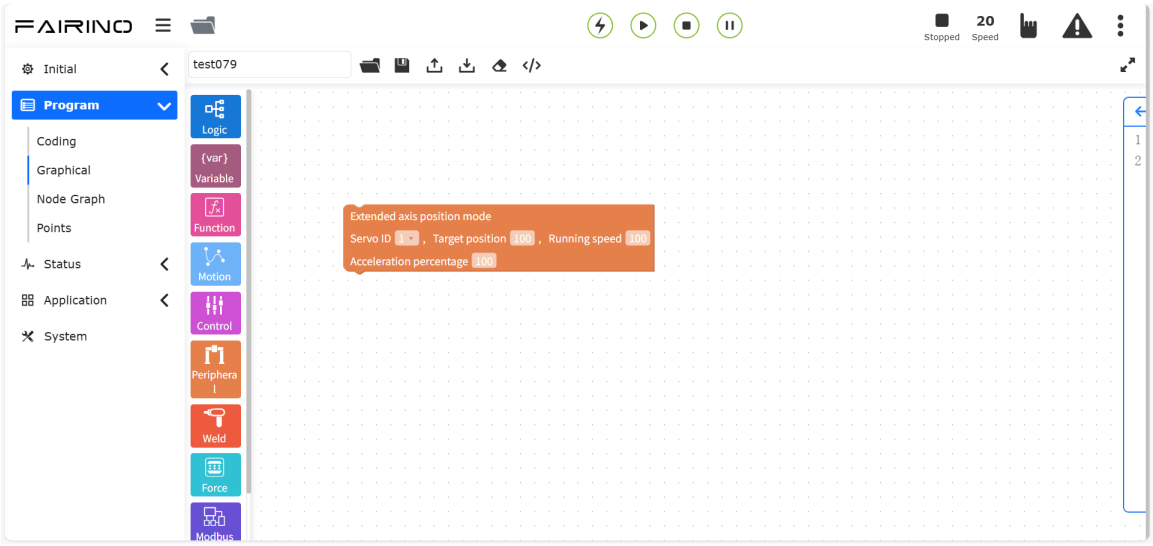


Chart 10.7-18 Position mode command code block

"Speed mode" command node, parameters:

- Servo ID: 1~15
- Target speed: unlimited
- Acceleration percentage: 1~100, default is 100

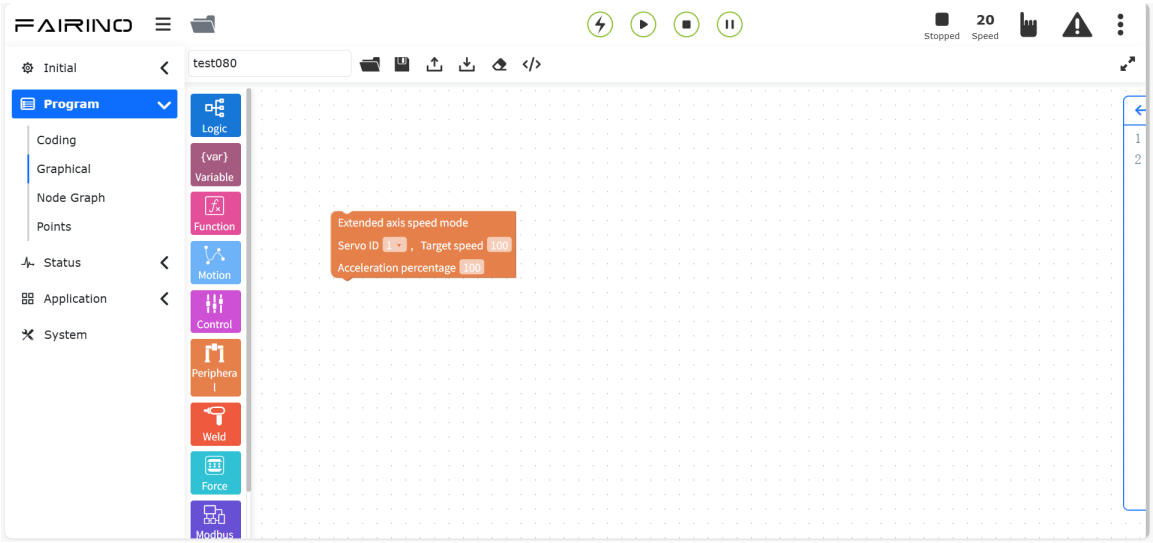


Chart 10.7-19 Speed mode command code block

1.3.10.7.5 Conveyor belt command

This command contains four commands: IO real-time detection, position real-time detection, tracking on and tracking off. See the robot peripherals chapter for details

“IO real-time detection” instruction node, parameters:

- Maximum waiting time: 0~10000

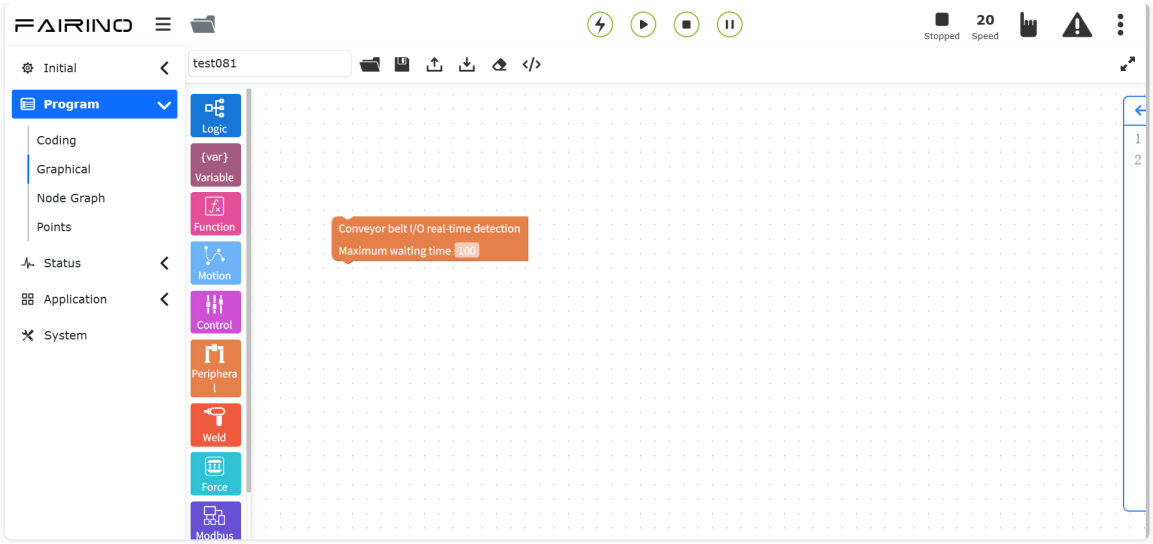


Figure 10.7-20 IO real-time detection instruction code block

“Position real-time detection” instruction node, parameters:

- Working mode: Tracking grabbing/Tracking motion/TPD tracking

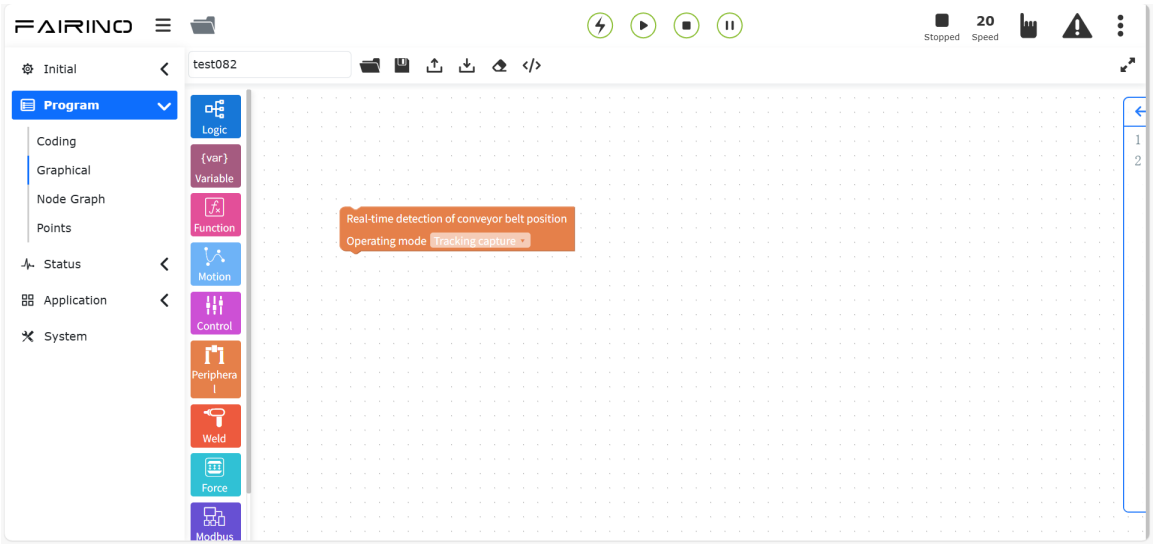


Figure 10.7-21 Position real-time detection instruction code block

“Tracking on”, “Tracking off” instruction nodes, parameters:

Working mode: Tracking grabbing/Tracking motion/TPD tracking

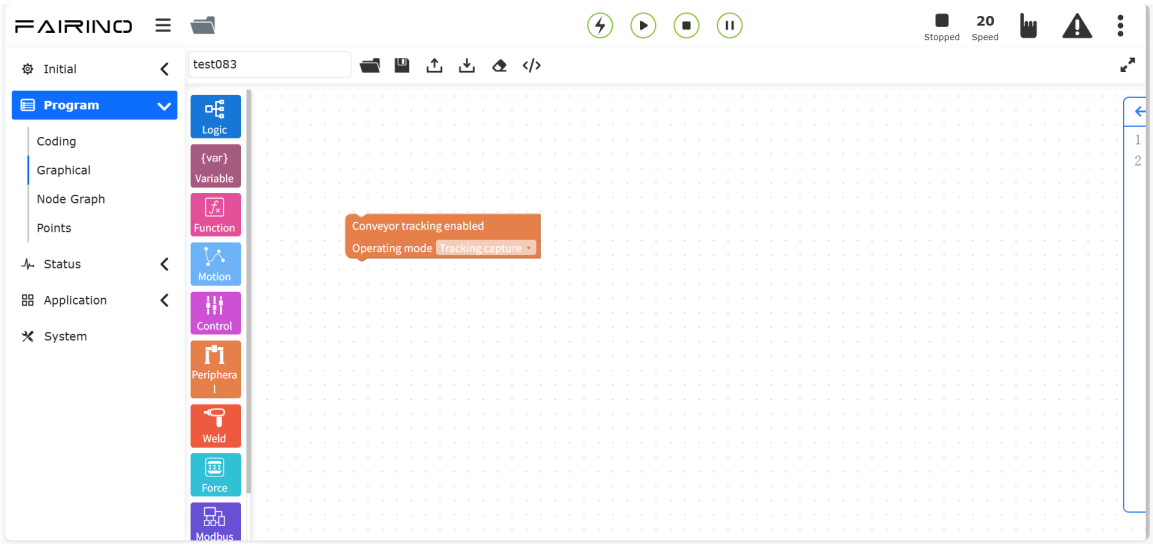


Figure 10.7-22 Tracking on/off instruction code block

1.3.10.7.6 Polishing instruction

Drag the “Polishing instruction” code block to enter the graphical editing interface workspace.

This instruction is used for polishing scenes. When using it, you need to unload the driver first and then load the driver, and then set the polishing device to enable.

Then set the speed, contact force, extension distance and control mode of the polishing device. At the same time, you can clear the polishing device errors and clear the device force sensor.

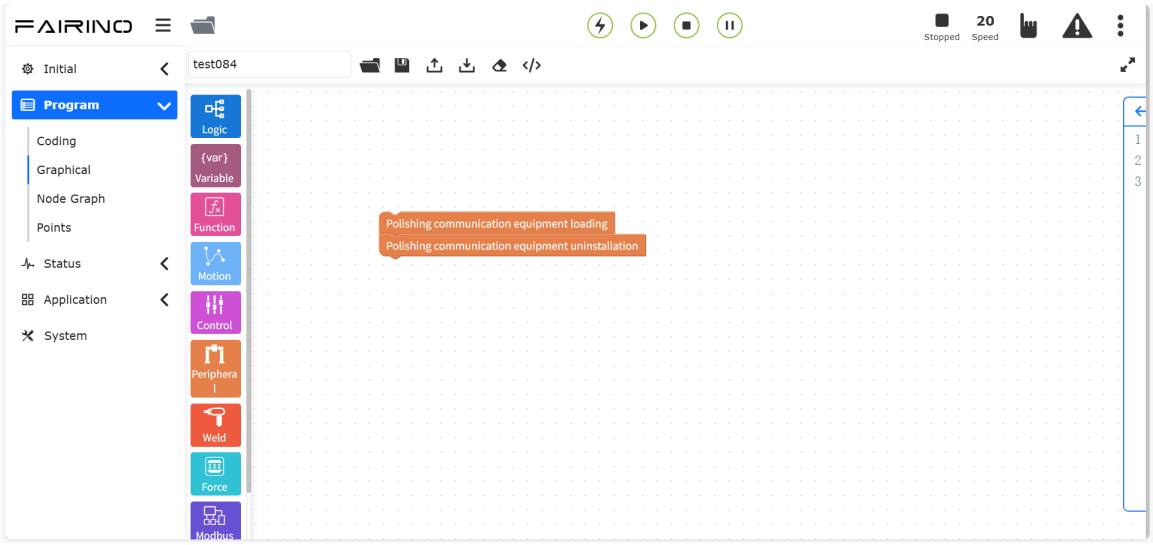


Figure 10.7-23 Communication driver load/unload instruction code block

“Device enable” instruction node, parameters:

- Device enable: up enable/down enable

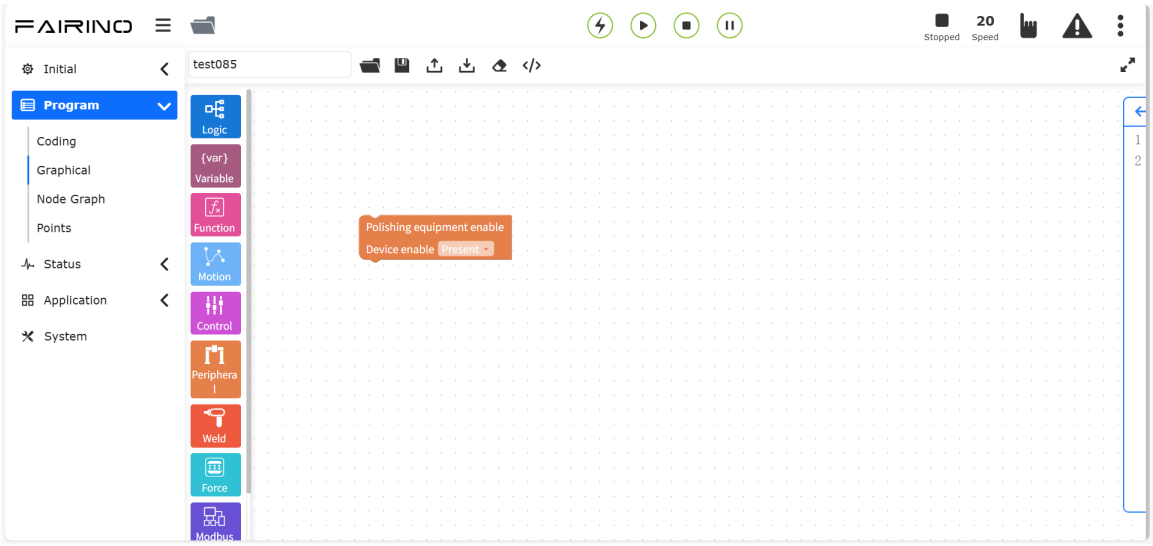


Figure 10.7-24 Device enable instruction code block

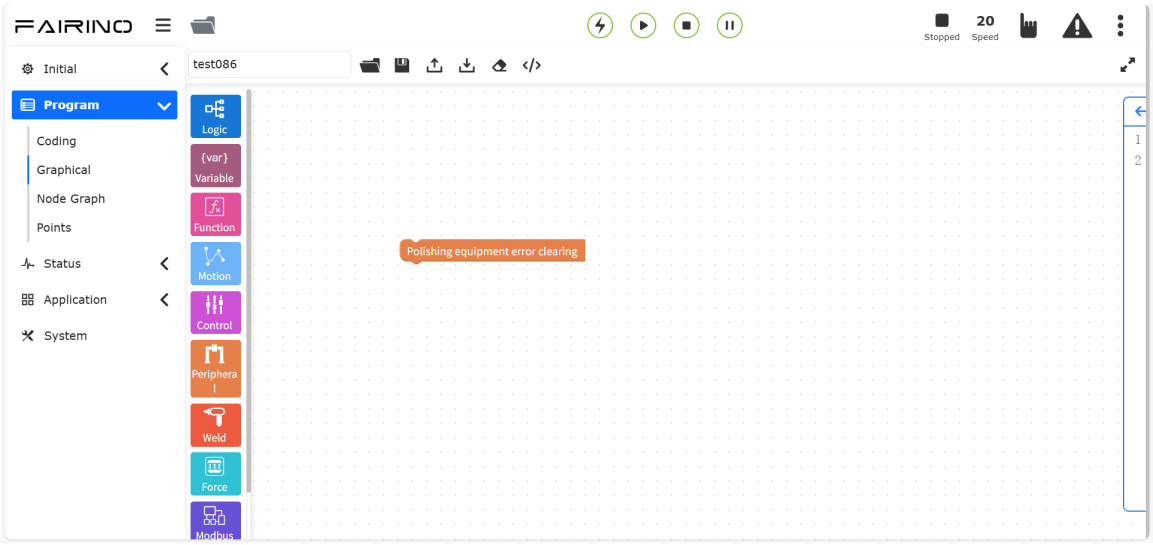


Figure 10.7-25 Device error clear instruction code block

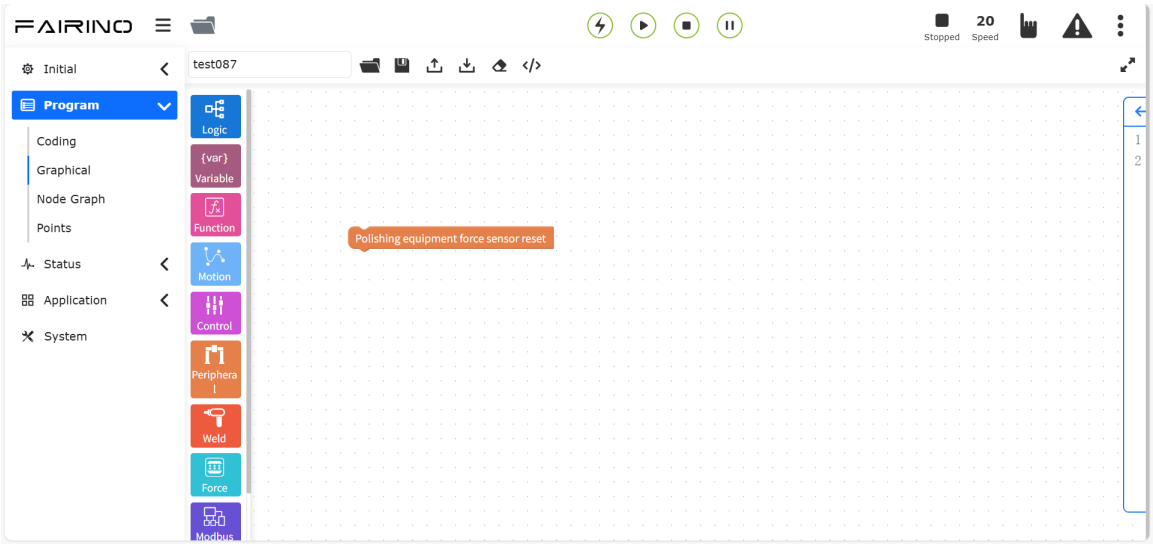


Figure 10.7-26 Device force sensor clear command code block

“Speed” command node, parameters:

- Speed: 0~5500

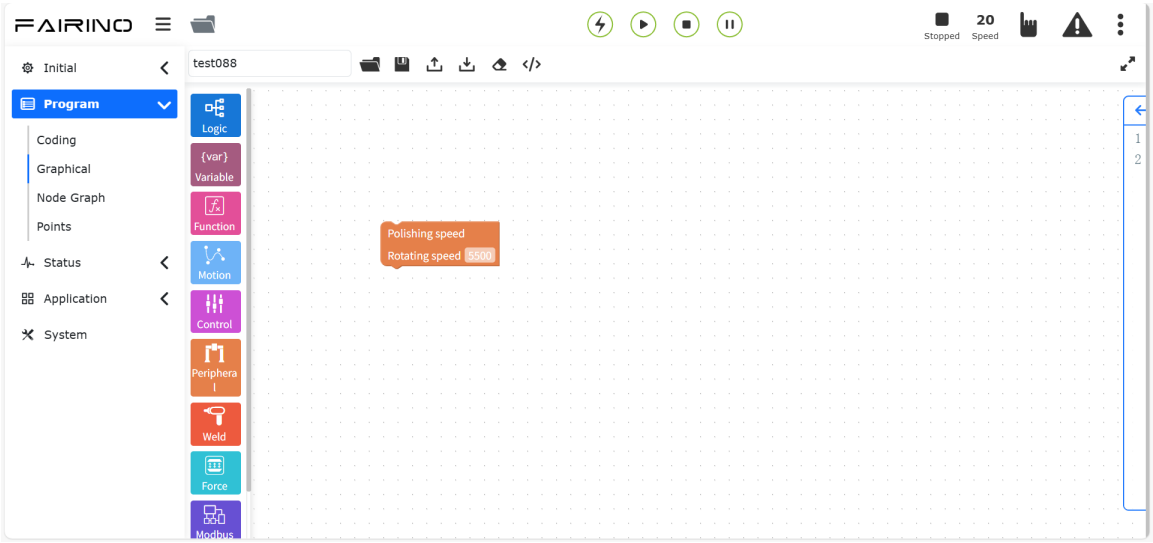


Figure 10.7-27 Device speed command code block

“Set force” command node, parameters:

- Set force: 0~200

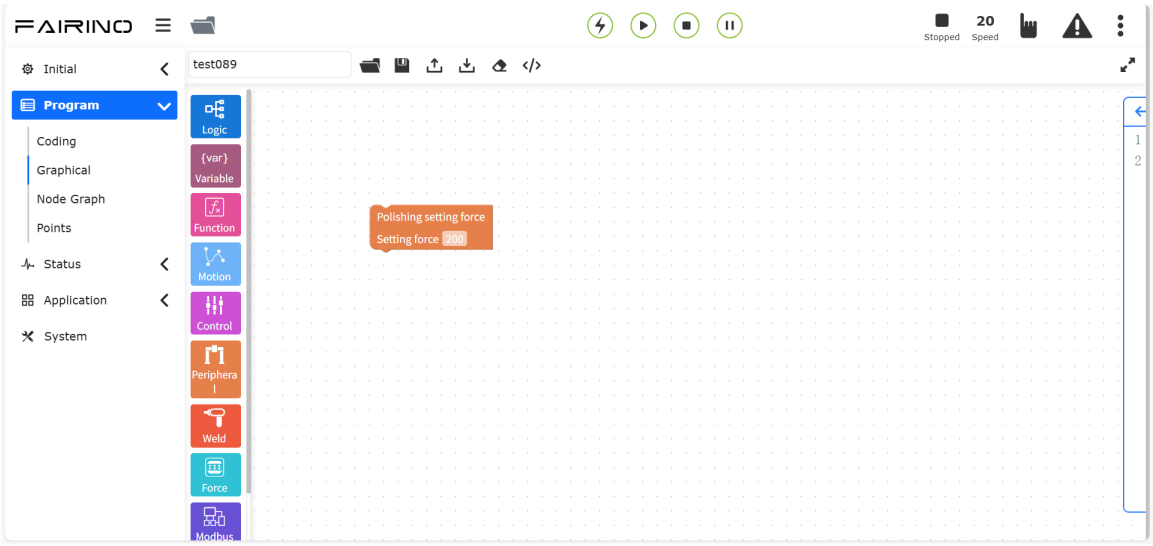


Figure 10.7-28 Set force command code block

“Extend distance” command node, parameters:

- Extend distance: 0~12

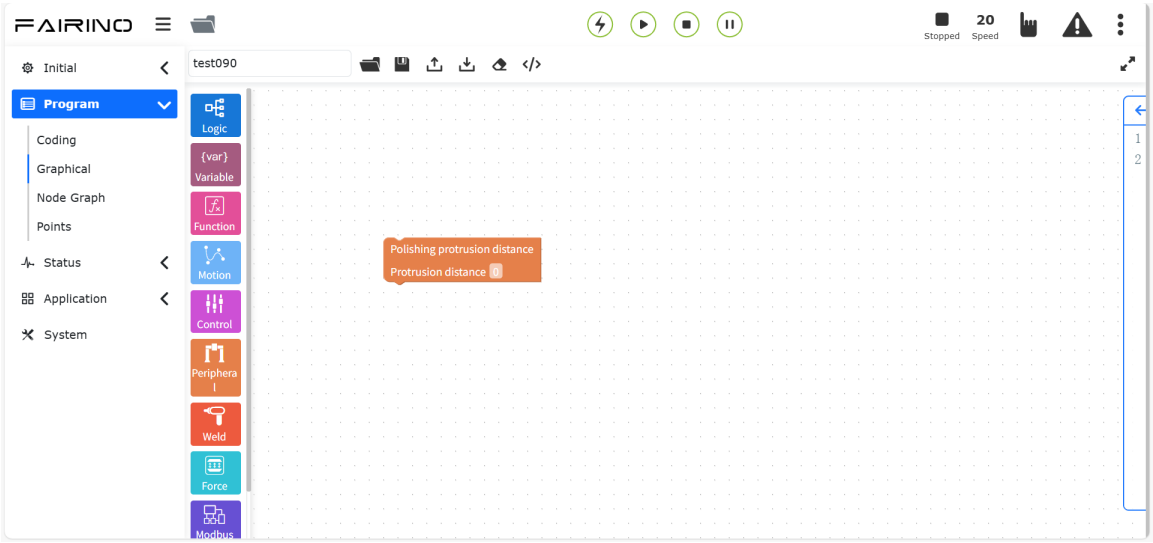


Figure 10.7-29 Extension distance instruction code block

“Polishing contact force” instruction node, parameters:

- Contact force: 0~10000

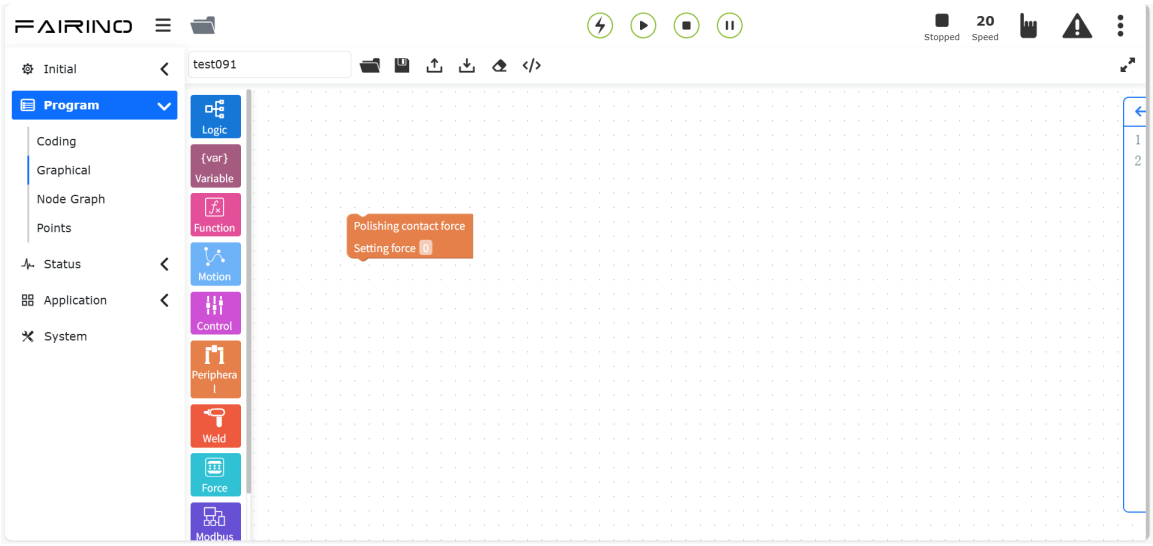


Figure 10.7-30 Polishing contact force instruction code block

“Set force transition time” instruction node, parameters:

- Set force transition time: 0~10000

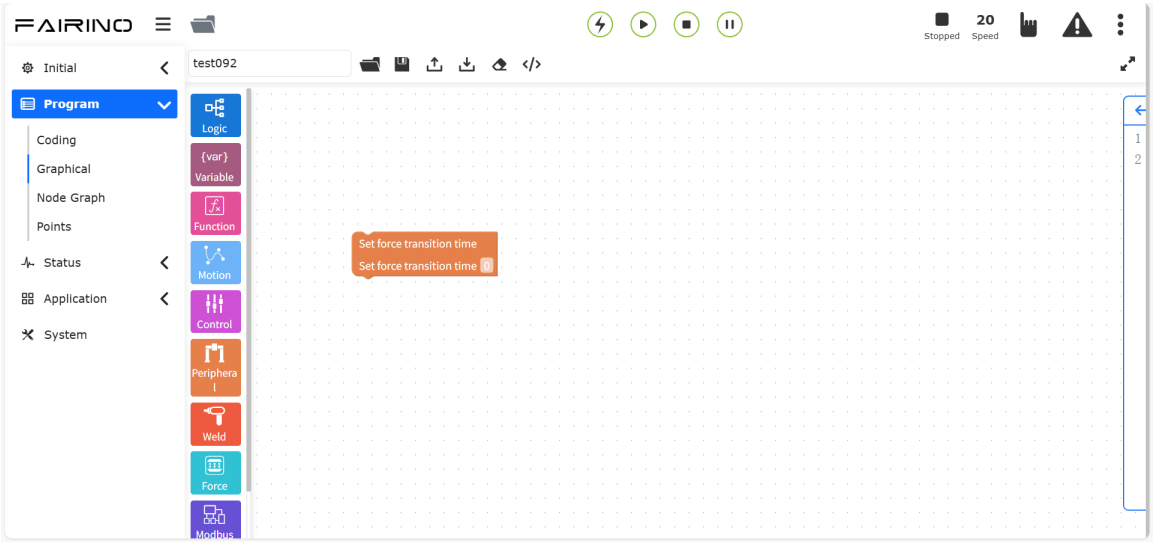


Figure 10.7-31 Set force transition time instruction code block

“Workpiece weight” instruction node, parameters:

- Workpiece weight: 0~10000

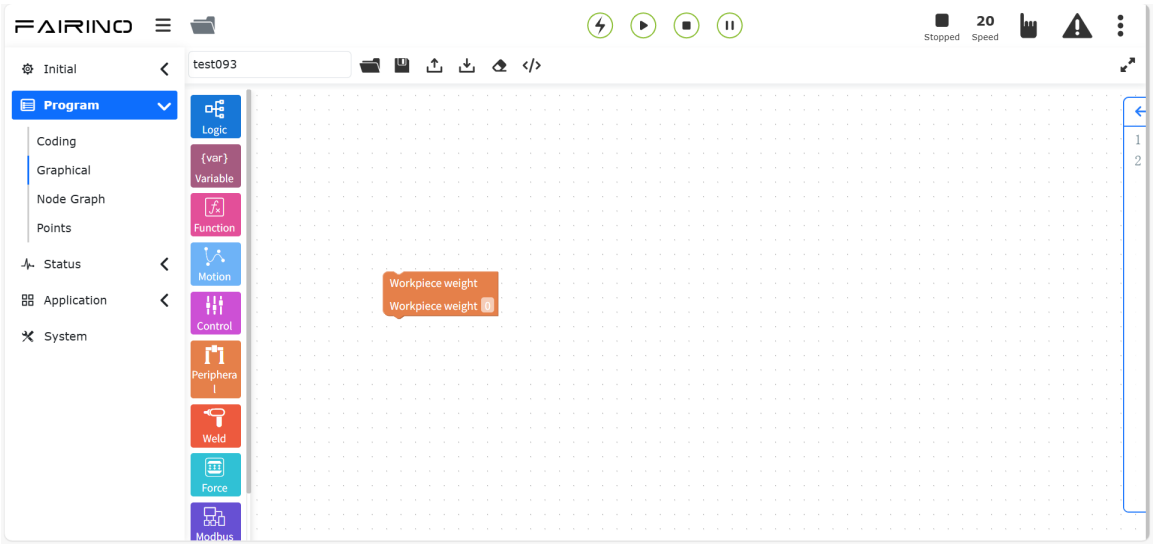


Figure 10.7-32 Workpiece weight instruction code block

“Control mode” instruction node, parameters:

- Control mode: return to zero mode/position mode/torque mode

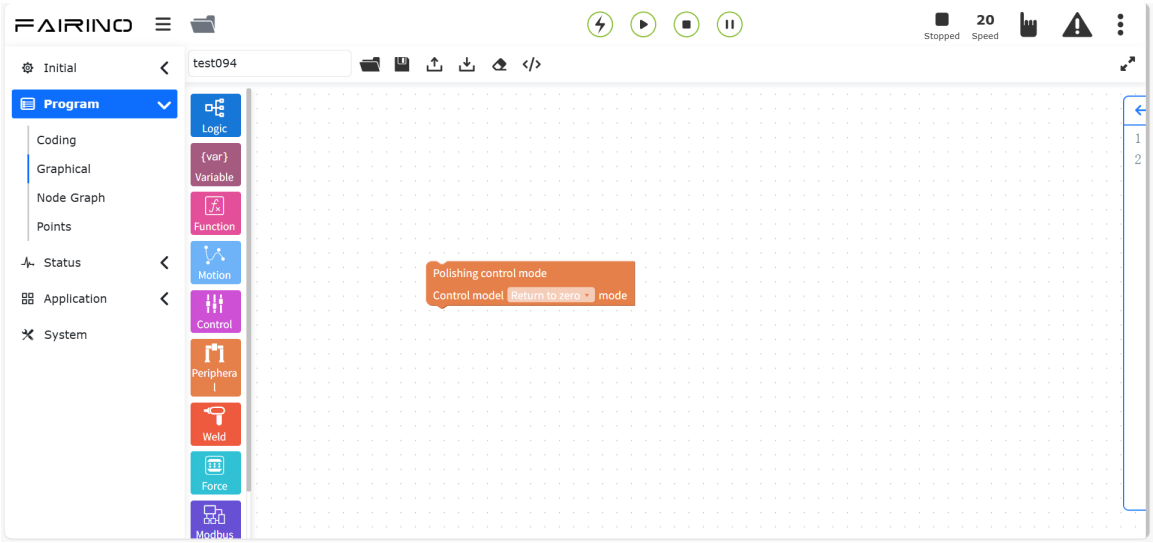


Figure 10.7-33 Control mode instruction code block

1.3.10.8 Welding Graphical Programming Commands

Welding Graphical Programming Commands include positioning, segment welding, welding, laser tracking and other welding commands.

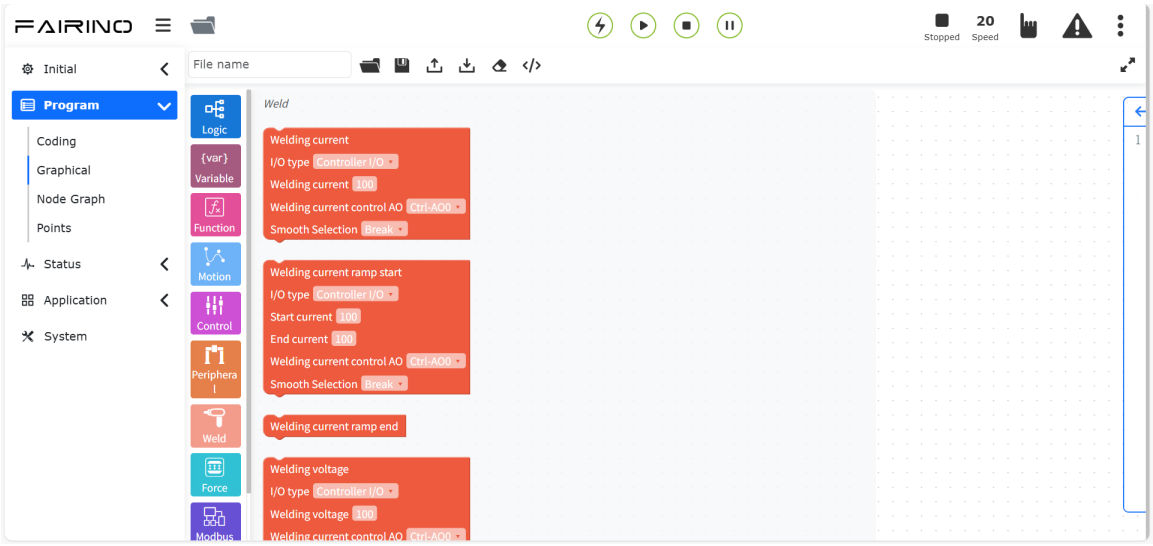


Figure 10.8 Welding Graphical Programming Commands

1.3.10.8.1 Segment welding instructions

Drag the “Segment welding instruction” code block to enter the graphical editing interface work area.

This instruction is a special welding instruction, mainly used for cyclic intermittent welding scenarios with one section of welding and one section of non-welding.

Between the starting point and the end point, use this instruction, select the segment welding mode, select the starting point and end point, set the debugging speed, set the DO port for arc starting, execution length, non-execution length, set the function mode according to the actual application scenario, and the swing selection and rounding rules can realize the segment welding function.

For detailed operations, see the program teaching page segment welding instruction.

1. “Arc off/arc on” command node, parameters:

- I/O type: controller IO/extension IO
- Welding process number: 0 ~ 7
- Maximum waiting time (ms): 0 ~ 10000

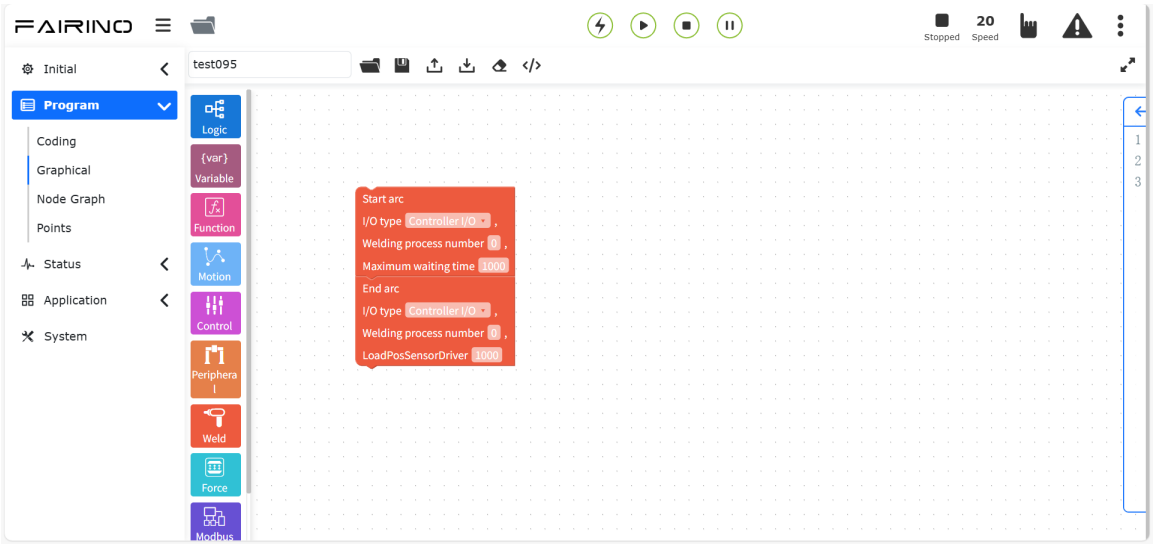


Figure 10.8-1 “Arc off/arc on” command code block

2. “Segment welding” command node, parameters:

- Segment welding mode: no change of posture/change of posture
- Starting point: teaching point
- End point: teaching point
- Debug speed (%): 0~100, default is 100
- Execution length: 0~1000
- Non-execution length: 0~1000
- Function mode: 0~100, default is 100
- Swing selection: execute segment without swing/execute segment swing
- Rounding rules: no rounding/loop rounding/single-segment rounding

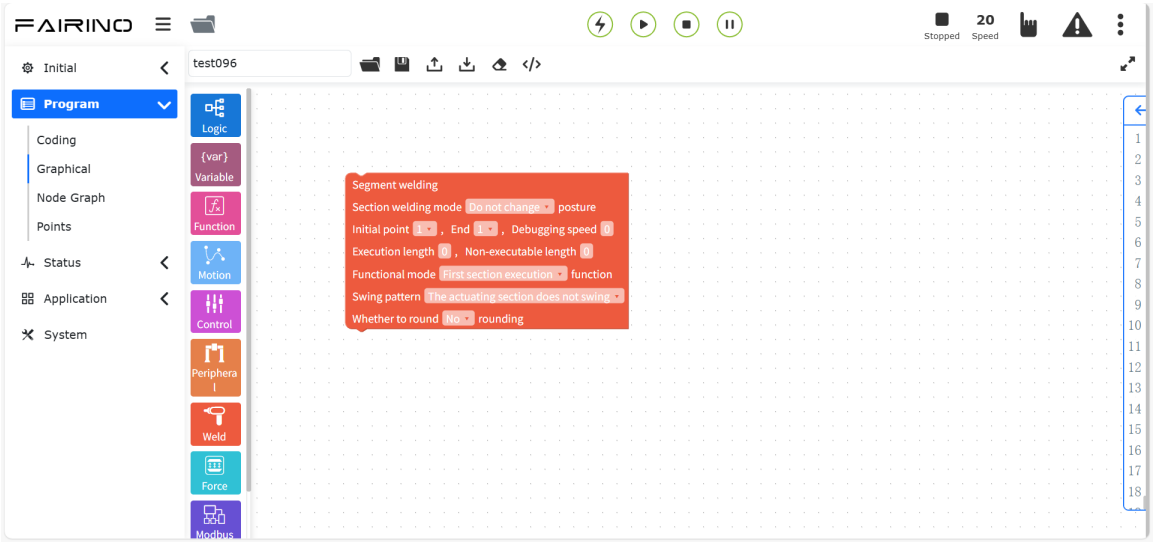


Figure 10.8-2 Segment welding instruction code block

1.3.10.8.2 Welding instruction

Drag the “welding instruction” code block to enter the graphical editing interface workspace.

This instruction is mainly used for welding machine peripherals. Before adding this instruction, please confirm whether the welding machine configuration in the user peripherals is complete. For details, see the robot peripherals chapter.

1. “Welding Voltage” Command Node, Parameters:

- I/O Type: Controller IO / Extended IO
- Welding Voltage: Minimum value is 0
- Welding Current Control AO: Ctrl-AO0 / Ctrl-AO1
- Smoothing Selection: Break / Serious

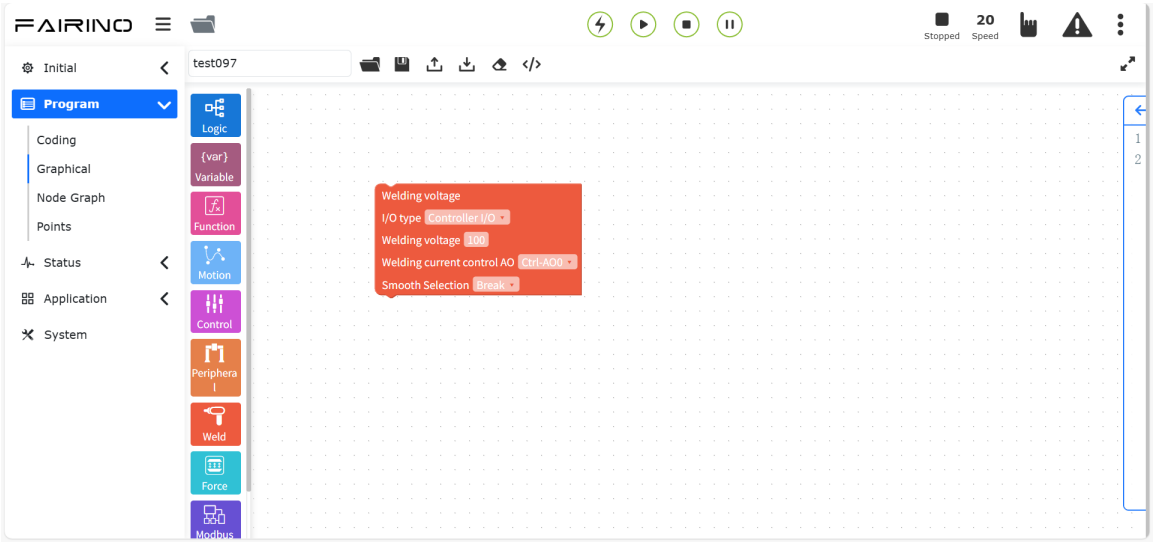


Figure 10.8-3 Welding Voltage Command Code Block

2. “Welding Current” Command Node, Parameters:

- I/O Type: Controller IO / Extended IO
- Welding Current: Minimum value is 0
- Welding Current Control AO: Ctrl-AO0 / Ctrl-AO1
- Smoothing Selection: Break / Serious

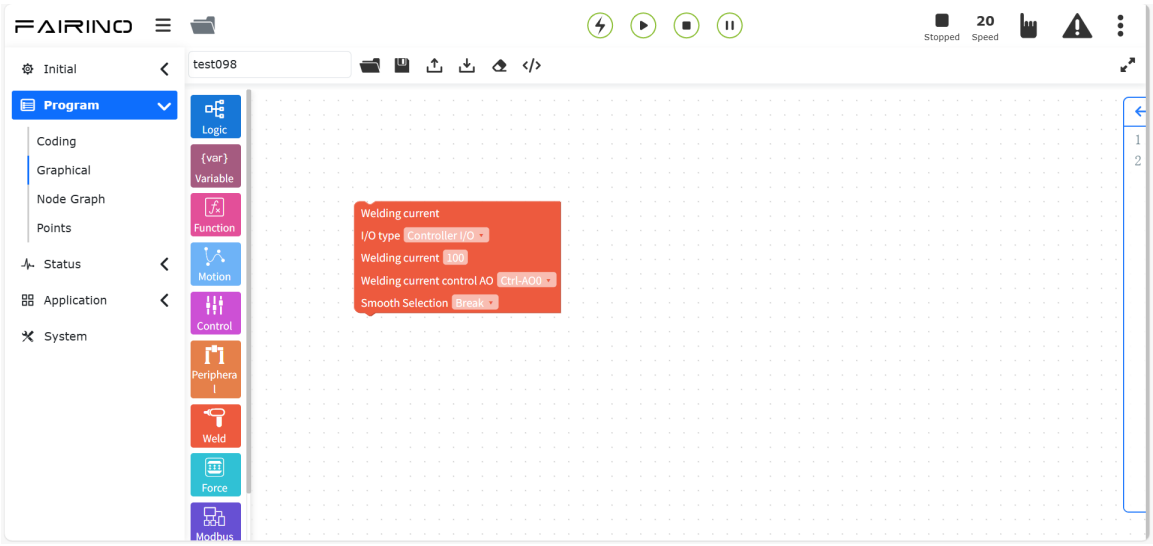


Figure 10.8-4 Welding machine current command code block

3. “Gas supply/gas off” command node, parameters:

- I/O type: controller IO/extension IO

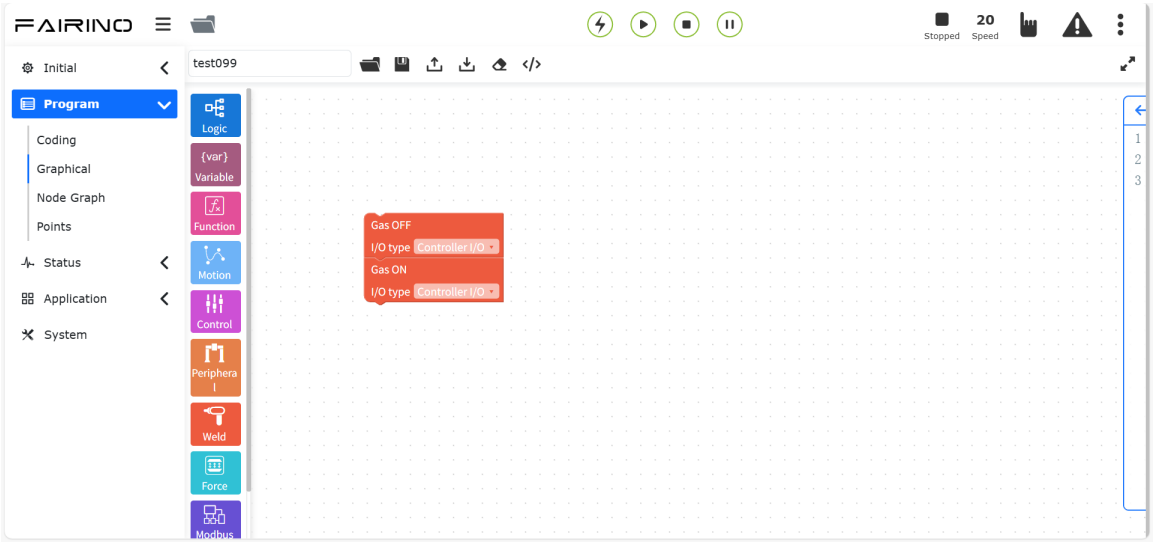


Figure 10.8-5 “Gas supply/gas off” command code block

4. “Forward wire feeding/stop forward wire feeding” command node, parameters:

- I/O type: controller IO/extension IO

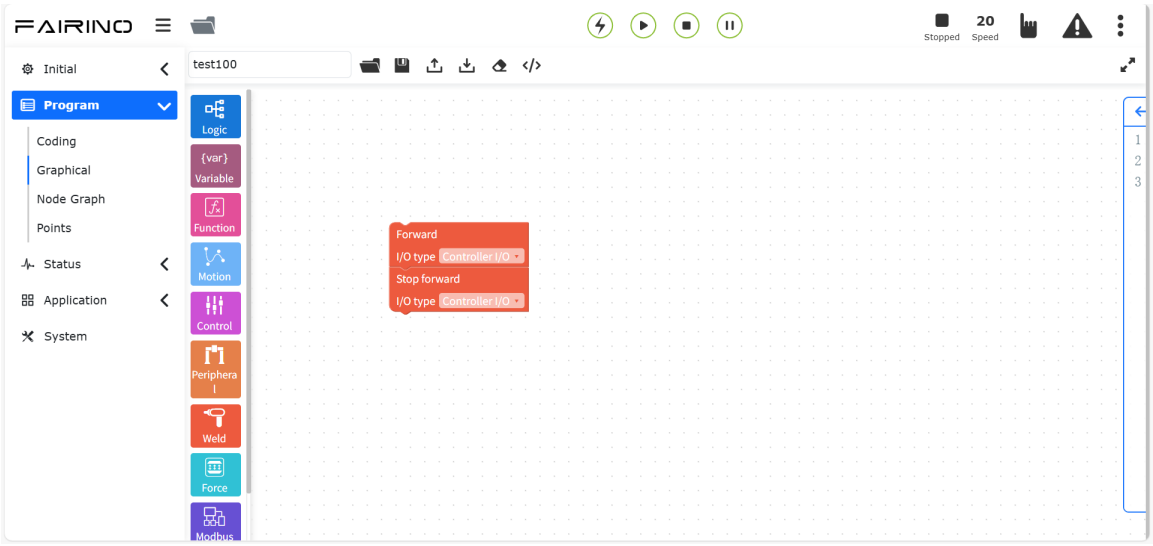


Figure 10.8-6 “Forward wire feeding/stop forward wire feeding” command code block

5. “Reverse wire feeding/stop reverse wire feeding” command node, parameters:

- I/O type: controller IO/extension IO

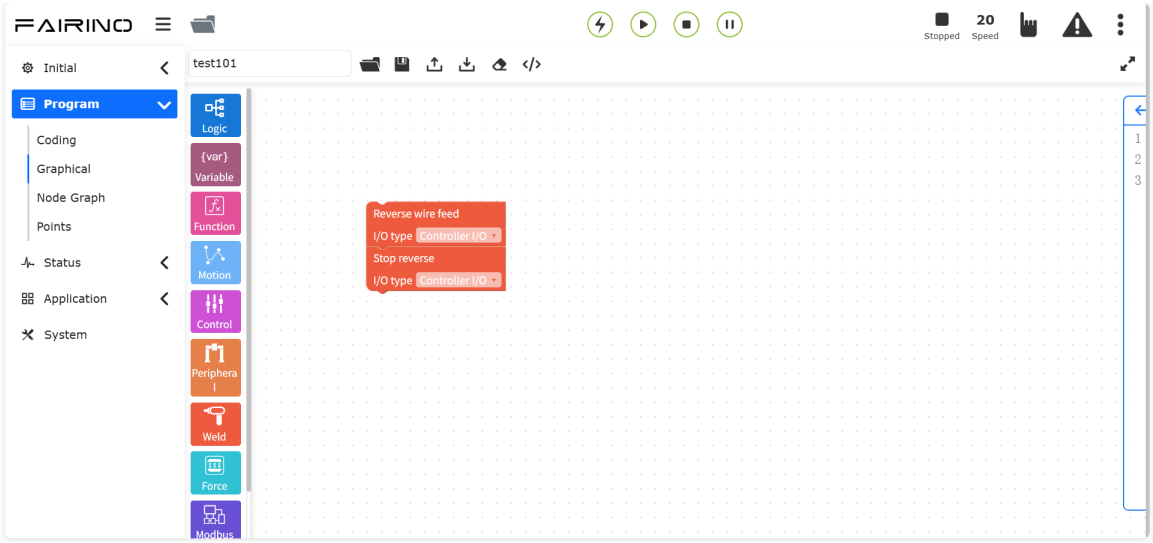


Figure 10.8-7 “Reverse wire feeding/stop reverse wire feeding” command code block

1.3.10.8.3 Laser tracking command

Drag the “Laser tracking command” code block to enter the graphical editing interface workspace.

This command contains three parts: laser command, tracking command and positioning command. Before adding this command, please confirm whether the laser tracking sensor in the user’s peripherals has been successfully configured. See the robot peripherals section for details.

1. “Turn sensor on/off” command node, parameters:

- Select weld type: 0 ~ 49

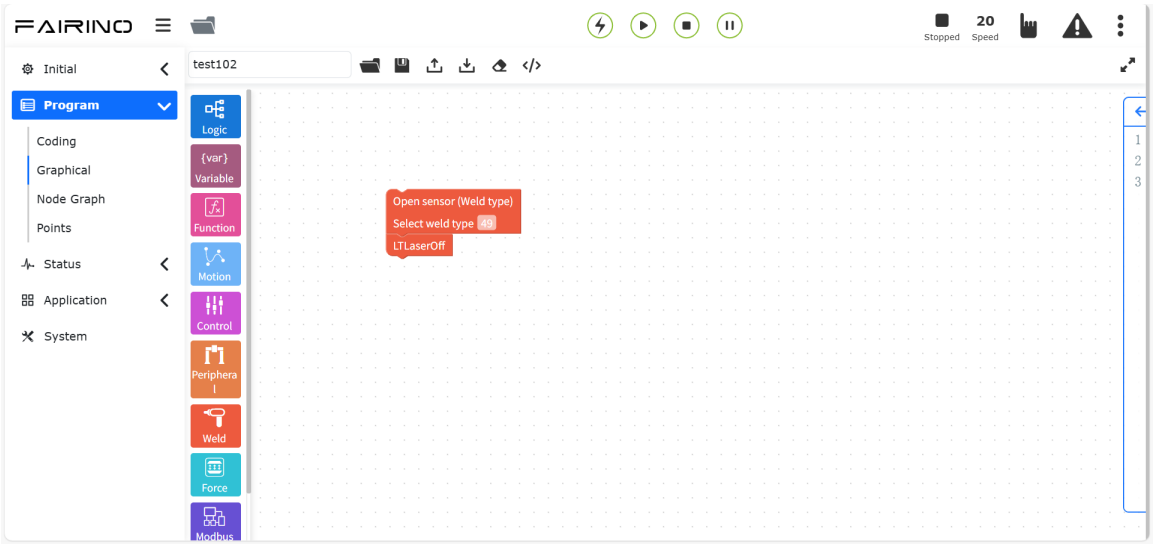


Figure 10.8-8 “Turn sensor on/off—weld type” command code block

- Select task number: 0 ~ 255

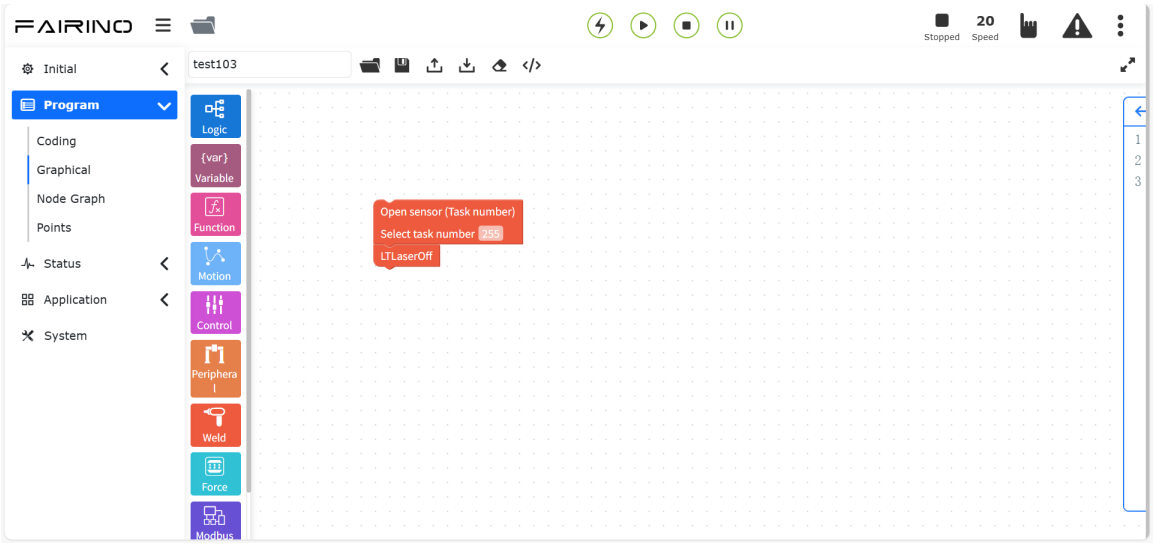


Figure 10.8-9 “Turn sensor on/off—task number” command code block

- Solution: 0 ~ 5

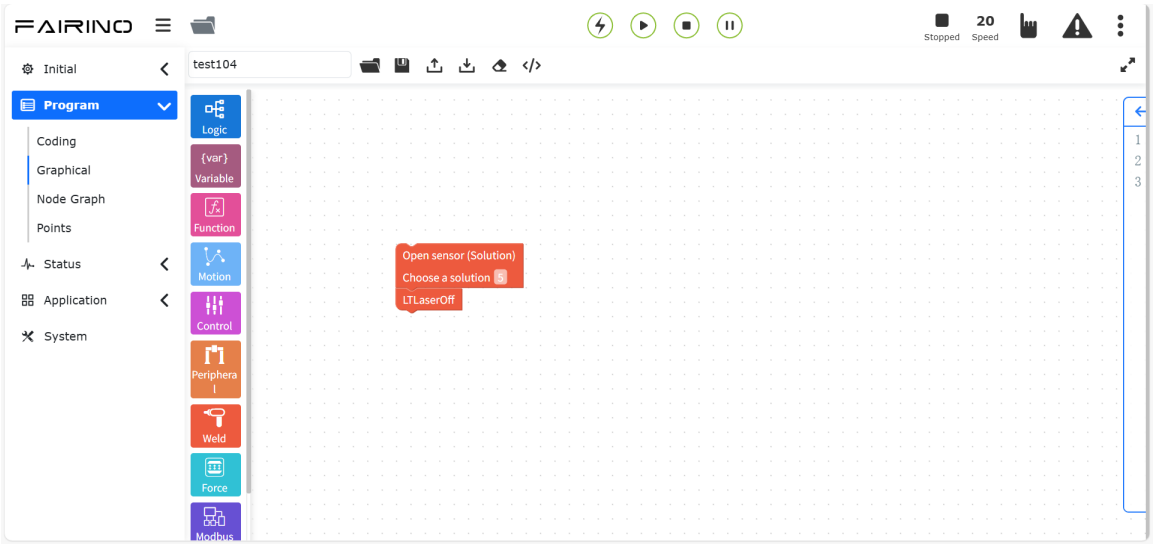


Figure 10.8-10 “Turn sensor on/off—solution” command code block

2. “Load/unload sensor” command node, parameters:

- Function selection: Ruiniu RRT-SV2-BP/Chuangxiang CXZK-RBTA4L

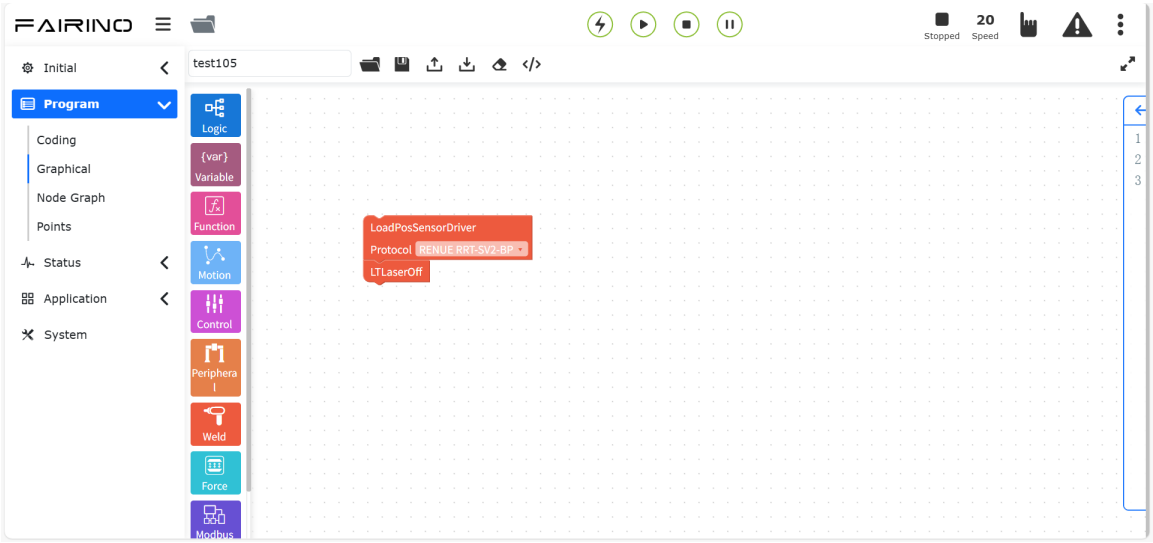


Figure 10.8-11 “Load/unload sensor” instruction code block

3. “Start/stop laser tracking” instruction node, parameters:

- Coordinate system name: custom configuration coordinate system

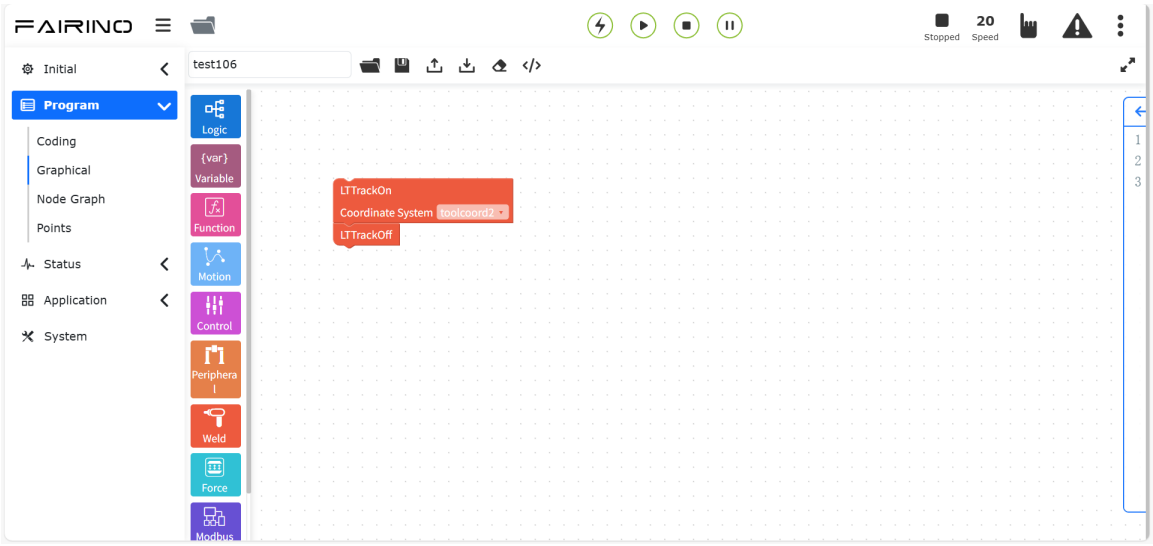


Figure 10.8-12 “Start/stop laser tracking” instruction code block

4. “Laser Sensor Record” Command Node, Parameters:

- Function Selection: Stop Recording / Real-time Tracking / Start Recording / Path Playback
- Function Selection: Delay Time / Delay Distance
- Time: 0 ~ 10000
- Extended Axis Number: 1 ~ 4
- Distance: 0 ~ 10000
- Compensation Sensitivity Coefficient: 0 ~ 1
- Speed: 0 ~ 100

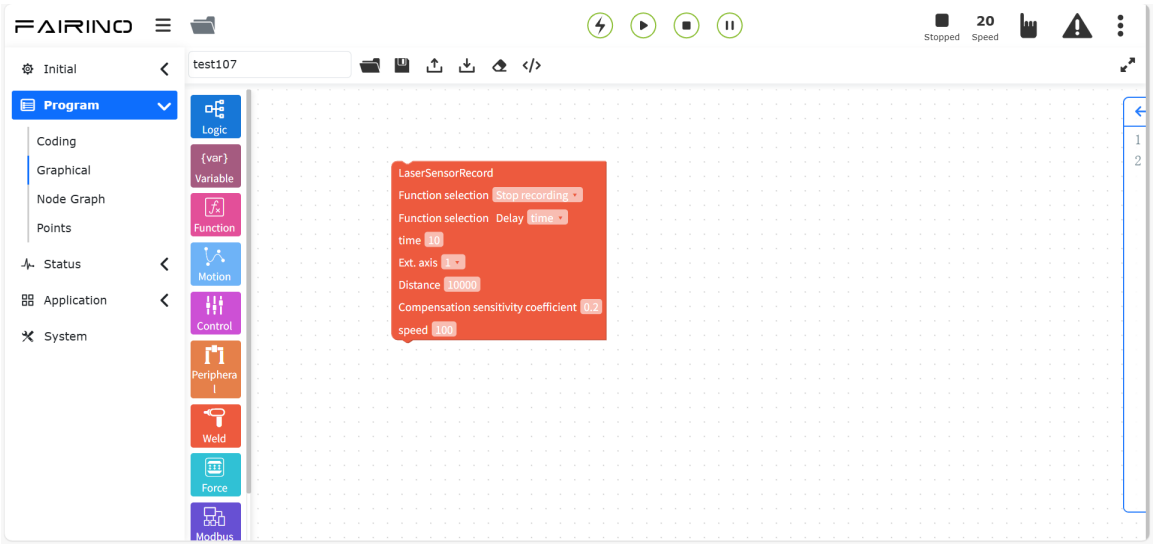


Figure 10.8-13 “Laser Sensor Record” Command Code Block

5. “Sensor Point Movement” instruction node, parameters:

- Coordinate system name: Custom configuration coordinate system
- Movement mode: PTP/Lin

- Debug speed (%): 0 ~ 100
- State reference point: Teaching Point

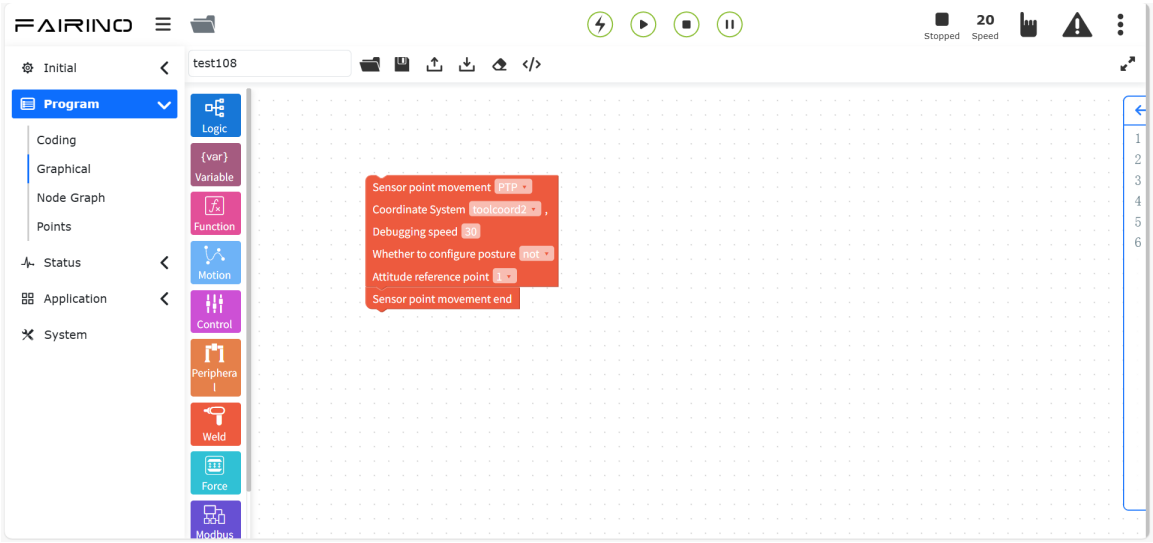


Figure 10.8-14 “Sensor Point Movement” instruction code block

6. “Laser Tracking Reappearance” instruction node, parameters:

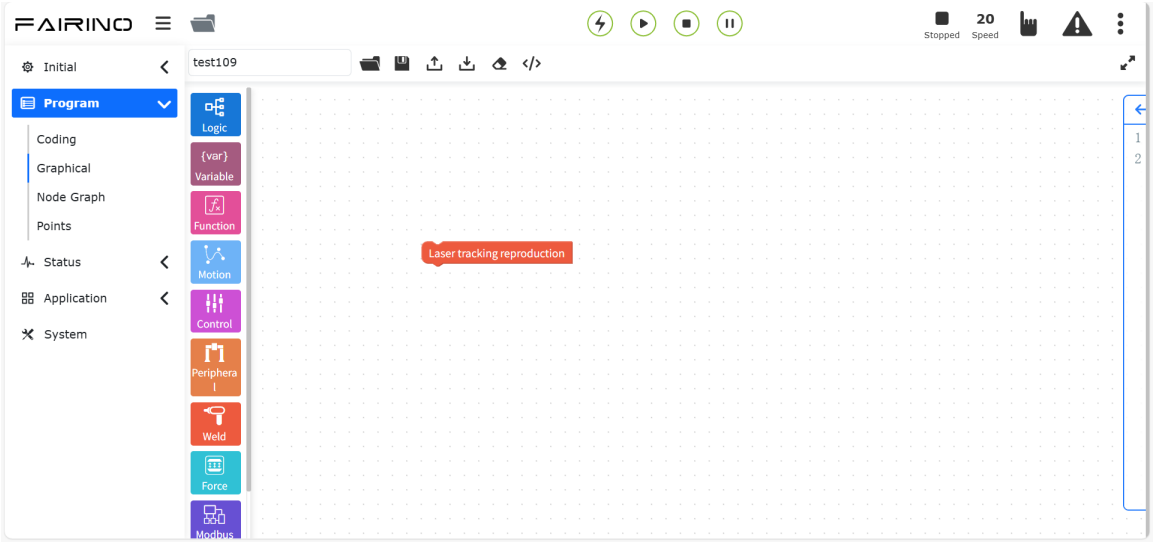


Figure 10.8-15 “Laser Tracking Reappearance” instruction code block

7. “Positioning Start/End” instruction node, parameters:

- Coordinate system name: Custom configuration coordinate system
- Direction: -x/-x/-y/-y/-z/-z/Specify direction
- Direction point: If “Specify direction” is not selected, the parameter is invalid
- Speed (%): 0 ~ 100
- Length (mm): 0 ~ 1000
- Maximum position seeking time (ms): 0 ~ 10000

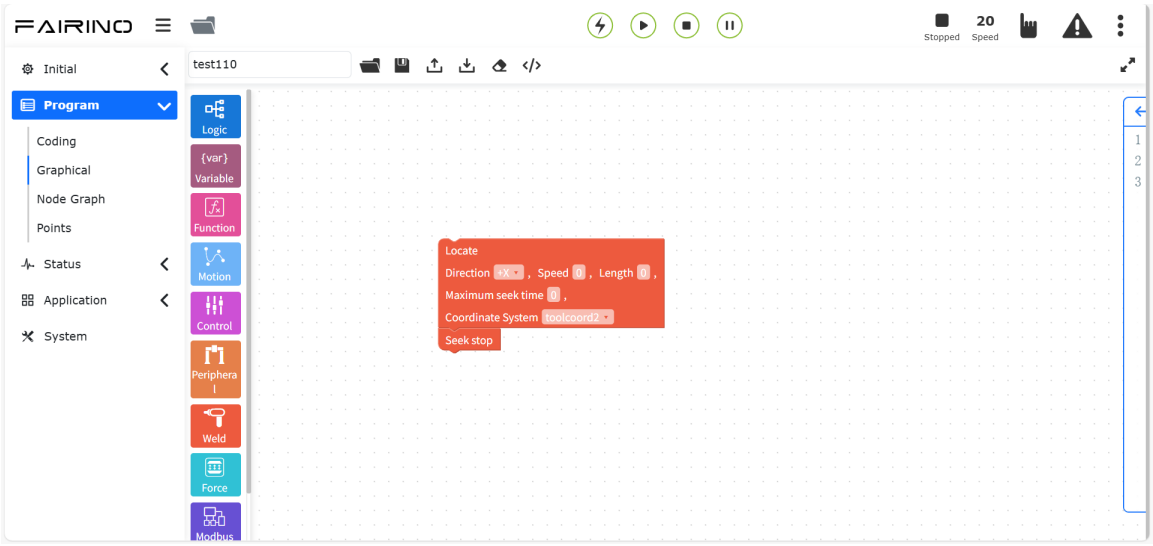


Figure 10.8-16 “Position seeking start/end” instruction code block

1.3.10.8.4 Laser recording instruction

Drag the “Laser recording instruction” code block to enter the graphical editing interface work area.

This instruction realizes the laser tracking and recording start and end point extraction function, so that the robot can automatically move to the starting position. It is suitable for occasions where the movement starts from the outside of the workpiece and the laser tracking and recording is performed. At the same time, the host computer can obtain the information of the start and end points in the recorded data for subsequent movement.

The adjustable laser tracking repetition speed function enables the robot to record at a very fast speed and then reproduce at the normal welding speed, which can improve work efficiency.

1. “Laser Sensor Record” Command Node, Parameters:

- Function Selection: Stop Recording / Real-time Tracking / Start Recording / Path Playback
- Function Selection: Delay Time / Delay Distance
- Time: 0 ~ 10000
- Extended Axis Number: 1 ~ 4
- Distance: 0 ~ 10000
- Compensation Sensitivity Coefficient: 0 ~ 1
- Speed: 0 ~ 100

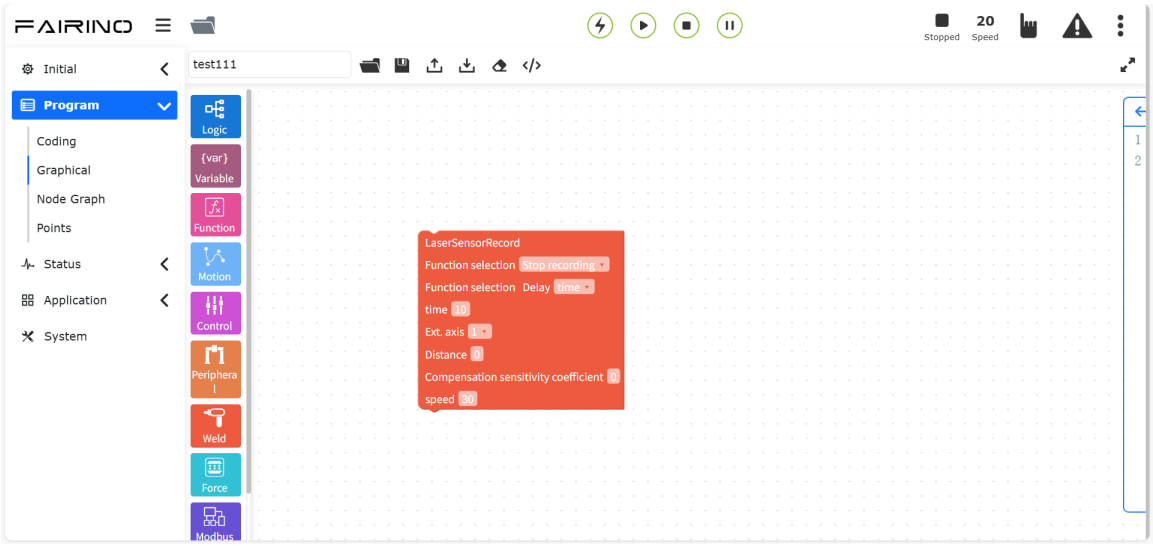


Figure 10.8-17 Weld data record command code block

2. “Get weld start/end point” command node, parameters:

- Movement mode: PTP/LIN
- Speed (%): 0~100, default is 30

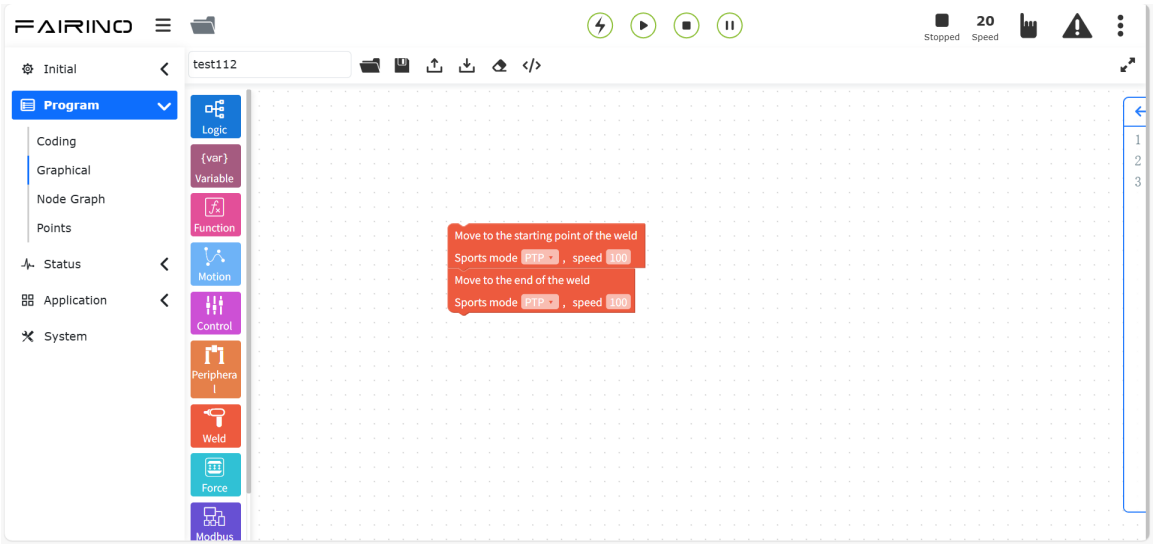


Figure 10.8-18 “Get weld start/end point” command code block

1.3.10.8.5 Welding wire positioning instruction

Drag the “Welding wire positioning instruction” code block to enter the graphical editing interface workspace.

This instruction is generally used in welding scenarios and requires the welding machine to be used in combination with the robot IO and motion instructions. It is divided into positioning start, positioning end, positioning point setting, offset calculation and contact point data writing.

“Wire position finding start/end” command node, parameters:

- Reference position: do not update/update

- Position finding speed: 0~100
- Position finding distance: 0~1000
- Automatic return mark: do not automatically return/automatically return
- Automatic return speed: 0~100
- Automatic return distance: 0~1000
- Position finding method: teaching point positioning/position finding with offset

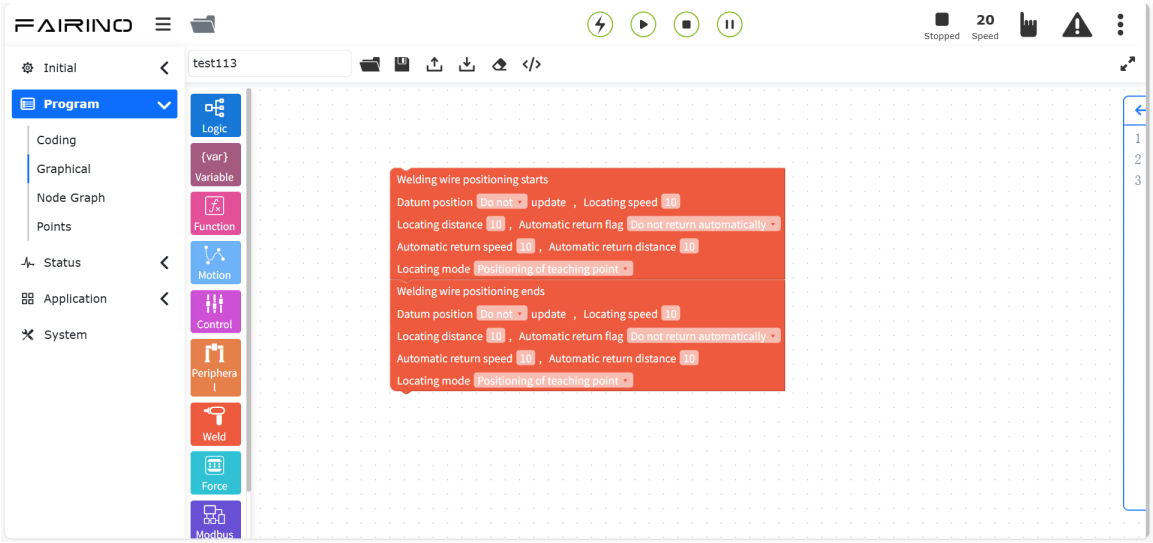


Figure 10.8-19 “Wire position finding start/end” command code block

The position finding point setting adds points according to the weld type and calculation method.

- When the type is fillet weld and the calculation method is 1D (one of xyz), the point to be added is selected from point a and point b;
- When the type is fillet weld and the calculation method is 2D (two of xyz), the point to be added is selected from point a, point b, point e, and point f;
- When the type is fillet weld and the calculation method is 3D (xyz), the point to be added is selected from point a, point b, point c, point d, point e, and point f;
- When the type is fillet weld and the calculation method is 2D- (two of xyz, one of rxryrz), the point to be added is selected from point a, point b, point c, point d, point e, and point f;
- When the type is inner and outer diameter and the calculation method is 2D2D (two of xyz), the point to be added is selected from point a and point b;
- When the type is point and the calculation method is 3D (xyz), the point to be added is selected from point a, point b, point c, point d, point e, and point f;
- When the type is camera and the calculation method is 3D-(xyzrxryrz), the point to be added is selected from point a and point b;
- When the type is surface and the calculation method is 3D-(xyzrxryrz), the point to be added is selected from point a and point b;

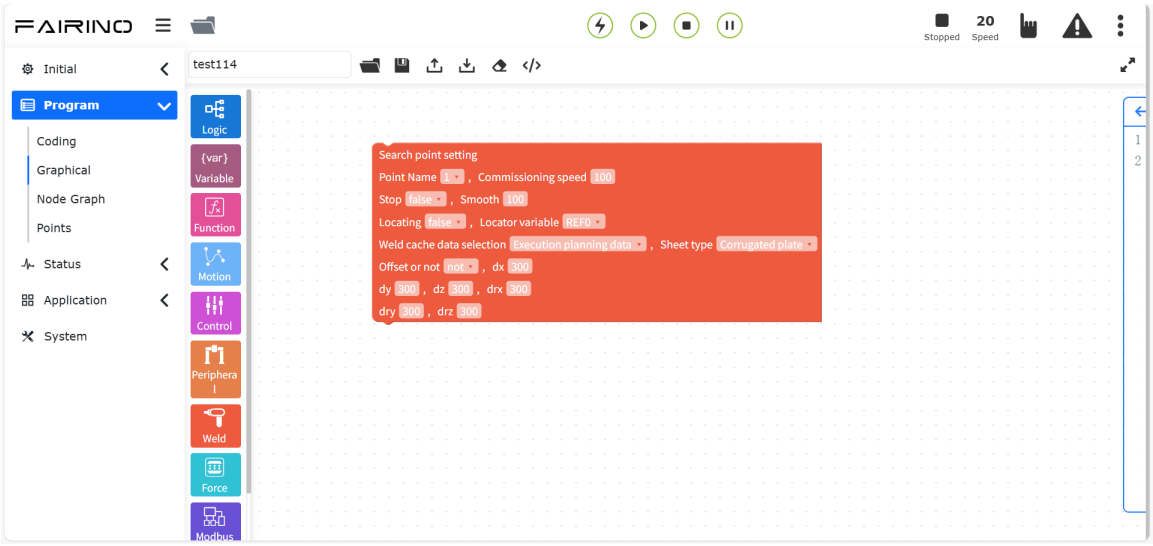


Figure 10.8-20 “Point Finding Setting Guide” instruction code block

Calculate the offset and set the reference point and contact point according to the weld type and calculation method.

- When the type is fillet weld and the calculation method is 1D (one of xyz), set reference point 1 and contact point 1;
- When the type is fillet weld and the calculation method is 2D (two of xyz), set reference point 1, reference point 2, contact point 1, and contact point 2;
- When the type is fillet weld and the calculation method is 3D (xyz), set reference point 1, reference point 2, reference point 3, contact point 1, contact point 2, and contact point 3;
- When the type is fillet weld and the calculation method is 2D- (two of xyz, one of rxryrz), set reference point 1, reference point 2, reference point 3, contact point 1, contact point 2, and contact point 3;
- When the type is inner and outer diameter and the calculation method is 2D2D (two of xyz), set reference point 1, reference point 2, reference point 3, contact point 1, contact point 2, and contact point 3;
- When the type is point and the calculation method is 3D (xyz), set contact point 1 and contact point 2;
- When the type is camera and the calculation method is 3D-(xyzrxryrz), set contact point 1 and contact point 2;
- When the type is surface and the calculation method is 3D-(xyzrxryrz), set contact point 1, contact point 2, contact point 3, contact point 4, contact point 5, and contact point 6;

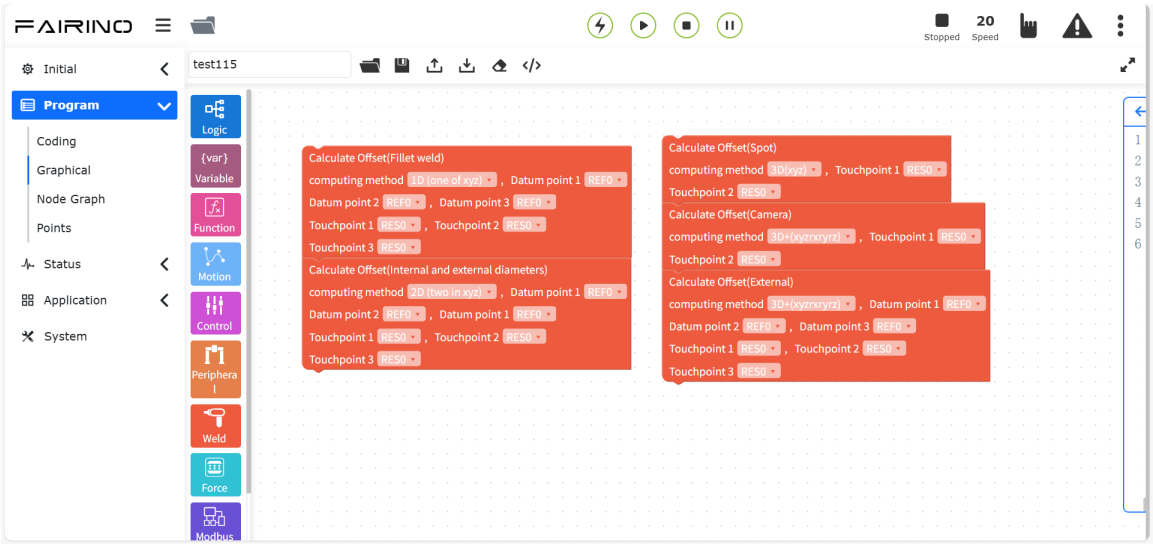


Figure 10.8-21 “Calculate offset” instruction code block

“Write contact point data” instruction node, parameters:

- Contact point name: RES0~99
- Contact point name: data format is {0,0,0,0,0,0};

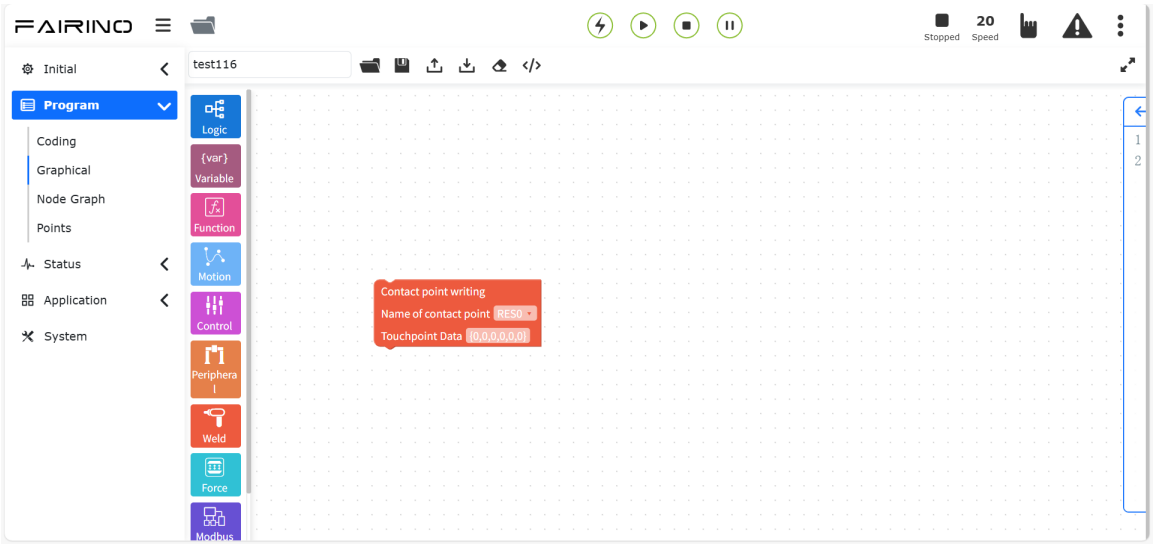


Figure 10.8-22 “Contact point data write” instruction code block

1.3.10.8.6 Arc tracking instruction

Drag the “Arc tracking instruction” code block to enter the graphical editing interface workspace.

This instruction implements robot weld tracking and uses weld deviation detection to compensate the trajectory. Arc sensors can be used to detect weld deviation.

“Arc tracking on/off” command node, parameters:

- Arc tracking lag time (ms): reference value 50
- Deviation compensation: off/on

- Adjustment coefficient: 0 ~ 300
- Compensation time (cyc): 0 ~ 300
- Maximum compensation amount per time (mm): 0 ~ 300
- Total maximum compensation amount (mm): 0 ~ 300
- Upper and lower coordinate system selection: swing
- Upper and lower reference current setting method: feedback/constant
- Upper and lower reference current (A): 0 ~ 300



Chart 10.8-23 Arc tracking instruction code block

1.3.10.8.7 Posture adjustment instruction

Drag the “Posture Adjustment Instruction” code block to enter the graphical editing interface workspace.

This instruction is for the welding tracking adaptive adjustment of the welding gun posture scene. You need to teach the three points PosA, PosB, and PosC first, otherwise you cannot add nodes.

After recording the three corresponding posture points, add the posture adaptive adjustment instruction according to the actual movement direction of the robot. See the robot peripherals chapter for details.

“Start posture adjustment” command node, parameters:

- Plate type: Corrugated board/Corrugated board/Fence board/Corrugated carapace steel
- Movement direction: From left to right/From right to left
- Posture adjustment time (ms): 0 ~ 1000
- First segment length (mm):
- Inflection point type: From top to bottom/From bottom to top
- Second segment length (mm):
- Third segment length (mm):
- Fourth segment length (mm):
- Fifth segment length (mm):

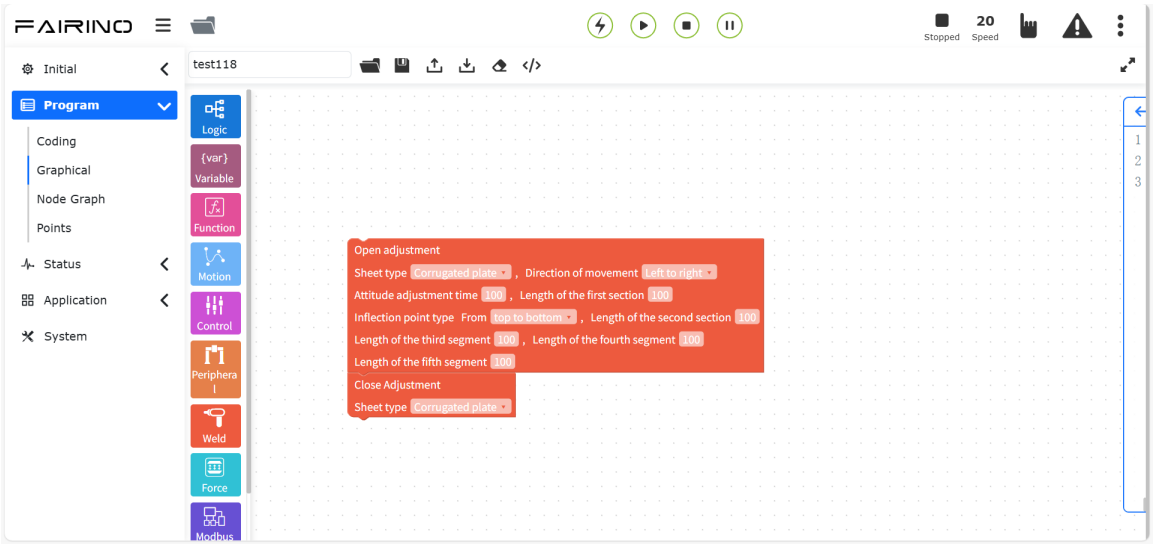


Figure 10.8-24 Posture adjustment command code block

1.3.10.9 Force control graphical programming commands

Force control graphical programming commands include force control set, torque recording and other force control commands.

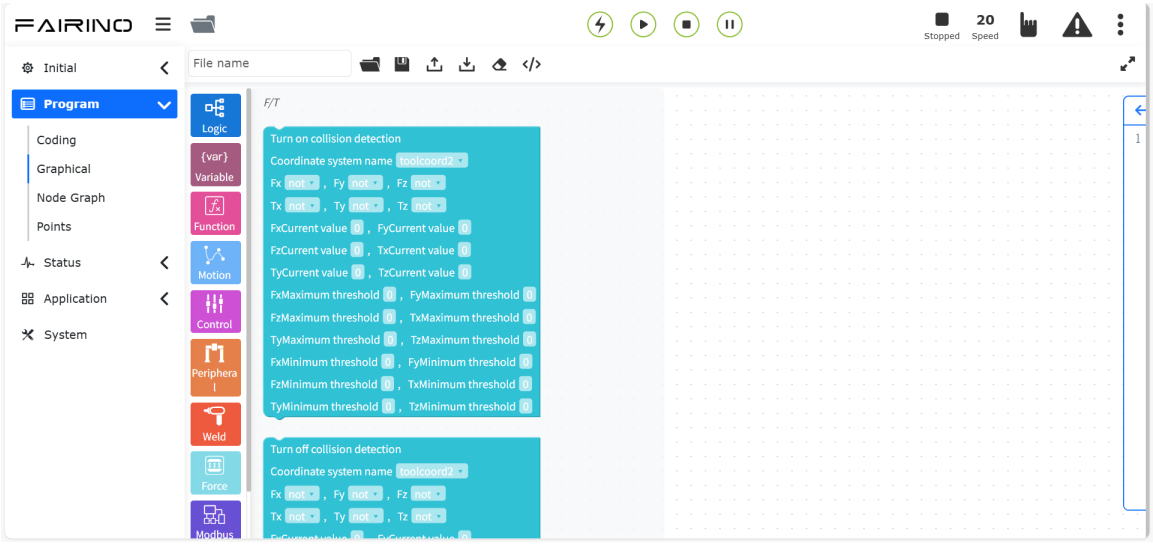


Figure 10.9 Force control graphical programming commands

1.3.10.9.1 Force control commands

Drag the “force control command” code block to enter the graphical editing interface workspace.

The command includes nine commands: FT_Guard (collision detection), FT_Control (constant force control), FT_Compliance (compliance control), FT_Spiral (spiral insertion), FT_Rot (rotation insertion), FT_Lin (linear insertion), FT_FindSurface (surface positioning), FT_CalCenter (center positioning), FT_Click (click force detection), see the robot peripherals chapter for details.

1. “Open/Close Collision Detection” command node, parameters:

- Coordinate system name: Custom configured coordinate system
- Fx-Tx true value: true/false
- Fx-Tx current value: Enter according to actual situation
- Fx-Tx maximum threshold: Enter according to actual situation
- Fx-Tx minimum threshold: Enter according to actual situation

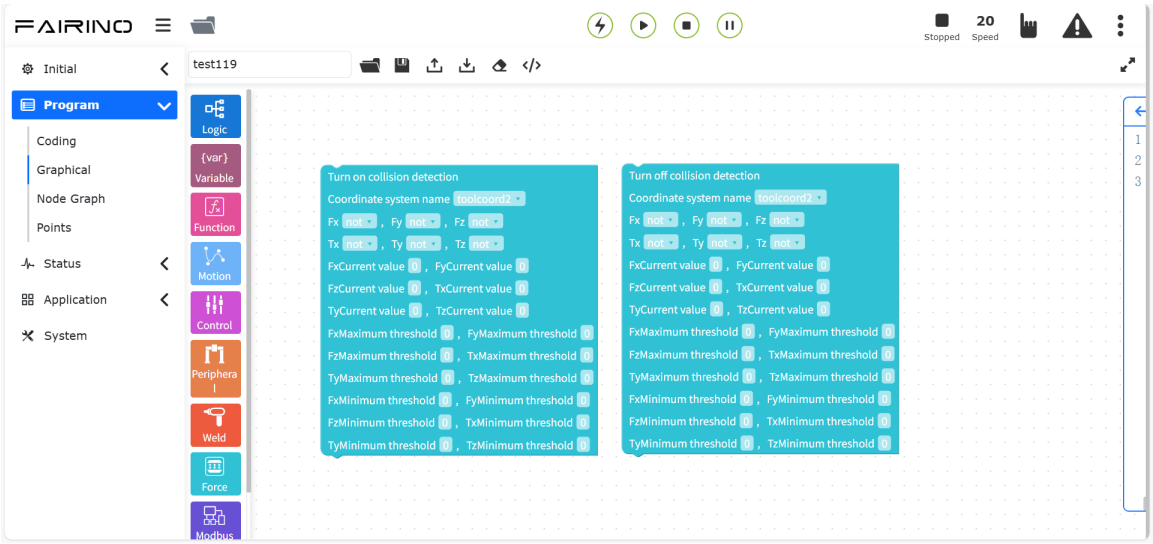


Figure 10.9-1 Open/Close Collision Detection Command Code Block

2. “Open/Close Control” command node, parameters:

- Coordinate system name: Custom configured coordinate system
- Fx-Tx true value: true/false
- Fx-Tx current value: Adjust according to actual situation
- F_P_gain - F_D_gain: Adjust according to actual situation, cannot be 0
- Adaptive start/stop state: Stop/Open
- ILC control start/stop status: stop/training/practical operation
- Maximum adjustment distance (mm): 0 ~ 1000
- Maximum adjustment angle (°): 0 ~ 1000



Figure 10.9-2 Open/close control instruction code block

3. “Open/close smooth control” instruction node, parameters:

- Send position adjustment coefficient: 0 ~ 1
- Smooth opening force threshold (N): 0 ~ 100

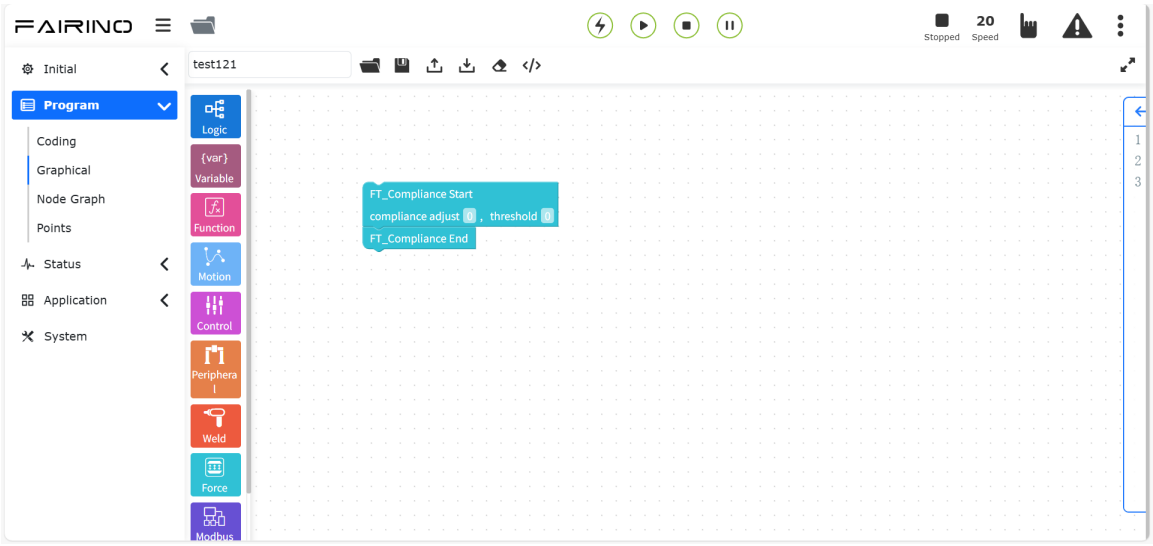


Figure 10.9-3 Open/close smooth control instruction code block

4. “Spiral insertion” instruction node, parameters:

- Coordinate system name: tool coordinate system/base coordinate system
- Radius feed per revolution (mm): 0 ~ 100, reference value: 0.7
- Force or torque threshold (N/Nm): 0 ~ 100, reference value: 50
- Maximum exploration time (ms): 0 ~ 60000, reference value: 60000
- Maximum linear velocity (mm/s): 0 ~ 100, reference value: 5

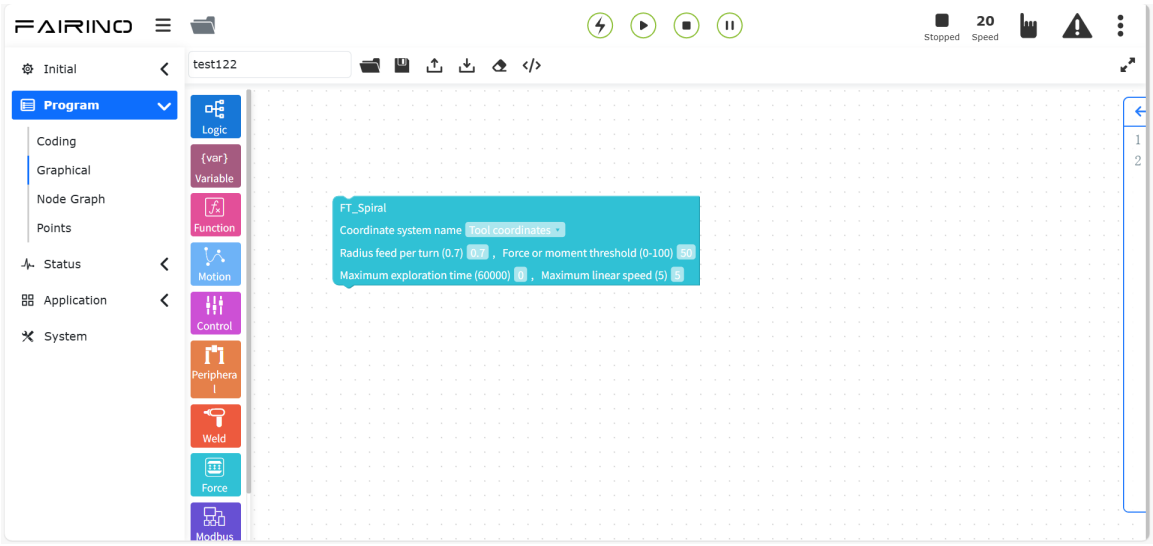


Figure 10.9-4 Helical insertion instruction code block

5. “Rotation insertion” instruction node, parameters:

- Coordinate system name: tool coordinate system/base coordinate system
- Rotation angular velocity (°/s): 0 ~ 100, reference value: 0.7
- Trigger force or end torque (N/Nm): 0 ~ 100, reference value: 50
- Maximum rotation angle (°): 0 ~ 100, reference value: 5
- Direction of force: direction z/direction mz
- Maximum rotation angular acceleration (°/s^2): 0 ~ 100
- Insertion direction: positive/negative

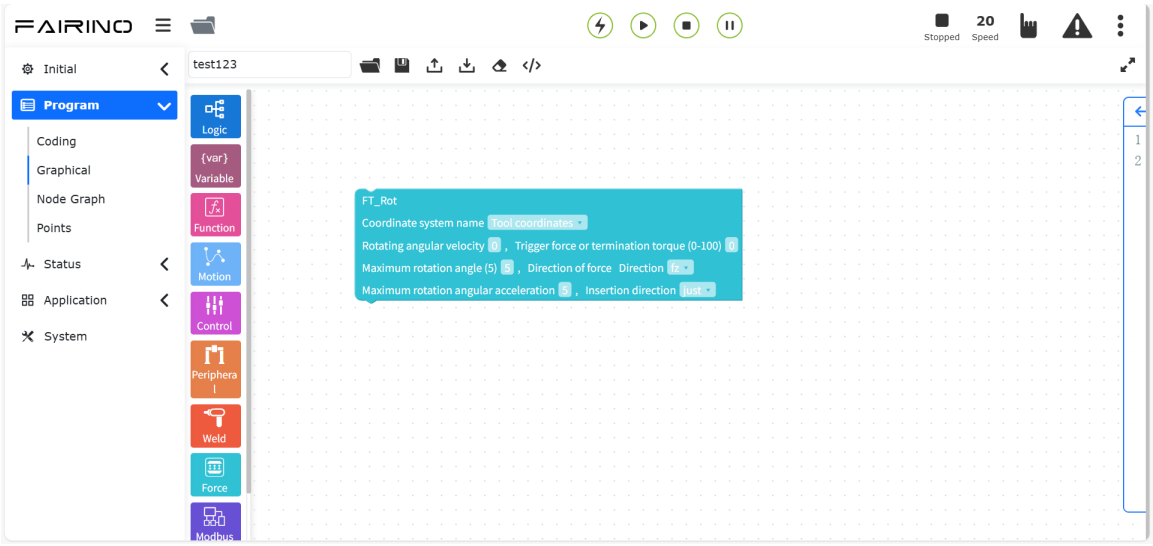


Figure 10.9-5 Rotational insertion instruction code block

6. “Linear insertion” instruction node, parameters:

- Coordinate system name: tool coordinate system/base coordinate system
- Action termination force threshold (N): 0 ~ 100

- Linear speed (mm/s): 0 ~ 100, reference value: 1
- Linear acceleration (°/s^2): 0 ~ 100
- Maximum insertion distance (mm): 0 ~ 100
- Insertion direction: positive/negative

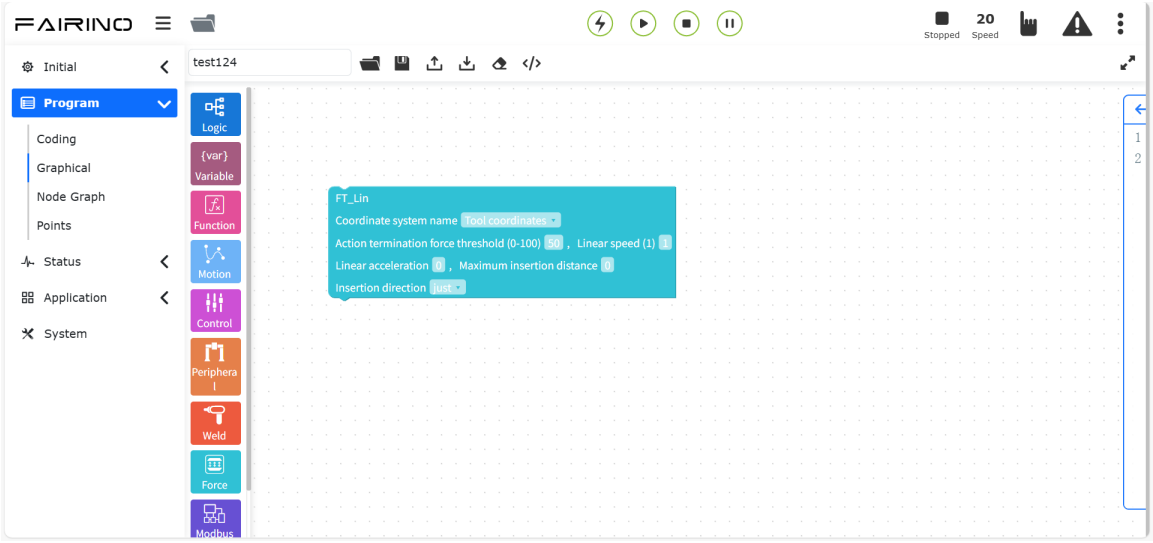


Figure 10.9-6 Linear insertion instruction code block

7. “Surface positioning” instruction node, parameters:

- Coordinate system name: tool coordinate system/base coordinate
- Moving direction: positive/negative
- Moving axis: X/Y/Z
- Exploration linear speed (mm/s): 0 ~ 100
- Exploration acceleration (mm/s^2): 0 ~ 100
- Maximum exploration distance (mm): 0 ~ 100
- Action termination force threshold (N): 0 ~ 100

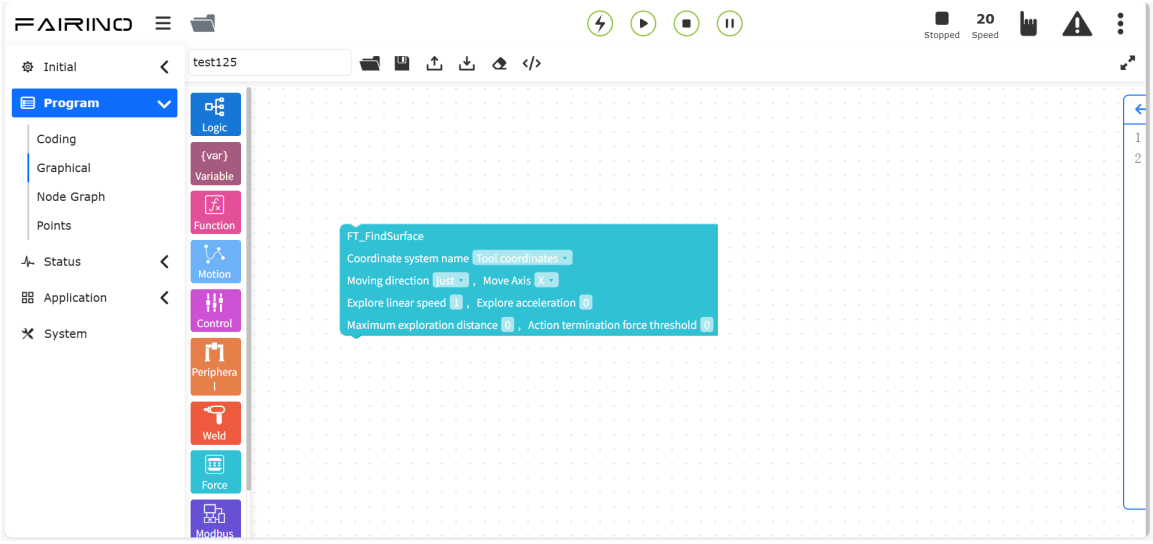


Figure 10.9-7 Surface positioning instruction code block

8. “Middle plane start/end calculation” instruction node:

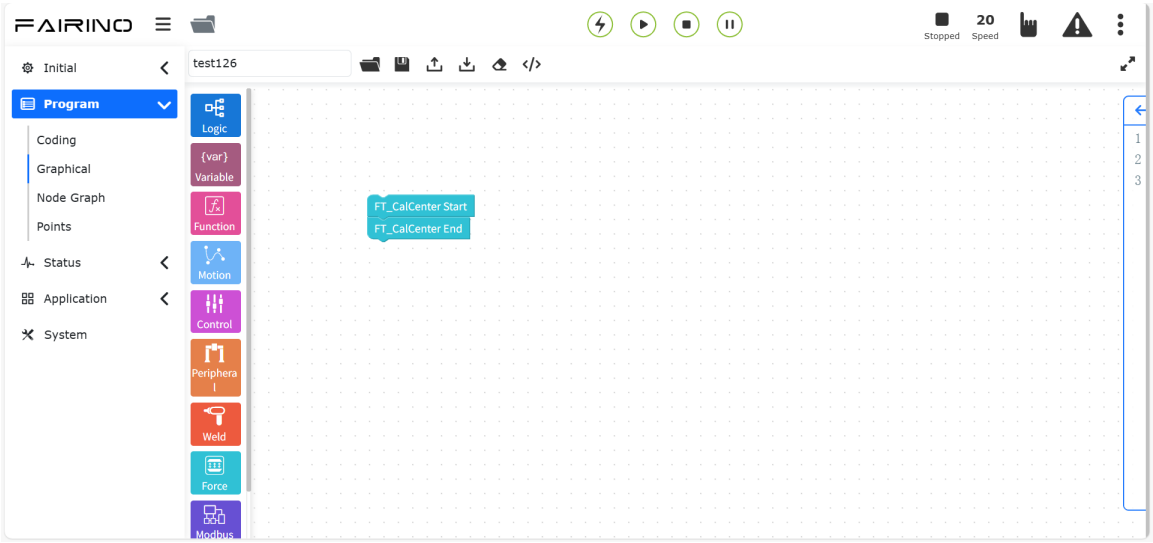


Figure 10.9-8 Middle plane start/end calculation instruction code block

1.3.10.9.2 Torque recording instruction

Drag the “Torque recording instruction” code block to enter the graphical editing interface workspace.

This instruction is a torque recording instruction, which includes three instructions: “Torque Recording Start/”Torque Recording Stop” and “Torque Recording Reset”.

Realize the torque real-time recording collision detection function.

Click the “Torque Recording Start” button to continuously record the collision situation during the operation of the motion instruction. The recorded real-time torque is used as the theoretical value of the collision detection judgment to reduce the probability of false alarms.

When the set threshold range is exceeded, the duration of the collision detection is recorded.

Click the “Torque Recording Stop” button to stop recording. Click “Torque Recording Reset” to restore the status to the default state.

1. “Torque Recording Start” instruction node, parameters:

- Smoothing selection: unsmoothed (original data)/smoothed (smoothed data)
- Joint negative threshold (Nm): -100 ~ 0
- Joint positive threshold (Nm): 0 ~ 100
- Joint continuous collision detection time (ms): 0 ~ 1000

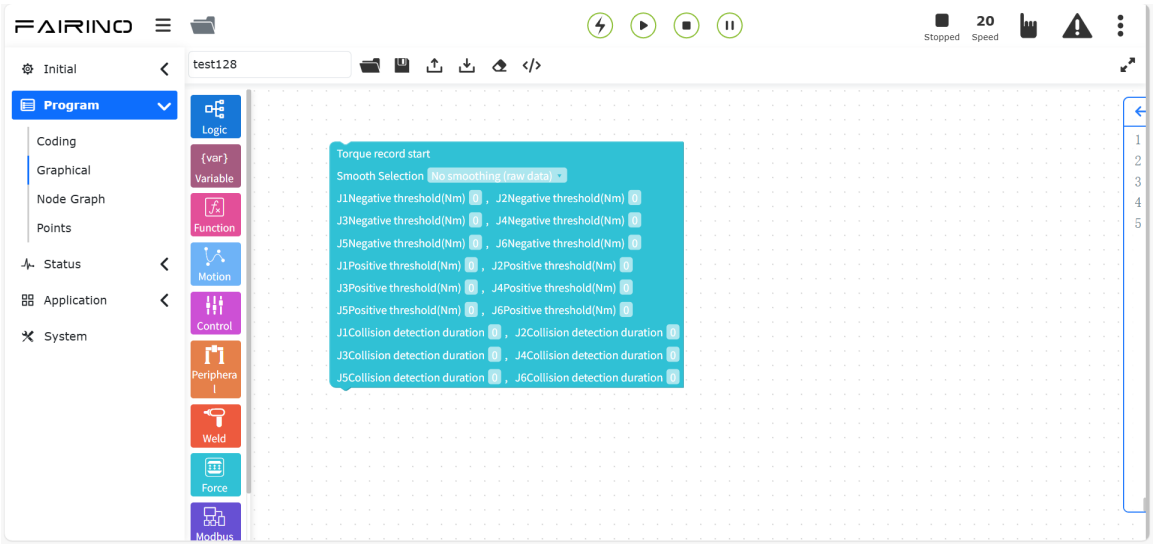


Chart 10.9-9 Torque record start instruction code block

2. “Torque record end” instruction node

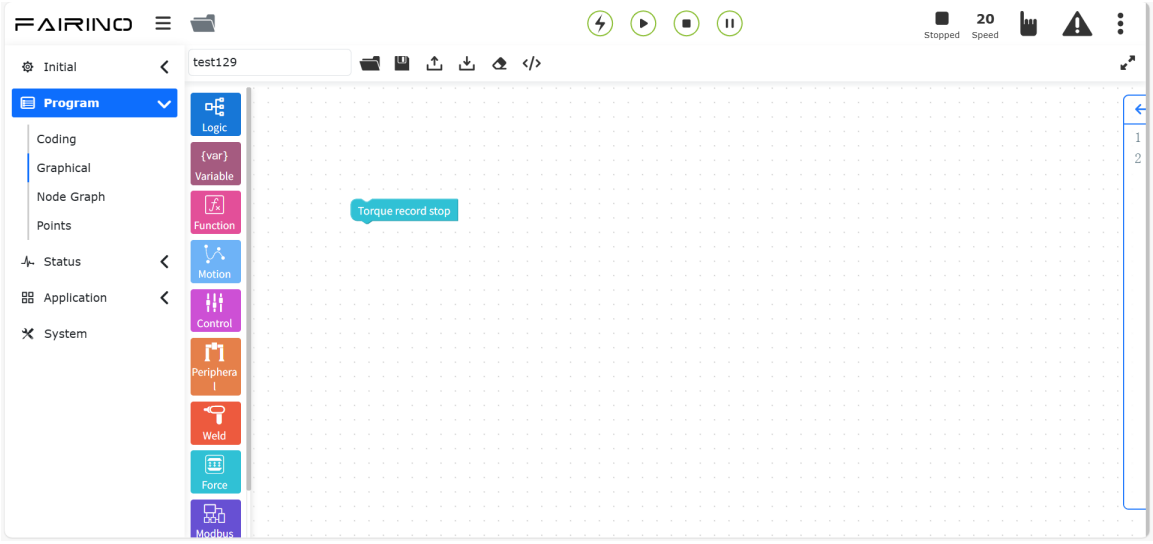


Figure 10.9-10 Torque record end instruction code block

3. “Torque record reset” instruction node

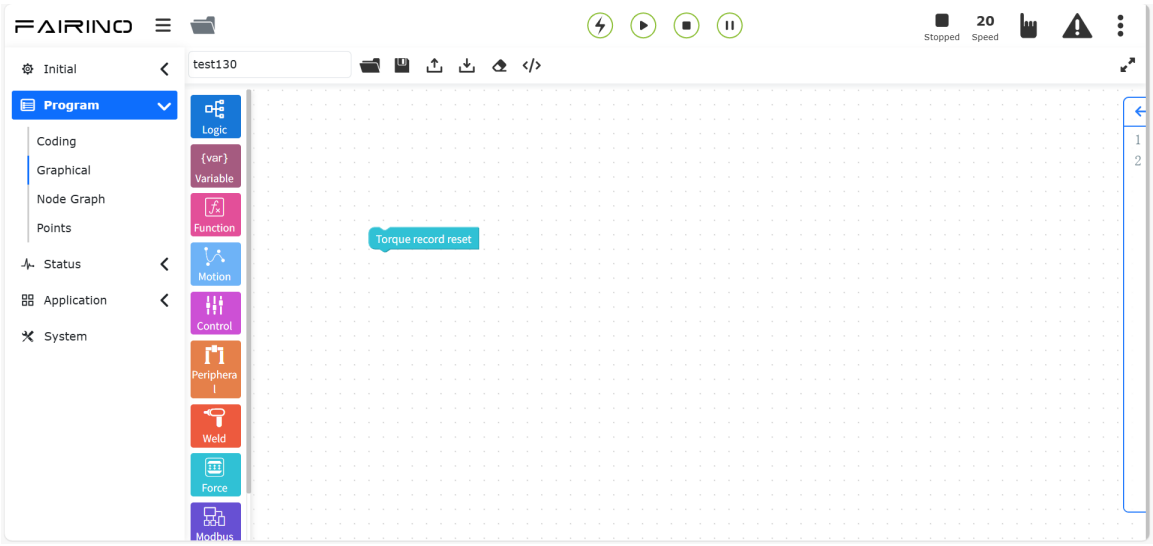


Figure 10.9-11 Torque record reset instruction code block

1.3.10.10 Communication graphic programming commands

Communication graphic programming commands include modbus master station settings (client), modbus slave station settings, register reading and other communication commands.

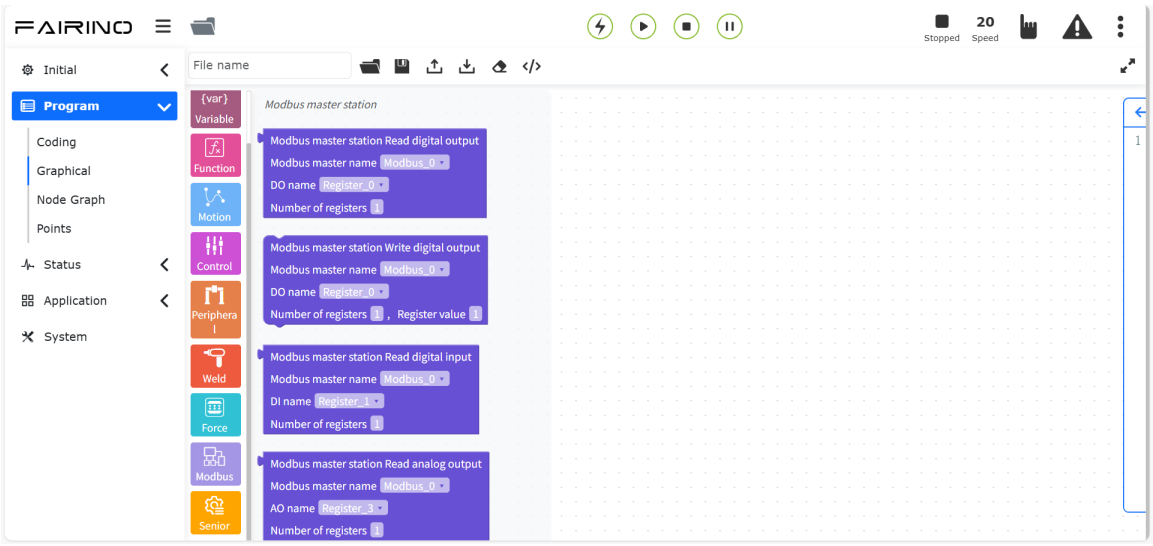


Figure 10.10 Communication Graphical Programming Commands

1.3.10.10.1 Modbus Commands

Drag the “Modbus Command” code block to enter the graphical editing interface workspace.

This command function is a bus function based on the ModbusTCP protocol. Users can control the robot to communicate with the ModbusTCP client or server (master and slave communication) through relevant commands, and read and write coils, discrete quantities, and registers. For more ModbusTCP operation functions, please contact us for consultation.

Before using the modbus node function, you need to configure the master station, slave station, and DI, DO, AI, and AO names in the teaching program ModbusTCP configuration.

1. Master digital output settings, parameters:

- Modbus master station name: Configure according to actual conditions
- DO name: Configure according to actual conditions
- Number of registers: Integer type 0 ~ 128
- Register value: Determined according to the number of registers, multiple values can be entered. For example, if the quantity is 3, the values are 1, 0, 1

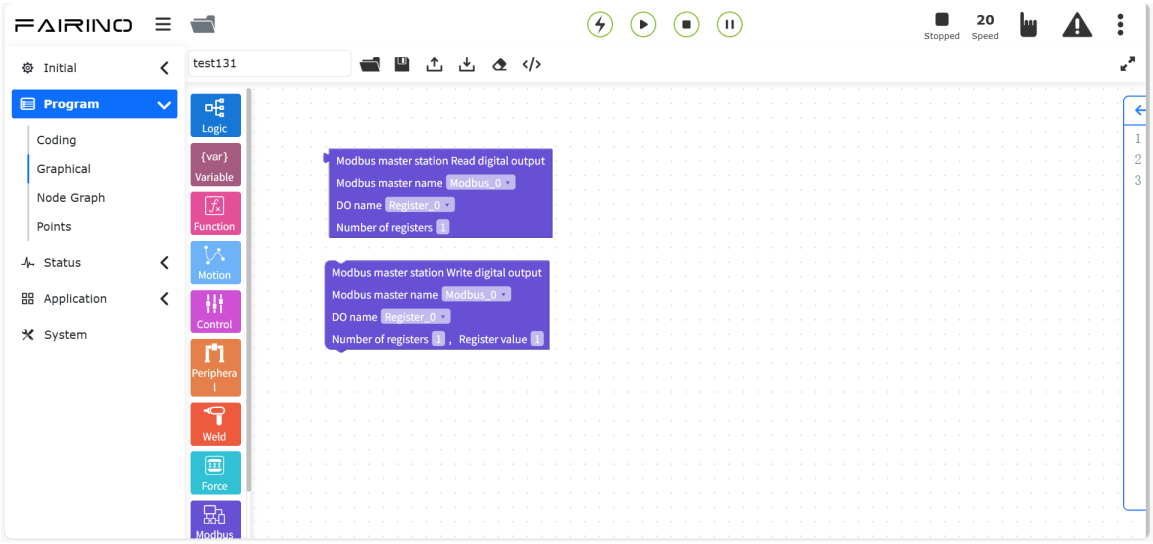


Figure 10.10-1 Master station “read/write digital output” instruction code block

2. Master station digital input settings, parameters:

- Modbus master station name: configure according to actual situation
- DI name: configure according to actual situation
- Number of registers: integer type 0 ~ 128

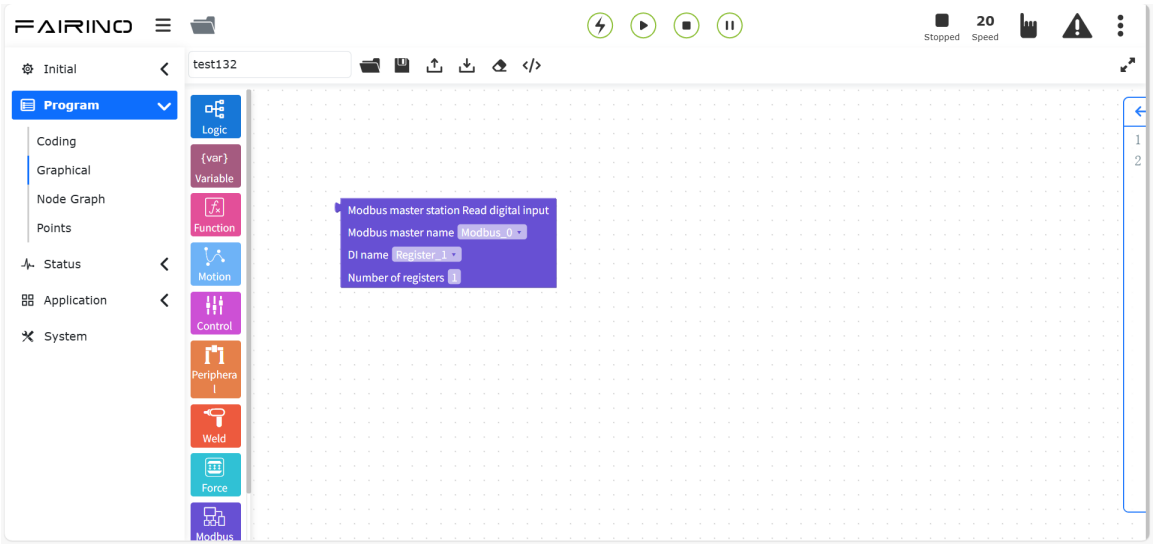


Figure 10.10-2 Master station “read digital input” instruction code block

3. Master station analog output settings, parameters:

- Modbus master station name: configure according to actual situation
- AO name: configure according to actual situation
- Number of registers: integer type 0 ~ 128
- Register value: Determined by the number of registers, multiple values can be entered. For example, if the quantity is 3, the values are 1, 0, 1

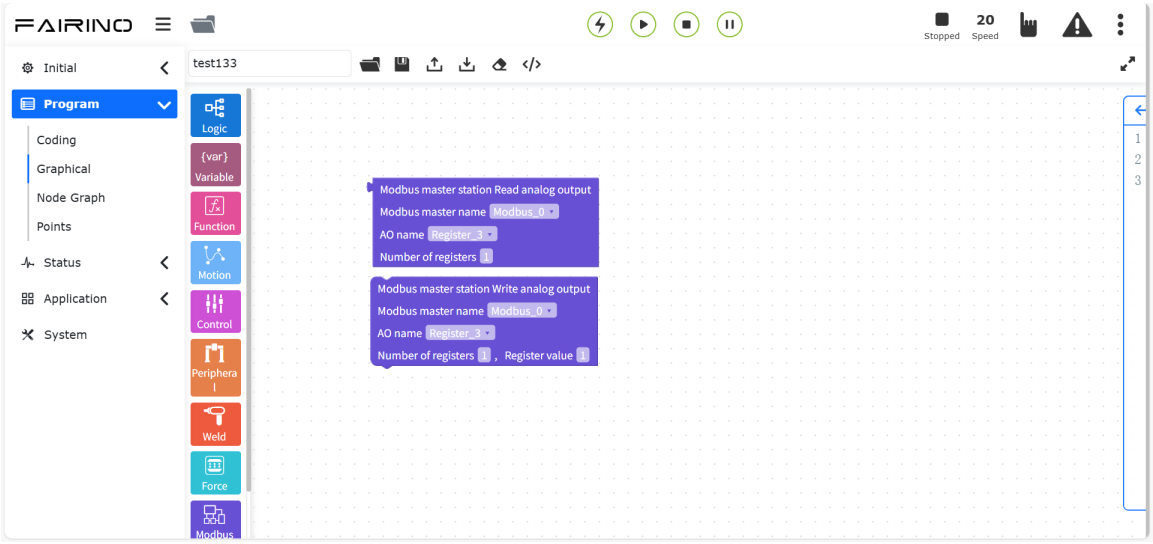


Figure 10.10-3 Master station “read/write analog output” instruction code block

4. Master station analog input setting, parameters:

- Modbus master station name: configure according to actual situation
- AI name: configure according to actual situation
- Number of registers: integer type 0 ~ 128

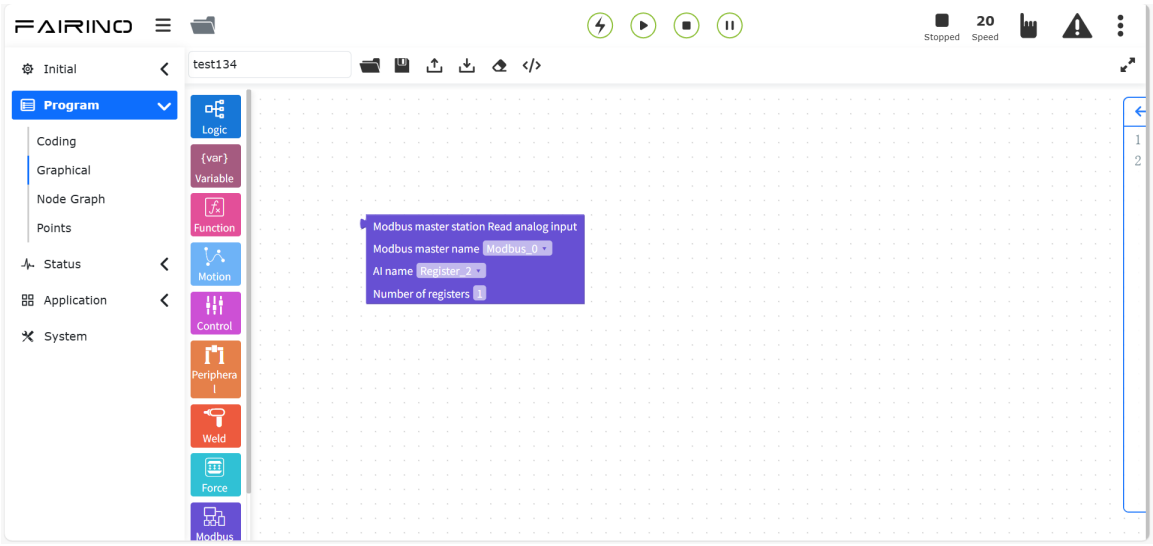


Figure 10.10-4 Master station “read analog input” instruction code block

5. Master station wait digital input setting, parameters:

- Modbus master station name: configure according to actual situation
- DI name: configure according to actual situation
- Waiting state: true/false
- Timeout time (ms): integer type 0 ~ 128

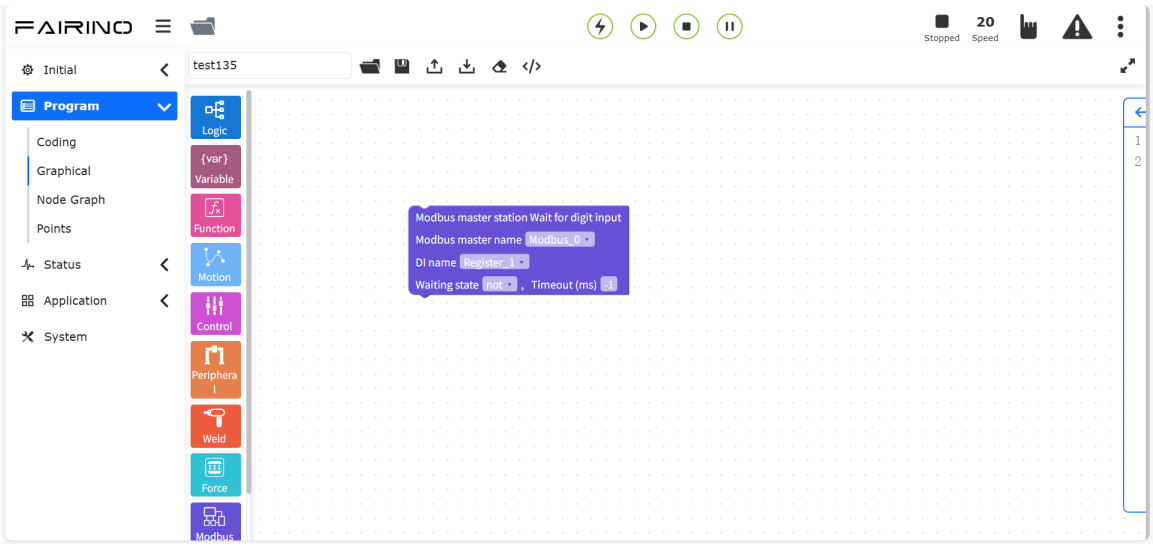


Figure 10.10-5 Master station “wait for digital input” instruction code block

6. Master station wait analog word input setting, parameters:

- Modbus master station name: Configure according to actual situation
- AI name: Configure according to actual situation
- Waiting state: greater than/less than
- Number of registers: Integer type 0 ~ 128
- Register value: Determined by the number of registers, multiple values can be entered.

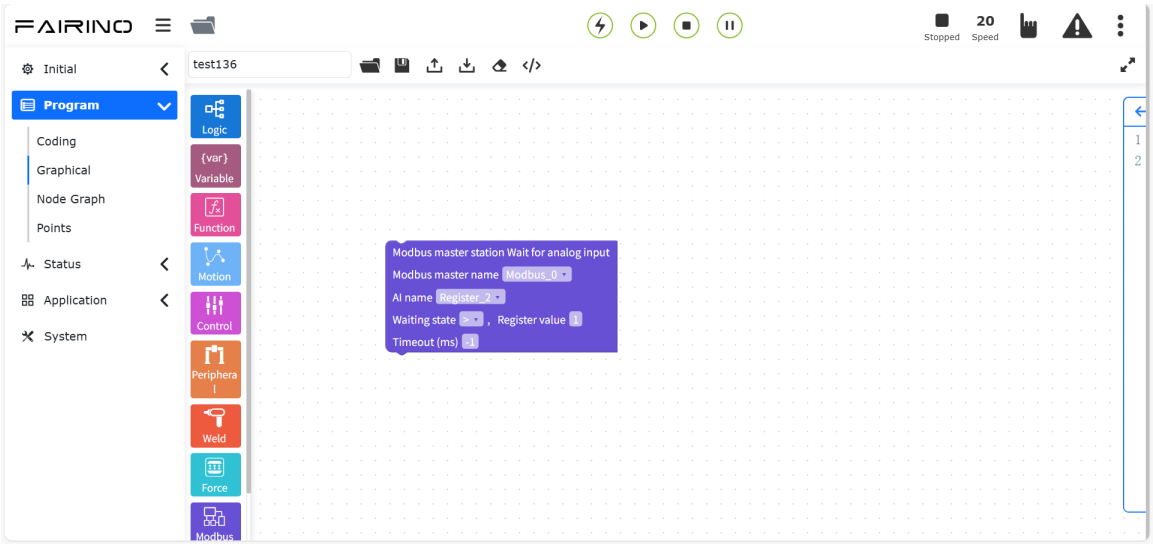


Figure 10.10-6 Master station "Wait for analog input instruction code block"

7. Slave station digital output settings, parameters:

- DO name: Configure according to actual situation
- Number of registers: Integer type 0 ~ 128
- Register value: Determined by the number of registers, multiple values can be entered. For example, the number is 3, the value is 1,0,1

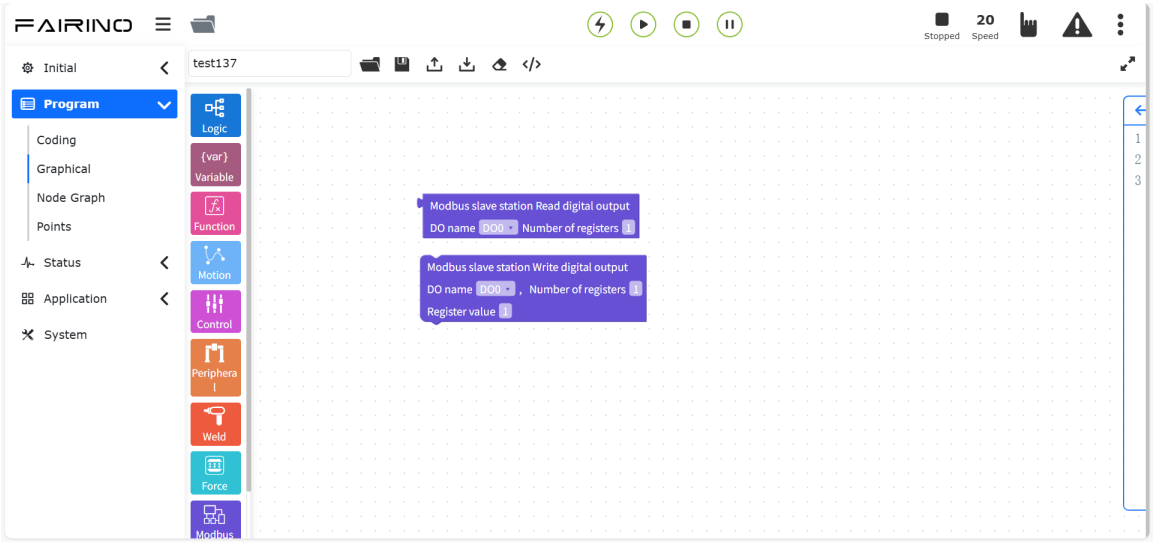


Figure 10.10-7 Slave station "Read/write digital output" instruction code block

8. Slave station digital input settings, parameters:

- DI name: Configure according to actual situation
- Number of registers: Integer type 0 ~ 128

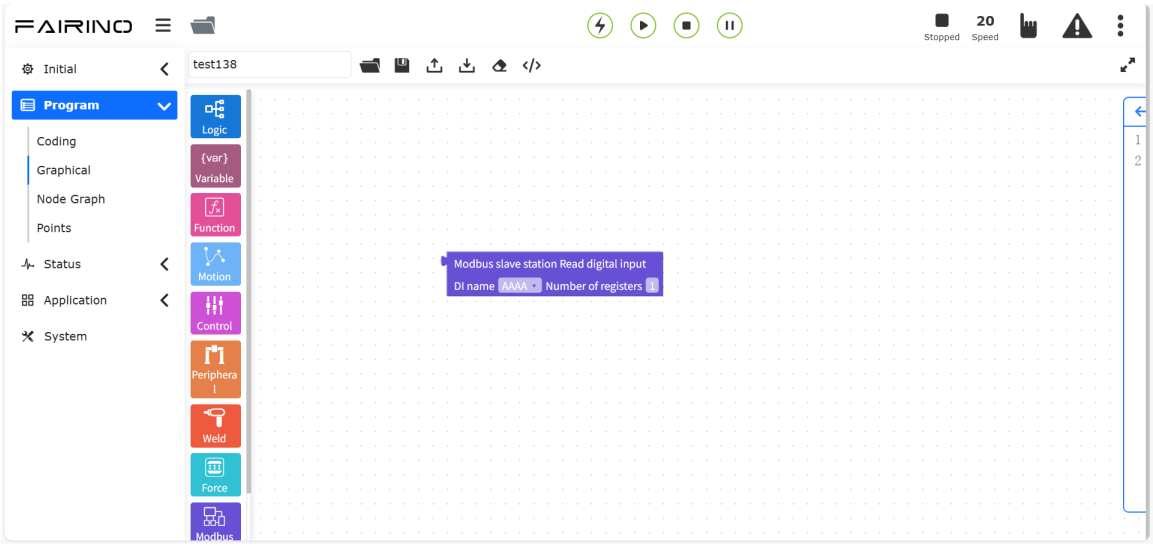


Figure 10.10-8 Slave "Read Digital Input" Instruction Code Block

9. Slave Analog Output Settings, Parameters:

- AO Name: Configure according to actual situation
- Number of Registers: Integer 0 ~ 128
- Register Value: Determined by the number of registers, multiple values can be entered. For example, if the number is 3, the values are 1, 0, 1

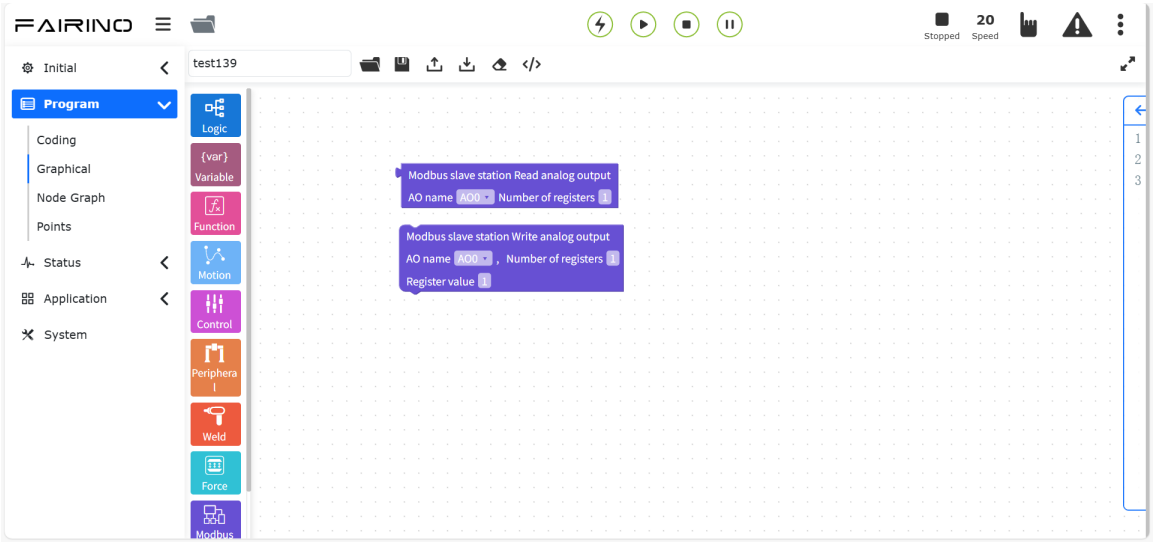


Figure 10.10-9 Slave "Read/Write Analog Output" Instruction Code Block

10. Slave Analog Input Settings, Parameters:

- AI Name: Configure according to actual situation
- Number of Registers: Integer 0 ~ 128

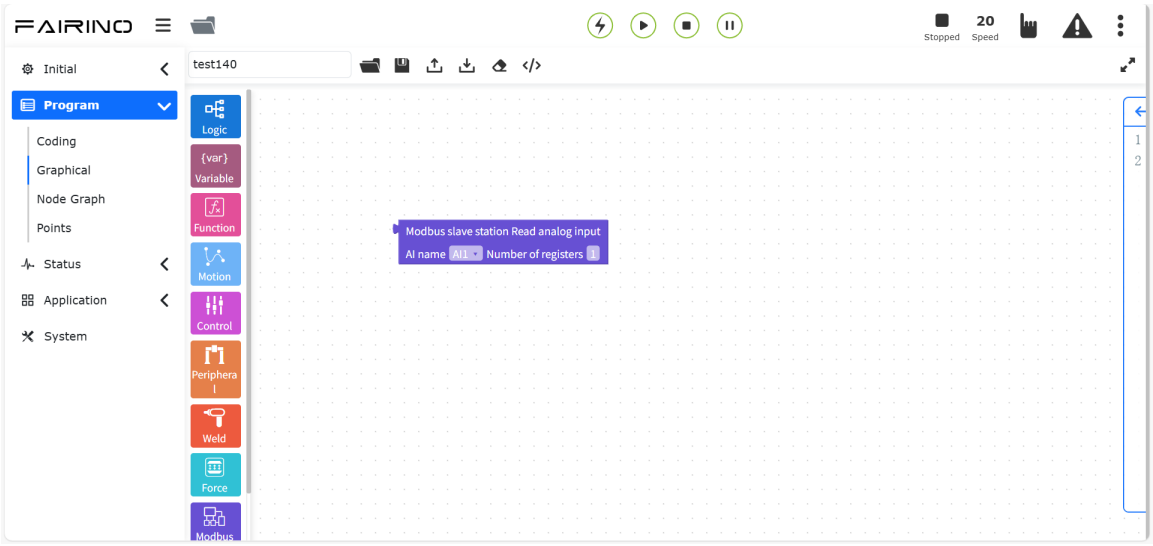


Figure 10.10-10 Slave station “read analog input” instruction code block

11. Slave station wait digital input setting, parameters:
    - DI name: configure according to actual situation
    - Waiting state: true/false
    - Timeout (ms): integer type

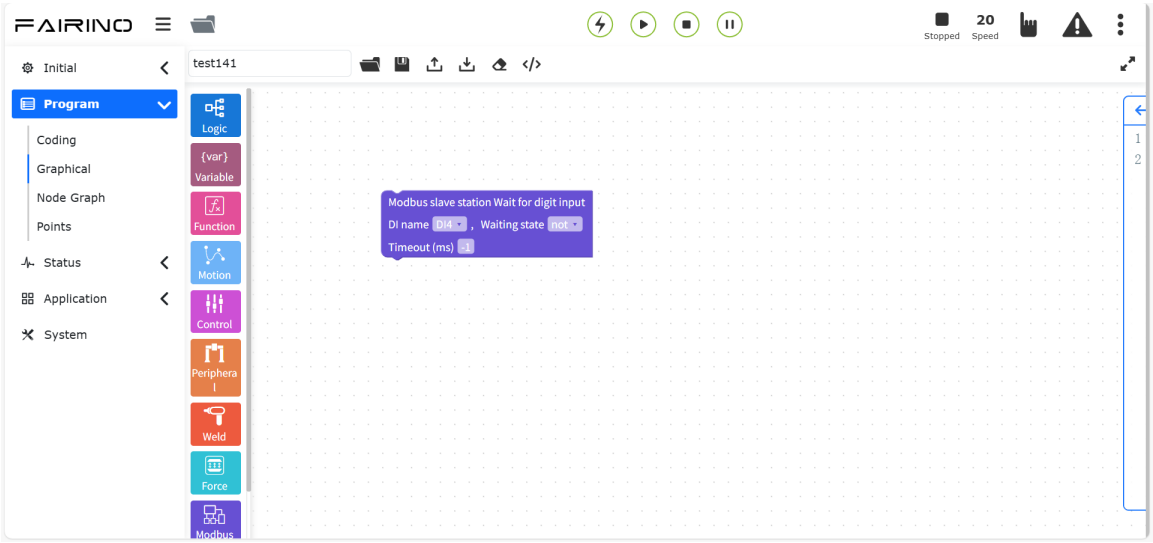


Chart 10.10-11 Slave station “wait digital input” instruction code block

12. Slave station wait analog input setting, parameters:
    - AI name: configure according to actual situation
    - Waiting state: greater than/less than
    - Number of registers: integer type 0 ~ 128
    - Register value: determined according to the number of registers, multiple values can be entered.

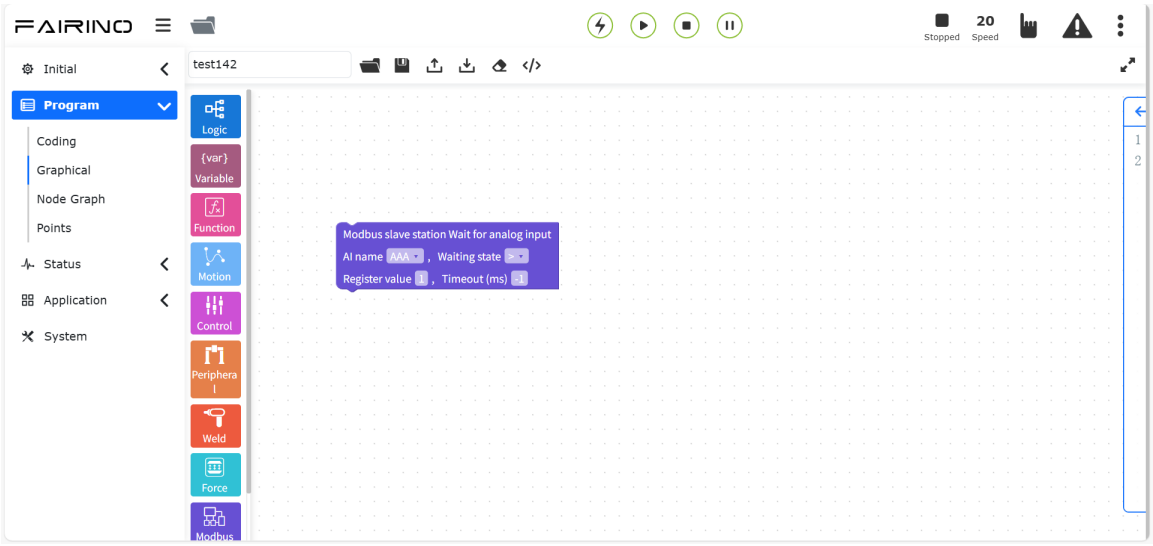


Figure 10.10-12 Slave “Wait for analog input” instruction code block

13. Read register instruction, parameters:

- Function code: 0x01-coil/0x02-discrete/0x03-holding register/0x04-input register
- Register, coil, discrete address: enter according to actual situation
- Register, coil, discrete quantity number: 0 ~ 255
- Address: enter according to actual situation
- Whether to apply thread: No/Yes

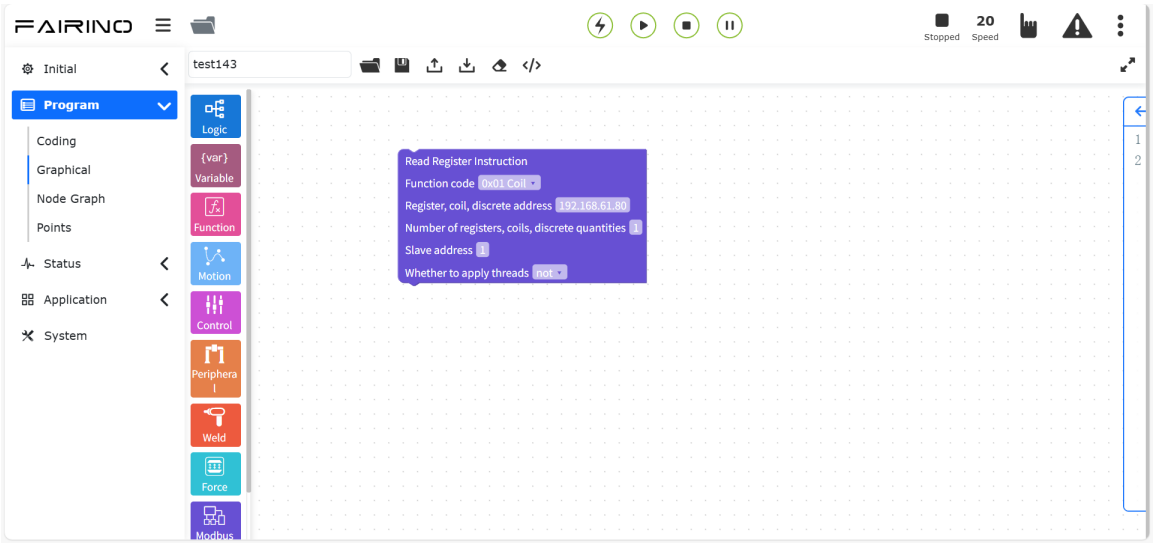


Figure 10.10-13 “Read register” instruction code block

14. Read register data instruction, parameters:

- Register, coil, discrete quantity number: 0 ~ 255
- Whether to apply thread: No/Yes

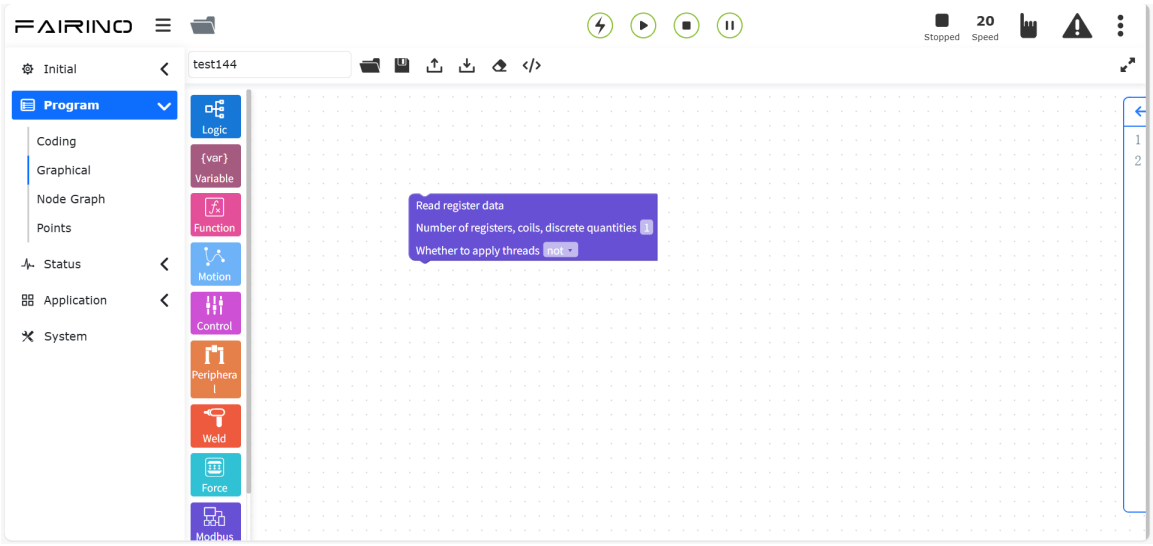


Figure 10.10-14 “Read register data” instruction code block

15. Write register instruction, parameters:

- Function code: 0x01-coil/0x02-discrete/0x03-holding register/0x04-input register
- Register, coil address: enter according to actual situation
- Number of registers, coils: 0 ~ 255
- Byte array: enter according to actual situation
- Address: enter according to actual situation
- Whether to apply thread: No/Yes

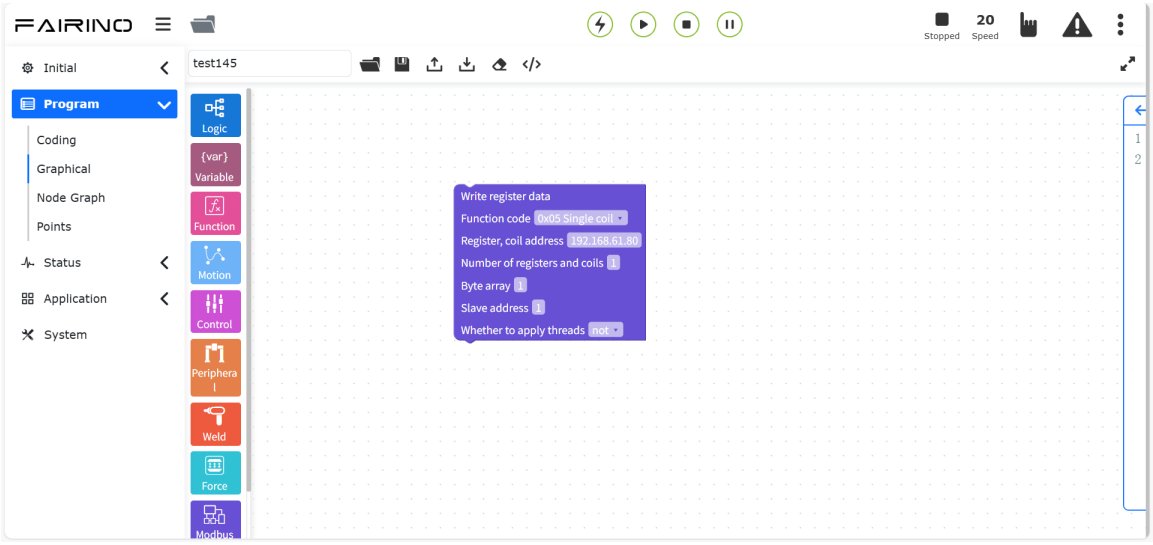


Figure 10.10-15 “Write register instruction” instruction code block

1.3.10.11 Advanced Graphical Programming Commands

Advanced graphical programming commands include advanced commands such as dofile calling subroutines, auxiliary threads, and folding instructions.

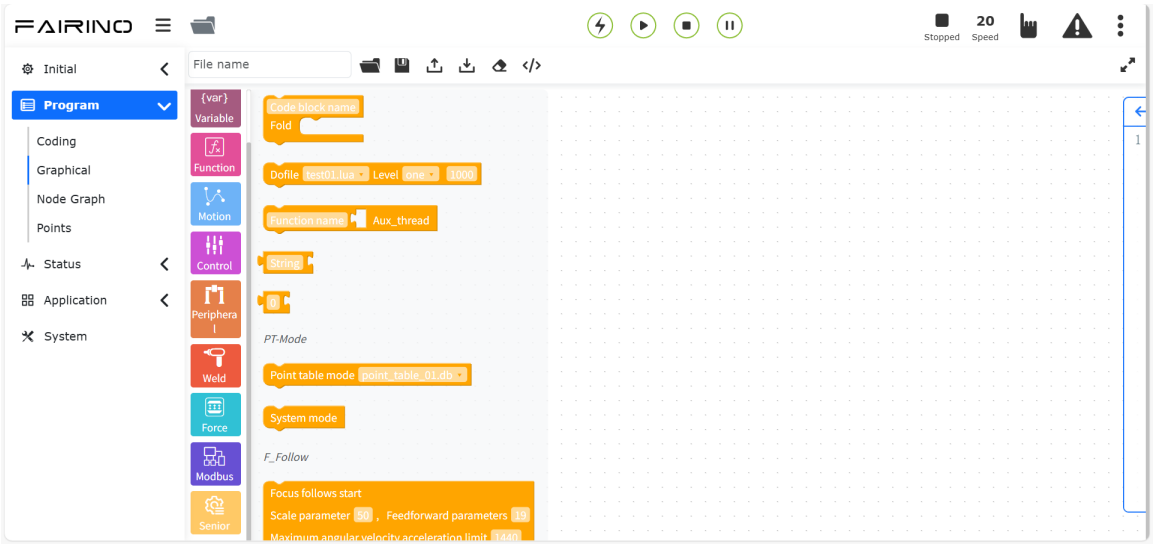


Figure 10.11 Advanced Graphical Programming Commands

1.3.10.11.1 Folding Instructions

Drag the “Folding Instructions” code block to enter the graphical editing interface workspace.

This instruction provides multi-line code block folding display, which is convenient for users to read code blocks.

“Fold” instruction node, parameters:

- Code block name: name the fold code block

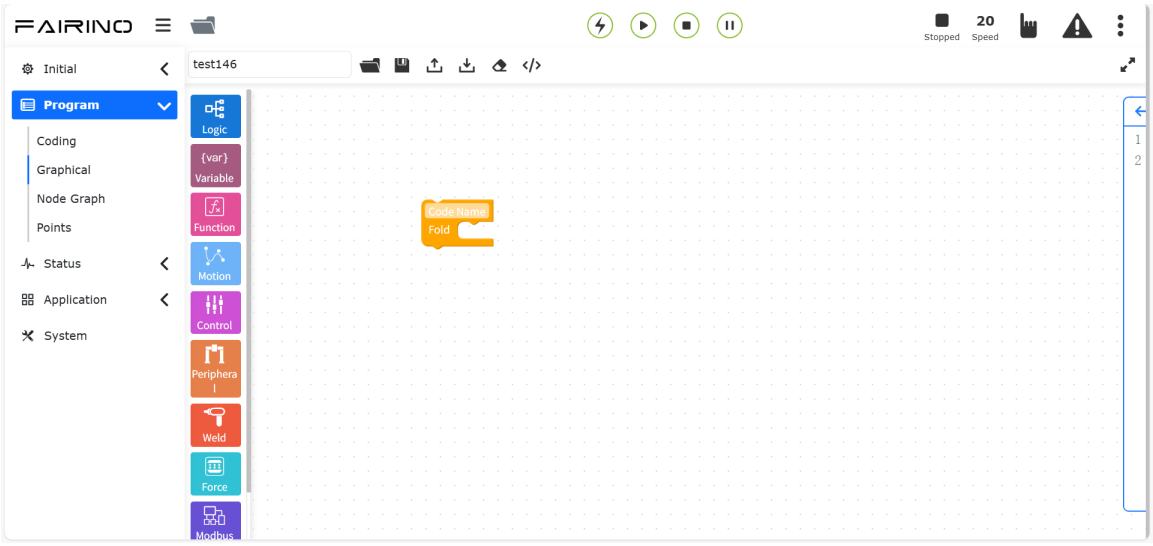


Figure 10.11-1 Fold instruction code block

1.3.10.11.2 Call subroutine instruction

Drag the “Call subroutine instruction” code block to enter the graphical editing interface workspace.

This instruction is the call subroutine instruction. Insert this instruction in the program. When the program executes this instruction, the robot will be in a paused state. If you want to continue running, click the “Pause/Resume” button in the control area.

“Call subroutine” instruction node, parameters:

- dofile file: create the generated file name
- Which layer to call: first layer/second layer
- id number: corresponding position id of the layer

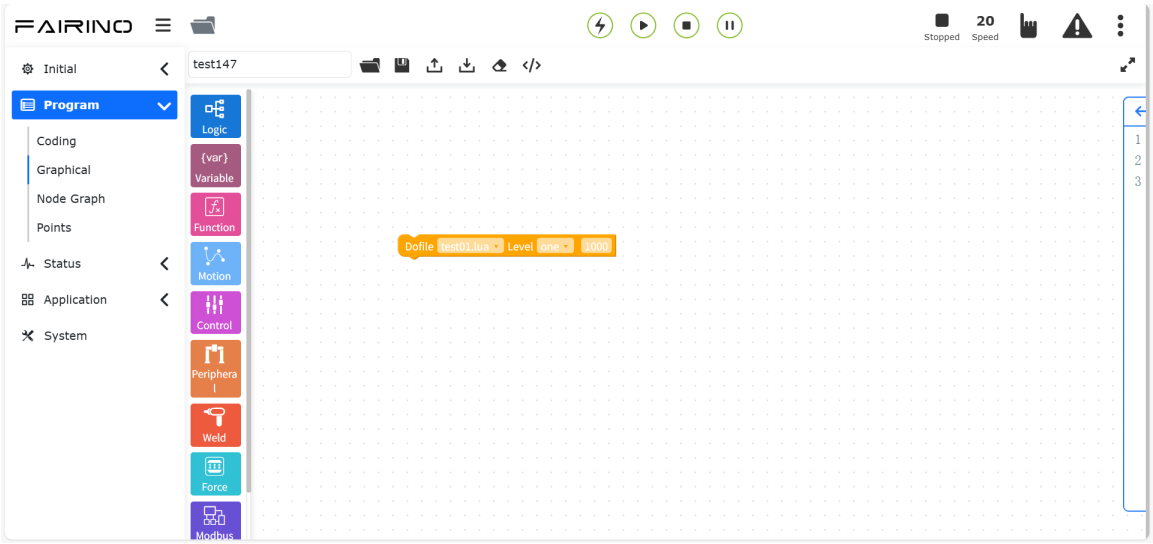


Figure 10.11-2 Call subroutine instruction code block

1.3.10.11.3 Auxiliary thread instruction

Drag the “auxiliary thread” code block to enter the graphical editing interface workspace.

The Thread command is an auxiliary thread function. Users can define an auxiliary thread to run simultaneously with the main thread. The auxiliary thread mainly interacts with external devices for data, supports socket communication, robot DI status acquisition, robot DO status setting, robot status information acquisition, and data interaction with the main thread. The data obtained by the main thread through the auxiliary thread is used to control the judgment of the robot’s motion logic.

“Auxiliary thread” instruction node, parameters:

- Method name: auxiliary thread name
- Call function: auxiliary thread call function value

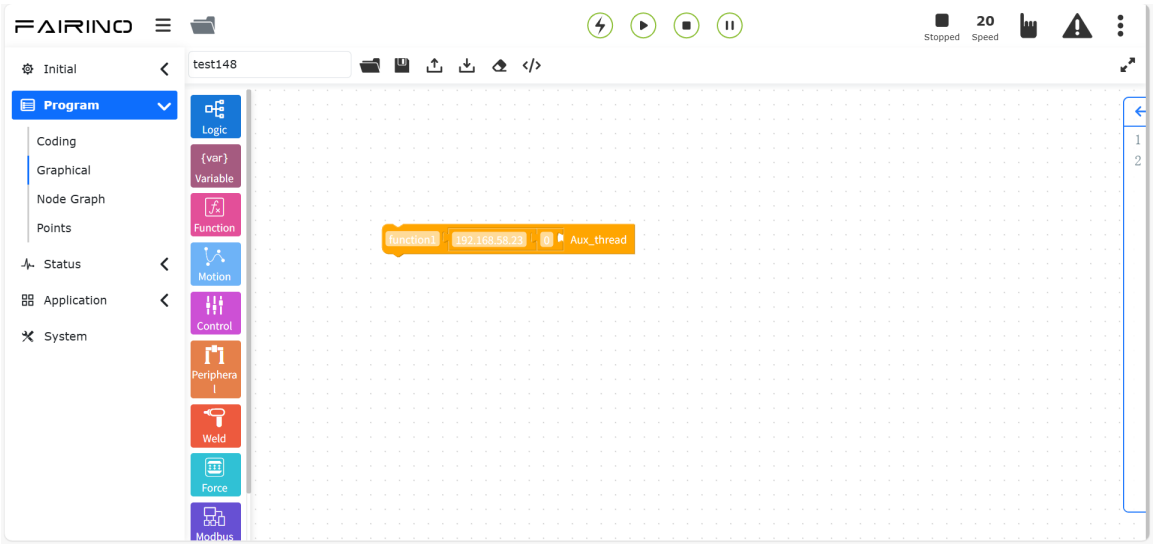


Figure 10.11-3 Auxiliary thread code block

1.3.10.11.4 Point table instruction

Drag the “point table” code block to enter the graphical editing interface workspace.

This instruction is mainly used to switch between system mode and point table mode. By switching the point table, the teaching points in different point tables are applied. For details, see Chapter 12 - Teaching Points.

“Point table” instruction node, parameters:

- Point table mode: switch different point table names

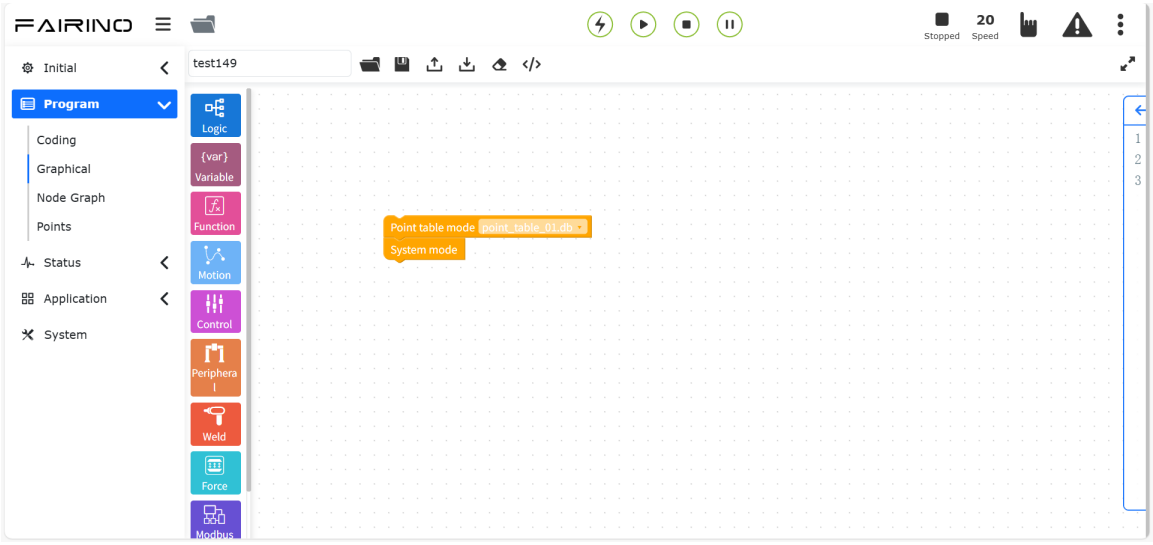


Figure 10.11-4 Point table code block

1.3.10.11.5 Focus follow instruction

Drag the “focus follow” code block to enter the graphical editing interface workspace.

This instruction is mainly used to focus on one point and follow the movement during the robot movement.

“Focus follow” command node, parameters:

- Parameter ratio: 0~100, default value 50
- Feedforward parameter: 0~1000, default value 19
- Maximum angular velocity acceleration limit: 0~10000, default value 1440
- Maximum angular velocity limit: 0~1000, default value 180
- Lock X-axis direction: reference input vector/horizontal/vertical

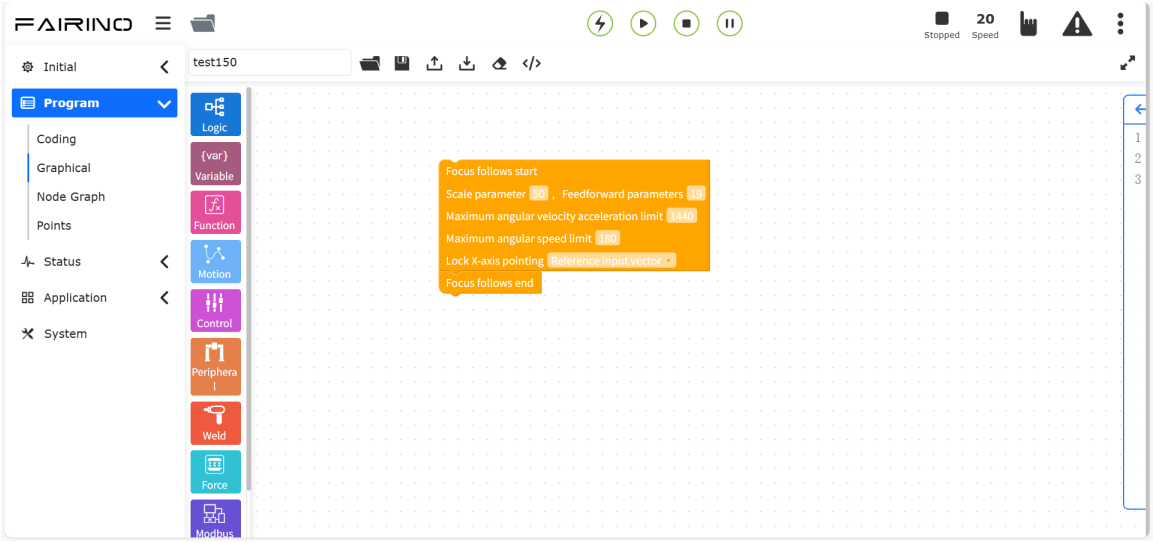


Figure 10.11-5 Focus follow code block

1.3.10.12 Graphical programming command usage example

After selecting the graphical programming type, click the graphical code block to be used, and you can drag and splice it in the workspace.

For example, select PTP and Lin motion instructions and control instruction Waitms for splicing. The outer layer can nest a folding advanced instruction and enter the comment name to achieve code block folding operation.

Click the drop-down box to select the instruction parameter type, and the input box can be filled in with instruction parameter data. The following are examples of graphical programming commands:

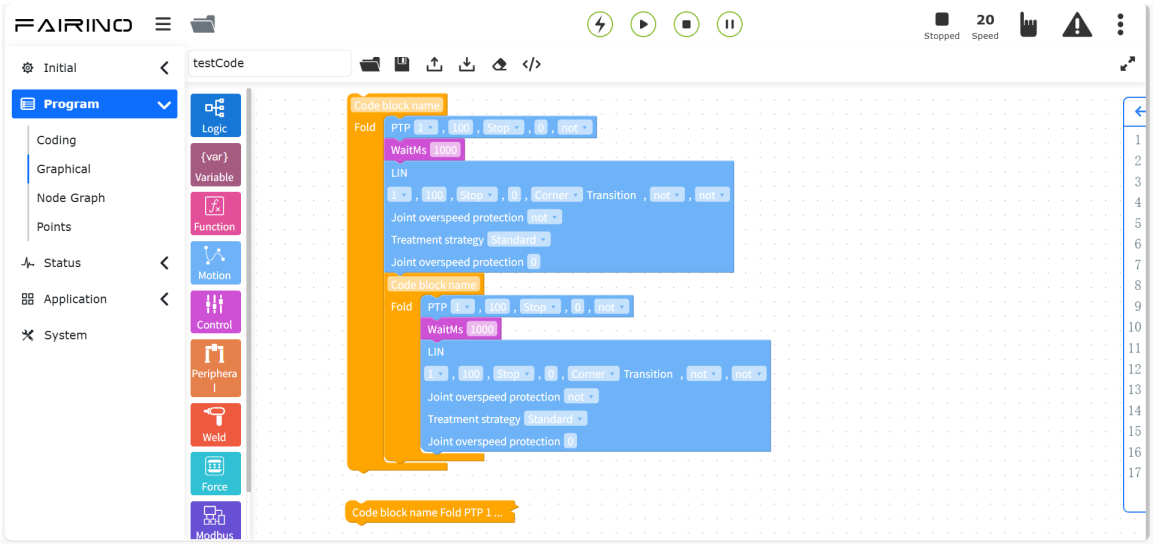


Figure 10.12-1 Example of graphical programming commands

After completing the splicing of graphical programming instructions and filling in parameters, fill in the workspace name and click the “Save” icon to save this program. Select the “workspace” that has been written, click Start Run, and you can execute this program.

1.3.10.12.1 Modularization of graphical programming code blocks

In order to improve the readability of graphical programming codes, the modularization function of graphical programming code blocks has been added, namely, advanced instructions: folding instruction code blocks.

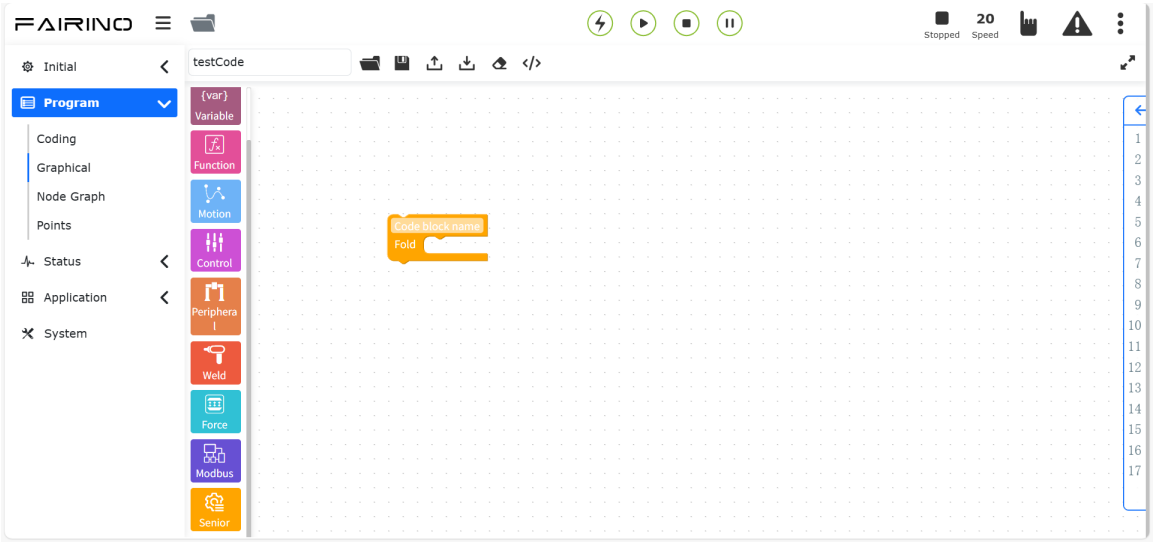


Figure 10.12-2 Folding instruction code block

1. Write a code block instruction, add a folding instruction code block in the outer layer, and write the remarks of the instruction in the input box.

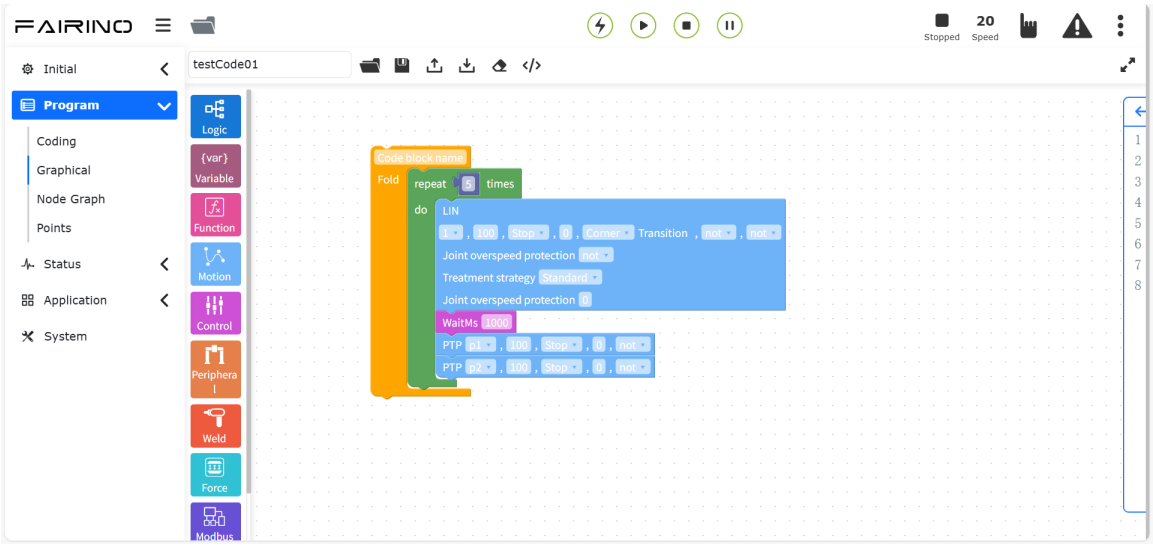


Figure 10.12-3 Folding instruction effect diagram

2. Right-click “Fold Block” in the right-click operation bar, and the instruction code block will be folded. The code block is folded into a line and the program can be executed correctly when folded.

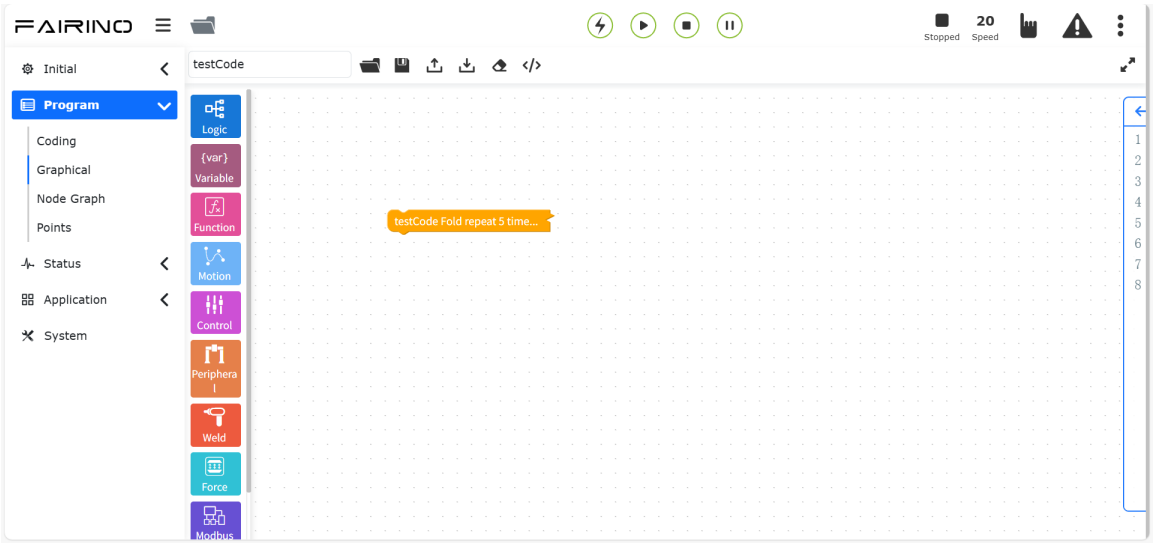


Figure 10.12-4 Folded effect diagram

3. Scroll the mouse to realize the page zoom function, the specific effect is as follows:

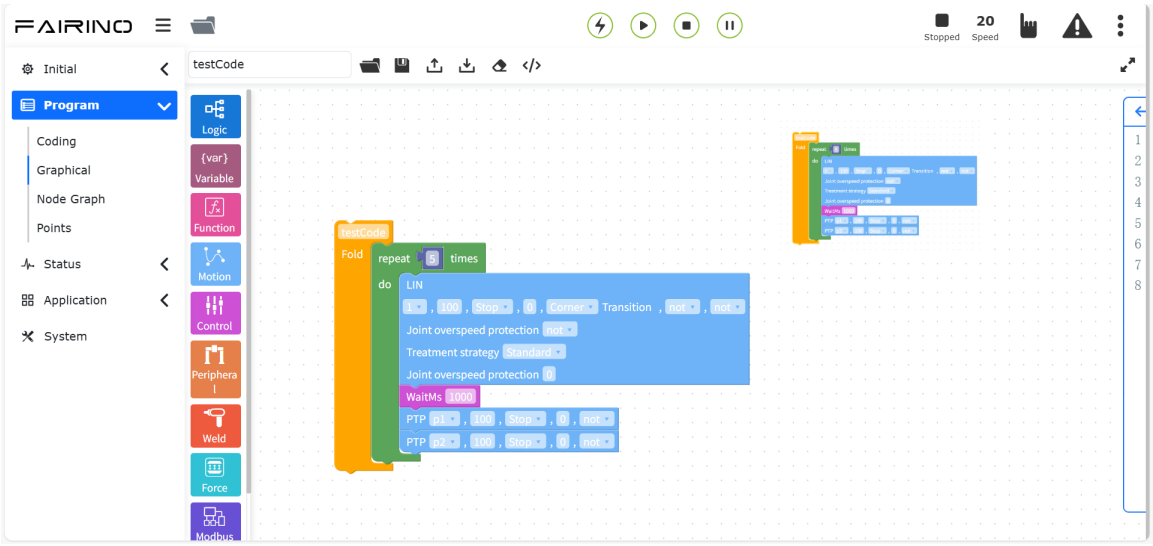


Figure 10.12-5 Page zoom function effect diagram

1.3.10.12.2 Graphical programming same name overwrite

On the graphical programming page, after creating/loading a file, change the workspace name and click Save. If the changed workspace name file already exists, the "Teaching point already exists" pop-up box will be triggered, as shown below.

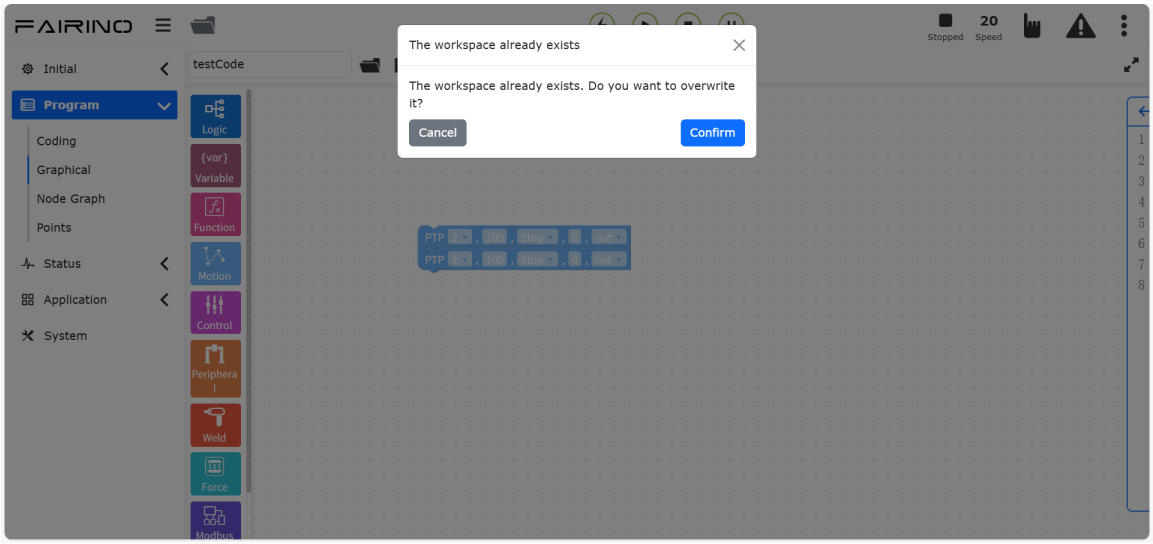


Figure 10.12-6 Graphical programming program overwriting

Step1: Click the "Cancel" button to continue the previous operation.

Step2: Click the "Synchronize and update teaching program" checkbox, and then click the "Overwrite" button, the lua program on the current graphical programming page will overwrite the lua program of the changed workspace file name.

1.3.10.12.3 Graphical programming program not saved verification

On the graphical programming page, after opening/creating a new program, if the graphical programming program is changed but not saved.

If you click the “Open” file operation, the “Save this program” pop-up box will be triggered, prompting “The current program has changed, do you want to save the changes to this program?”, as shown in the figure below.

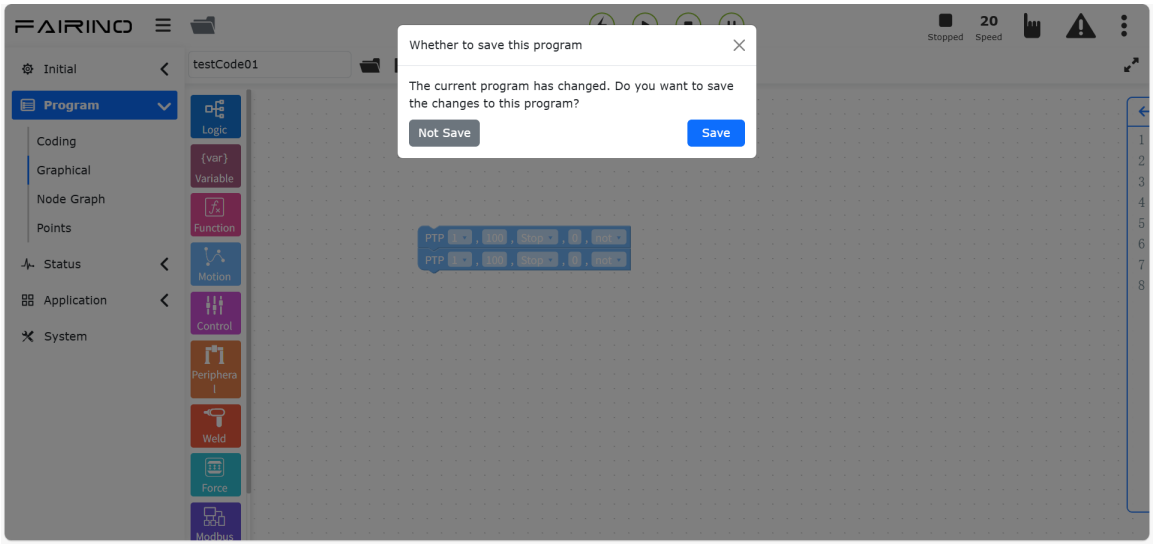


Figure 10.12-7 Current page program is not saved verification

Step1: Click the “Don’t Save” button to continue the previous “Open” file operation.

Step2: Click the “Save” button, the unsaved Lua program is saved successfully, and the previous “Open” file operation is continued.

If you leave the graphical programming page and switch to other pages, the “Do you want to save this program” prompt will also be triggered, and you will still stay on the current graphical programming page, as shown below.

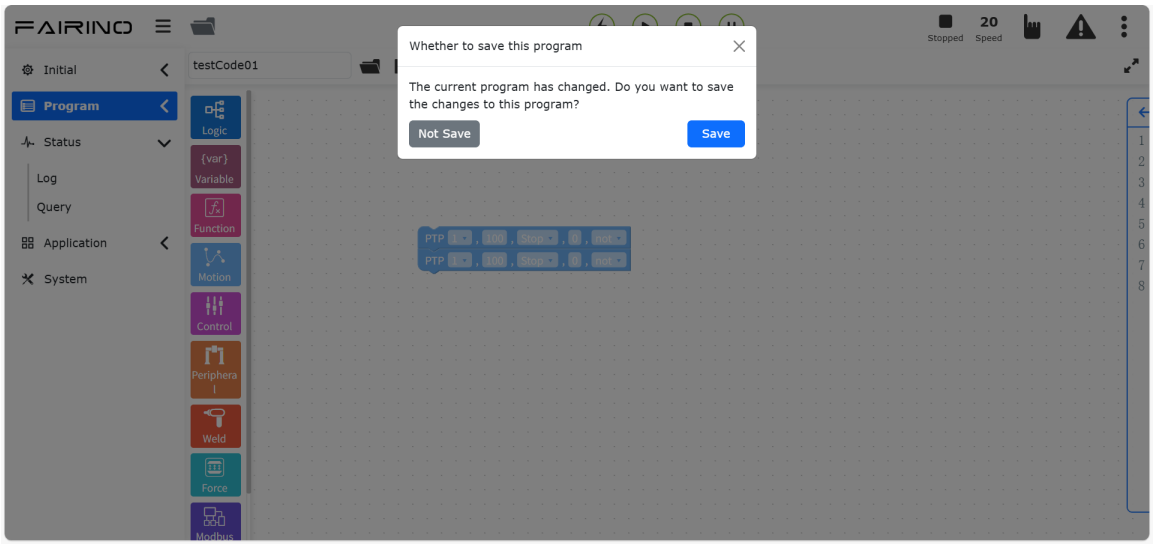


Figure 10.12-8 Switch page program is not saved verification

Step1: Click the “Don’t Save” button to jump to the previously selected page.

Step2: Click the "Save" button, the unsaved Lua program is saved successfully, and jumps to the previously selected page. If the saved program name already exists, it prompts that the teaching point already exists, whether to overwrite. After canceling/overwriting, jump to the previously selected page.

1.3.11 Node graph programming

1.3.11.1 Basic information

1.3.11.1.1 Introduction

Node graph programming is programming software developed for robots. Its main functions and technical features are as follows:

- The connections between nodes better present the contextual logic of the program;
- Through operations such as creating nodes, connecting nodes, and editing node parameters, robot programming can be completed with just dragging operations and a small amount of parameter input;
- Helps to better visualize code and script complex and repetitive tasks faster;

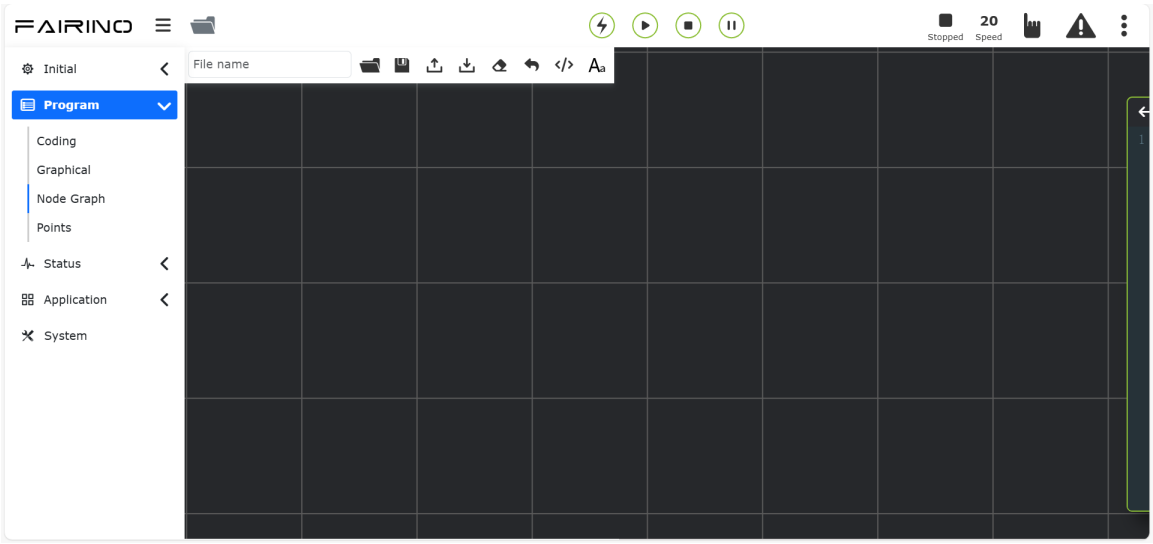


Figure 11.1-1 Node graph programming interface

1.3.11.1.2 Toolbar

Use the toolbar at the top left of the Node Graph Programming page.

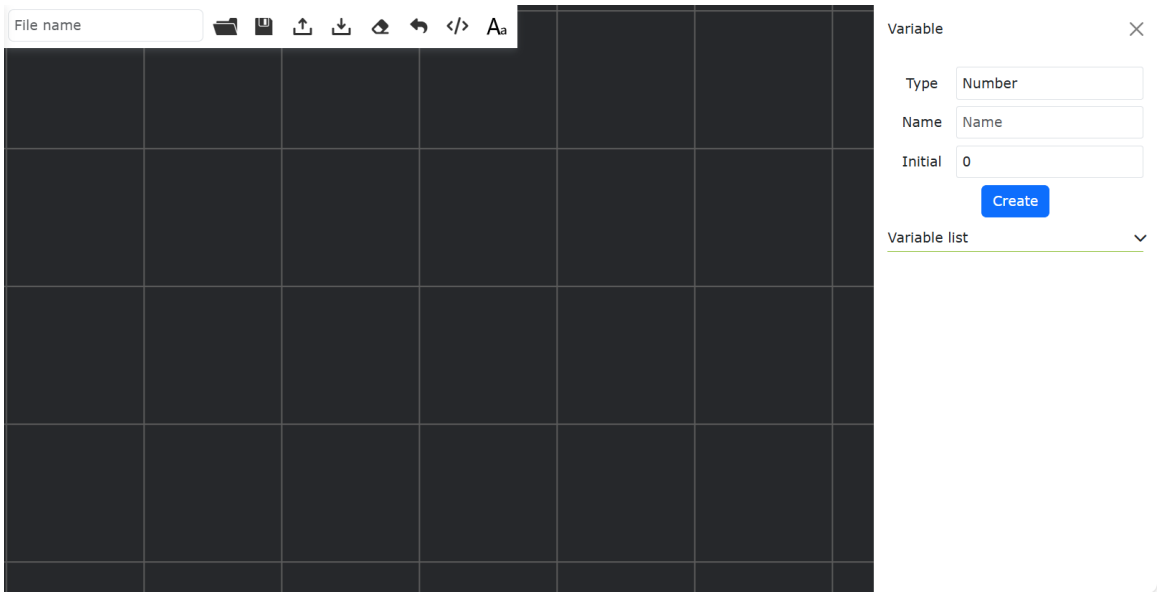


Figure 11.1-2 Toolbar



Note:

name:**Open**

effect:Open user program file, choose to load or delete files in the pop-up box



Note:

name:**Save**

effect:Save node graph edits



Note:

name:**Overload**

effect:Reload the node graph content of the last operation to the local



Note:

name:**New**

effect:Create a new node graph programming file



Note:

name:**Export**

effect:After creating/opening the node graph programming file, click the "Export" button to pop up the "Export Node Graph Programming" pop-up box, select the workspace file name to export the file (json format)



Note:

name:**Import**

effect:Click the "Import" button to pop up the import prompt box. Select the file to be imported and click Import. The file content will be displayed in the node graph programming workspace



Note:

name:**Code**

effect:After the node graph is connected, Lua code is generated

1.3.11.2 Node graph operations

1.3.11.2.1 Node program

For node programs, you need to right-click the mouse in a blank space to open the node program selection bar. Program instructions are mainly divided into logic instructions, motion instructions, force control instructions, control instructions, Modbus instructions, expansion axis and other instructions.

The input box above the node program selection bar allows fuzzy search to quickly locate the required node instructions.

The specific node program operation process is as follows:

- Click “Begin” to start the node and create the starting node programming location;
- Click on the selected program instruction node, the corresponding node diagram is displayed in the workspace, and the instruction parameters can be selected and entered in the drop-down box;
- The functions of the arrows on the right side of the command node: 1. A single arrow icon connects to the next node; 2. Multiple arrow icons, the first “Body” arrow icon connects to the content node, and the second “Completed” icon connects to the next node;
- Connect the “Begin” start node to the completed node program to end the node programming operation;

1.3.11.3 If/Else instruction

Click the “If/Else” related command node to enter the node diagram editing interface. (This command requires a certain programming foundation. If you need help, please contact us)

“If/Else” command:

- First: Connect the node instructions within the if condition
- Second: If only two judgment conditions are entered on the left, it means connecting the node instructions in the else condition; if all three judgment conditions on the left exist, it means connecting the node instructions in the elseif condition.
- Third: If all three judgment conditions on the left exist, it means connecting the else condition.
- Completed: Connect subsequent node instructions

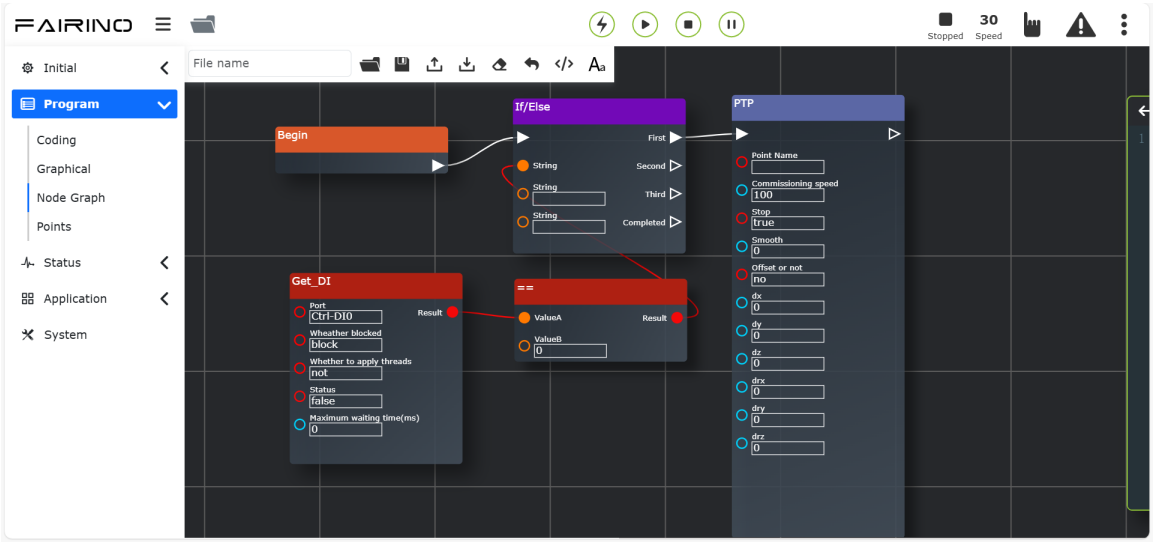


Figure 11.3-1 “If/Else” command node interface

1.3.11.4 While instruction

Click the “While” related command node to enter the node graph editing interface.

Enter the waiting condition in the input box behind While, enter the action instructions during the loop in the input box behind do, and click Save. (For the convenience of operation, you can enter any do content and edit other instructions in the program to insert them instead).

“While” command:

- Condition: while loop condition

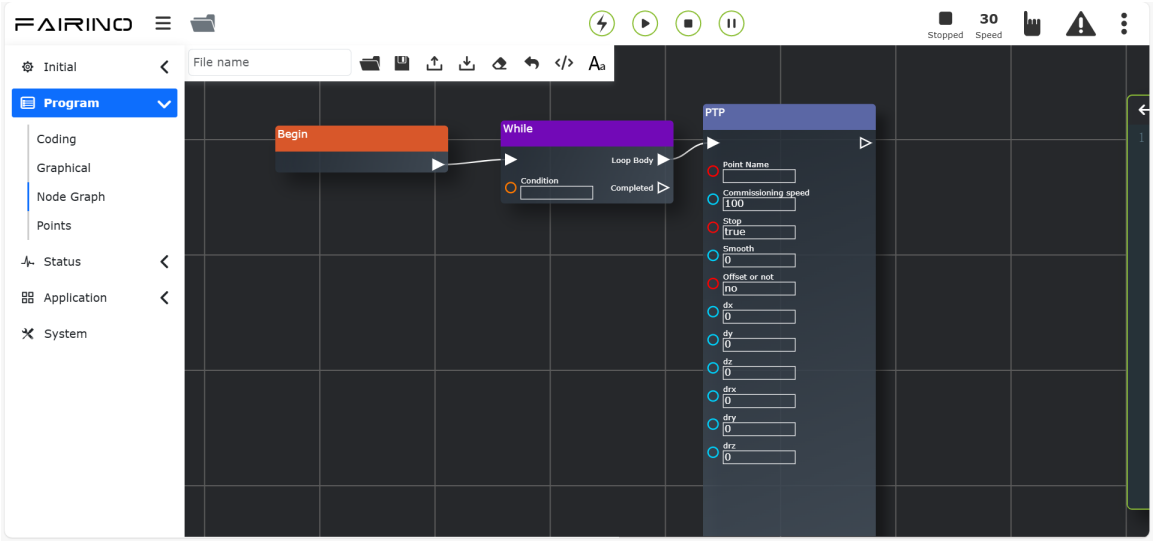


Figure 11.4-1 “While” command node interface

1.3.11.5 Goto instruction

Click “Goto” on the relevant command node to enter the node graph editing interface.

“Goto” command, the first “Body” arrow icon connects to the body content node, and the second “Completed” arrow icon connects to the subsequent jump position goto instruction node. (This command requires a certain programming foundation. If you need help, please contact us).

- Jump name: Enter the jump name to determine the jump location

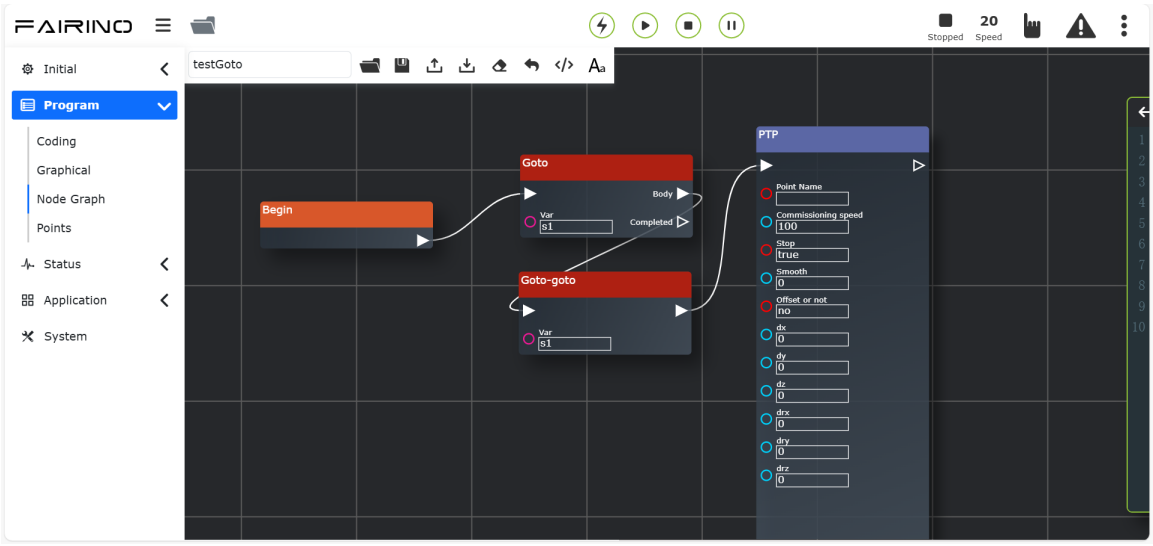


Figure 11.5-1 “Goto” command node interface

Important: Jump name cannot start with a number.

1.3.11.6 Wait instruction

Click the “Wait” related command node to enter the node diagram editing interface.

This instruction is a delay instruction, divided into four parts: “WaitMs”, “WaitDI”, “WaitMultiDI” and “WaitAI”.

1. “Wait” command node, parameters:

- Waiting time (ms): The unit of delay waiting time is milliseconds. Enter the number of milliseconds to wait.

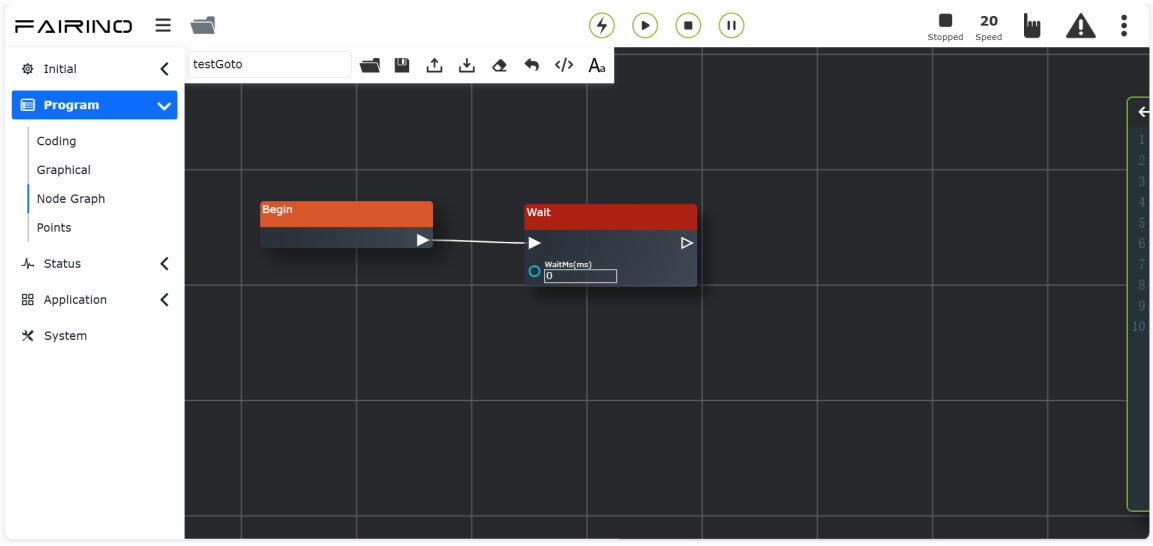


Figure 11.6-1 “Wait” command node interface

2. “WaitDI” command node, parameters:

- DI port number: Ctrl-DI0 ~ Ctrl-CI7(WaitDI,[0~15]), End-DI0 ~ End-DI1(WaitToolDI,[0~1])

- Status: false/true
- Maximum time (ms): 0 ~ 10000
- Waiting for timeout processing: stop reporting errors/continue execution/wait forever

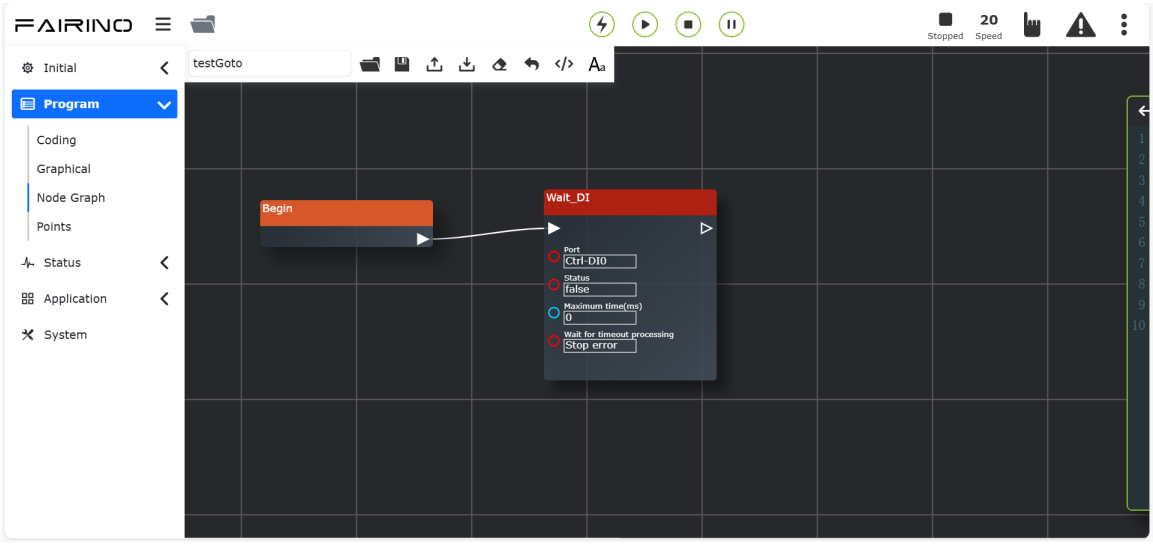


Figure 11.6-2 “WaitMultiDI” command node interface

3. “WaitMultiDI” command node, parameters:

- Condition: and/or
- Conditional selection: Select the port number that the bit status is enabled, separated by commas, for example DI0, DI1
- True value corresponding port: Select the port number of the true value, separated by commas, for example DI0, DI1
- Maximum time (ms): 0 ~ 10000, maximum waiting time
- Waiting for timeout processing: stop reporting errors/continue execution/wait forever

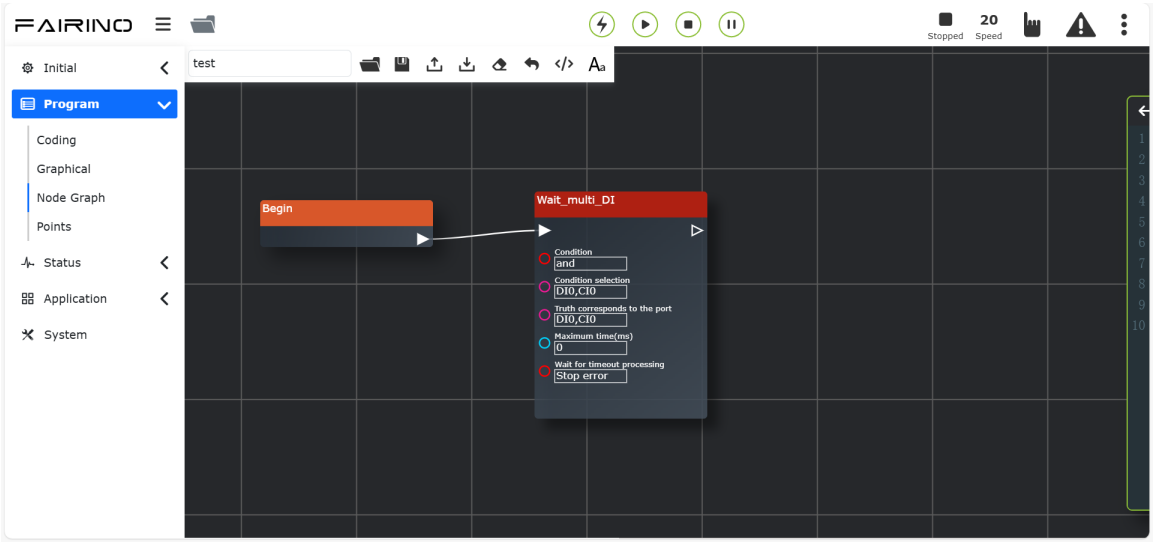


Figure 11.6-3 “WaitMultiDI” command node interface

4. “WaitAI” command node, parameters:

- Condition: and/or
- AI port number: Ctrl-AI0 ~ Ctrl-AI1(WaitAI,[0~1]), End-AI0(WaitToolAI,[0])
- Condition: greater than/less than
- Value (%): 1 ~ 100
- Maximum time (ms): 0 ~ 10000
- Waiting for timeout processing: Stop error reporting/continue execution/keep waiting. When waiting for timeout processing and keep waiting, the maximum time defaults to 0.

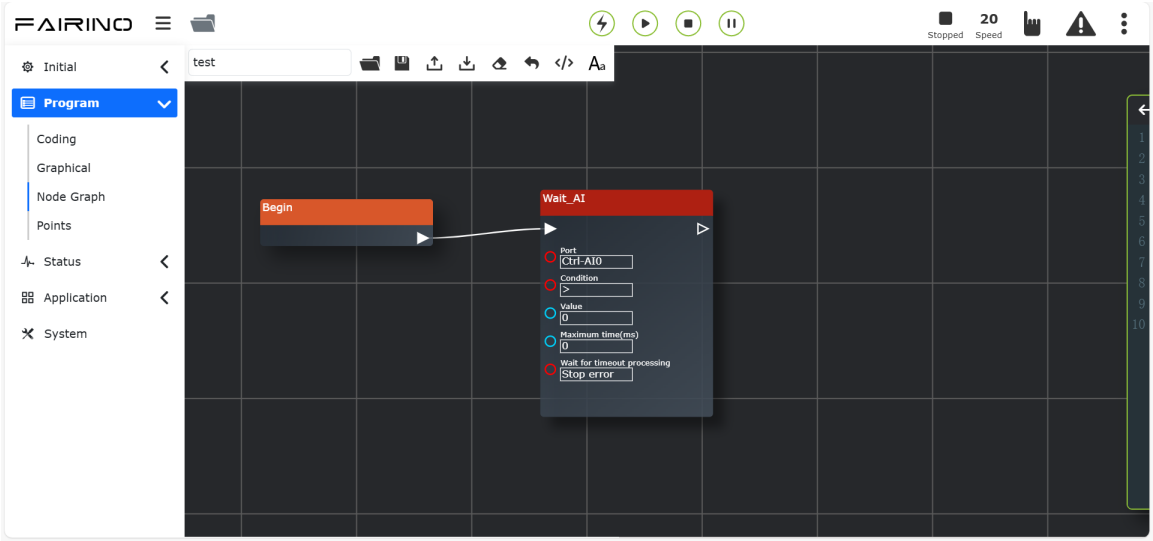


Figure 11.6-4 “WaitAI” command node interface

1.3.11.7 Pause instruction

Click the “Pause” command node to enter the node graph editing interface.

This command is a pause command. Insert this command in the program. When the program executes this command, the robot will be in a paused state. If you want to continue running, click the “Pause/Resume” button in the control area.

“Pause” command node, parameters:

- Pause type: no function, cylinder not in place, etc.

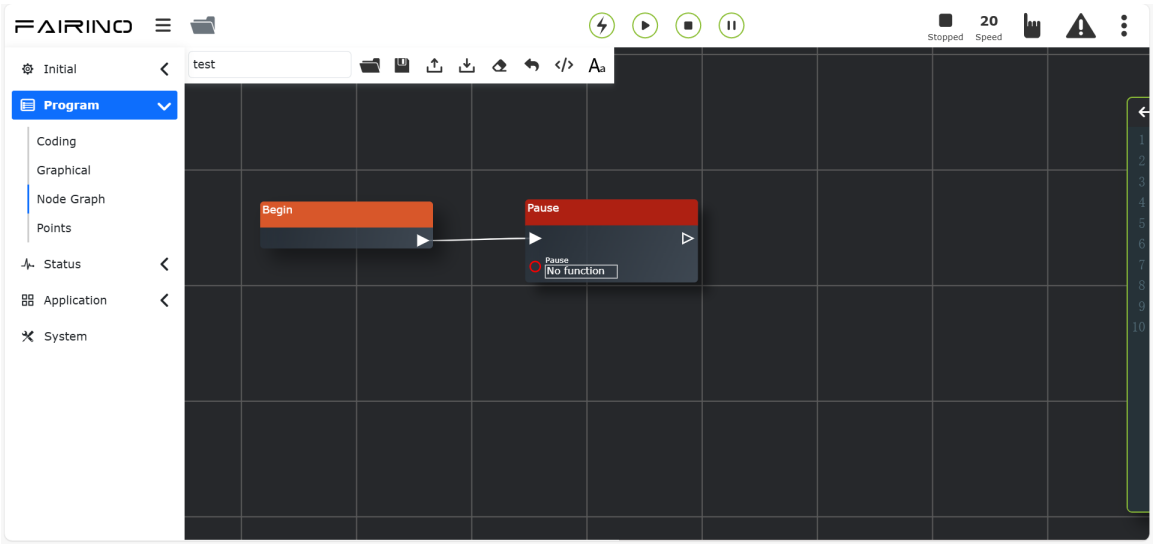


Figure 11.7-1 “Pause” command node interface

1.3.11.8 Dofile instruction

Click the “Dofile” command node to enter the node graph editing interface.

This command is a dofile command. Insert this command in the program. When the program executes this command, the robot will be in a paused state. If you want to continue running, click the “Pause/Resume” button in the control area.

“Dofile” command node, parameters:

- dofile: create the generated file name
- Which layer to call: first layer/second layer
- ID number: the corresponding position id of the level to which it belongs

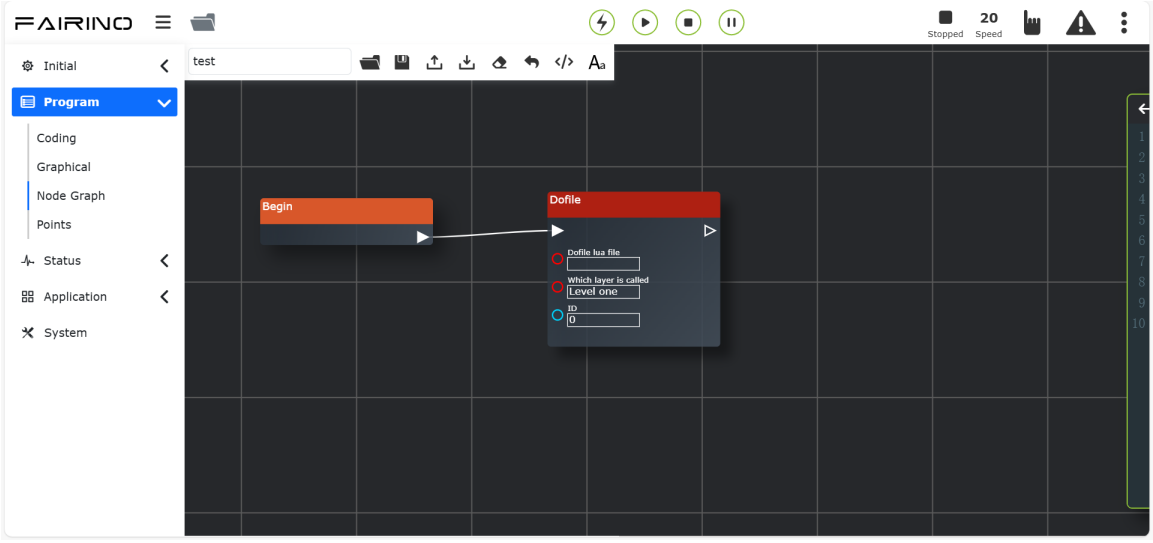


Figure 11.8-1 “Dofile” command node interface

1.3.11.9 Set system variable command

Click the “Set system variable command” command node to enter the node graph editing interface.

This instruction is a setting system variable instruction, which is divided into setting system variables and obtaining system variables. It is used in conjunction with while, if-else and other instructions.

“Configure system variable command” command node, parameters:

- Var: custom variable name
- Value: Enter according to actual situation

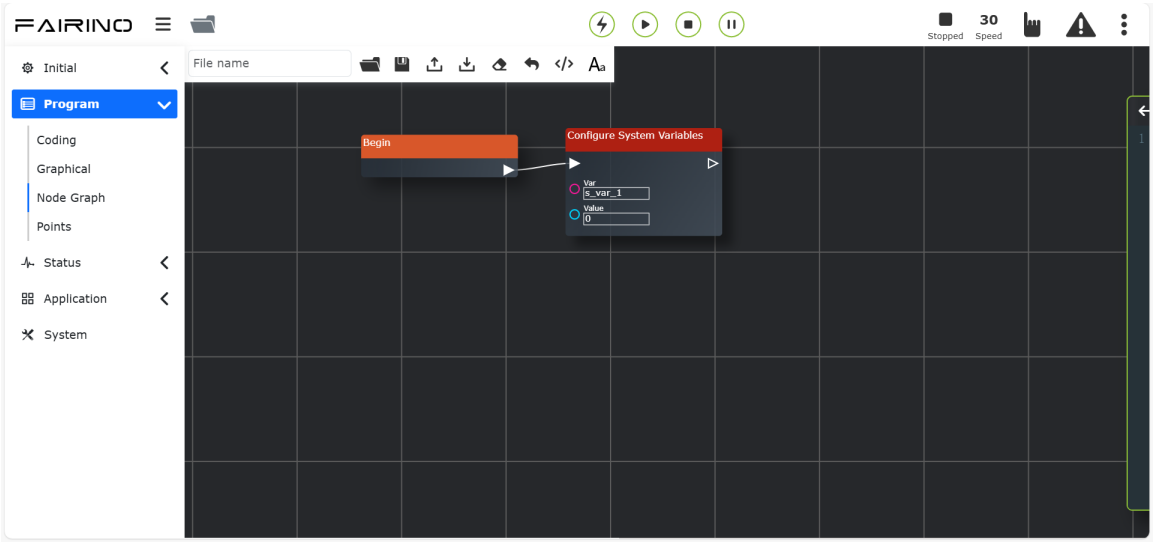


Figure 11.9-1 “Configure system variable command” command node interface

“Obtain system variable command” command node, parameters:

- Var: custom variable name

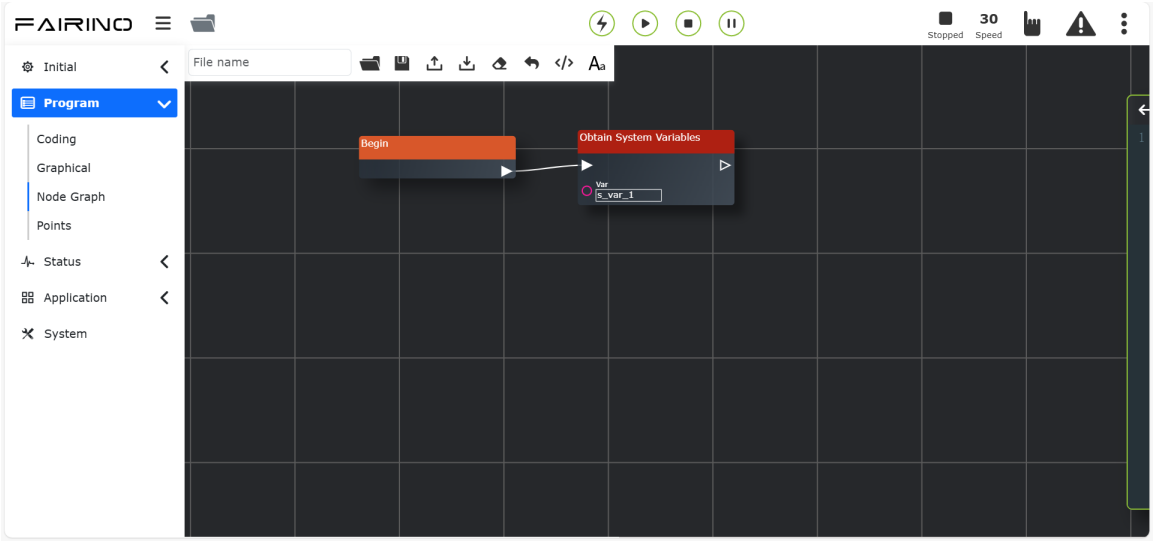


Figure 11.9-2 “Obtain system variable command” command node interface

Important: The variable name must be a defined name.

1.3.11.10 PTP instruction

Click the "PTP" command node to enter the node graph editing interface.

You can select the point you need to reach. The smooth transition time setting can realize the continuous movement from this point to the next point. Whether to set the offset, you can choose the offset based on the base coordinate system and the offset based on the tool coordinate, and x, y will pop up. z, rx, ry, rz offset settings, the specific PTP path is the optimal path automatically planned by the motion controller.

"PTP" command node, parameters:

- Point name: Teaching point
- Debugging speed (%): 0 ~ 100
- Stop: false/true
- Smooth transition (ms): smooth transition time 0 ~ 500
- Whether to offset No/base coordinate offset/tool coordinate offset. When No is selected, the dx~drz parameter values will not take effect.
- dx~drz: offset

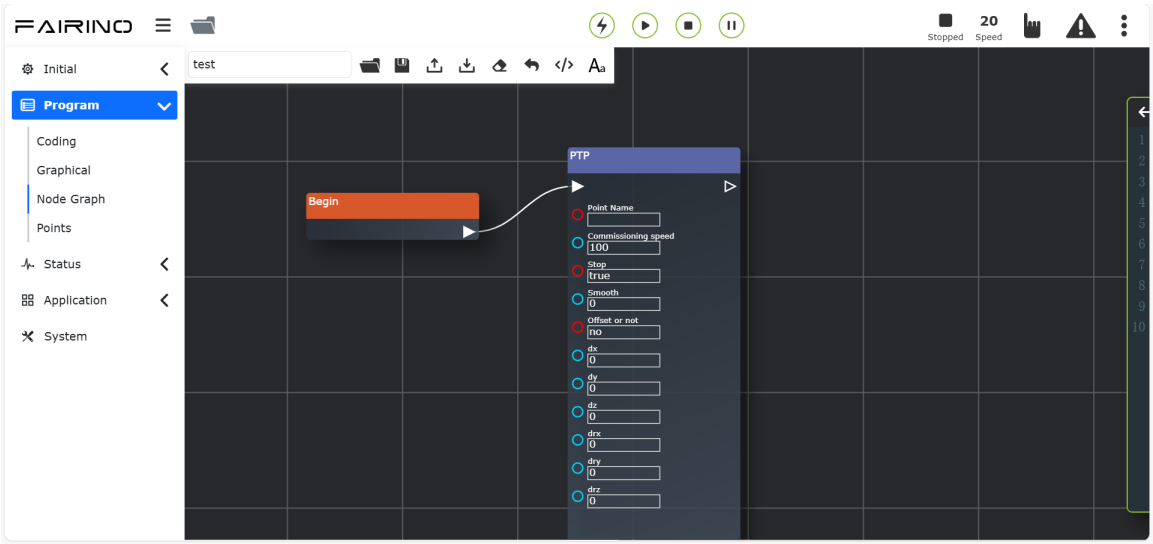


Figure 11.10-1 "PTP" command node interface

1.3.11.11 LIN instruction

Click the “LIN” command node to enter the node graph editing interface.

The function of this command is similar to the “point to point” command, but the path to the point reached by this command is a straight line.

“LIN” command node, parameters:

- Point name: Teaching point
- Debugging speed (%): 0 ~ 100
- Stop: false/true. When true is selected, the smooth transition parameter value does not take effect
- Smooth transition (mm): Smooth transition radius 0 ~ 1000
- Whether to seek position: false/true
- Position search point variables: REF0~99/RES0~99. When false is selected for position search, the parameters will not take effect
- Whether to offset: No/base coordinate offset/tool coordinate offset. When No is selected, the dx~drz parameter values will not take effect
- dx~drz: Offset.

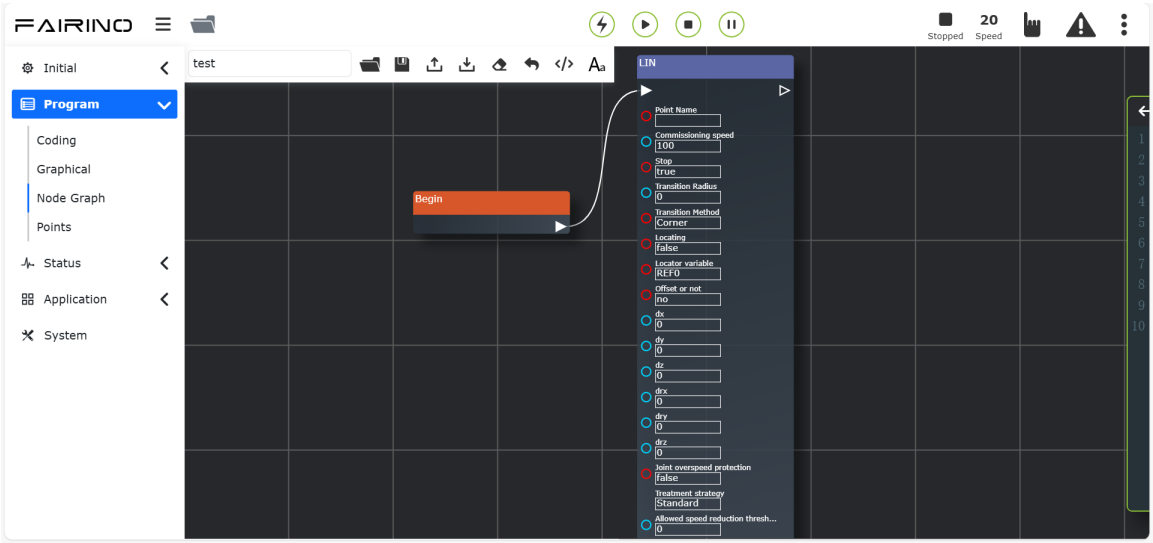


Figure 11.11-1 “LIN” command node interface

1.3.11.12 LIN(seamPos) instruction

Click the “LIN(seamPos)” command node to enter the node graph editing interface.

This command function is used in welding scenarios using laser sensors.

“LIN(seamPos)” command node, parameters:

- Point name: Teaching point
- Debugging speed (%): 0 ~ 100
- Stop: false/true. When true is selected, the smooth transition parameter value does not take effect
- Smooth transition (mm): Smooth transition radius 0 ~ 1000

- Weld cache data selection: execution planning data/execution record data
- Board type: corrugated board/corrugated board/fence board/oil drum/corrugated shell steel
- Whether to offset: No/base coordinate offset/tool coordinate offset/Laser raw data offset. When No is selected, the dx~drz parameter values will not take effect.
- dx~drz: Offset.

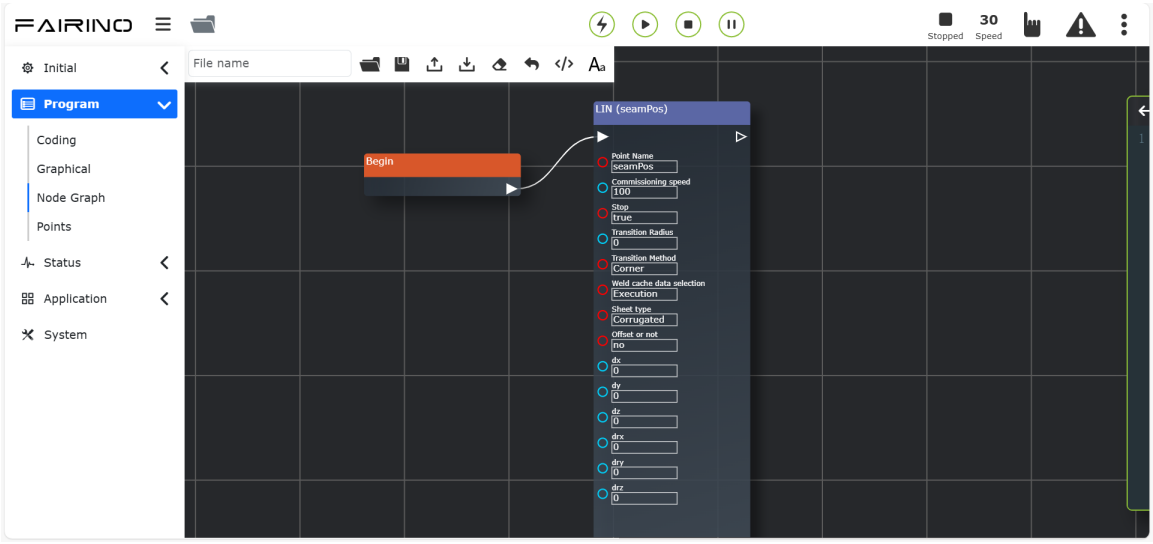


Figure 11.12-1 “LIN(seamPos)” command node interface

1.3.11.13 ARC instruction

Click the “ARC” command node to enter the node graph editing interface.

The arc motion contains two points. The first point is the transition point in the middle of the arc, and the second point is the end point. Both the transition point and the end point can be offset or not. You can choose to offset based on the base coordinate system or offset based on the tool coordinate. Shift, set x, y, z, rx, ry, rz offset, the end point can set a smooth transition radius to achieve continuous motion effect.

“ARC” command node, parameters:

- Arc middle point: teaching point
- Whether to offset: No/base coordinate offset/tool coordinate offset. When No is selected, the dx~drz parameter values will not take effect
- dx~drz: offset
- Arc end point: teaching point
- Whether to offset: No/base coordinate offset/tool coordinate offset. When No is selected, the dx~drz parameter values will not take effect
- dx~drz: offset
- Debugging speed (%): 0 ~ 100
- Stop: false/true. When true is selected, the smooth transition parameter value does not take effect.
- Smooth transition (mm): Smooth transition radius 0 ~ 1000

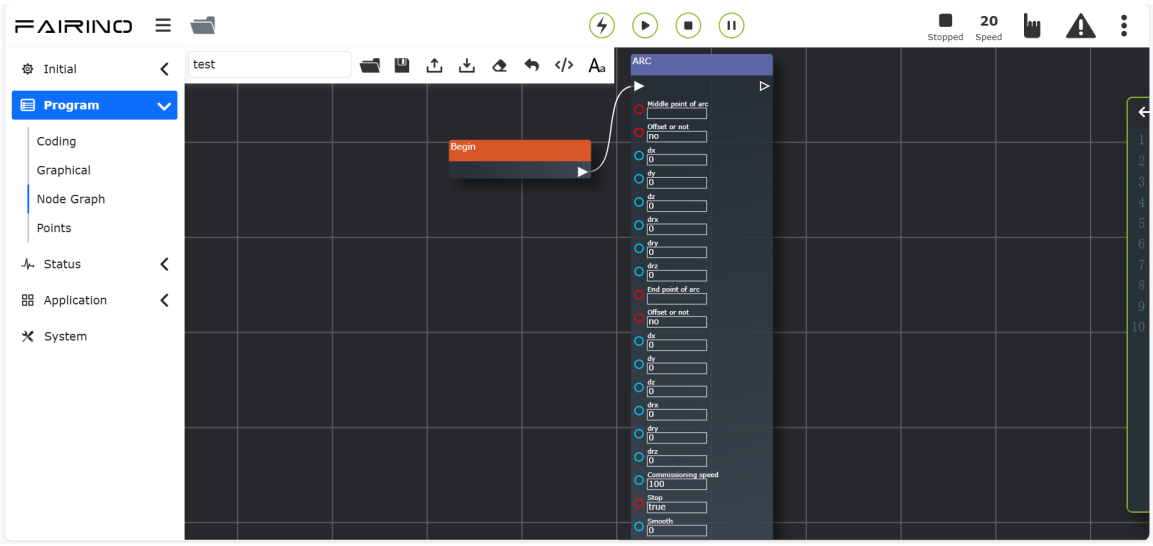


Figure 11.13-1 “ARC” command node interface

1.3.11.14 Circle instruction

Click the “Circle” command node to enter the node graph editing interface.

The full circle motion includes two points. The first point is the middle transition point 1 of the full circle, and the second point is the middle transition point 2 of the full circle. Transition point 2 can be set to be offset. This offset takes effect on both transition point 1 and Transition point 2.

“Circle” command node, parameters:

- Middle point 1 of the whole circle: teaching point
- Middle point 2 of the whole circle: teaching point
- Debugging speed (%): 0 ~ 100
- Whether to offset: No/base coordinate offset/tool coordinate offset. When No is selected, the dx~drz parameter values will not take effect.
- dx~drz: offset

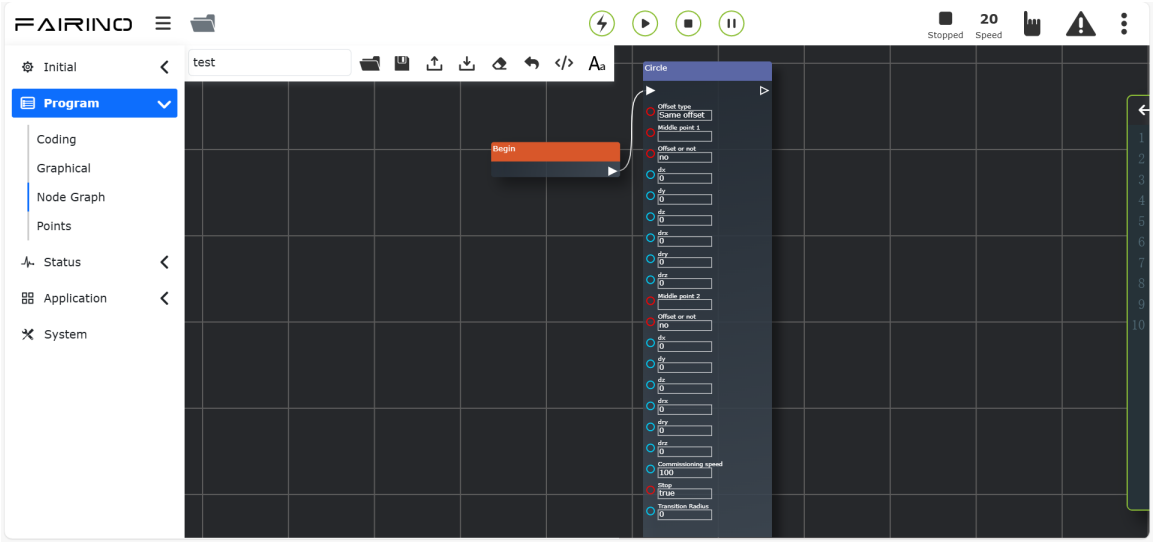


Figure 11.14-1 “Circle” command node interface

1.3.11.15 Spiral instruction

Click the “Spiral” command node to enter the node graph editing interface.

The spiral motion contains three points, which form a circle. On the third point setting page, the parameter settings include the number of spiral turns, attitude correction angle, radius increment and rotation axis direction increment. The number of spiral turns is The number of movement circles of the spiral. The attitude correction angle corrects the attitude at the end of the spiral and the attitude of the first point of the spiral. The radius increment is the increment of the radius of each circle. The rotation axis direction increment is the direction of the spiral axis. Increment. Set whether to offset, the offset takes effect on the entire spiral trajectory.

“Spiral” command node, parameters:

- Spiral middle point 1: Teaching point
- Spiral middle point 2: teaching point
- Spiral middle point 3: teaching point
- Debugging speed (%): 0 ~ 100
- Whether to offset: No/base coordinate offset/tool coordinate offset. When No is selected, the dx~drz parameter values will not take effect.
- dx~drz: offset
- Number of spiral turns: 0 ~ 100
- Attitude angle correction rx(°): -1000 ~ 1000
- Attitude angle correction (°): -1000 ~ 1000
- Attitude angle correction rz(°): -1000 ~ 1000
- Radius increment (mm): -100 ~ 100
- Increment in rotation axis direction (mm): -100 ~ 100

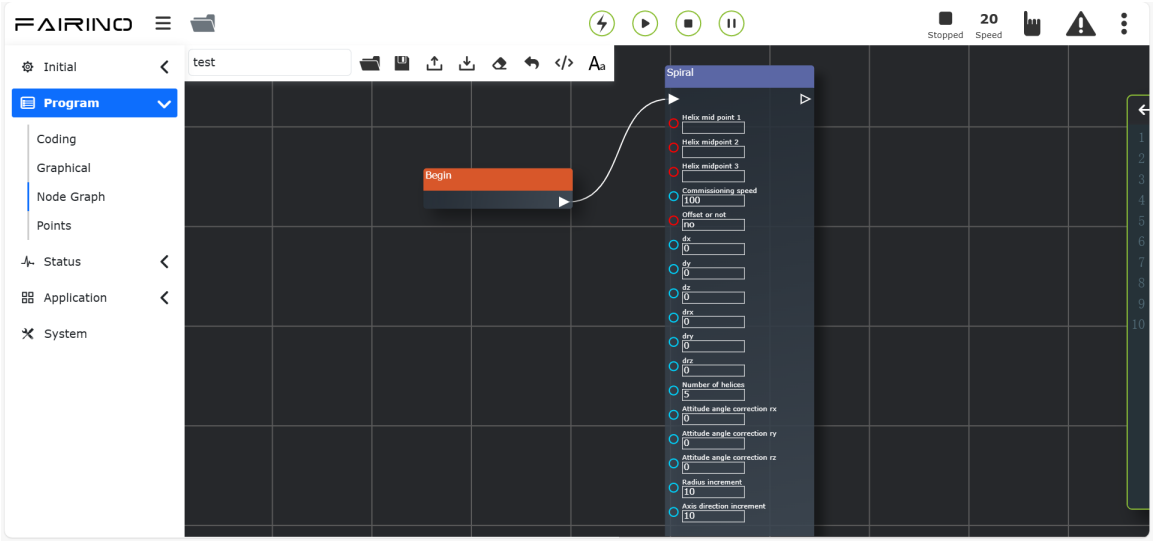


Figure 11.15-1 “Spiral” command node interface

1.3.11.16 N-Spiral instruction

Click the “N-Spiral” command node to enter the node graph editing interface.

The new spiral motion is an optimized version of spiral motion. This instruction only requires one point and the configuration of each parameter to achieve spiral motion. The robot takes the current position as the starting point, and the user sets the debugging speed, offset or not, the number of spiral turns, the spiral inclination angle, the initial radius, the radius increment, the rotation axis direction increment and the rotation direction. The number of spiral turns is the length of the spiral. The number of movement circles, the spiral inclination angle is the angle between the Z axis of the tool and the horizontal direction, the attitude correction angle corrects the attitude at the end of the spiral and the attitude of the first point of the spiral, the initial radius is the radius of the first circle, and the radius increment That is, the increment of the radius of each circle, the increment of the rotation axis direction is the increment of the spiral axis direction, and the rotation direction is clockwise and counterclockwise.

“N-Spiral” command node, parameters:

- Starting point of spiral: teaching point
- Debugging speed (%): 0 ~ 100
- Whether to offset: No/base coordinate offset/tool coordinate offset. When No is selected, the dx~drz parameter values will not take effect.
- dx~drz: offset
- Number of spiral turns: 0 ~ 100
- Helix inclination angle (°): -100 ~ 100
- Initial radius: 0 ~ 100
- Radius increment (mm): -100 ~ 100
- Increment in rotation axis direction (mm): -100 ~ 100
- Direction of rotation: clockwise/counterclockwise

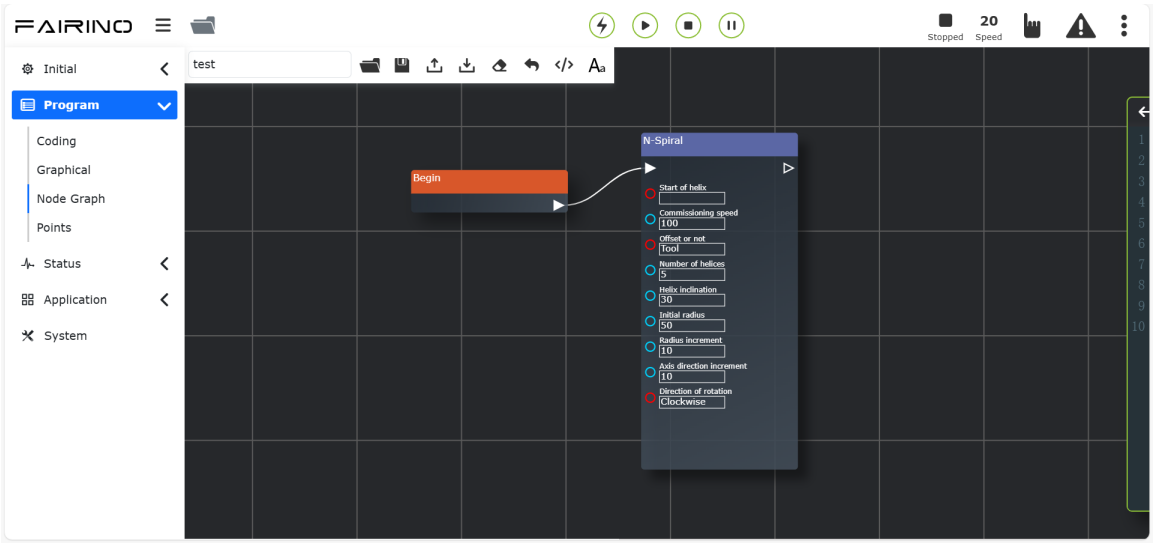


Figure 11.16-1 “N-Spiral” command node interface

1.3.11.17 H-Spiral instruction

Click the “H-Spiral” icon to enter the H-Spiral command editing interface.

The “H-Spiral” command is a horizontal space spiral motion. This command is set after the single-segment motion (straight line) command.

“H-Spiral” command node, parameters:

- Spiral radius: 0~100mm
- Helix angular speed: 0~2rev/s
- Direction of rotation: spiral clockwise/counterclockwise
- Helix inclination angle: 0~40°

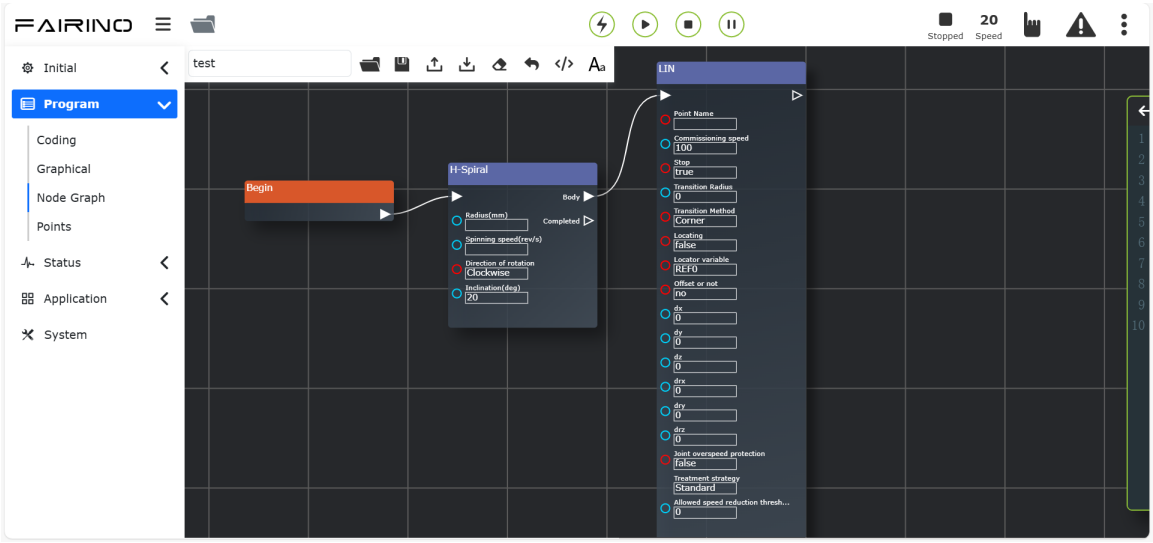


Figure 11.17-1 “H-Spiral” command node interface

1.3.11.18 Spline instruction

Click the “Spline” command node to enter the node graph editing interface.

This command is divided into three parts: the start of the spline group, the spline segment and the end of the spline group. The start of the spline group is the starting mark of the spline movement. The current node graph of the spline segment only contains the SPL segment. The end of the spline group is The end mark of the spline movement.

“Spline-SPTP” command node, parameters:

- Point name: Teaching point
- Debugging speed (%): 0 ~ 100

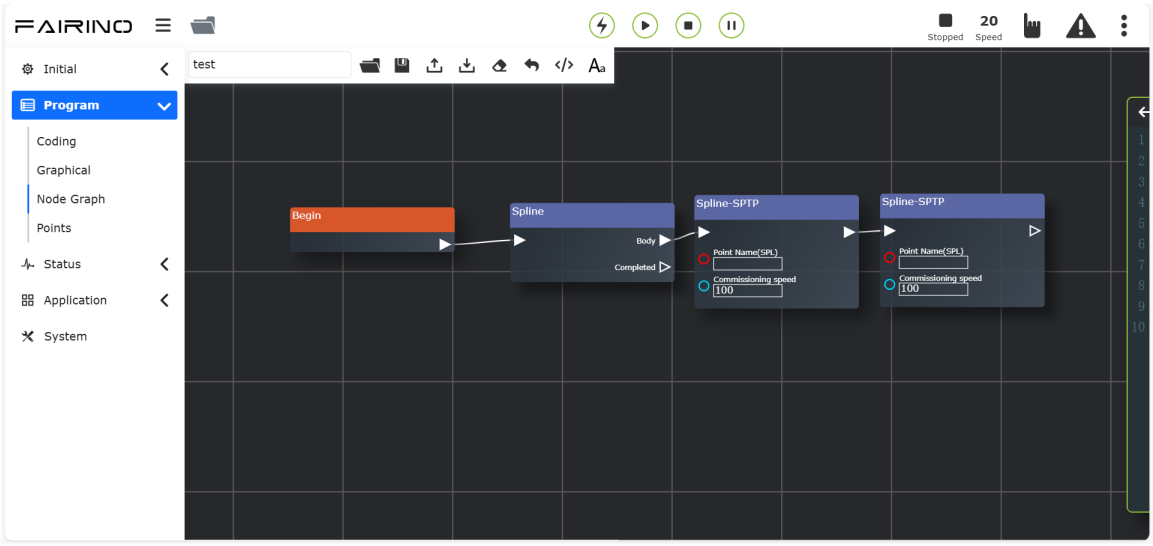


Figure 11.18-1 “Spline” command node interface

1.3.11.19 N-Spline instruction

Click the “N-Spline” command node to enter the node graph editing interface.

This instruction is a spline instruction algorithm optimization instruction. It will replace the existing spline instruction later. This instruction is divided into three parts: multi-point trajectory start, multi-point trajectory segment and multi-point trajectory end. The multi-point trajectory starts with multi-point trajectory. The starting mark of the movement. The multi-point track segment means setting each track point. Click the icon to enter the point adding interface. The end of the multi-point track is the end mark of the multi-point track movement. Here you can set the control mode and debugging speed. The control mode is divided into for a given control point and a given path point.

“N-Spline” command node, parameters:

- Control mode: teaching point
- Global average connection time: integer, greater than 10, default value is 2000 ms

“N-Spline-SPL” command node, parameters:

- Point name: Teaching point
- Debugging speed (%): 0 ~ 100
- Smooth transition radius: 0 ~ 1000
- whether last point: no/yes

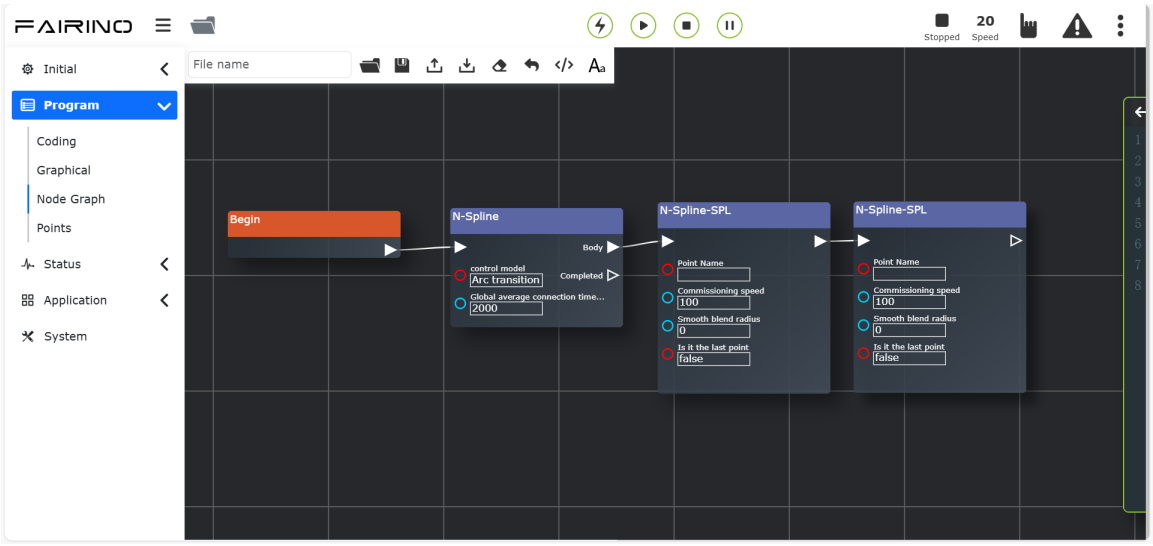


Figure 11.19-1 “N-Spline” command node interface

1.3.11.20 Weave instruction

Click the “Weave” command node to enter the node graph editing interface.

This command consists of two parts. The first part selects the weaving number of the configured parameters. The program for connecting the Body to represent the connection node is executed between “Start Weaving” and “Stop Weaving”.

“Weave” command node, parameters:

- Number: 0~7

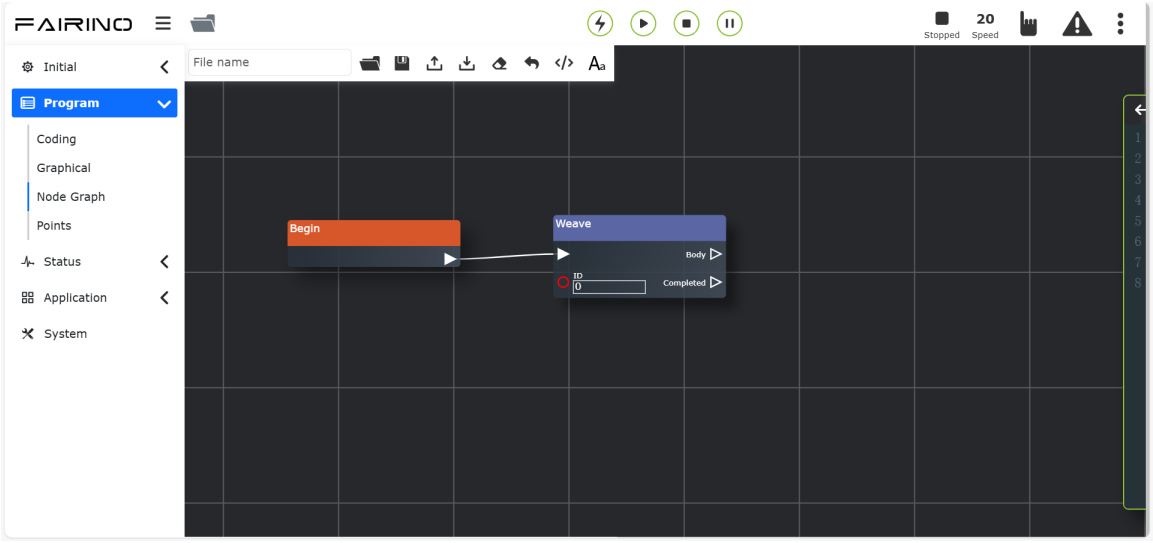


Figure 11.20-1 “Weave” command node interface

1.3.11.21 TPD instruction

Click the “TPD” command node to enter the node graph editing interface.

In this command, the user first needs to have a recorded trajectory.

When programming, first use the point-to-point command to reach the starting point of the corresponding trajectory, then select the trajectory in the trajectory recurrence command, select the smooth trajectory, and set the debugging speed. The trajectory loading command is mainly used to read the trajectory file in advance and extract it into trajectory instructions, which can be better applied to conveyor belt tracking scenarios.

“TPD” command node, parameters:

- Track name: recorded track
- Smooth trajectory: No/Yes
- Debugging speed (%): 0 ~ 100, default value is 25

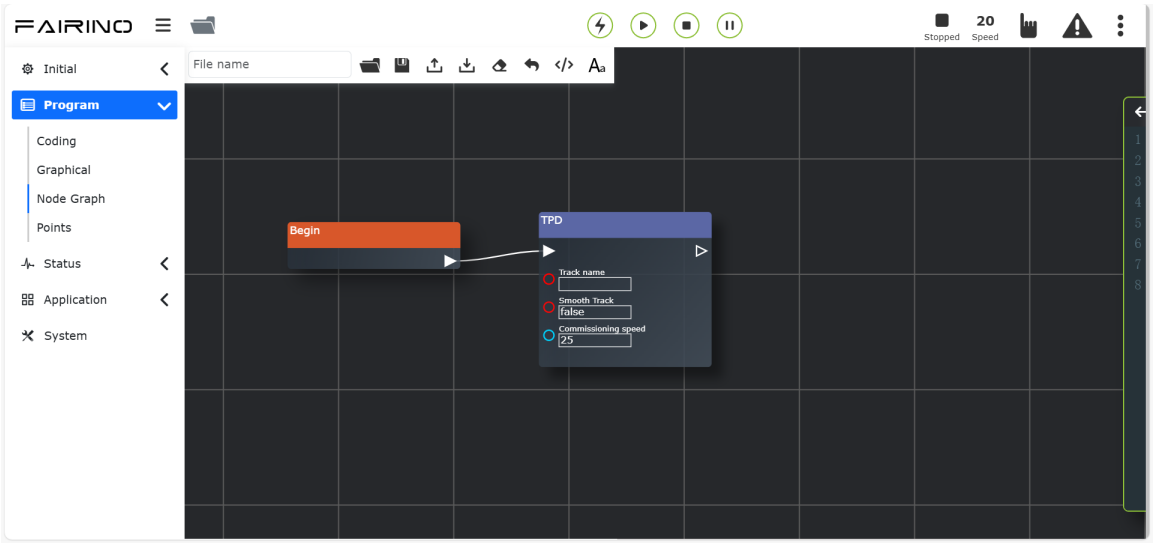


Figure 11.21-1 “TPD” command node interface

1.3.11.22 Offset instruction

Click the “Offset” command node to enter the node graph editing interface.

This command is an overall offset command. Enter each offset. The program connecting the Body to represent the connection node is executed between start and close. The motion command in the middle will be offset based on the base coordinates (or workpiece coordinates).

“Offset” command node, parameters:

- x: offset, -300~300
- y: offset, -300~300
- z: offset, -300~300
- rx: offset, -300~300
- ry: offset, -300~300
- rz: offset, -300~300

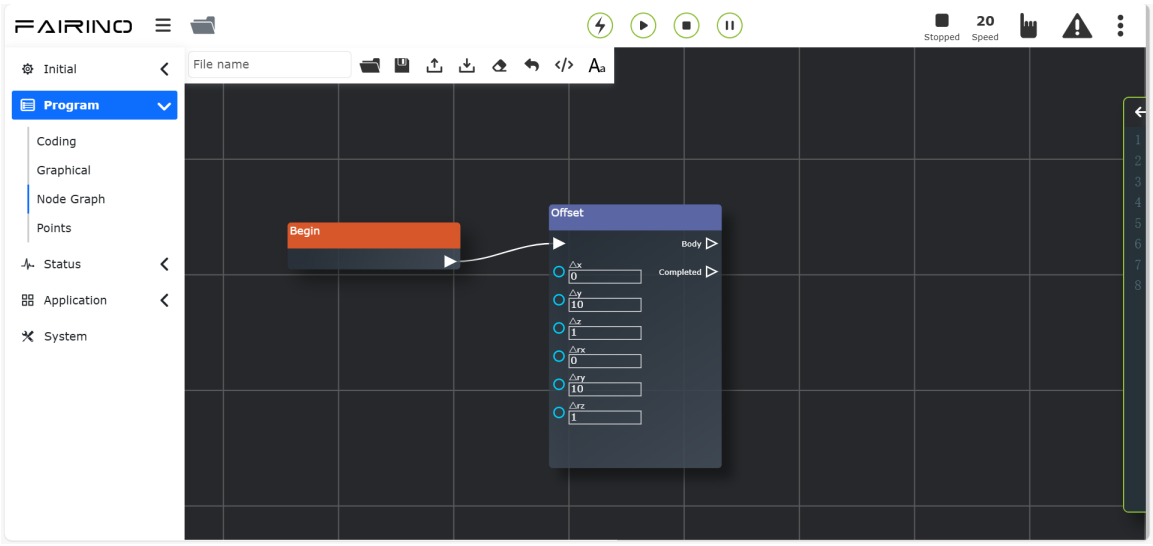


Figure 11.22-1 “Offset” command node interface

1.3.11.23 ServoCart instruction

Click the “ServoCart” command node to enter the node diagram editing interface.

Servo control (Cartesian space motion) instruction, which can control robot movement through absolute pose control or based on the current pose offset.

“ServoCart” command node, parameters:

- Movement mode: absolute position/base coordinate offset/tool coordinate offset
- x: offset, -300~300
- y: offset, -300~300
- z: offset, -300~300
- rx: offset, -300~300
- ry: offset, -300~300
- rz: offset, -300~300
- Proportional coefficient x: 0~1
- Proportional coefficient y: 0~1
- Proportional coefficient z: 0~1
- Proportional coefficient rx: 0~1
- Proportional coefficient ry: 0~1
- Proportional coefficient rz: 0~1
- Acceleration (%): 0~100
- Speed (%): 0~100
- Instruction cycle (s): 0.001~0.016
- Filter time (s): 0~1
- Scale up: 0~100

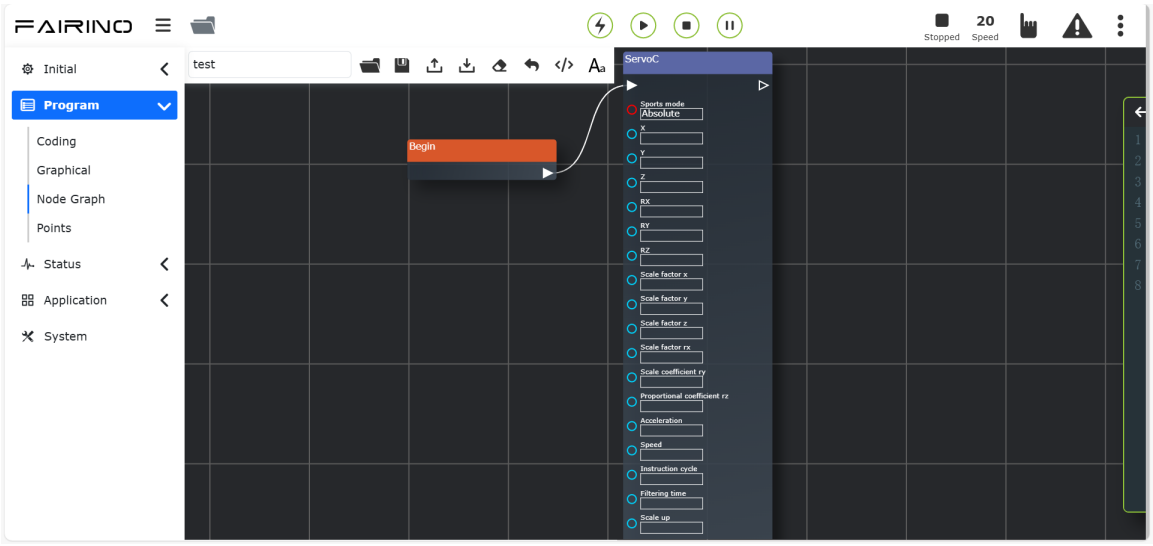


Figure 11.23-1 “ServoCart” command node interface

1.3.11.24 Trajectory instruction

Click the “Trajectory” command node to enter the node graph editing interface.

In this command, the user first needs to have a recorded trajectory.

“Trajectory” command node, parameters:

- Select track file: recorded track
- Debugging speed (%): 0 ~ 100, default value is 25

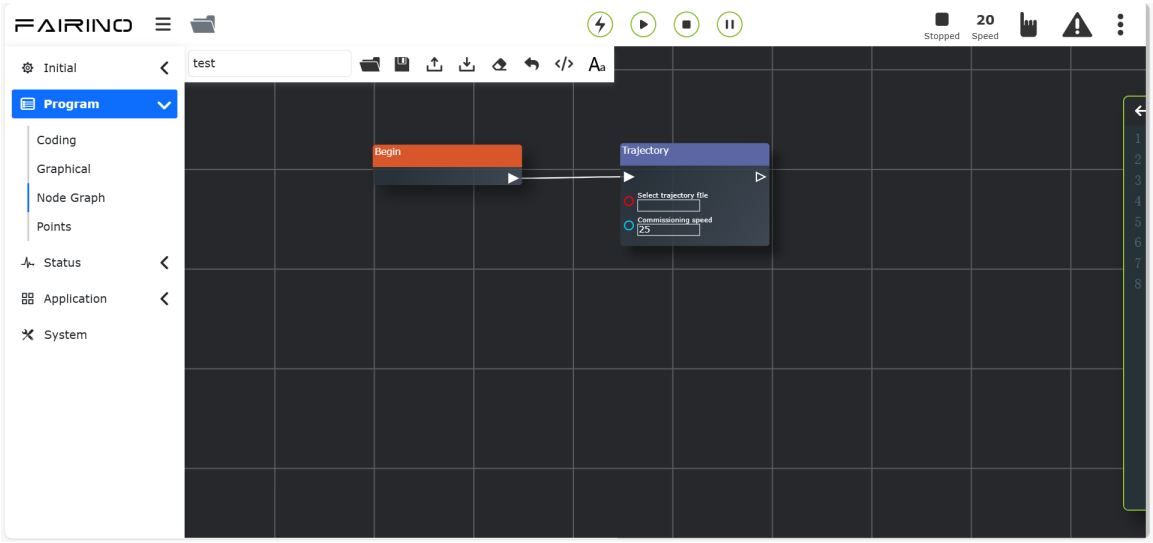


Figure 11.24-1 “Trajectory” command node interface

1.3.11.25 TrajectoryJ instruction

Click the “TrajectoryJ” command node to enter the node graph editing interface.

In this command, the user first needs to have a recorded trajectory, and the trajectory file can be imported in advance in the teaching program interface. The trajectory command and trajectory J command are suitable for the general interface of the camera directly giving the trajectory. When there is a discrete trajectory point file in a fixed format, it can be imported into the system to make the robot move according to the trajectory of the imported file.

“TrajectoryJ” command node, parameters:

- Select track file: recorded track
- Debugging speed (%): 0 ~ 100, default value is 25
- Track mode: path points/control points

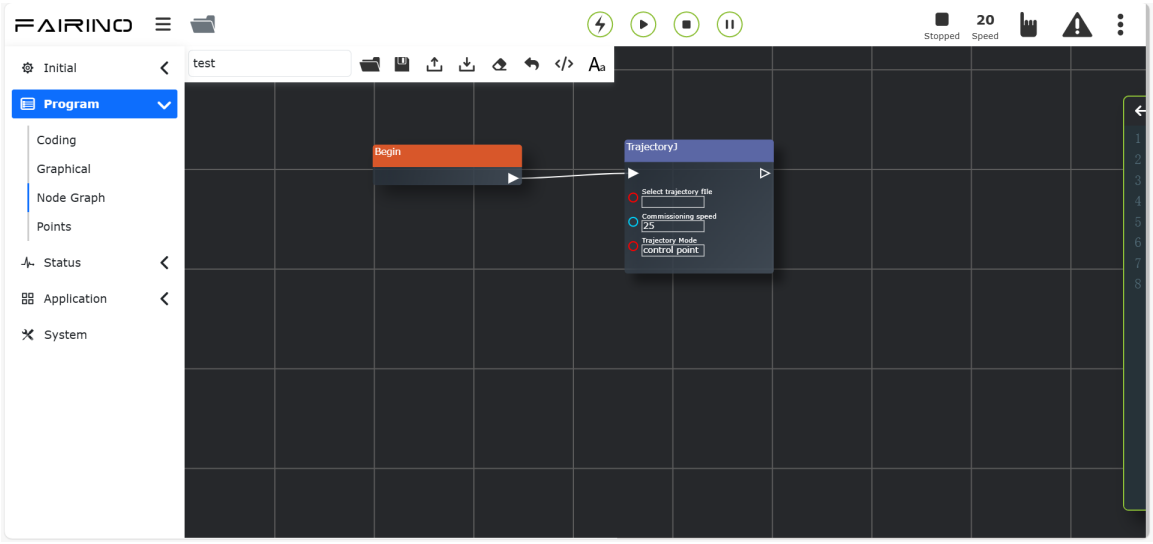


Figure 11.25-1 “TrajectoryJ” command node interface

1.3.11.26 DMP instruction

Click the “DMP” command node to enter the node graph editing interface.

DMP is a trajectory imitation learning method that requires planning a reference trajectory in advance. In the command editing interface, select the teaching point as the new starting point, click “Add” and “Apply” to save the command. The specific path of DMP is a new trajectory that imitates the reference trajectory with a new starting point.

“DMP” command node, parameters:

- Point name: teaching point
- Debugging speed (%): 0 ~ 100, default value is 100

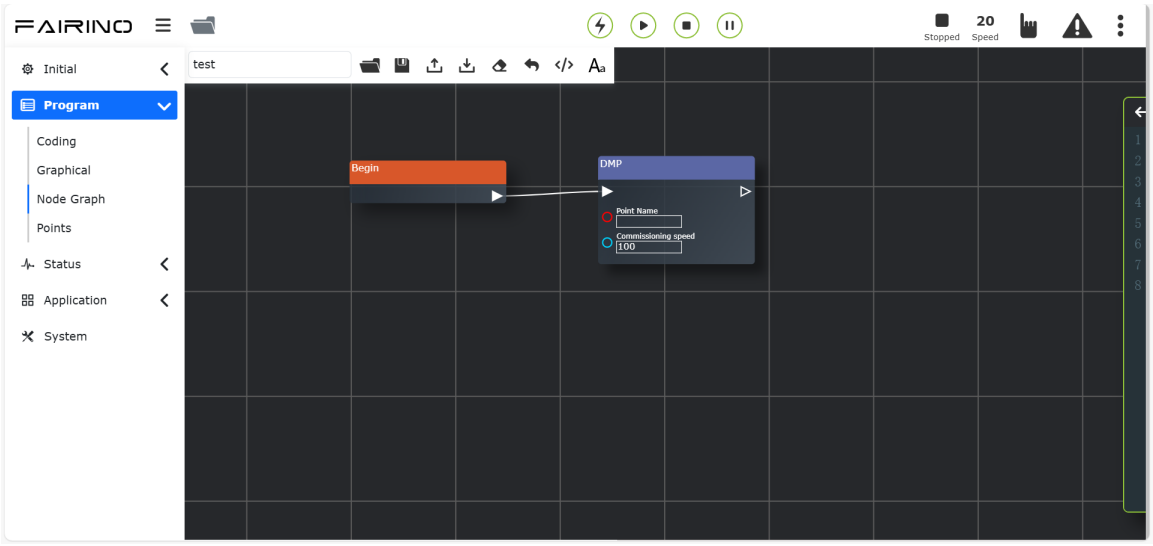


Figure 11.26-1 “DMP” command node interface

1.3.11.27 WPSTrsf instruction

Click the “WPSTrsf” command node to enter the node graph editing interface.

Select the workpiece coordinate system to be automatically converted, click “Add” and “Apply” to save the instruction. When adding PTP and LIN instructions, connect them to the Body to implement execution within the instruction, and the points in the workpiece coordinate system are automatically converted.

“WPSTrsf” command node, parameters:

- Workpiece coordinate system: workpiece coordinate series list

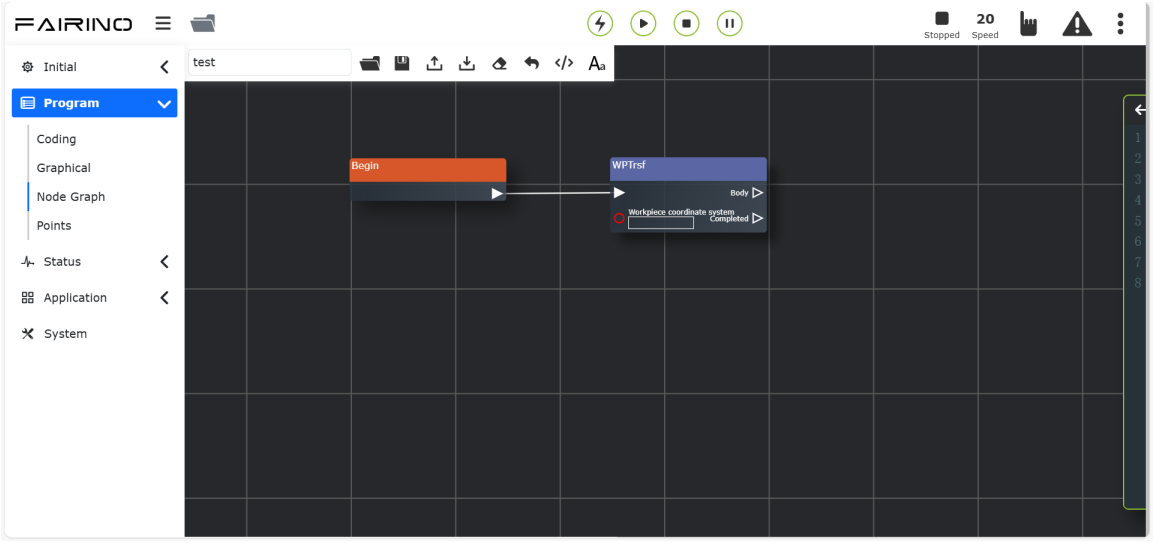


Figure 11.27-1 “WPSTrsf” command node interface

1.3.11.28 ToolTrst instruction

Click the “ToolTrst” command node to enter the node graph editing interface.

Select the tool coordinate system to be automatically converted, click “Add” and “Apply” to save the instruction. When adding PTP and LIN instructions, connect them to the Body to implement execution within the instruction, and the points in the tool coordinate system are automatically converted.

“ToolTrst” command node, parameters:

- Tool coordinate system: tool coordinate series list

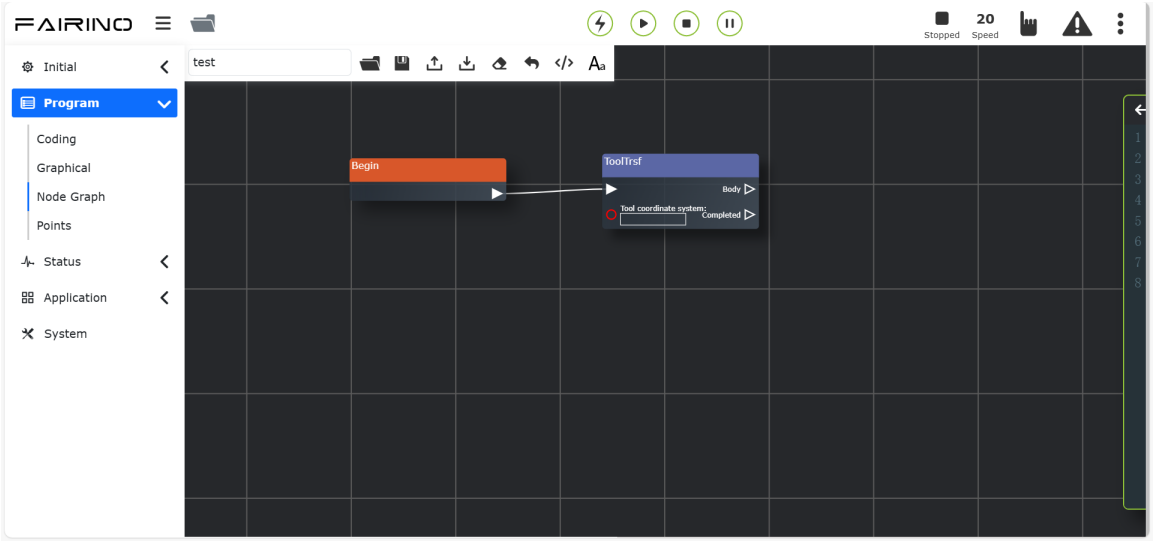


Figure 11.28-1 “ToolTrst” command node interface

1.3.11.29 Digital IO instruction node

Click the “Set DO”/”Get DI” command node to enter the node diagram editing interface

This instruction is an IO instruction, which is divided into two parts: setting IO (SetDO/SPLCSetDO) and getting IO (GetDI/SPLCGetDI).

1. “SetDO” command node, parameters:

- Port: Ctrl-DO0 ~ Ctrl-CO7 (blocking: SetDO, non-blocking: SPLCSetDO, [0~15]), End-DO0 ~ End-DO1 (blocking: SetToolDO, non-blocking: SPLCSetToolDO, [0~1])
- Status: false/true
- Whether to block: blocking/non-blocking
- Smooth trajectory: Break/Serious
- Whether to apply threads: No/Yes

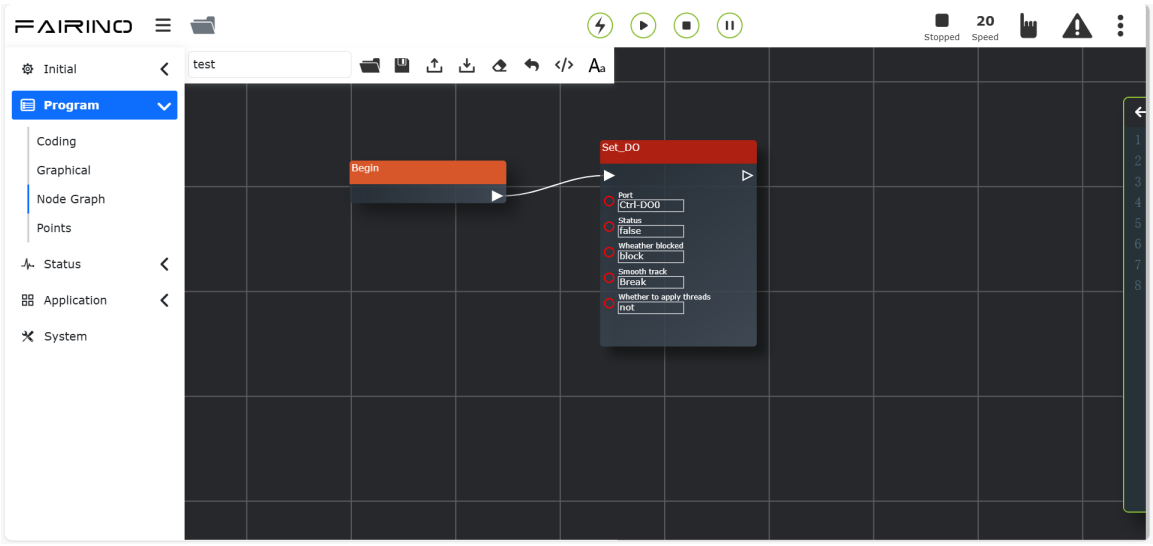


Figure 11.29-1 “SetDO” command node interface

2. “GetDI” command node, parameters:

- Port: Ctrl-DI0 ~ Ctrl-CI7 (blocking: GetDI, non-blocking: SPLCGetDI, [0~15]), End-DI0 ~ End-DI1 (blocking: GetToolDI, non-blocking: SPLCGetToolDI, [0~1])
- Whether to block: blocking/non-blocking
- Status: false/true
- Maximum waiting time (ms): 0 ~ 10000
- Whether to apply threads: No/Yes

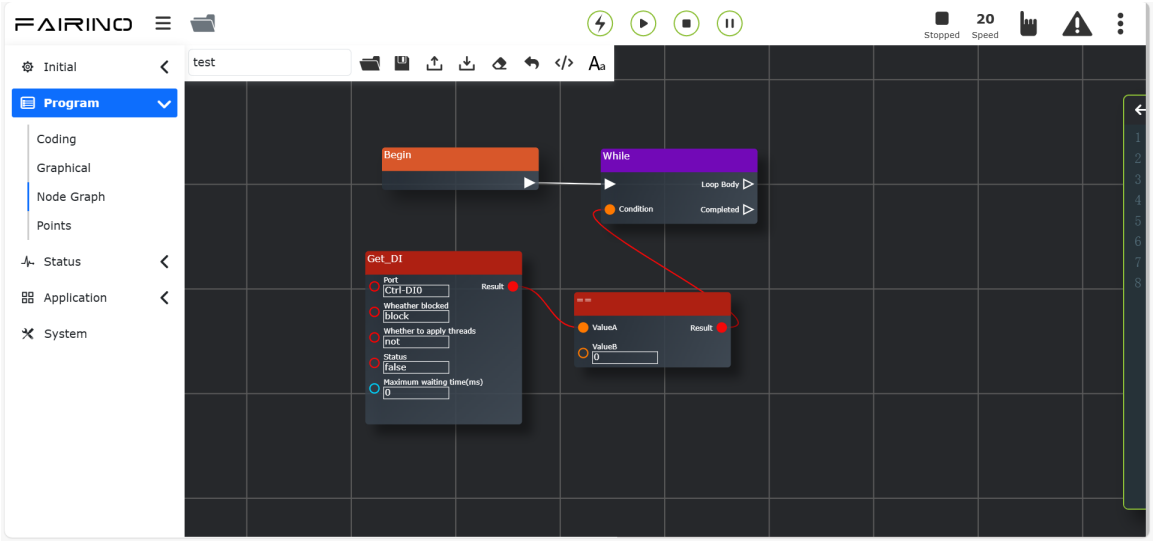


Figure 11.29-2 “GetDI” command node interface

1.3.11.30 Simulate AI commands

Click the “Set AO”/”Get AI” command node to enter the node graph editing interface.

In this command, it is divided into two functions: setting analog output (SetAO/SPLCSetAO) and getting analog input (GetAI/SPLCGetAI).

1. “SetAO” command node, parameters:

- Port: Ctrl-AO0 ~ Ctrl-AO1 (blocking: SetAO, non-blocking: SPLCSetAO, [0~1]), End-AO0 (blocking: SetToolAO, non-blocking: SPLCSetToolAO, [0])
- Value (%): 0 ~ 100
- Whether to block: blocking/non-blocking
- Whether to apply threads: No/Yes

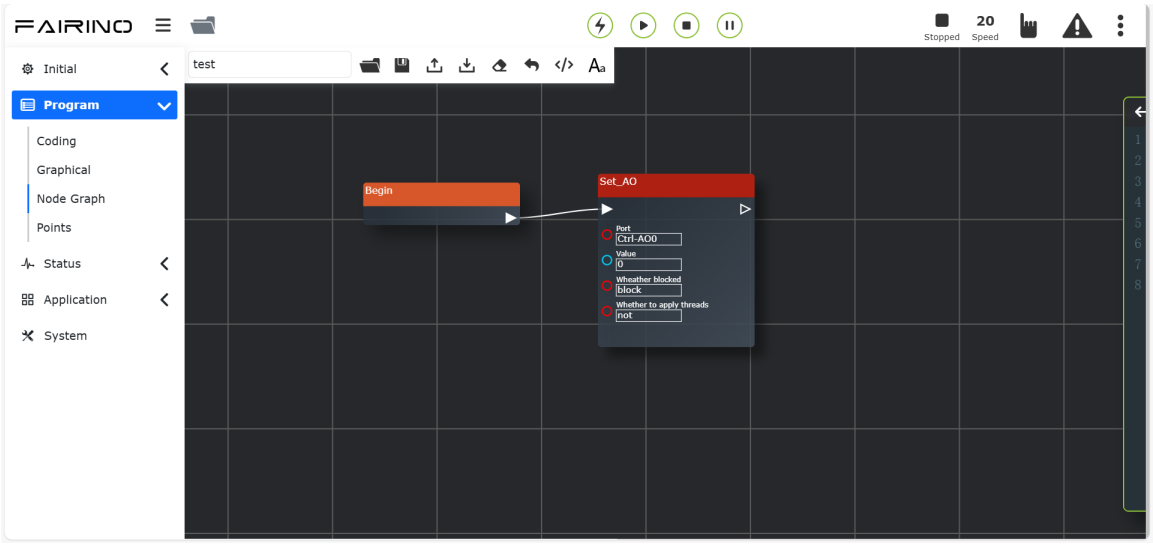


Figure 11.30-1 “SetAO” command node interface

2. “GetAI” command node, parameters:

- Port: Ctrl-AI0 ~ Ctrl-DI1 (blocking: GetAI, non-blocking: SPLCGetAI, [0~1]), End-AI0 (blocking: GetToolAI, non-blocking: SPLCGetToolAI, [0])
- Condition: greater than/less than
- Value (%): 0 ~ 100
- Maximum time (ms): 0 ~ 10000
- Whether to block: blocking/non-blocking
- Whether to apply threads: No/Yes

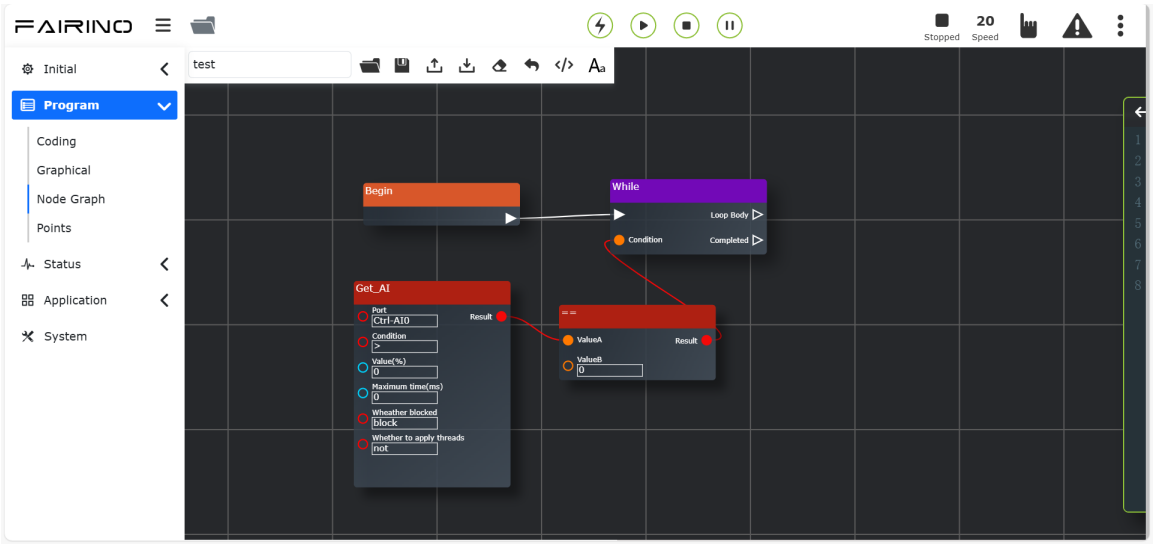


Figure 11.30-2 “GetAI” command node interface

1.3.11.31 Virtual IO command node

Click the “Configure Simulated External DI”/”Configure Simulated External AI” command node to enter the node diagram editing interface.

This instruction is a virtual IO control instruction that can set the simulated external DI and AI status and obtain the simulated DI and AI status.

1. “Configure simulated external DI” command node, parameters:

- Port: Vir-Ctrl-DI0 ~ Vir-Ctrl-DI15(SetVirtualDI,[0~15]), Vir-End-DI0 ~ Vir-End-DI1(SetVirtualToolDI,[1~2])
- Status: false/true

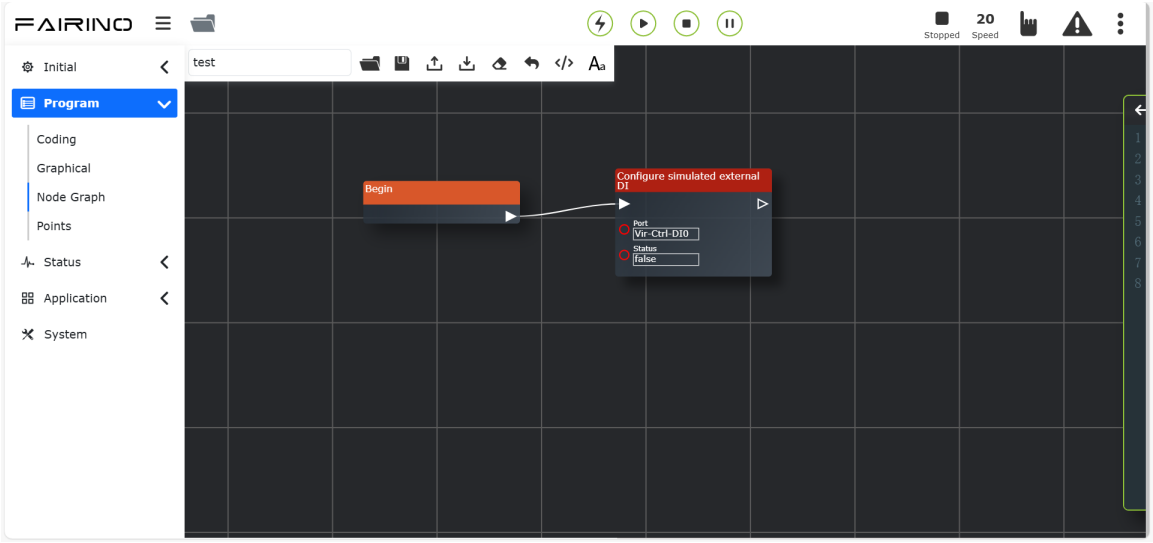


Figure 11.31-1 “Configure simulated external DI” command node interface

2. “Configure simulated external AI” command node, parameters:

- Port: Vir-Ctrl-AI0 ~ Vir-Ctrl-AI0(SetVirtualAI,[0~1]), Vir-End-AI0(SetVirtualToolAI,[0])

- Value (v/ma): 0 ~ 20

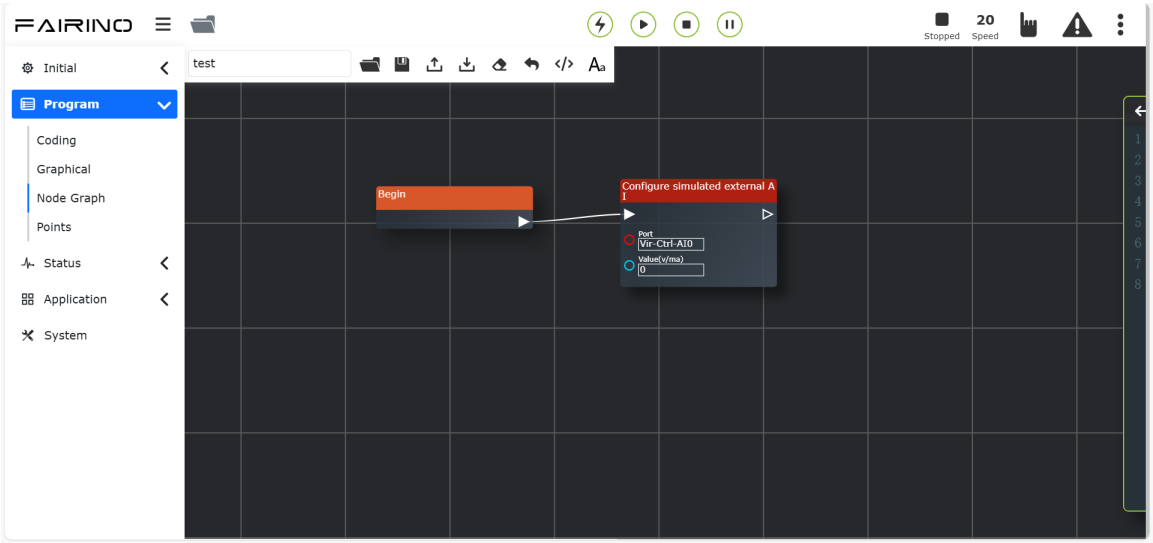


Figure 11.31-2 “Configure simulated external AI” command node interface

1.3.11.32 Extended IO command node

Click the “Obtain simulated external DI”/”Obtain simulated external AI” command node to enter the node diagram editing interface.

Aux-IO is a command function used by the robot to communicate with the PLC to control external expansion IO. It requires the robot to establish UDP communication with the PLC.

1. “Obtain simulated external DI” command node, parameters:

- Port: Vir-Ctrl-DI0 ~ Vir-Ctrl-DI15(GetVirtualDI,[0~15]), Vir-End-DI0 ~ Vir-End-DI1(GetVirtualToolDI,[1~2])

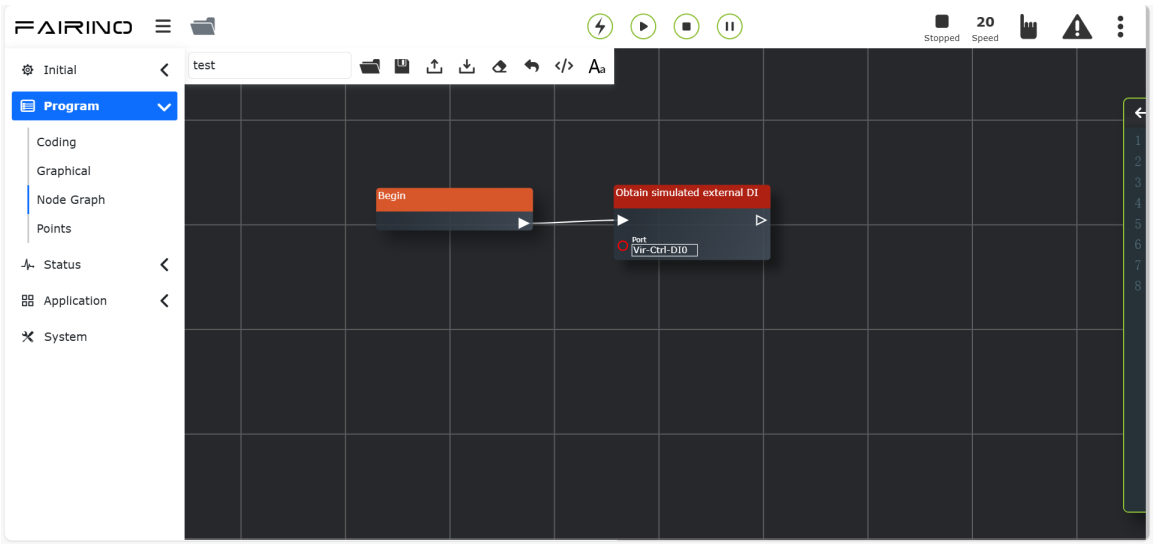


Figure 11.32-1 “Obtain simulated external DI” command node interface

2. “Obtain simulated external AI” command node, parameters:

- Port: Vir-Ctrl-AI0 ~ Vir-Ctrl-AI0(GetVirtualAI,[0~1]), Vir-End-AI0(GetVirtualToolAI,[0])

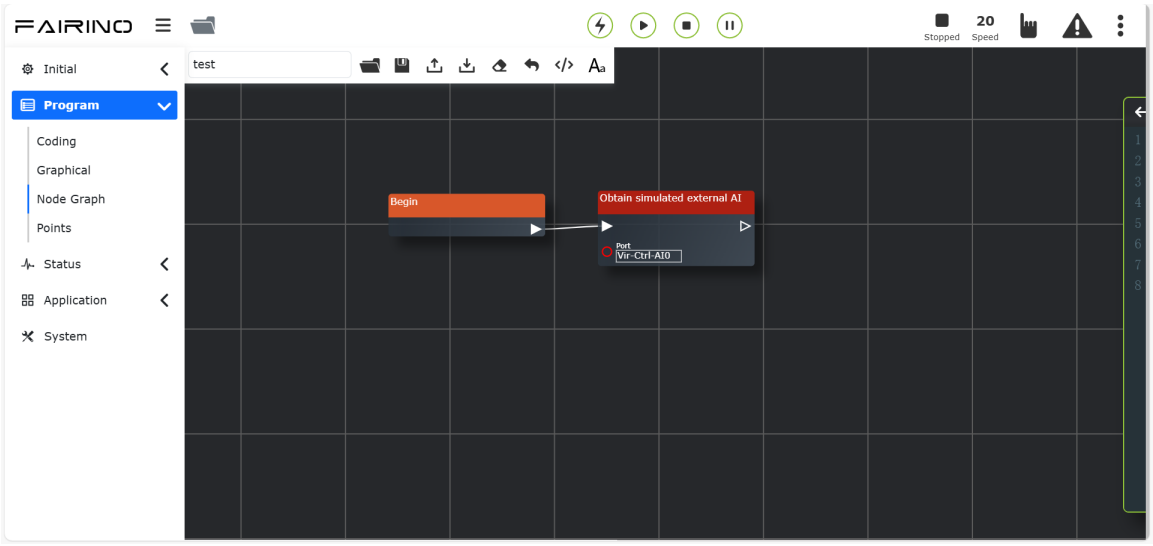


Figure 11.32-2 “Obtain simulated external AI” command node interface

3. “Configure UDP communication” command node, parameters:

- ip: ip address
- port: port number
- Communication cycle (ms): 0 ~ 10000

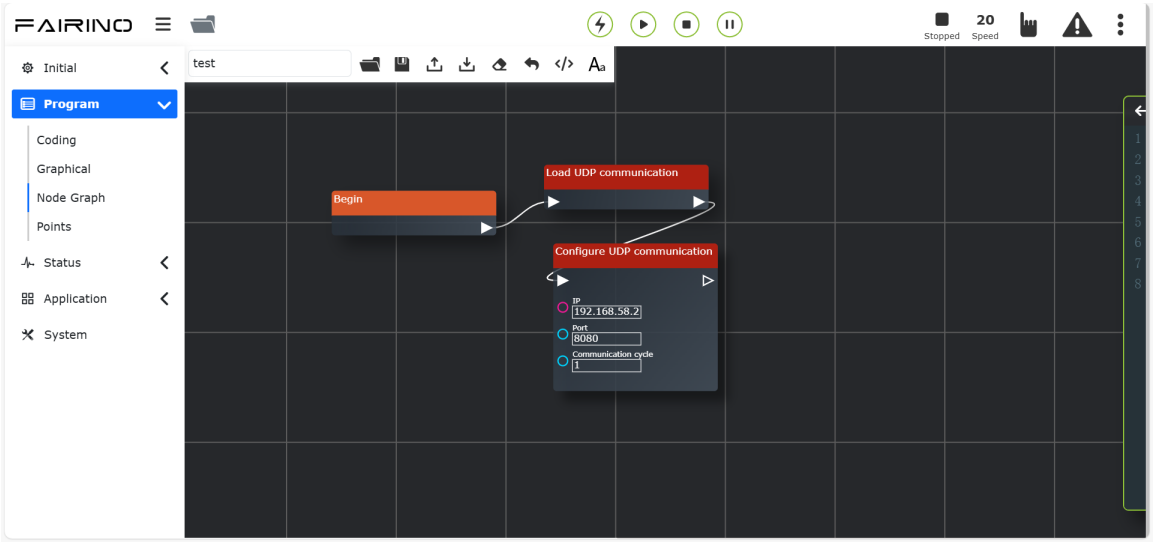


Figure 11.32-3 “Configure UDP Communication” command node interface

1.3.11.33 MoveDO instruction

Click the “MoveDO” command node to enter the node graph editing interface.

This instruction implements the function of continuously outputting DO signals according to the set interval during linear motion.

“Motion DO continuous output” command node, parameters:

- Port: Ctrl-DO0 ~ Ctrl-DO0(MoveDOStart,[0~15]), End-DO1(MoveDOStart,[0~1])
- Setting interval (mm): 0 ~ 500
- Output pulse duty cycle (%): 0 ~ 99

“Motion DO single output” command node, parameters:

- Port: Ctrl-DO0 ~ Ctrl-DO0(MoveDOOnceStart,[0~15]), End-DO1(MoveDOOnceStart,[0~1])
- Output mode: constant speed section output/free configuration
- Setting time (ms): 0 ~ 1000 (The default output mode of the constant speed segment is -1)
- Reset time (ms): 0 ~ 1000 (The default output mode of the constant speed segment is -1)

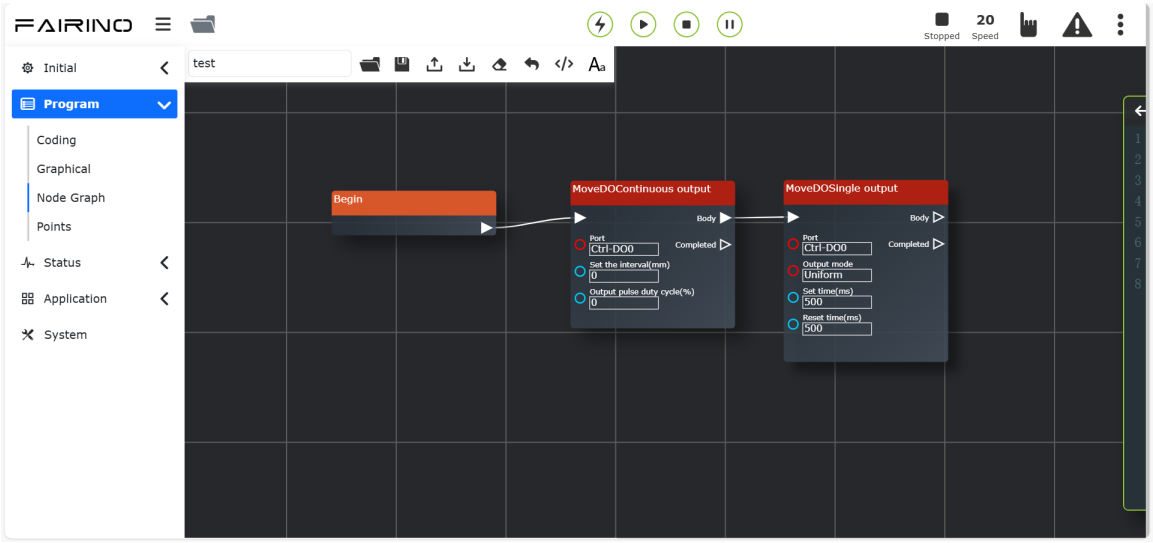


Figure 11.33-1 “Motion DO single/continuous output” command node interface

1.3.11.34 ToolList instruction

Click the “SetToolList”/”SetWobToolList” related command node to enter the node diagram editing interface.

In this command, it is divided into two functions: “SetToolList” and “SetWobToolList”.

Select the tool coordinate system name and click “Apply” to add this instruction to the program. When the program runs this statement, the tool coordinate system of the robot will be set.

1. “SetToolList” command node, parameters:

- Tool coordinate system name: toolcoord1 ~ toolcoord19(SetToolList, [0~19]), etoolcoord0 ~ etoolcoord14(SetExToolList, [0~14])

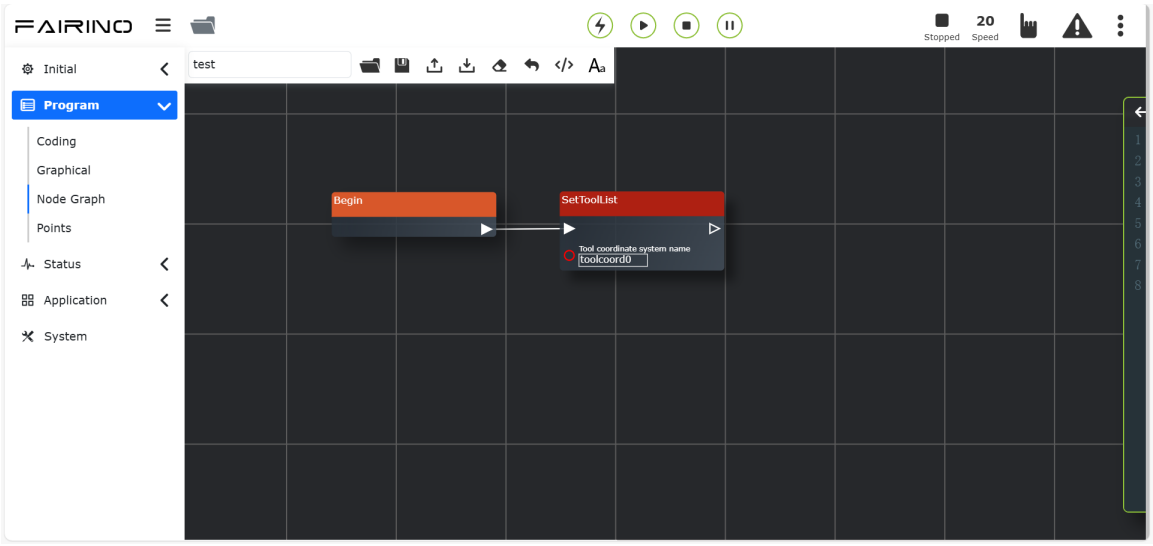


Figure 11.34-1 “SetToolList” command node interface

2. “SetWobToolList” command node, parameters:

- Workpiece coordinate system name: wobjcoord1 ~ wobjcoord14

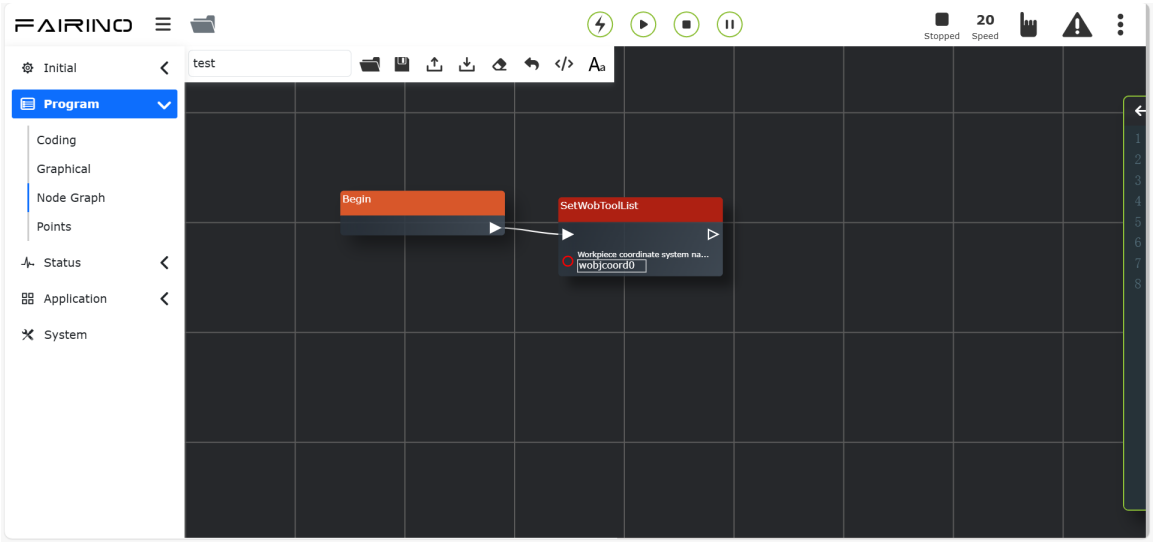


Figure 11.34-2 “SetWobToolList” command node interface

1.3.11.35 Mode instruction

Click the “Mode” command node to enter the node graph programming interface

This instruction can switch the robot to manual mode. It is usually added at the end of a program so that the user can automatically switch the robot to manual mode and drag the robot after the program is finished.

“Mode” command node, parameters:

- Mode switch: manual mode

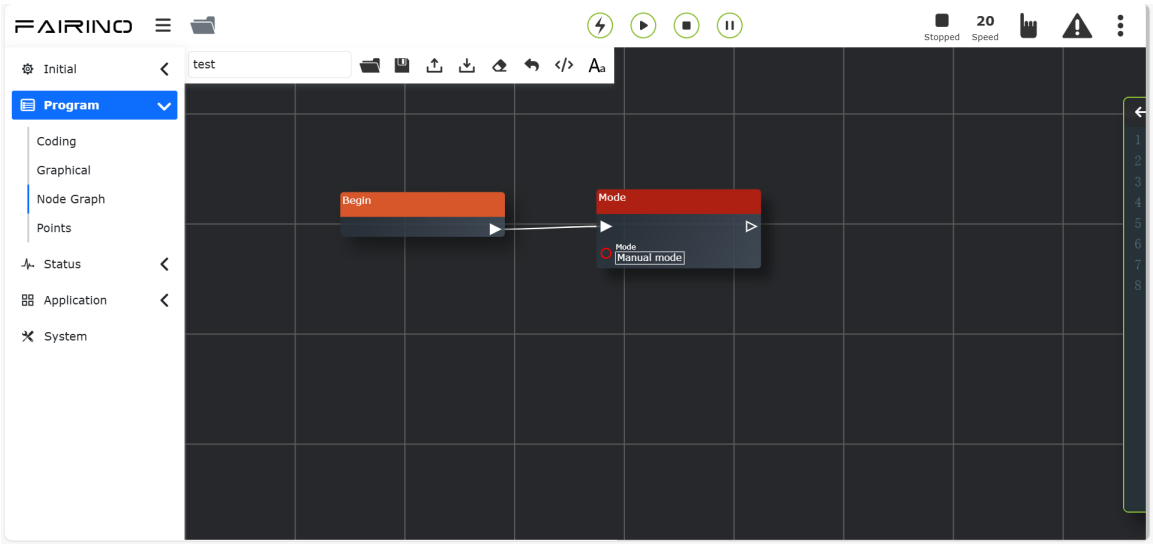


Figure 11.35-1 “Mode” command node interface

1.3.11.36 Collision instruction

Click the “Collision” command node to enter the node graph programming interface

This command sets the collision level. Through this command, the collision level of each axis can be adjusted in real time while the program is running, allowing for more flexible deployment of application scenarios.

“Collision” command node, parameters:

- Standard Level: Standard Level/Customized Percentage
- joint1-joint6(N): 0 ~ 100, collision threshold, array type

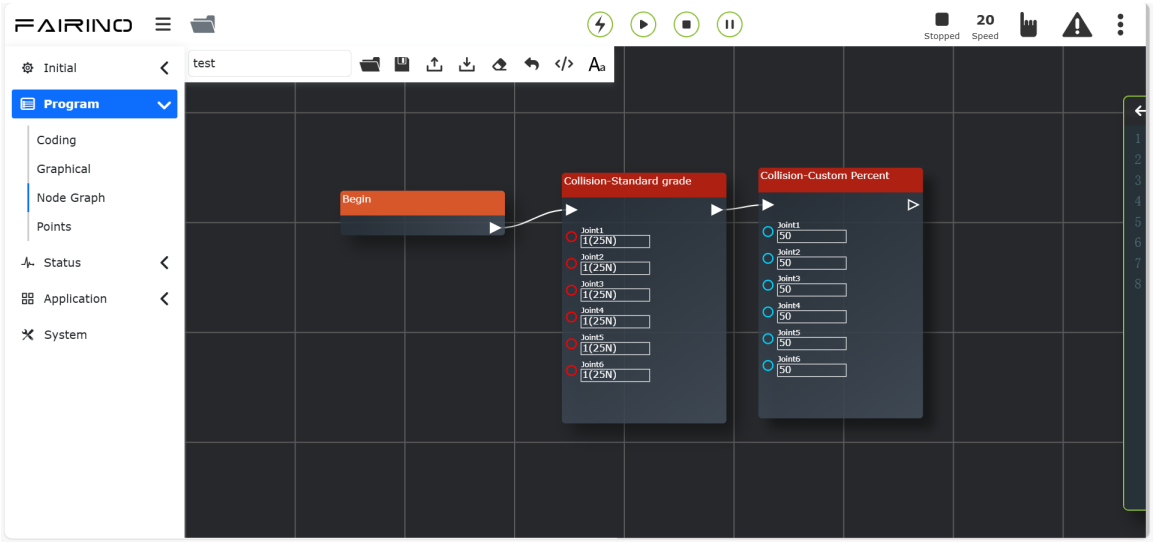


Figure 11.36-1 “Collision” command node interface

1.3.11.37 Acc instruction

Click the “Acc” command node to enter the node graph programming interface

The “Acc” command is a function that allows the robot’s acceleration to be set independently. By adjusting the motion command acceleration scaling factor, the acceleration and deceleration time can be increased or decreased, and the robot’s action beat time can be adjusted.

“Acc” command node, parameters:

- Acceleration percentage (%): 0 ~ 100

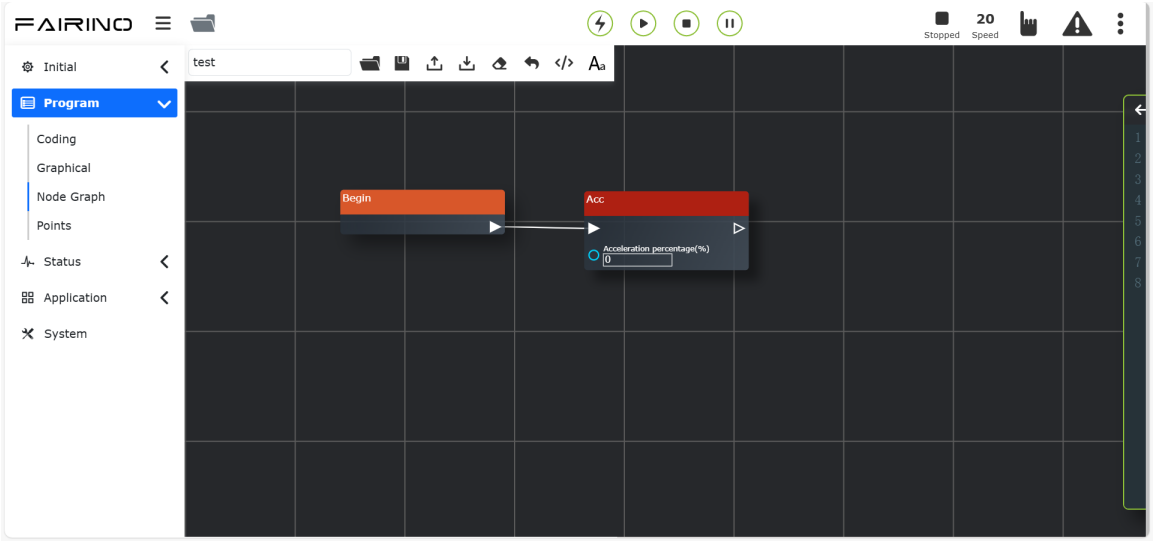


Figure 11.37-1 “Acc” command node interface

1.3.11.38 Gripper instruction

The command is divided into “gripper movement”, “gripper activation” and “gripper reset”.

In the command, the number of the gripper that has been configured and activated is displayed. The settings for the opening and closing, opening and closing speed and opening and closing torque of the gripper. The value is a percentage. Whether to block the function option. If blocking is selected, the gripper movement needs to wait for the previous one. The motion command is executed only after it is executed. Select non-blocking, that is, the gripper motion is parallel to the previous motion command.

“Gripper Movement” node, parameters:

- Gripper number: The number of the activated gripper
- Clamp position: 0~100
- Opening and closing speed: 0~100
- Opening and closing torque: 0~100
- Maximum time (ms): 0~30000
- Whether to block: false/true

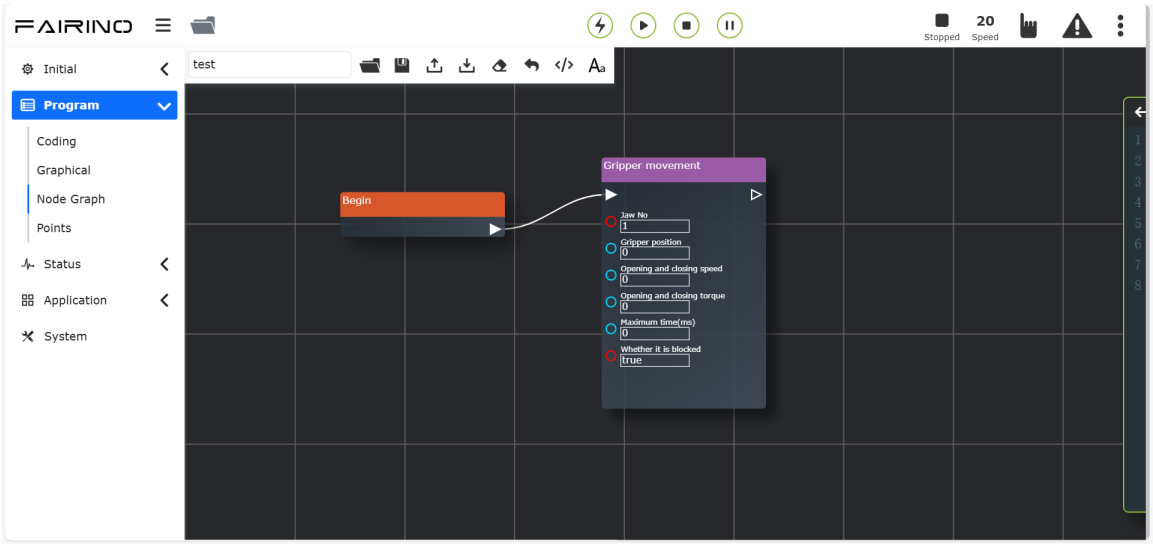


Figure 11.38-1 “Gripper Movement” node interface

The gripper reset command displays the configured gripper number. You can add the gripper reset command to the program.

“Gripper reset” node, parameters:

- Gripper number: The number of the activated gripper

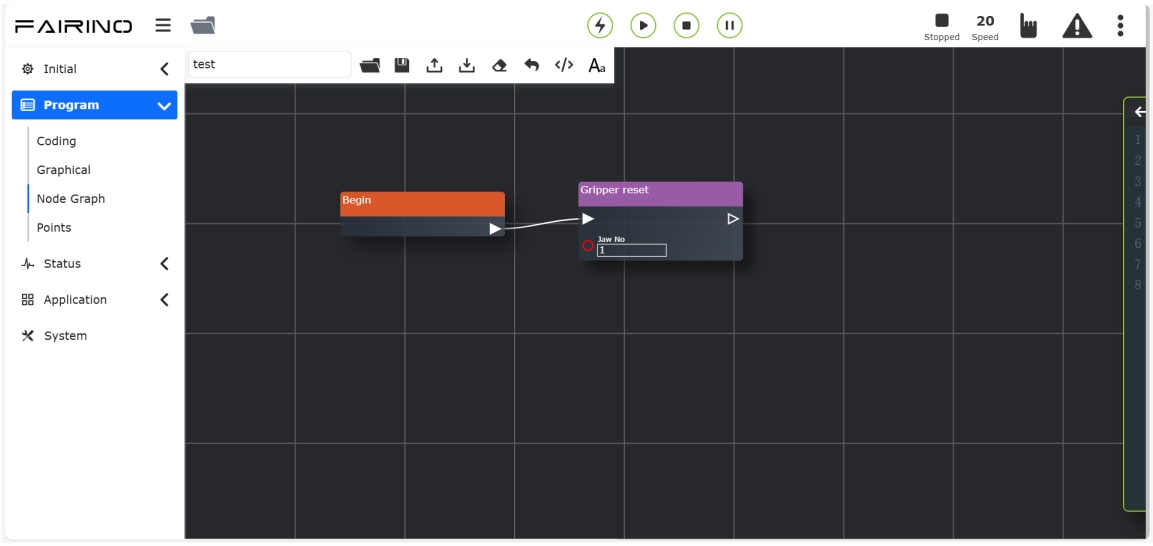


Figure 11.38-2 “Gripper reset” node interface

The gripper activation command displays the configured gripper number. You can add the gripper activation command to the program.

“Gripper activation” node, parameters:

- Gripper number: The number of the activated gripper

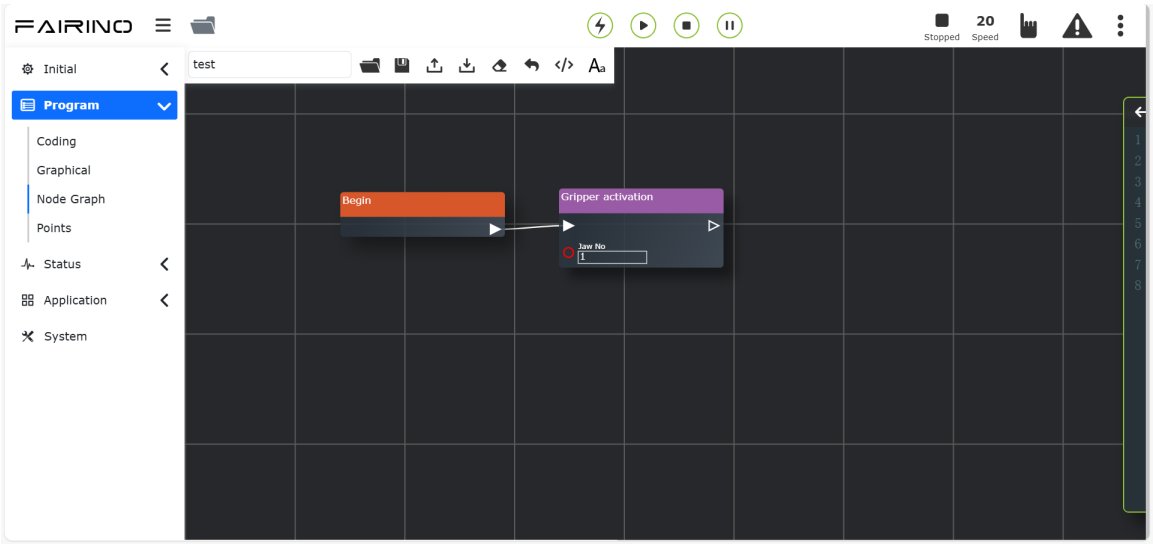


Figure 11.38-3 “Gripper activation” node interface

1.3.11.39 Spray instruction

This command is a spray-related command that controls the spray gun to “start spraying”, “stop spraying”, “start clearing the gun” and “stop clearing the gun”. When editing the relevant nodes of this program, you need to confirm that the spray gun peripherals have been configured, otherwise it cannot be saved. See the Robot Peripherals chapter for details.

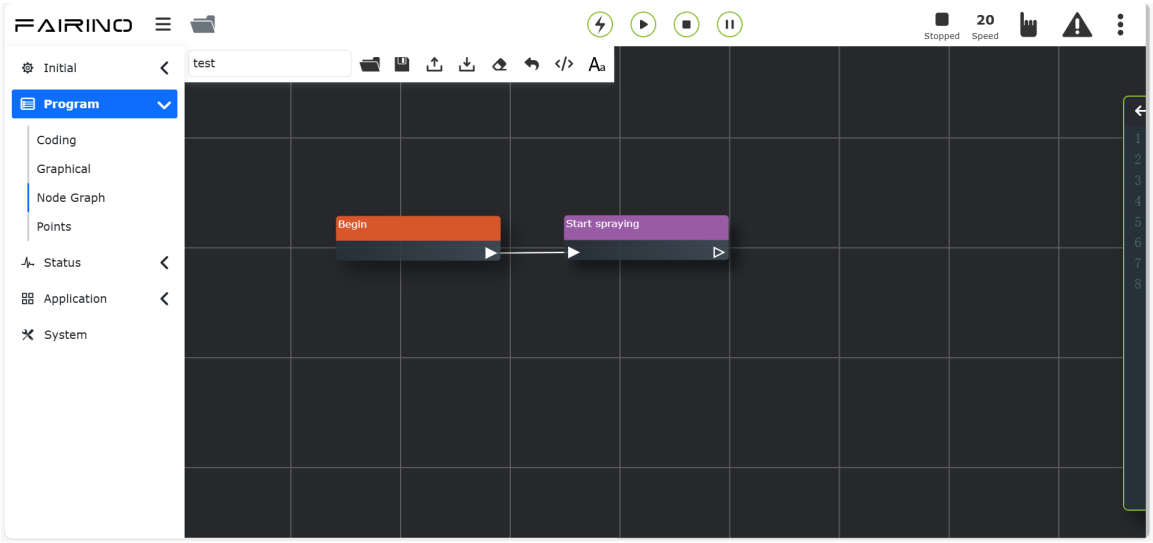


Figure 11.39-1 “Start spraying” command node interface

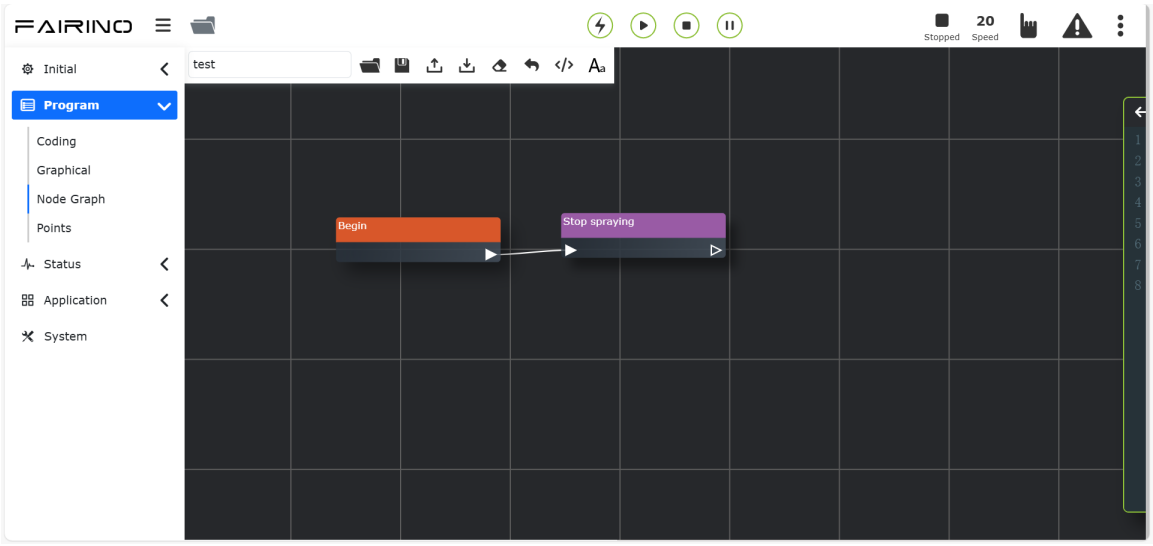


Figure 11.39-2 “Stop spraying” command node interface

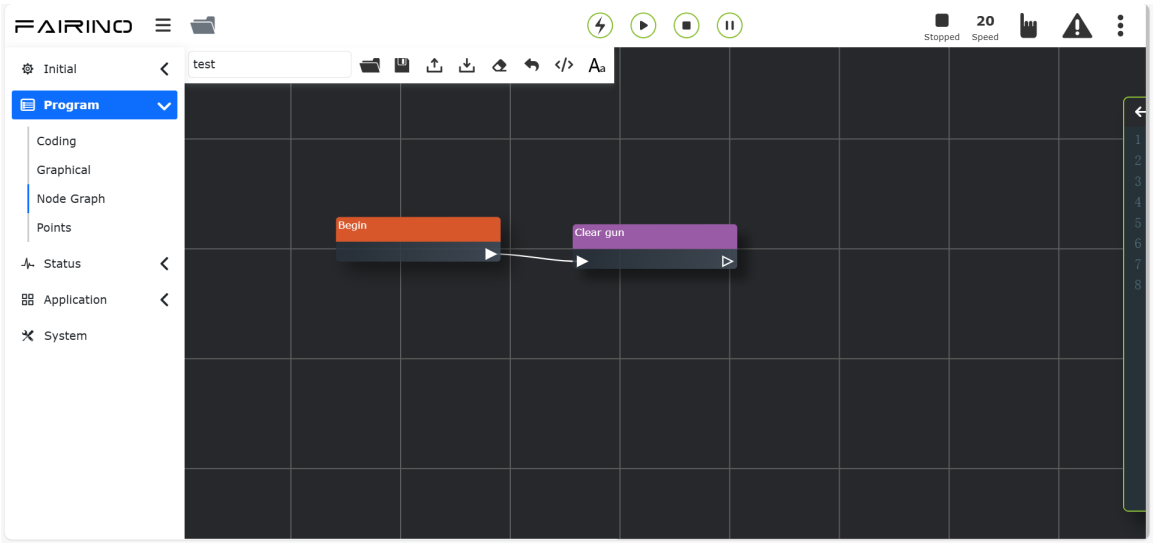


Figure 11.39-3 “Clearing gun” command node interface

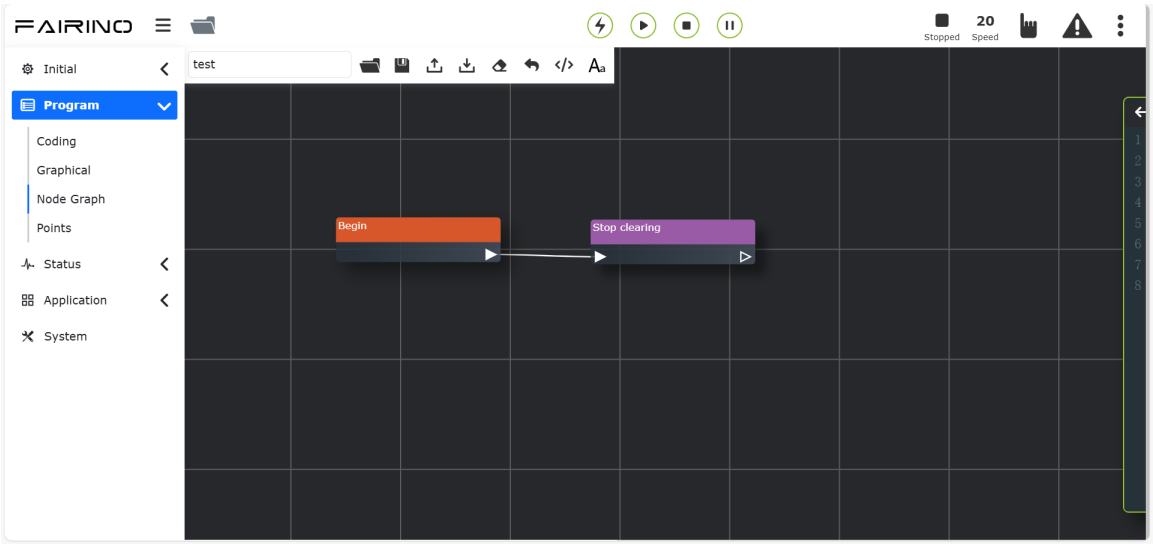


Figure 11.39-4 “Stop clearing” command node interface

1.3.11.40 Extended axis instructions (controller + PLC)

This instruction is aimed at scenarios where external axes are used. Used in combination with the PTP instruction, it can decompose the movement of a point in space in the X-axis direction into external axis motion. Select the external axis number, select synchronization as the motion mode, and select the point you want to reach.

It is divided into UDP communication loading/configuration, asynchronous movement, synchronous PTP/LIN movement, synchronous ARC movement, zero return command and enable command.

“Extended axis UDP communication configuration” command node, enter the IP address, port number and communication cycle.

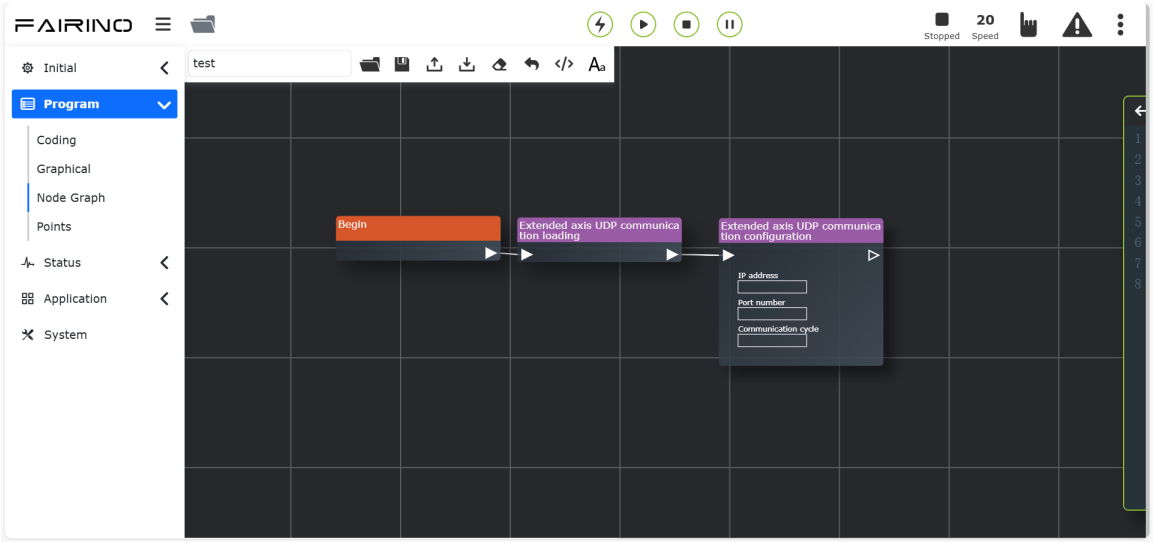


Figure 11.40-1 “Extended axis UDP communication configuration” command node interface

“Extended axis asynchronous motion” command node, parameters:

- Point name: Teaching point
- Debugging speed (%): 0~100

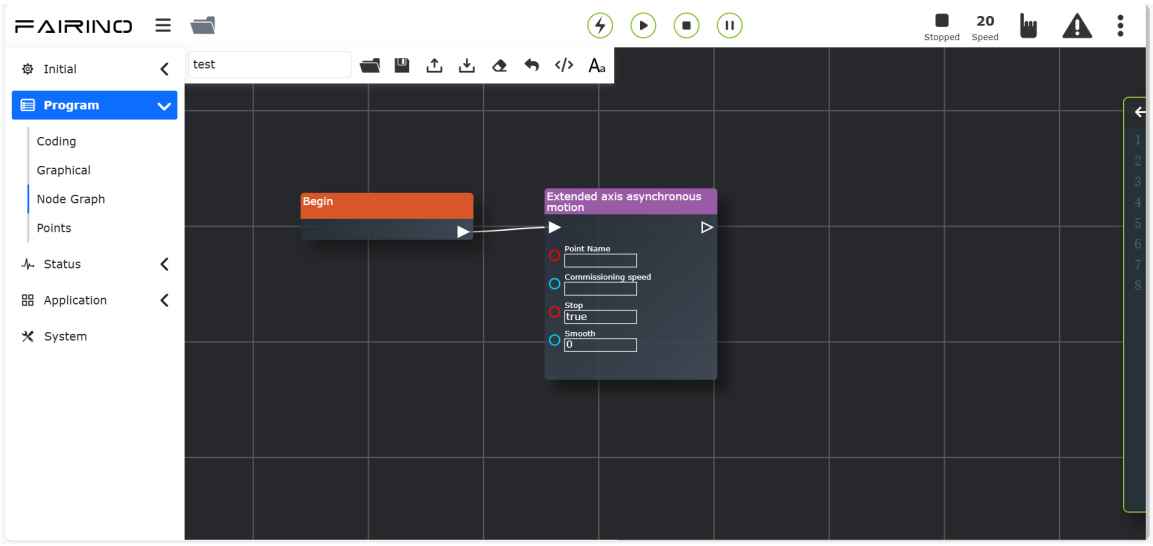


Figure 11.40-2 “Extended axis asynchronous motion” command node interface

“Synchronized PTP/LIN motion” command node, parameters:

- Sport selection: PTP/LIN
- Point name: Teaching point
- Debugging speed (%): 0~100

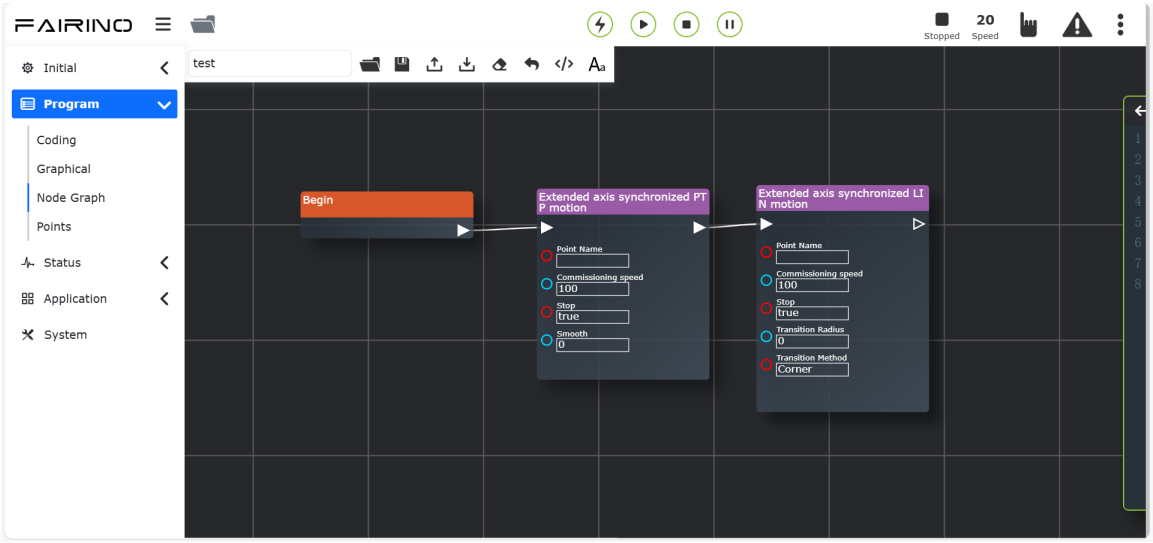


Figure 11.40-3 “Synchronized PTP/LIN motion” command node interface

“Synchronized ARC motion” command node, the default motion mode is ARC, parameters:

- Point name: Teaching point
- Debugging speed (%): 0~100

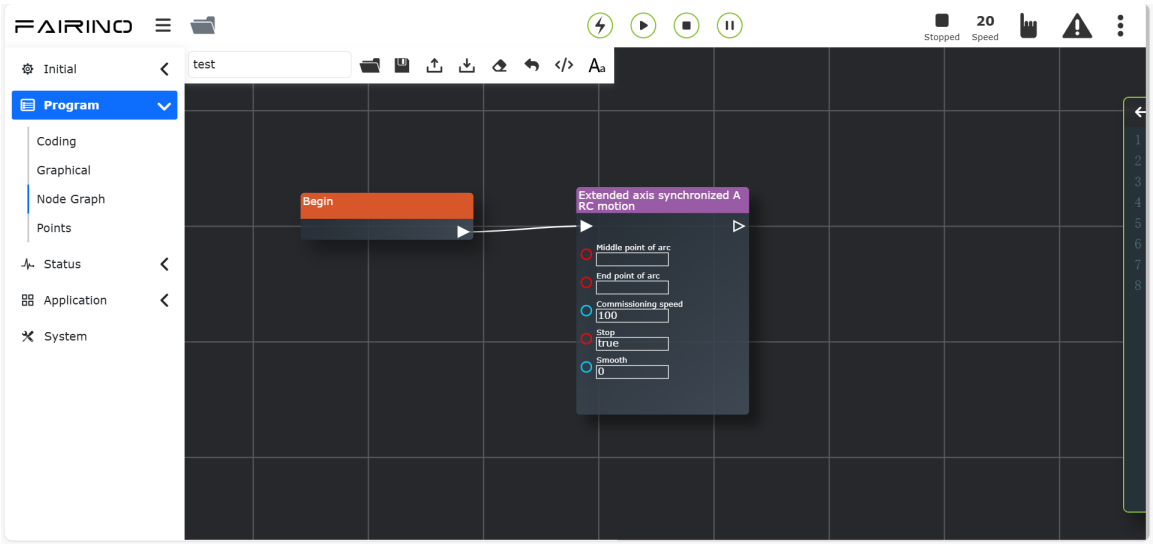


Figure 11.40-4 “Synchronized ARC motion” command node interface

“Return to zero” command node, parameters:

- Expansion axis number: 1~4
- Zero return method: current position zero return/negative limit zero return/positive limit zero return
- Homing speed: 0~2000, default bit 5
- Zero point hoop speed: 0~2000, default is 1

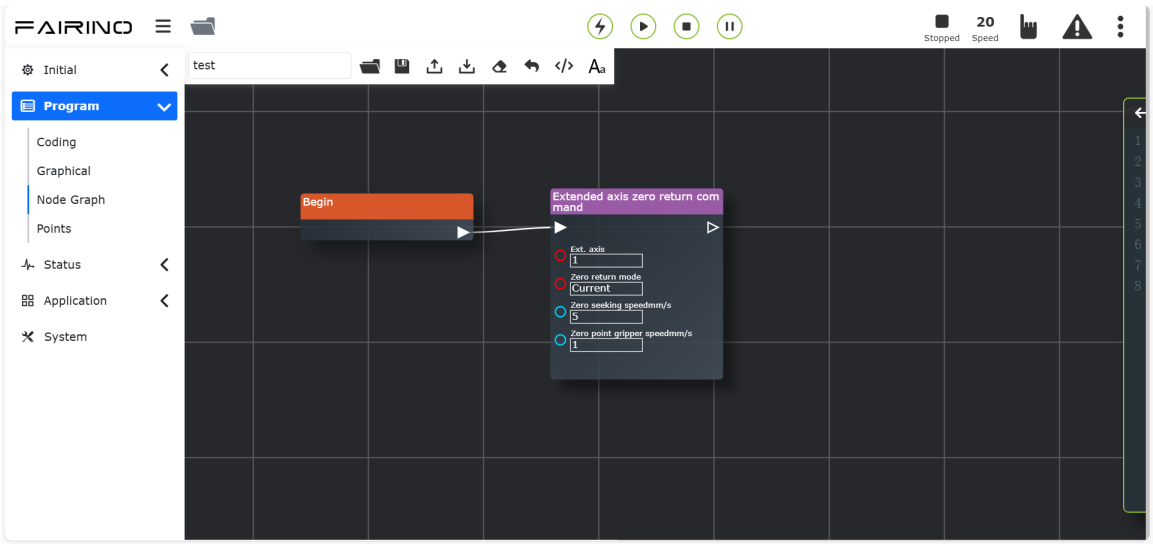


Figure 11.40-5 “Return to zero” command node interface

“Enable” command node,,parameters:

- Expansion axis number: 1~4

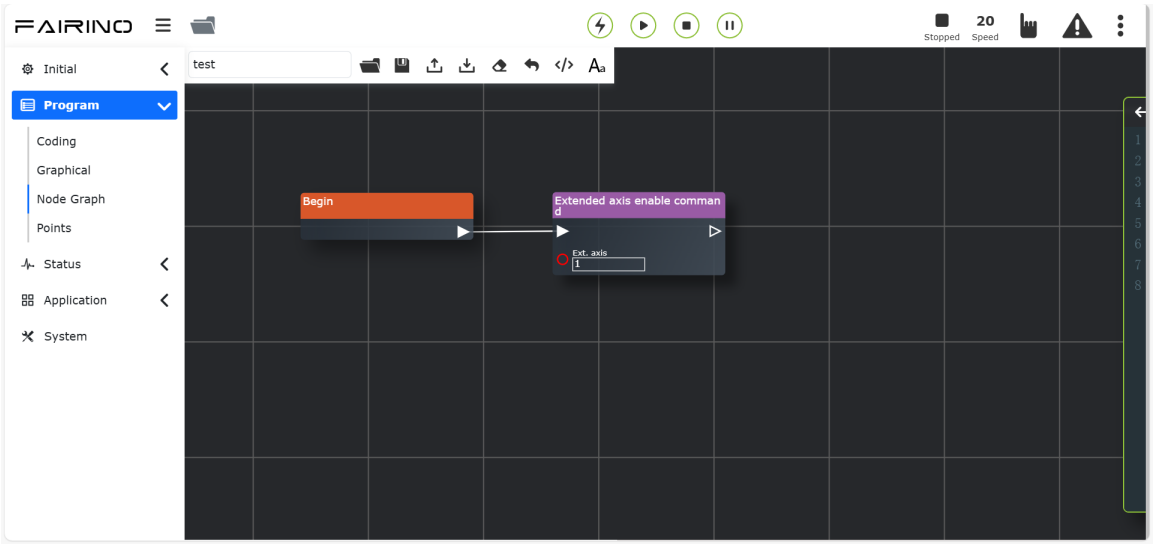


Figure 11.40-6 “Enable” command node interface

1.3.11.41 Extended axis instructions (controller + servo drive)

This command can configure extended axis parameters. Set different parameters according to different control modes. The configured expansion axis can be set to its zero point.

It is divided into servo ID, control mode, servo enable and servo zero return; the control mode is further divided into position mode and speed mode. These two nodes need to be used in conjunction with the control mode, otherwise adding them separately will not take effect.

“Servo ID” command node, parameters:

- Servo ID: 1~15

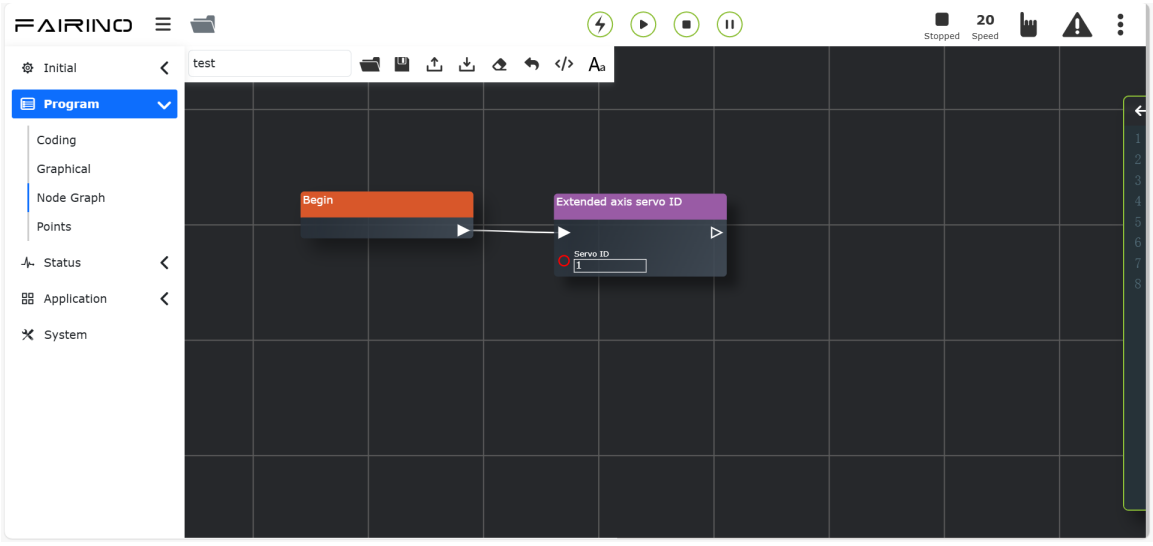


Figure 11.41-1 “Servo ID” command node interface

“Control mode” command node, parameters:

- Servo ID: 1~15
- Control mode: position mode/speed mode

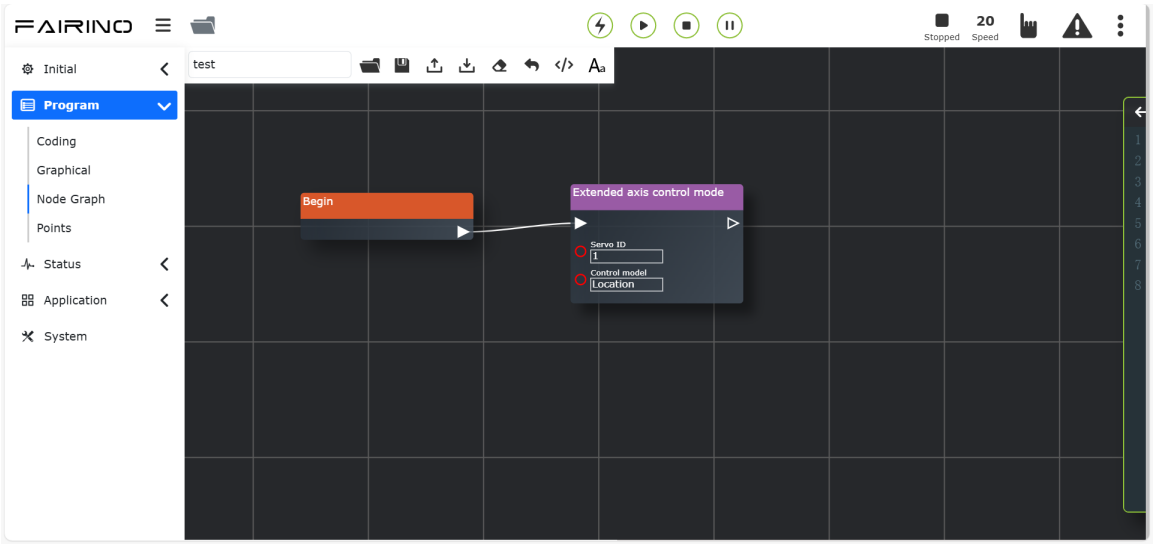


Figure 11.41-2 “Control Mode” command node interface

“Servo enable” command node, parameters:

- Servo ID: 1~15
- Servo enable: servo enable/remove enable

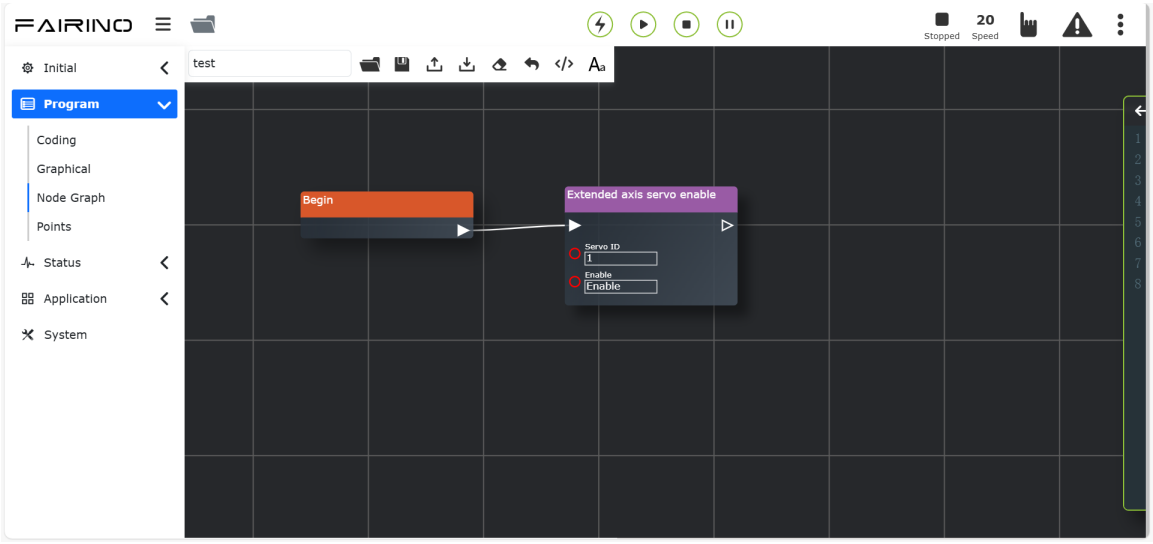


Figure 11.41-3 “Servo enable” command node interface

“Servo return to zero” command node, parameters:

- Servo ID: 1~15
- Zero return method: current position zero return/negative limit zero return/positive limit zero return
- Homing speed: 0~2000, default bit 5
- Zero point hoop speed: 0~2000, default is 1
- Acceleration percentage: 1-100

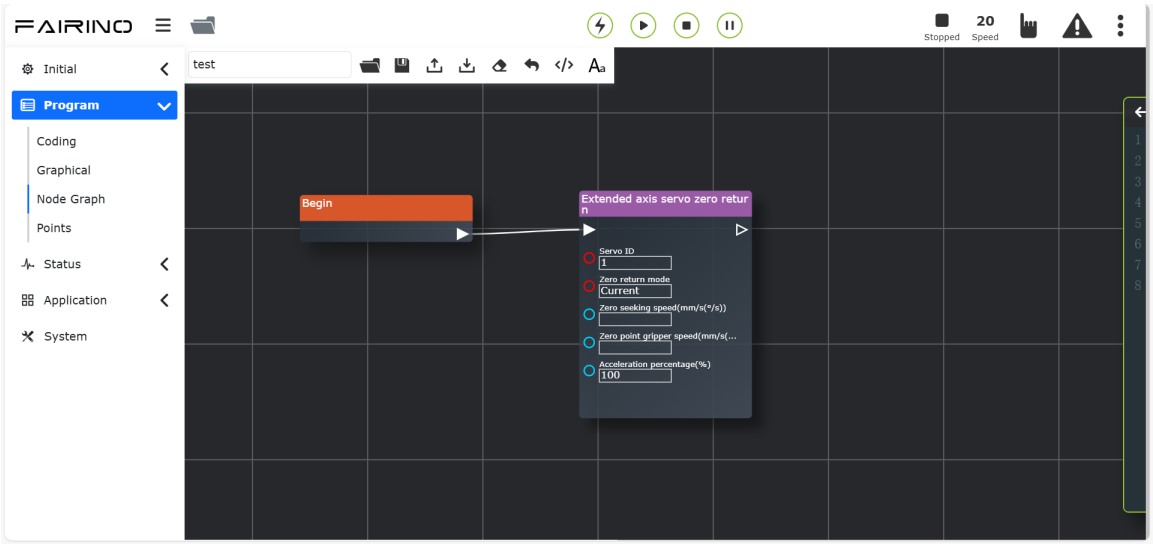


Figure 11.41-4 “Servo return to zero” command node interface

“Position mode” command node, parameters:

- Servo ID: 1~15
- Target location: unlimited
- Homing speed: unlimited
- Acceleration percentage: 1-100

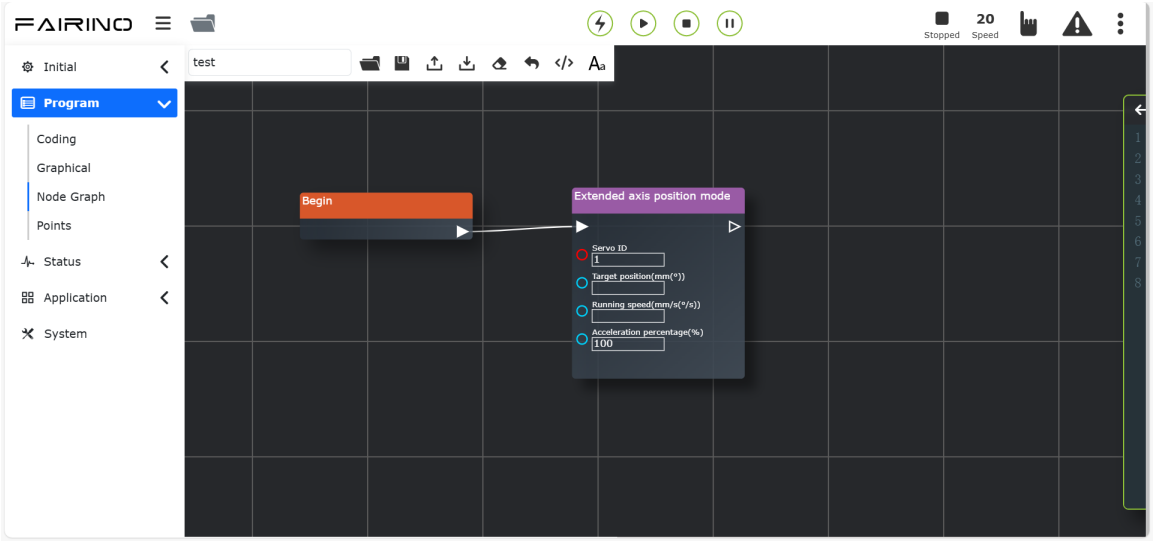


Figure 11.41-5 “Position Mode” command node interface

“Speed mode” command node, parameters:

- Servo ID: 1~15
- Target speed: unlimited
- Acceleration percentage: 1-100

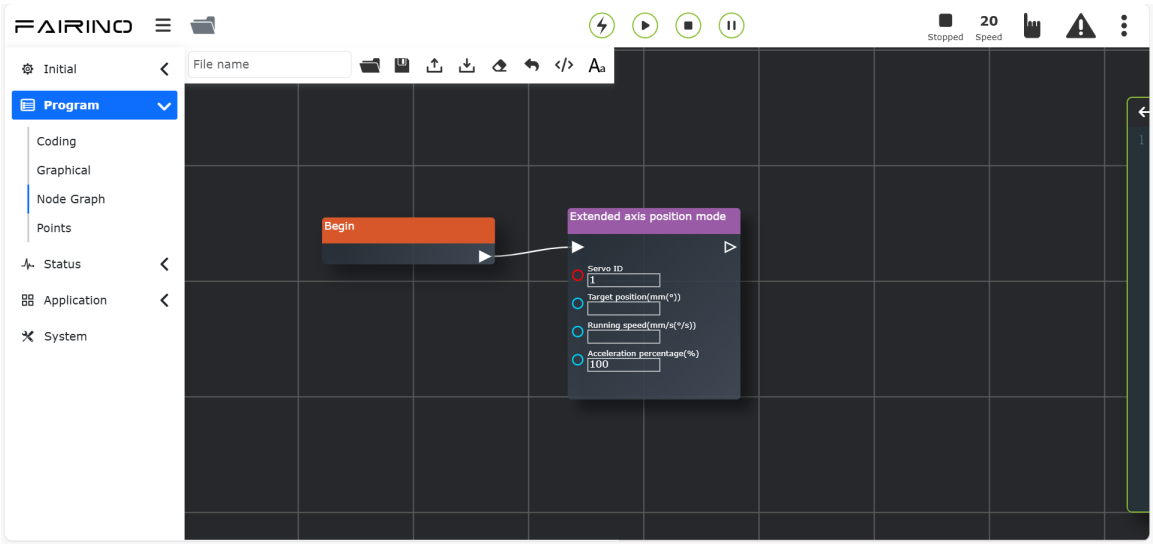


Figure 11.41-6 “Speed Mode” command node interface

1.3.11.42 Conveyor belt instruction

This instruction includes four commands: Conveyor belt IO real-time detection, position real-time detection, tracking on and tracking off. See the Robot Peripherals chapter for details.

“Conveyor belt IO real-time detection” command node, parameters:

- Maximum waiting time: 0~10000

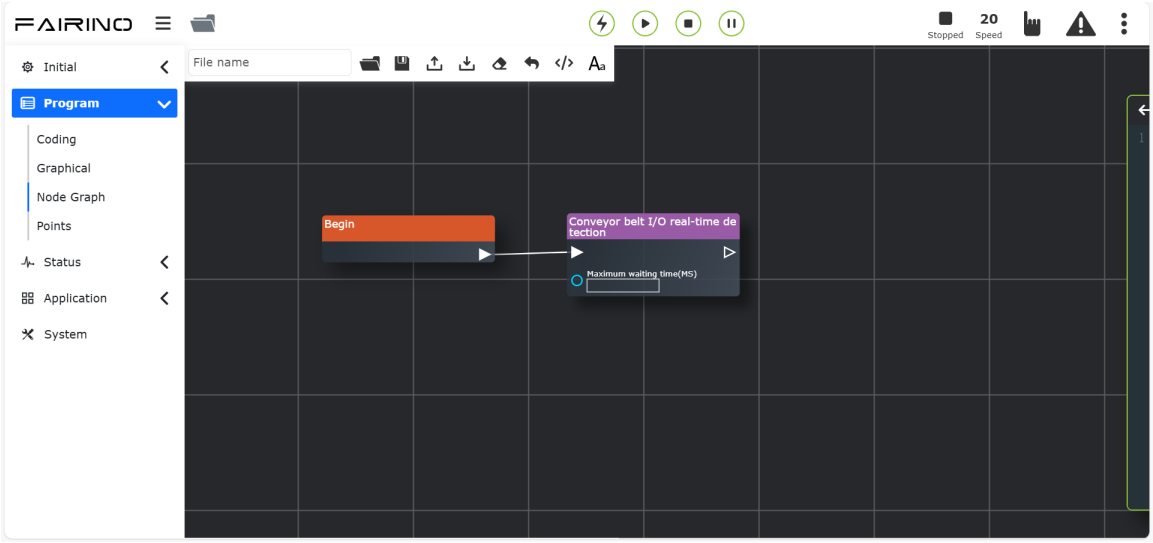


Figure 11.42-1 “Conveyor belt IO real-time detection” command node interface

“Real-time detection of conveyor belt position” command node, parameters:

- Operating mode: Tracking capture/Tracking motion/TPD tracking

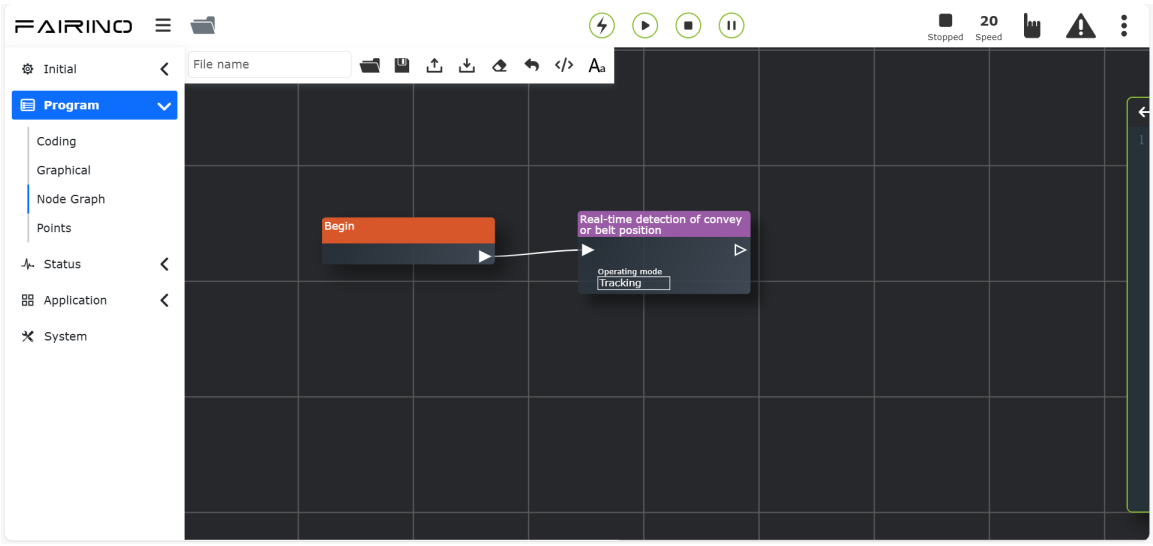


Figure 11.42-2 “Real-time detection of conveyor belt position” command node interface

“Conveyor tracking enabled” command node, parameters:

- Operating mode: Tracking capture/Tracking movement/TPD tracking

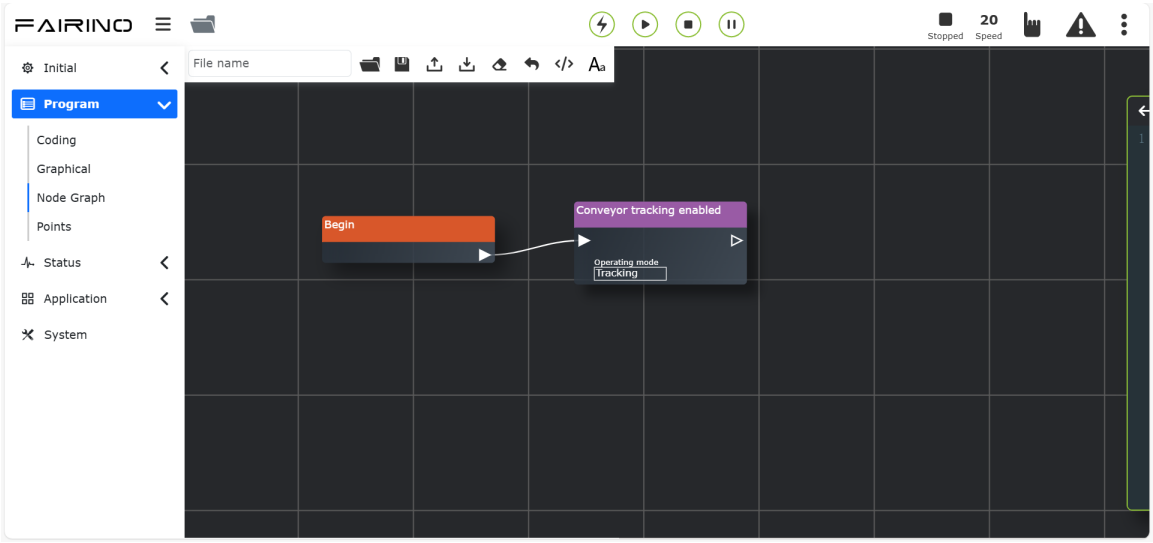


Figure 11.42-3 “Conveyor tracking enabled” command node interface

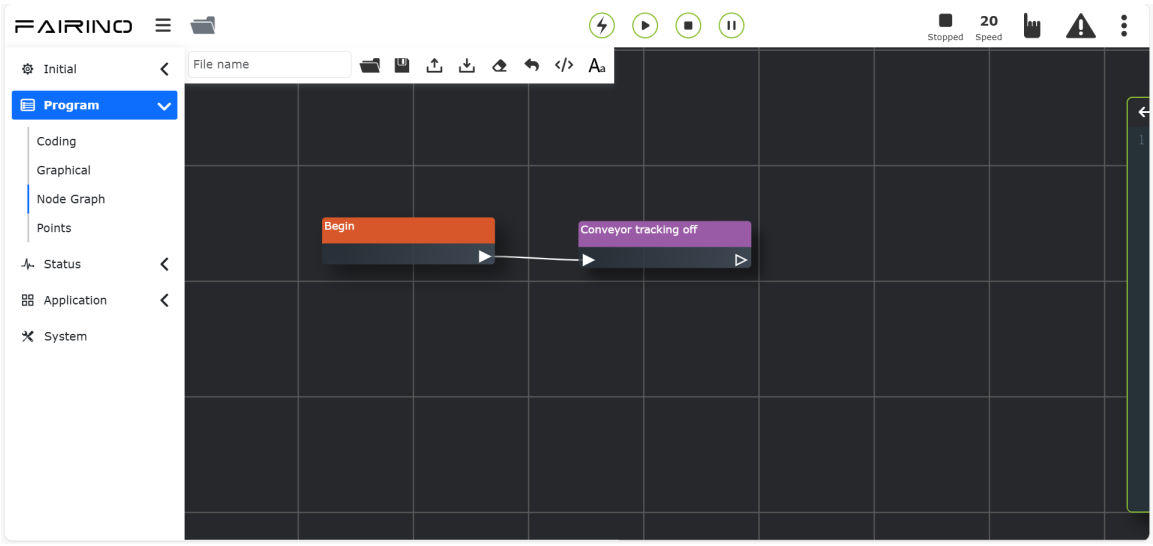


Figure 11.42-4 “Conveyor tracking off” command node interface

1.3.11.43 Polish instruction

This command is used for polishing scenes. When using it, you need to uninstall and then load the driver, and then set the polishing device to be enabled. Then set the rotation speed, contact force, extension distance and control mode of the grinding equipment. At the same time, you can clear the grinding equipment errors and clear the equipment force sensor.

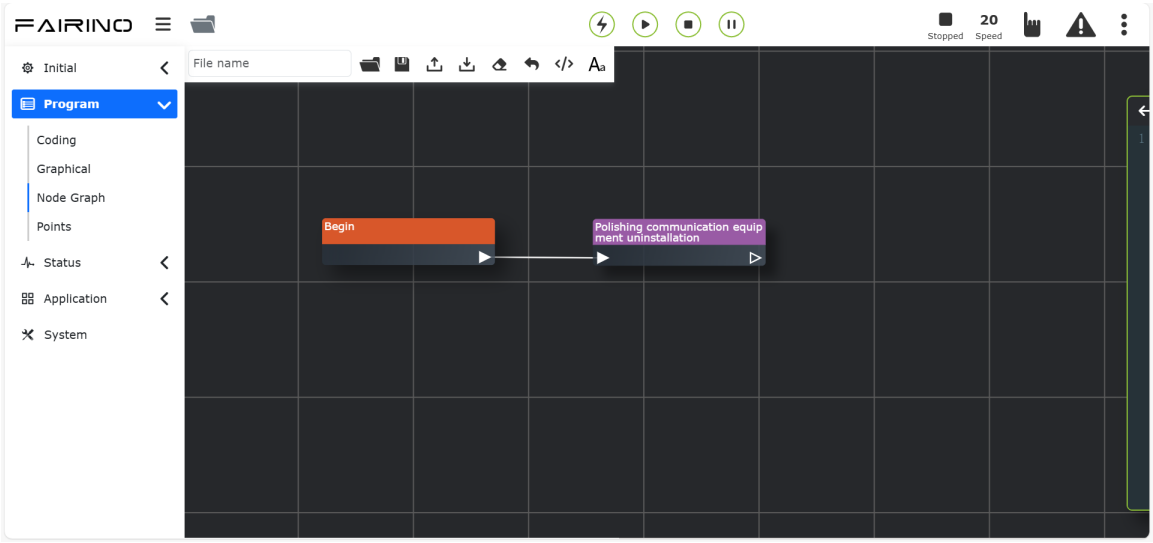


Figure 11.43-1 “Polishing communication equipment uninstallation” command node interface

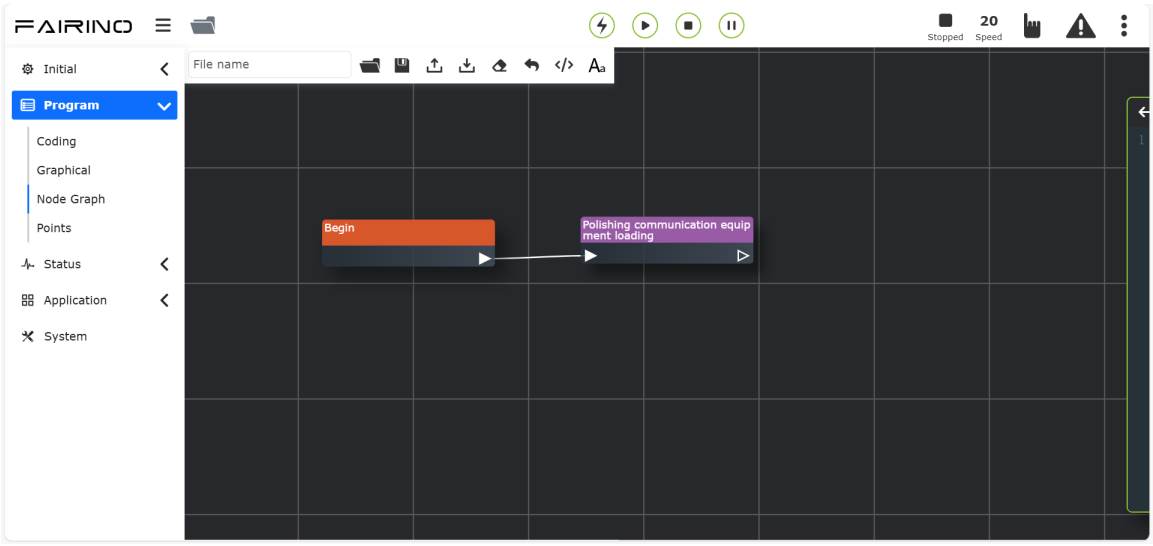


Figure 11.43-2 “Polishing communication equipment loading” command node interface

“Polishing equipment enable” command node, parameters:

- Device enable: Present/Descend

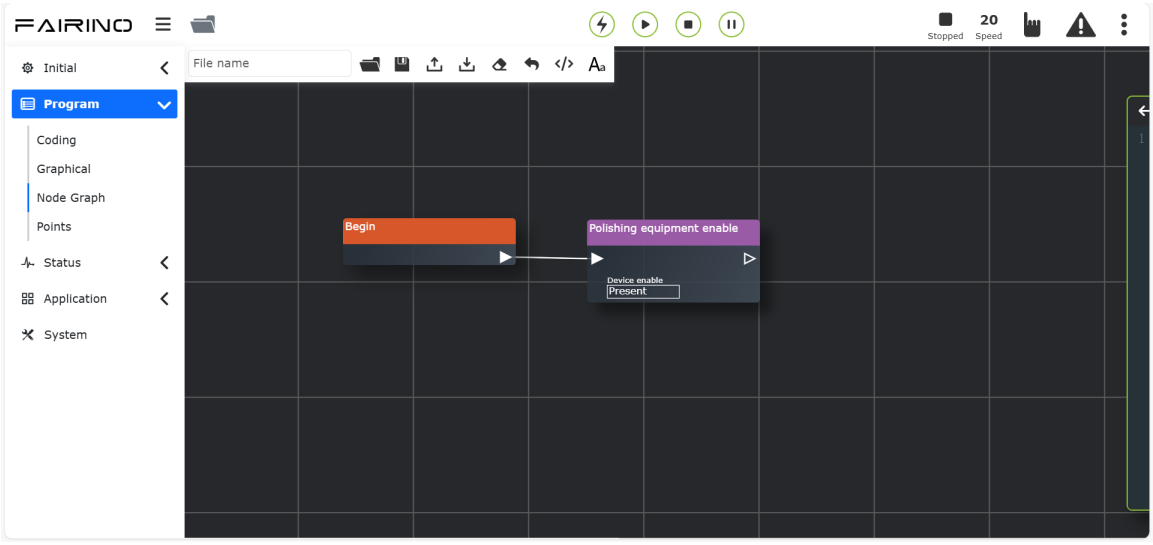


Figure 11.43-3 “Polishing equipment enable” command node interface

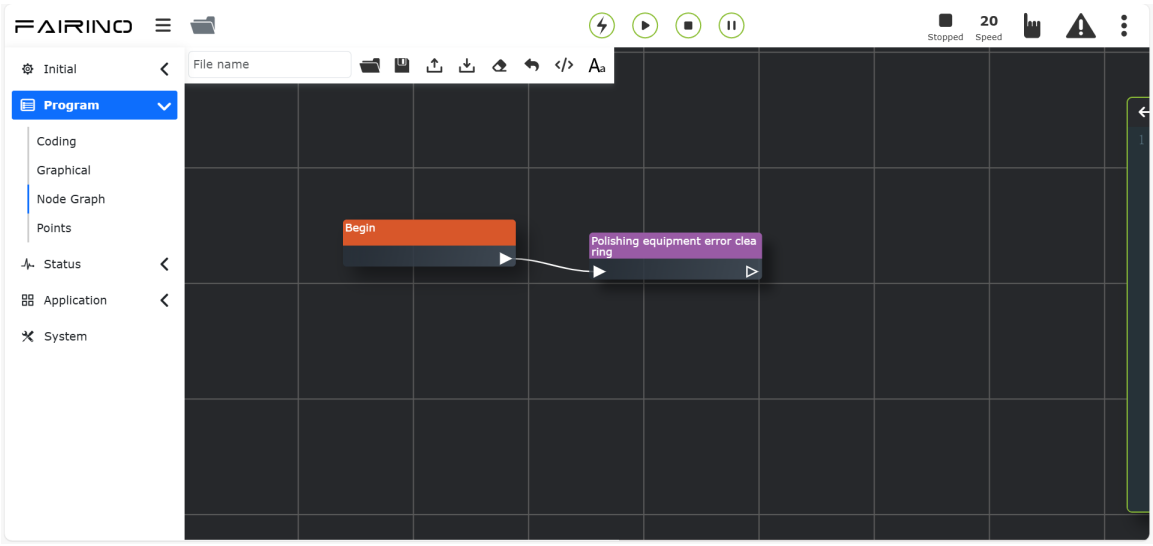


Figure 11.43-4 “Polishing equipment error clearing” command node interface

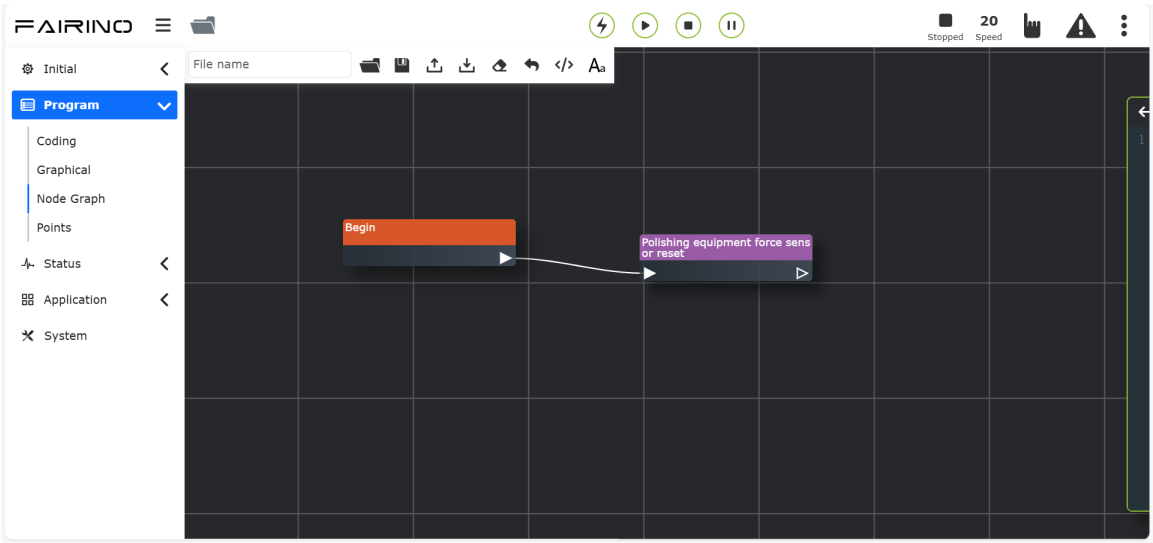


Figure 11.43-5 “Polishing equipment force sensor clear” command node interface

“Polishing speed” command node, parameters:

- Polishing speed: 0~5500

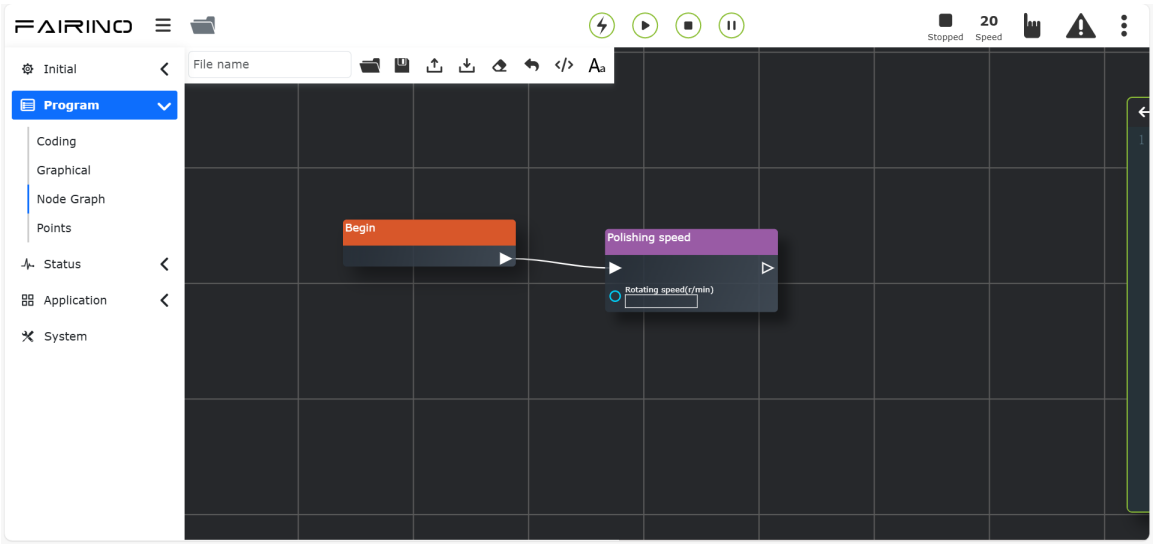


Figure 11.43-6 “Speed” command node interface

“Polishing contact force” command node, parameters:

- Contact force: 0~200

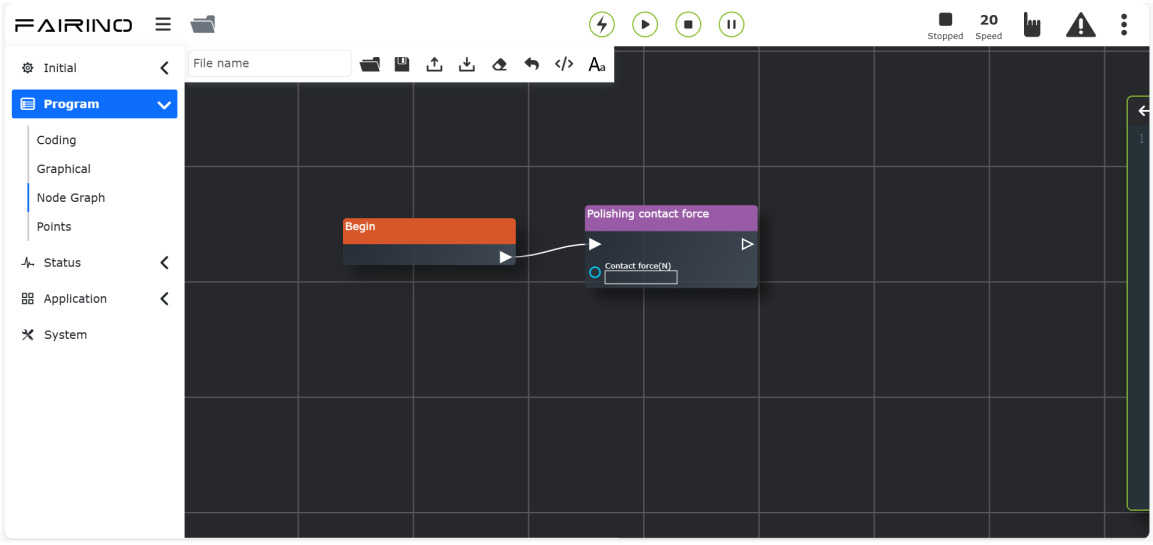


Figure 11.43-7 “Polishing contact force” command node interface

“Polishing protrusion distance” command node, parameters:

-Polishing protrusion distance: 0~12

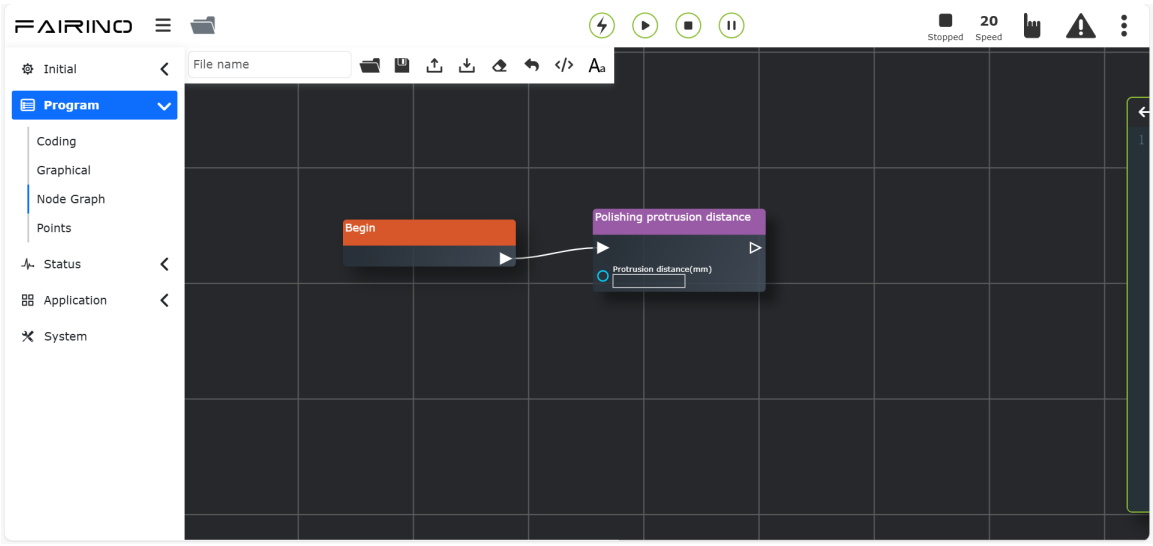


Figure 11.43-8 “Polishing protrusion distance” command node interface

“Polishing control mode” command node, parameters:

- Control mode: zero return mode/position mode/torque mode

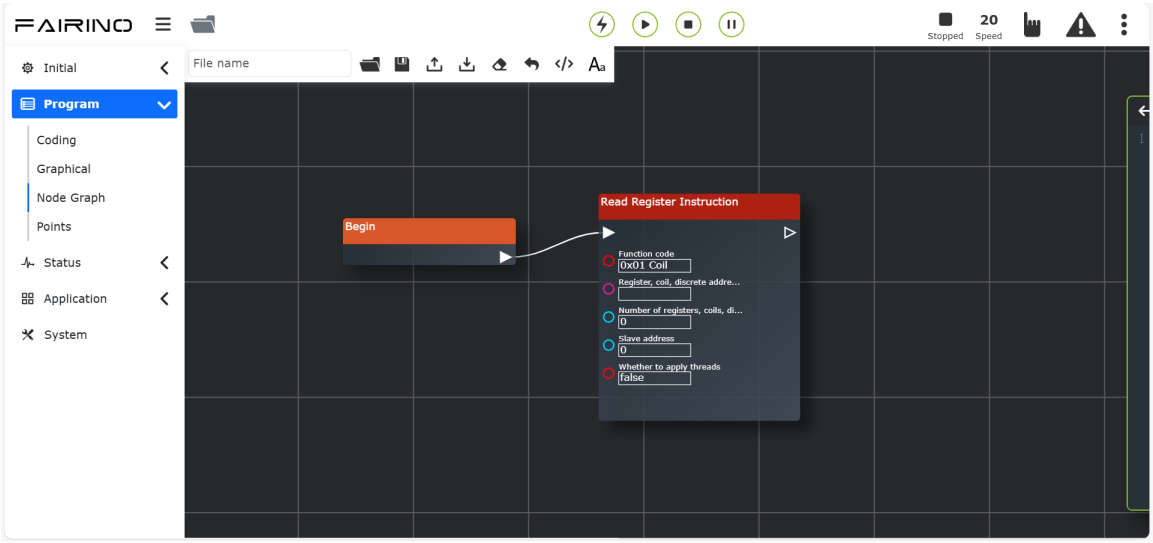


Figure 11.43-9 “Polishing control mode” command node interface

1.3.11.44 Weld command

Click “Welding related command node” to enter the node graph programming interface.

This command is mainly used for welding machine peripherals. Before adding this command, please confirm whether the welding machine configuration is completed in the user peripherals. For details, see the Robot Peripherals chapter.

1. “Welding machine voltage” command node, parameters:

- Welding machine voltage: minimum value is 0

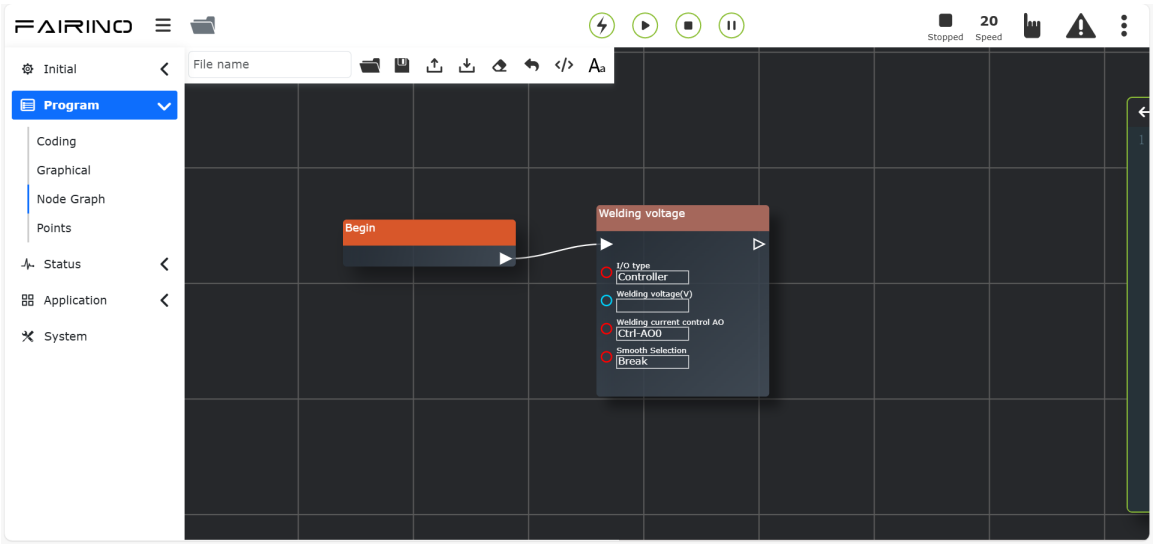


Figure 11.44-1 “Welding machine voltage” command node interface

2. “Welding machine current” command node, parameters:

- Welding machine current: minimum value is 0

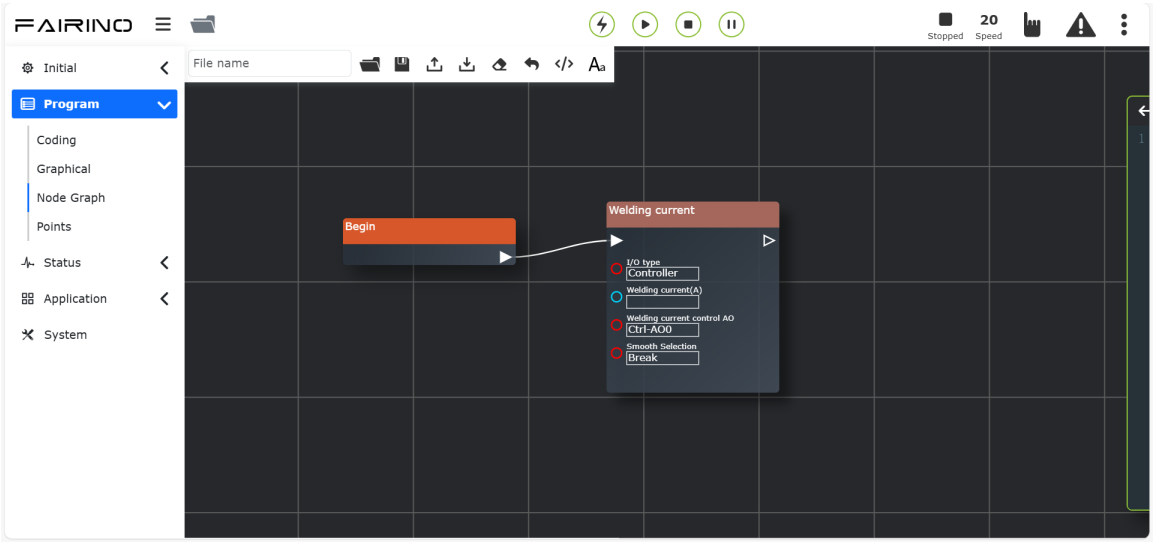


Figure 11.44-2 “Welding machine current” command node interface

3. “Arc closing/arc starting” command node, parameters:

- I/O type: controller IO/expansion IO
- Welding process number: 0 ~ 7
- Maximum waiting time (ms): 0 ~ 10000

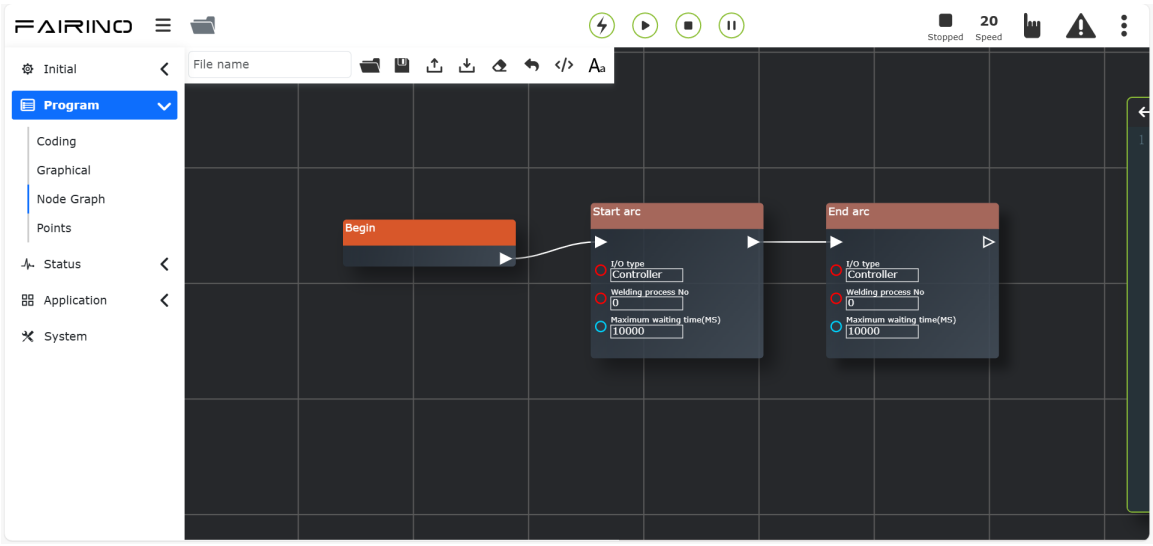


Figure 11.44-3 “Arc closing/arc starting” command node interface

4. “Gas ON/OFF” command node, parameters:

- I/O type: controller IO/expansion IO

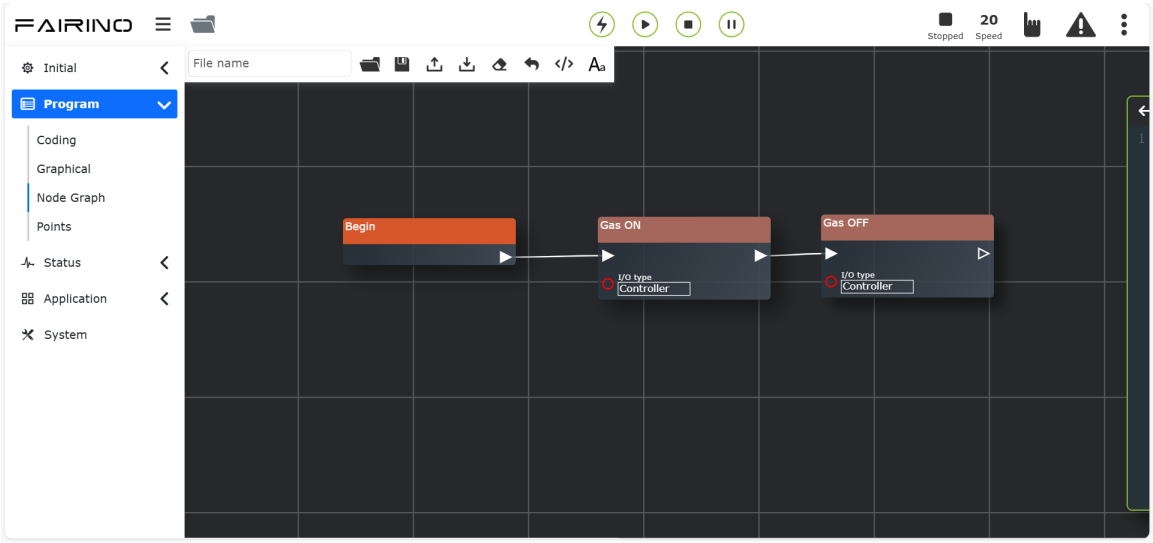


Figure 11.44-4 “Gas on/off” command node interface

5. “Forward/Stop forward” command node, parameters:

- I/O type: controller IO/expansion IO

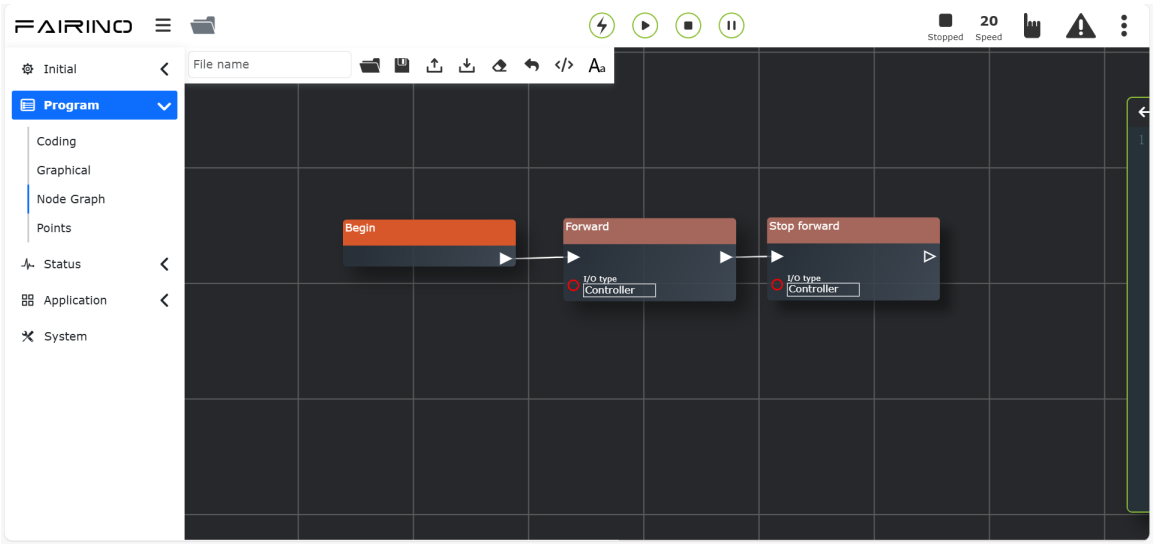


Figure 11.44-5 “Forward/Stop forward” command node interface

6. “Reverse wire feed/Stop reverse” command node, parameters:

- I/O type: controller IO/expansion IO

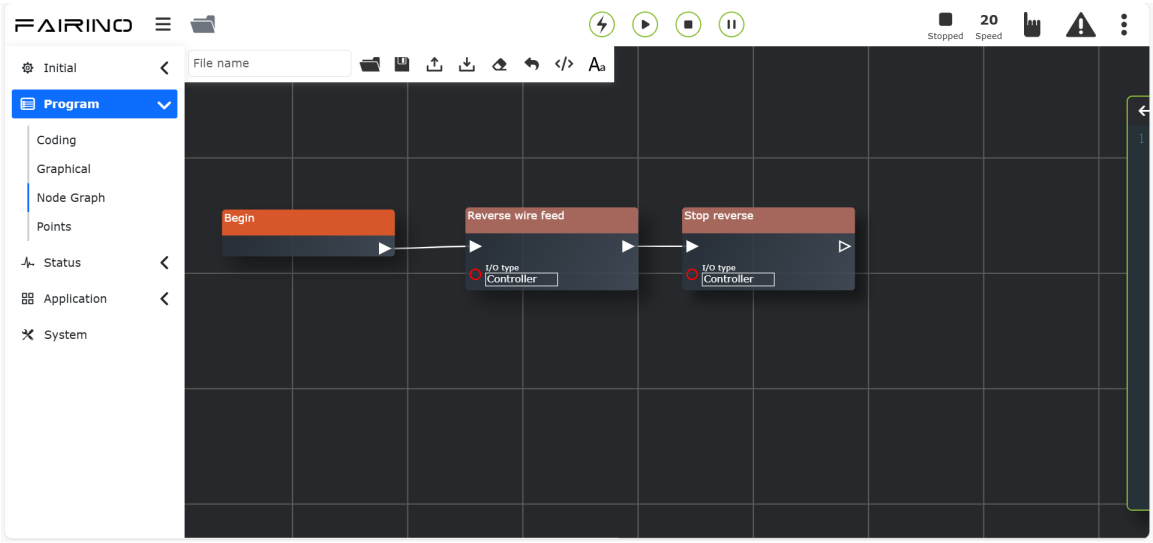


Figure 11.44-6 “Reverse wire feed/Stop reverse” command node interface

1.3.11.45 Segment instruction

This command is a special command for welding, mainly used in intermittent welding scenarios where one period of welding and one period of non-welding are cycled. Between the starting point and the end point, use this command to select the starting point and the end point, set the debugging speed, set the DO port of arc starting, execution length, non-execution length, set the function mode, swing selection and rounding rules according to the actual application scenario. Realize the segment welding function, detailed operations can be found in the segment welding instructions on the program teaching page.

“Segment welding” command node, parameters:

- Segment welding mode: no change in attitude/change in attitude

- Starting point: teaching point
- End point: teaching point
- Debugging speed (%): 0~100, default is 100
- Execution length: 0~1000
- Non-executable length: 0~1000
- Function mode: 0~100, default is 100
- Swing selection: execution section does not swing/execution section swings
- Rounding rules: no rounding/circular rounding/single segment rounding

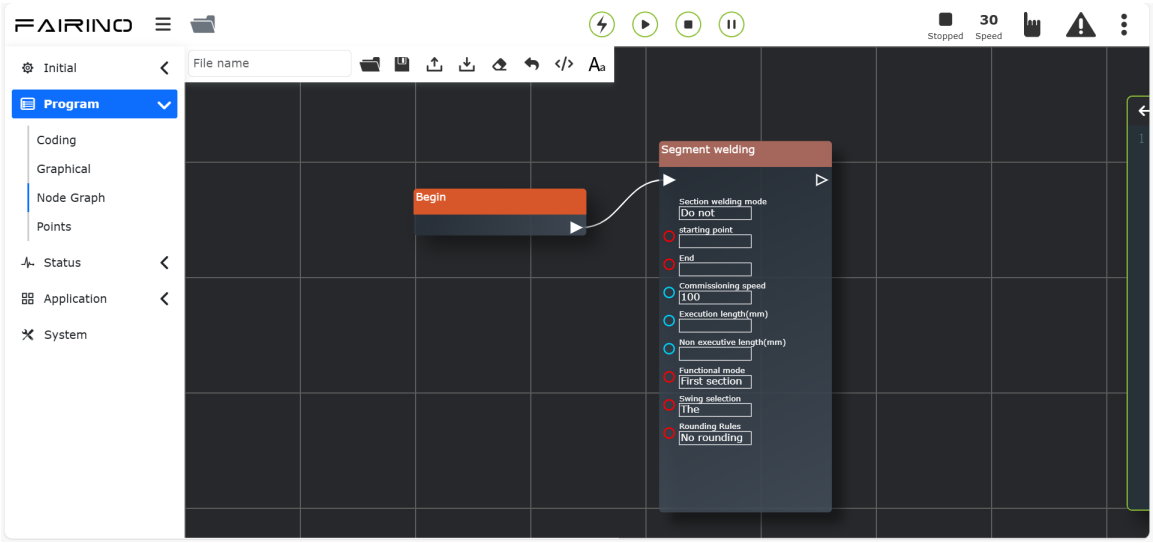


Figure 11.45-1 “Segment welding” command node interface

1.3.11.46 Laser instruction

Click the “Laser” command node to enter the node graph programming interface

This command contains three parts: laser command, tracking command and positioning command. Before adding this command, please confirm whether the laser tracking sensor in the user peripheral has been configured successfully. See the Robot Peripherals chapter for details.

1. “Open/Close sensor” command node, parameters:

- Select weld type: 0 ~ 49

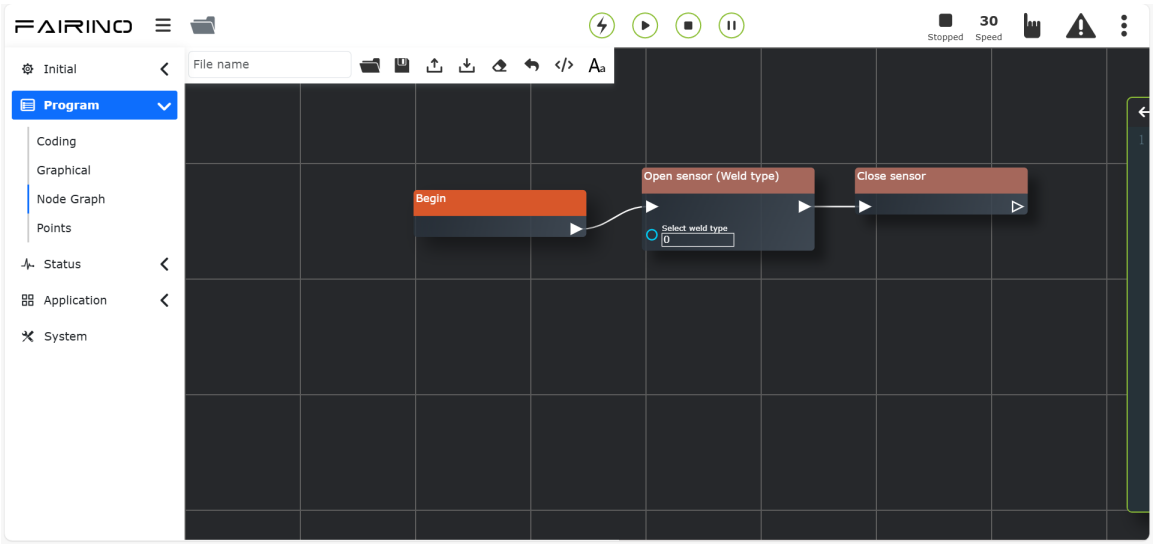


Figure 11.46-1 “Open/Close sensor” command node interface – Weld type

- Select task number: 0 ~ 255

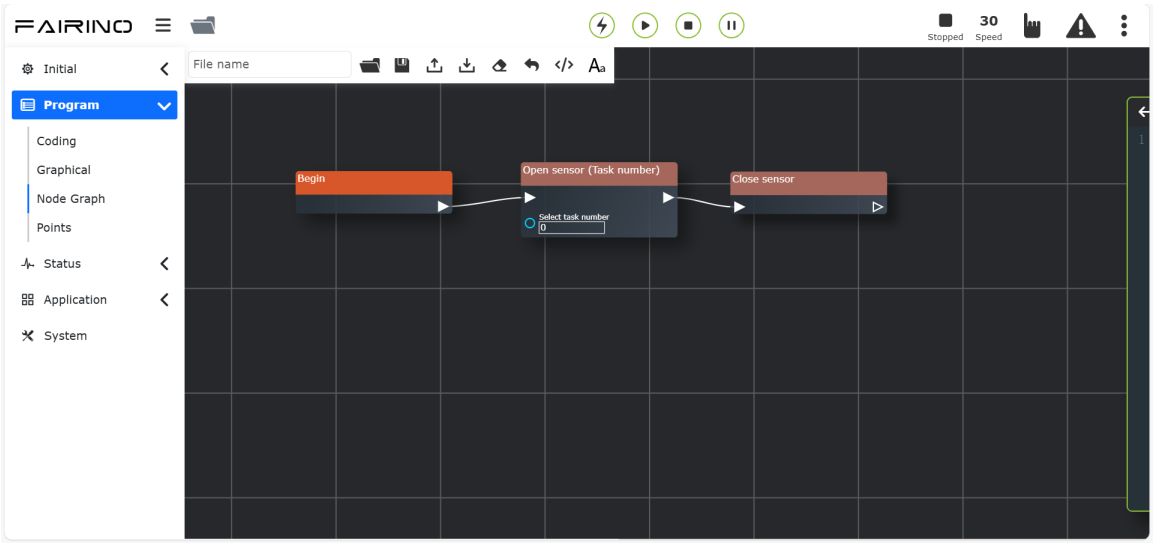


Figure 11.46-2 “Open/Close sensor” command node interface – Task number

2. “Load/unload sensor” command node, parameters:

- Function selection: Ruiniu RRT-SV2-BP/CXZK-RBTA4L

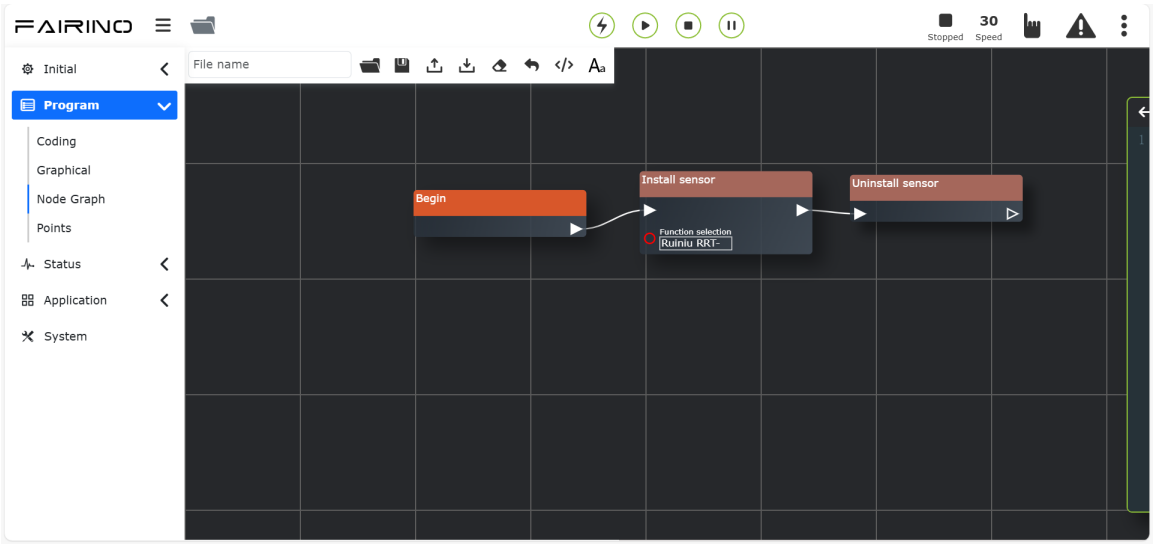


Figure 11.46-3 “Load/unload sensor” command node interface

3. “Start/Stop Tracking” command node, parameters:

- Coordinate system name: Customized configuration coordinate system

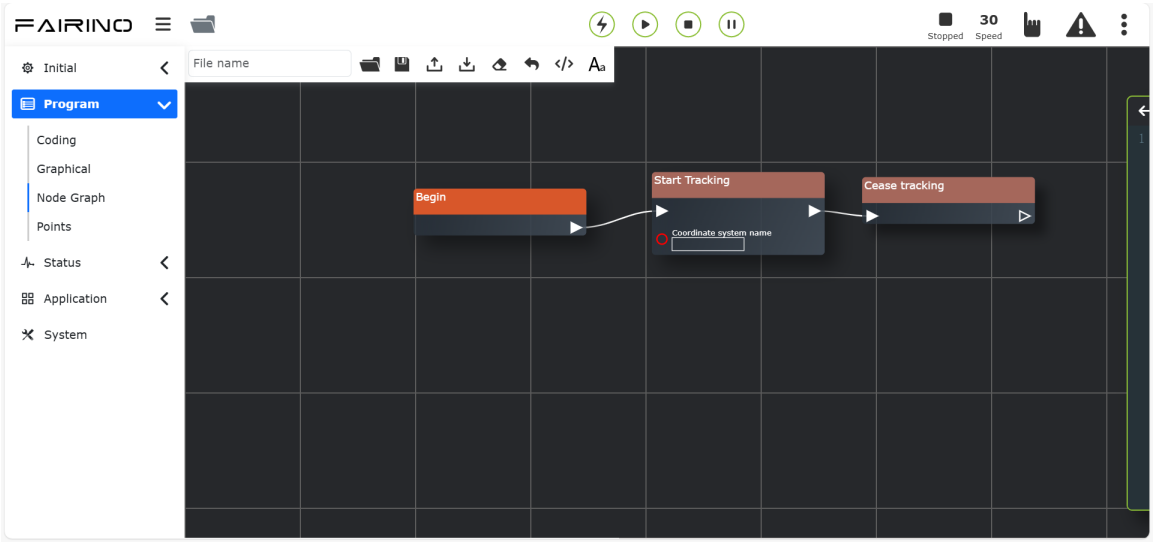


Figure 11.46-4 “Start/stop tracking” command node interface

4. “Data Record” command node, parameters:

- Function selection: stop recording/real-time tracking/start recording/track recurrence
- Waiting time (ms): 0 ~ 10000

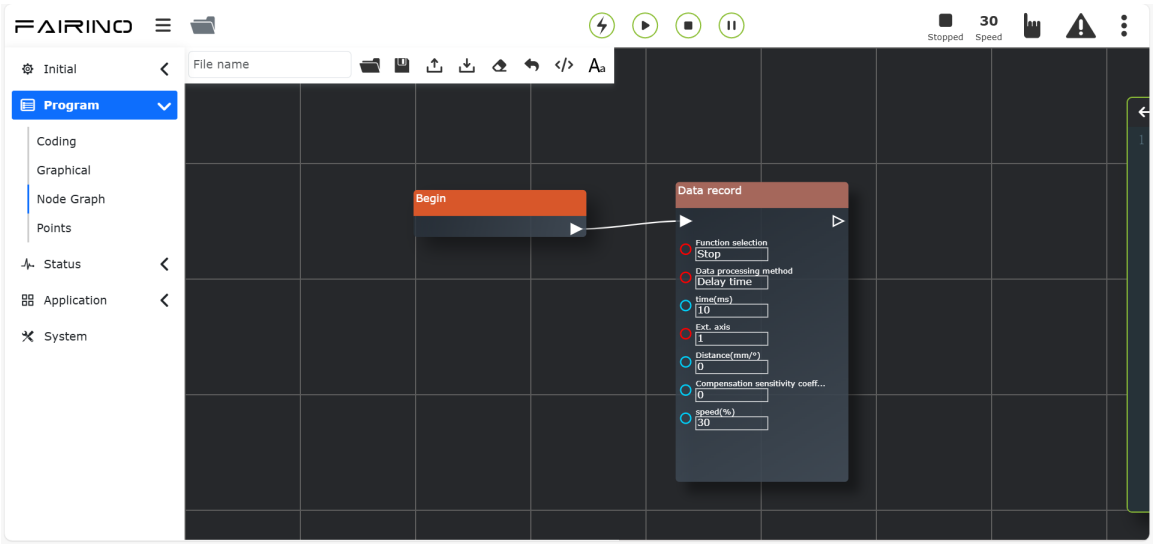


Figure 11.46-5 “Data Logging” command node interface

5. “Laser tracking recurrence” command node, parameters:

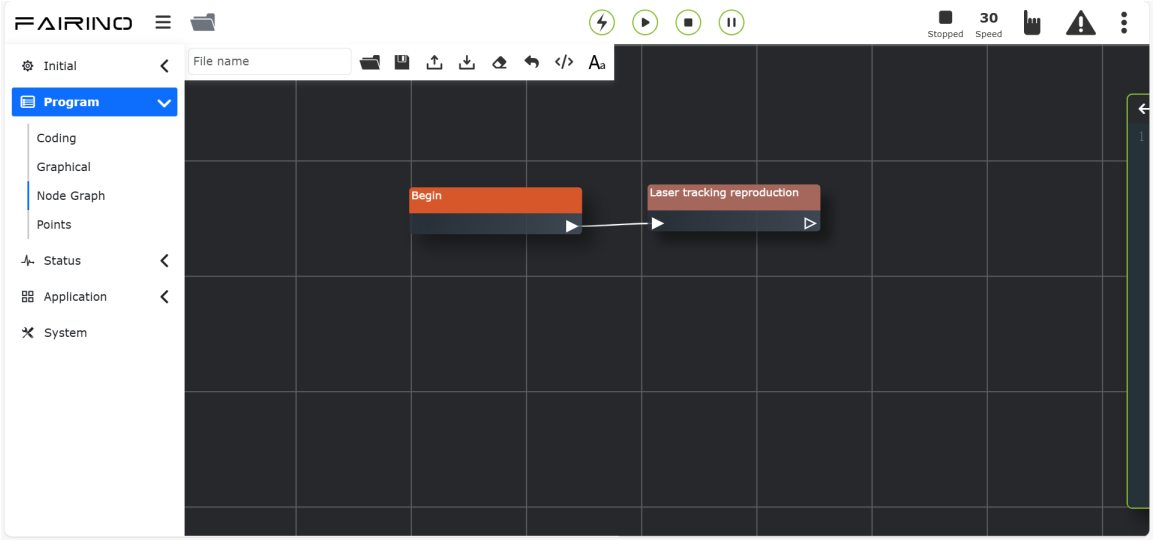


Figure 11.46-6 “Laser Tracking Reappearance” command node interface

6. “Sensor point acquisition movement” command node, parameters:

- Coordinate system name: Customized configuration coordinate system
- Sports mode: PTP/Lin
- Debugging speed (%): 0 ~ 100

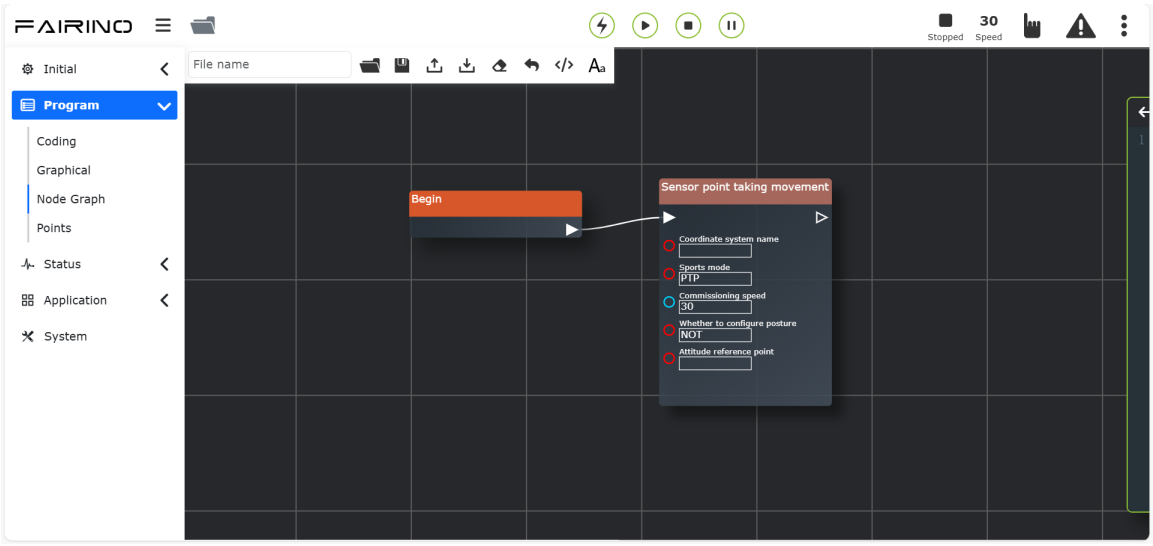


Figure 11.46-7 “Data Logging” command node interface

7. “Start/End locating” command node, parameters:

- Coordinate system name: Customized configuration coordinate system
- Direction: -x/-x/-y/-y/-z/-z/specifies the direction
- Direction point: When “Specify direction” is not selected, the parameter is invalid
- Speed (%): 0 ~ 100
- Length (mm): 0 ~ 1000
- Maximum search time (ms): 0 ~ 10000

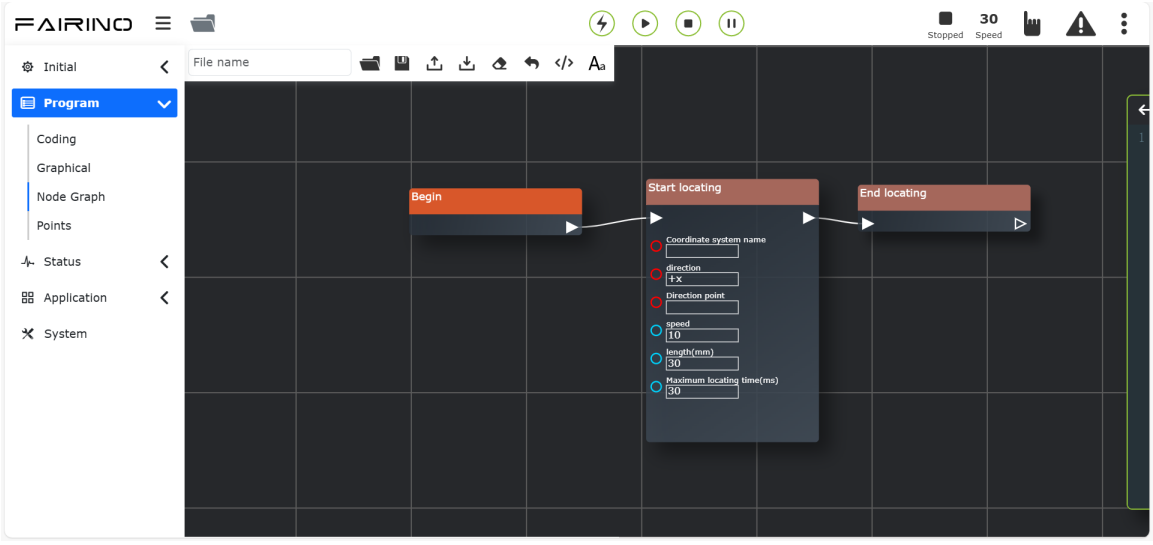


Figure 11.46-8 “Start/End locating” command node interface

1.3.11.47 Laser recording instructions

This command implements the function of taking out the starting point and end point of laser tracking and recording, so that the robot can automatically move to the starting point. It is suitable for situations where the movement starts from outside the workpiece and laser tracking and recording is performed. At the same time, the host computer can obtain the information of the starting point and end point in the recorded data. for subsequent exercise.

Implementing the adjustable laser tracking reproduction speed function allows the robot to record at a very fast speed and then reproduce at the normal welding speed, which can improve operating efficiency.

“Weld data record” command node, parameters:

- Function selection: stop recording/real-time tracking/start recording/track recurrence
- Waiting time (ms): 0~10000, default is 10
- Speed (%): 0~100, default is 30, this parameter takes effect when selecting trajectory recurrence

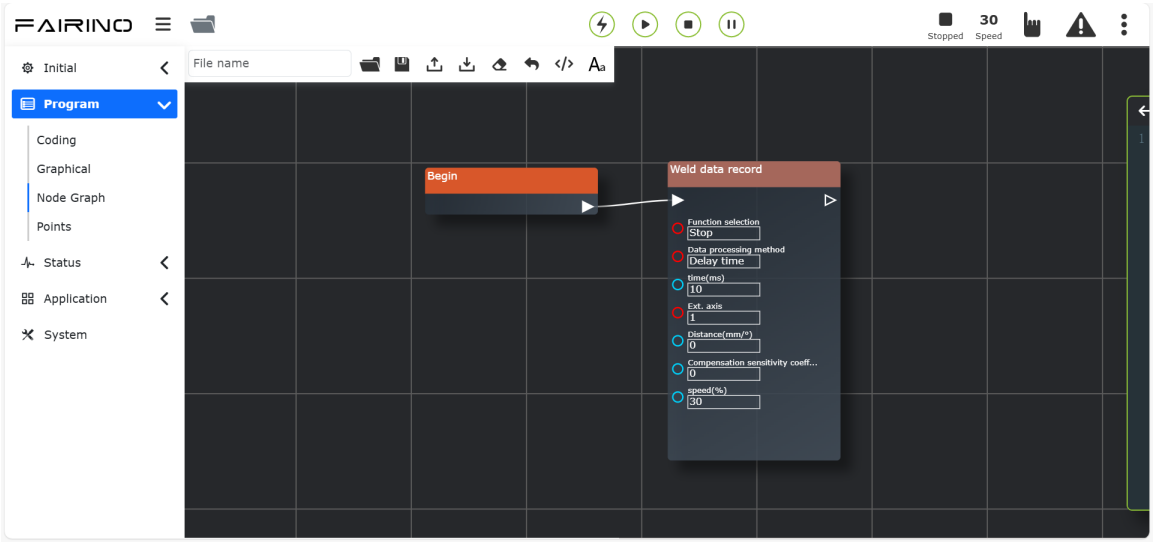


Figure 11.47-1 “Weld data record” command node interface

“Move to the starting/end point of weld” command node, parameters:

- Sports mode: PTP/LIN
- Speed (%): 0~100, default is 30

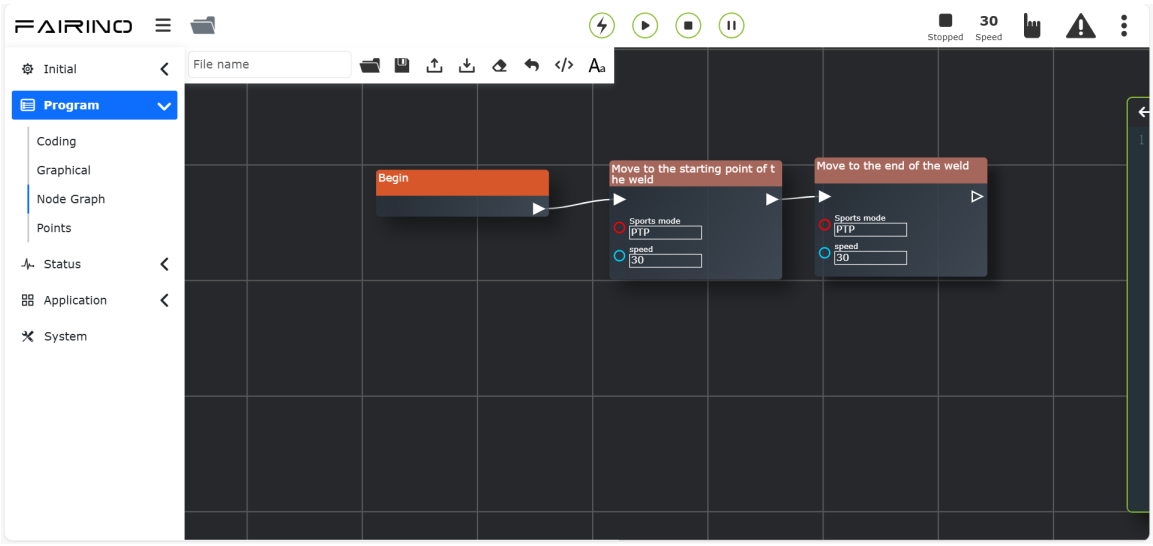


Figure 11.47-2 “Obtain weld starting point/end point” command node interface

1.3.11.48 W-Search instruction

This command is generally used in welding scenarios and requires the welding machine to be used in combination with robot IO and motion commands. It is divided into search start, search end, search point setting, offset calculation and contact point data writing.

“Welding wire positioning start/end” command node, parameters:

- Base position: do not update/update
- Position search speed: 0~100
- Search distance: 0~1000
- Automatic return flag: no automatic return/automatic return
- Automatic return speed: 0~100
- Automatic return distance: 0~1000
- Positioning method: teaching point positioning/positioning with offset

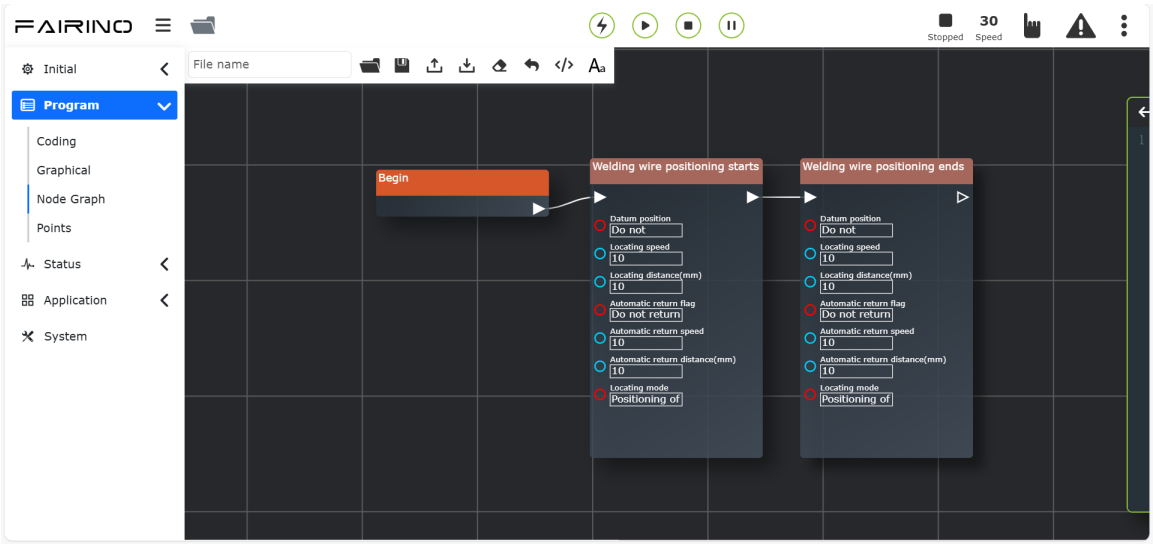


Figure 11.48-1 “Welding wire positioning start/end” command node interface

The search point setting adds points based on the weld type and calculation method.

- When the type is fillet weld and the calculation method is 1D (one of xyz), the point addition is selected from point a and point b
- When the type is fillet weld and the calculation method is 2D (two in xyz), the point addition is selected from point a, point b, point e, and point f
- When the type is fillet weld and the calculation method is 3D (xyz), the point addition is selected from point a, point b, point c, point d, point e, and point f
- When the type is fillet weld and the calculation method is 2D- (two in xyz, one in rxryrz), the point addition is selected from point a, point b, point c, point d, point e, and point f
- When the type is inner and outer diameter and the calculation method is 2D2D (two in xyz), the point addition is selected from point a and point b
- When the type is point and the calculation method is 3D (xyz), the point addition is selected from point a, point b, point c, point d, point e, and point f
- When the type is camera and the calculation method is 3D- (xyzrxryrz), the point addition is selected from point a and point b
- When the type is surface and the calculation method is 3D- (xyzrxryrz), the point addition is selected from point a and point b

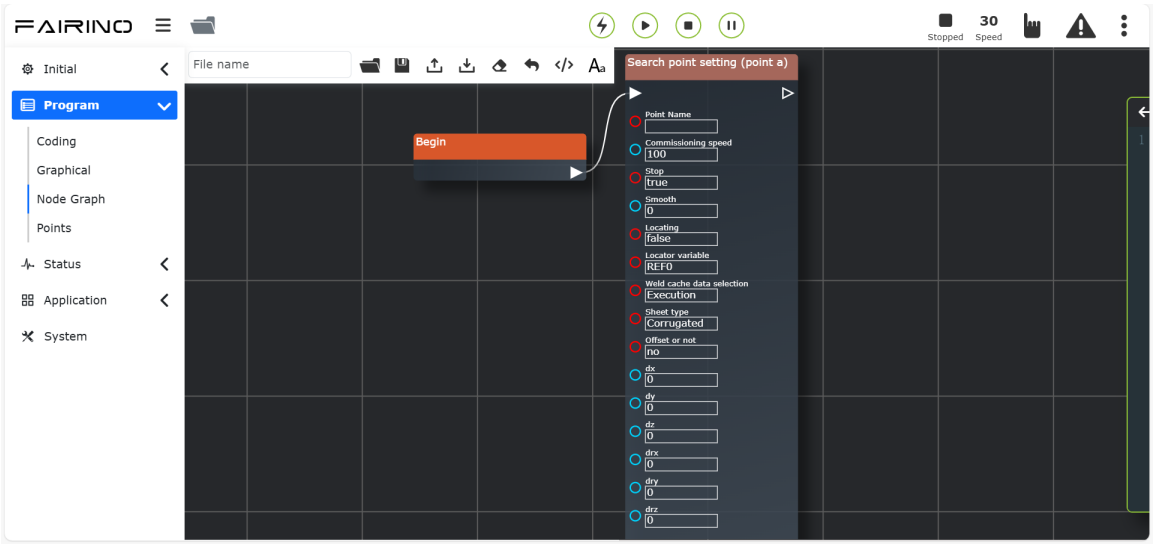


Figure 11.48-2 “Search point setting” command node interface

Calculate the offset to set the datum and contact points based on the weld type and calculation method.

- When the type is fillet weld and the calculation method is 2D (two in xyz), set datum point 1, datum point 2, contact point 1, and contact point 2
- When the type is fillet weld and the calculation method is 3D (xyz), set datum point 1, datum point 2, datum point 3, contact point 1, contact point 2, and contact point 3
- When the type is fillet weld and the calculation method is 2D- (two in xyz, one in rxryrz), set reference point 1, reference point 2, reference point 3, contact point 1, contact point 2, contact point 3
- When the type is inner and outer diameter and the calculation method is 2D2D (two in xyz), set datum point 1, datum point 2, datum point 3, contact point 1, contact point 2, and contact point 3
- When the type is point and the calculation method is 3D (xyz), set contact point 1 and contact point 2
- When the type is camera and the calculation method is 3D- (xyzrxryrz), set contact point 1 and contact point 2
- When the type is face and the calculation method is 3D- (xyzrxryrz), set contact point 1, contact point 2, contact point 3, contact point 4, contact point 5, and contact point 6

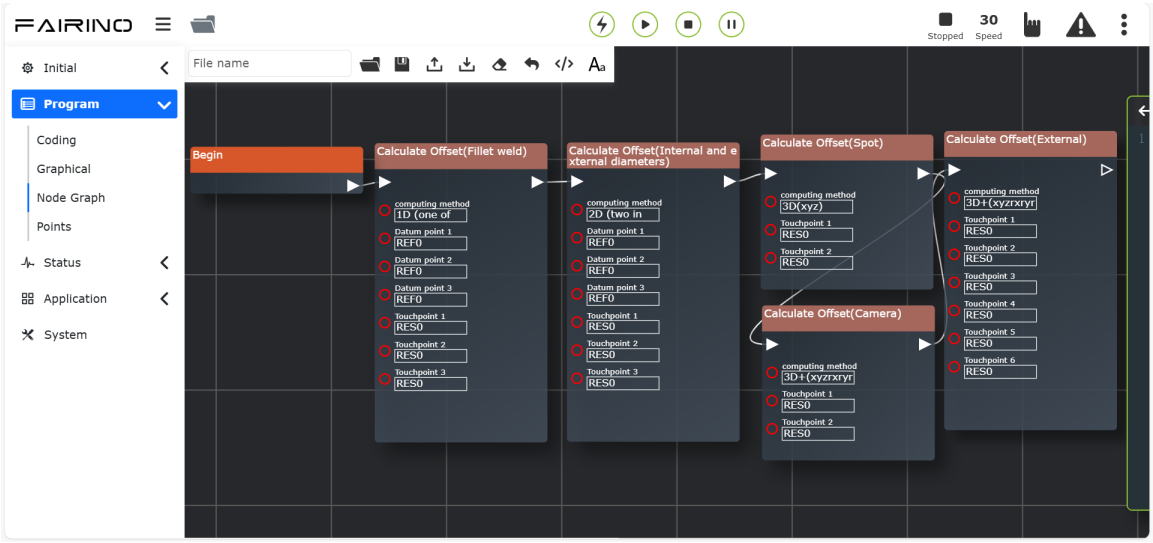


Figure 11.48-3 “Calculate offset” command node interface

“Contact point data write” command node, parameters:

- Contact point name: RES0~99
- Contact point name: The data format is {0,0,0,0,0,0}

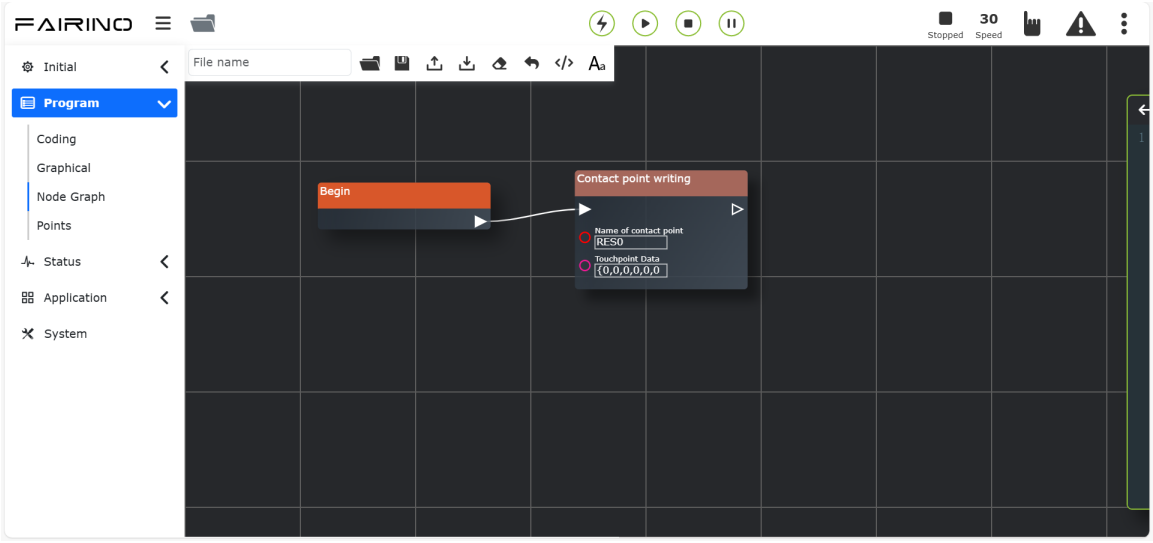


Figure 11.48-4 “Contact Point Data Write” command node interface

1.3.11.49 Weld-Trc instruction

Click the “Weld-Trc” command node to enter the node graph programming interface

This instruction implements robot welding seam tracking using welding seam deviation detection to compensate for the trajectory. Arc sensors can be used to detect welding seam deviations.

“Arc tracking on/off” command node, parameters:

- Arc tracking lag time (ms): reference value 50
- Deviation compensation: off/on
- Adjustment coefficient: 0 ~ 300
- Compensation time (cyc): 0 ~ 300
- Maximum compensation amount each time (mm): 0 ~ 300
- Total maximum compensation amount (mm): 0 ~ 300
- Up and down coordinate system selection: swing
- Upper and lower reference current setting method: feedback/constant
- Upper and lower reference current (A): 0 ~ 300

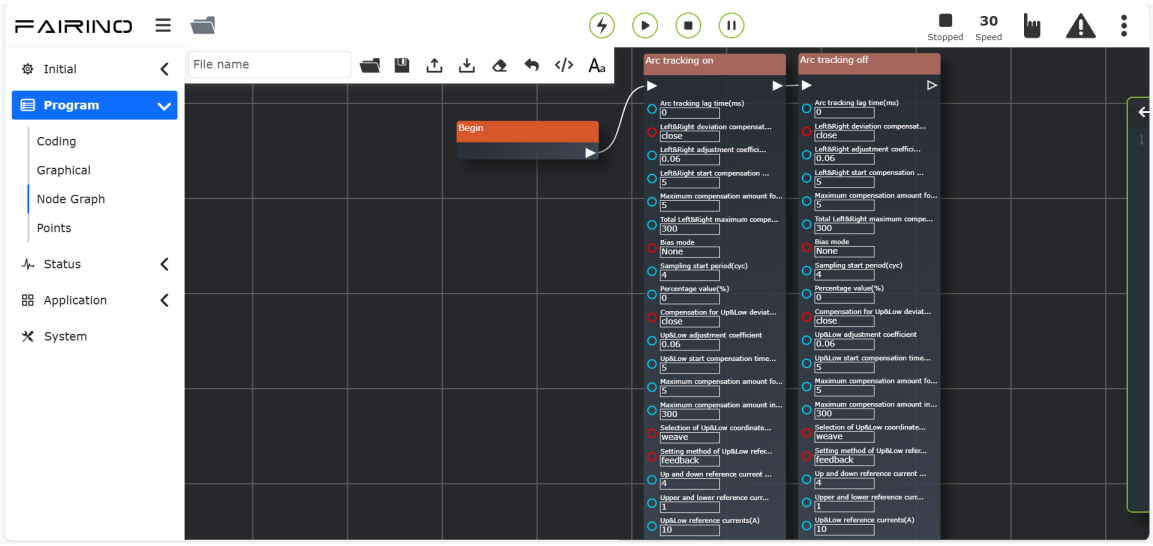


Figure 11.49-1 “Arc tracking on/off” command node interface

1.3.11.50 Attitude adjustment instructions

Click on the “Attitude Adjustment” related command node to enter the node graph programming interface.

This command adaptively adjusts the welding gun posture for welding tracking scenarios. It is necessary to teach the three points PosA, PosB, and PosC first, otherwise nodes cannot be added.

After recording the three corresponding posture points, add posture adaptive adjustment instructions according to the actual movement direction of the robot. See the Robot Peripherals chapter for details.

“Turn on attitude adjustment” command node, parameters:

- Board type: corrugated board/corrugated board/fence board/corrugated shell steel
- Direction of movement: left to right/right to left
- Attitude adjustment time (ms): 0 ~ 1000
- Length of first section (mm):
- Inflection point type: top to bottom/bottom to top
- Length of second section (mm):
- Length of third section (mm):
- Length of the fourth section (mm):
- Length of the fifth section (mm):

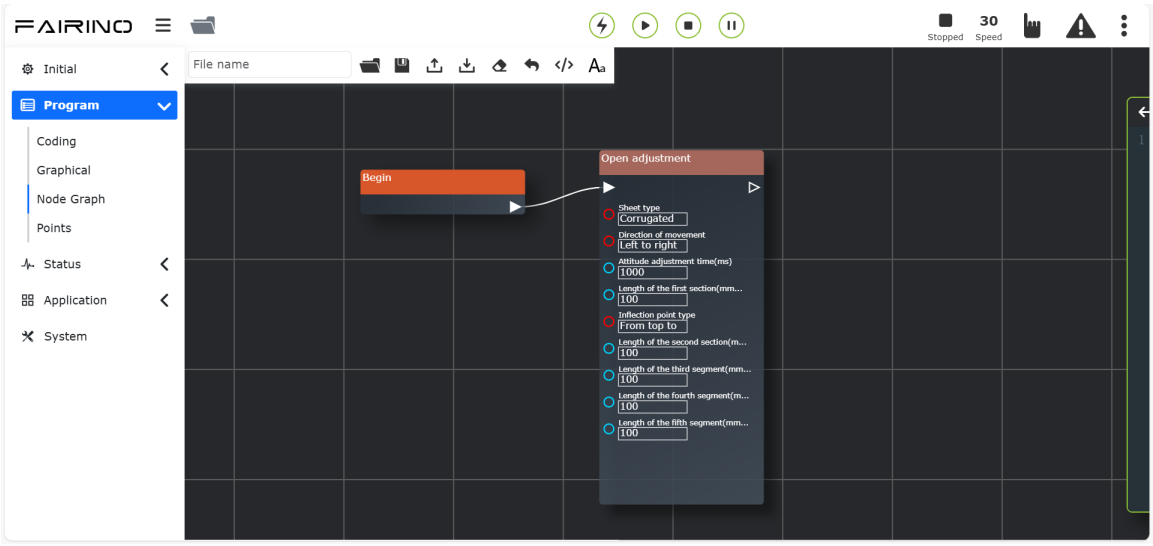


Figure 11.50-1 “Turn on attitude adjustment” command node interface

“Turn off attitude adjustment” command node, parameters:

- Board type: corrugated board/corrugated board/fence board/corrugated shell steel

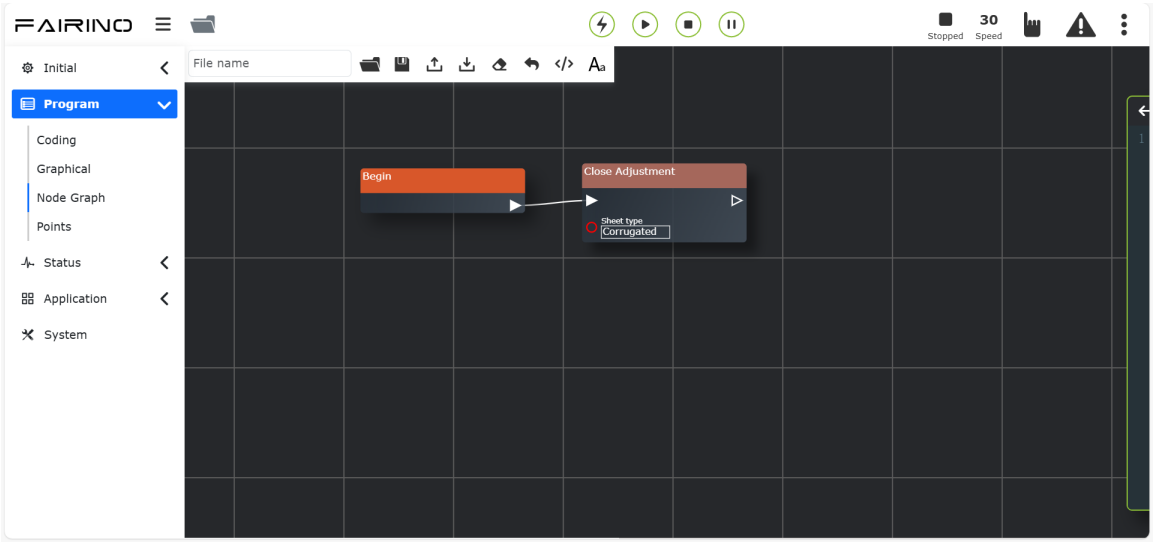


Figure 11.50-2 “Turn off attitude adjustment” command node interface

1.3.11.51 F/T Instruction

Click on the command node related to the “F/T” command to enter the node graph programming interface.

This command includes FT_Guard (collision detection), FT_Control (constant force control), FT_Compliance (compliance control), FT_Spiral (spiral insertion), FT_Rot (rotation insertion), FT_Lin (linear insertion), FT_FindSurface (surface positioning), FT_CalCenter (center positioning)), FT_Click (click force detection) nine instructions, see the robot peripherals chapter for details.

1. “Turn on/off collision detection” command node, parameters:

- Coordinate system name: Custom configured coordinate system
- Fx-Tx truth value: true/false

- Fx-Tx current value: input according to actual situation
- Fx-Tx maximum threshold: input according to actual situation
- Fx-Tx minimum threshold: input according to actual situation

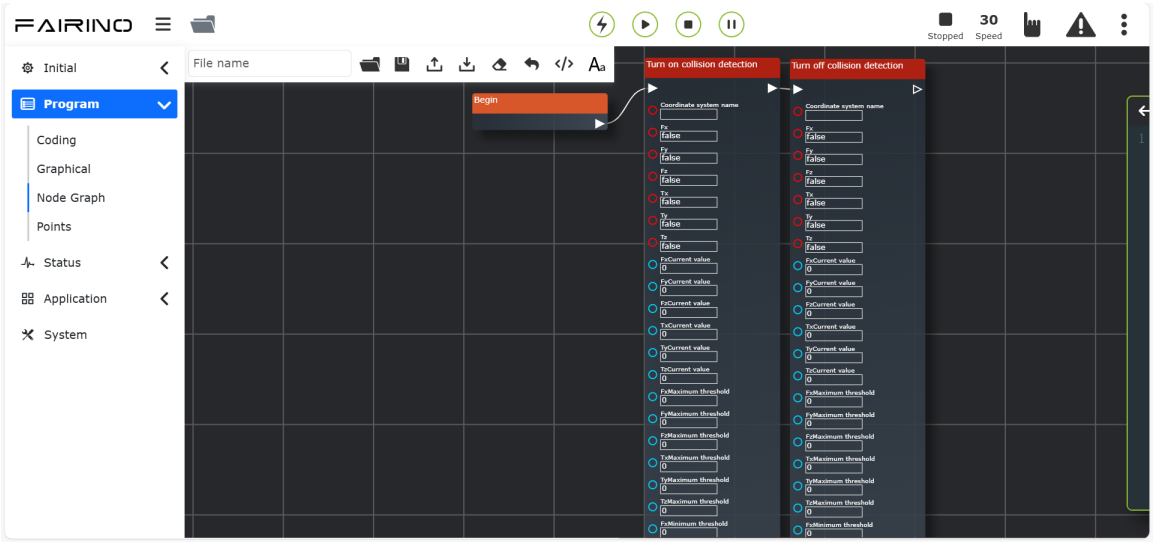


Figure 11.51-1 “Turn on/off collision detection” command node interface

2. “Turn on/off control” command node, parameters:

- Coordinate system name: Custom configured coordinate system
- Fx-Tx truth value: true/false
- Current value of Fx-Tx: adjusted according to actual situation
- F_P_gain - F_D_gain: adjusted according to actual situation, cannot be 0
- Adaptive start and stop status: stop/start
- ILC control start and stop status: stop/training/practical operation
- Maximum adjustment distance (mm): 0 ~ 1000
- Maximum adjustment angle (°): 0 ~ 1000

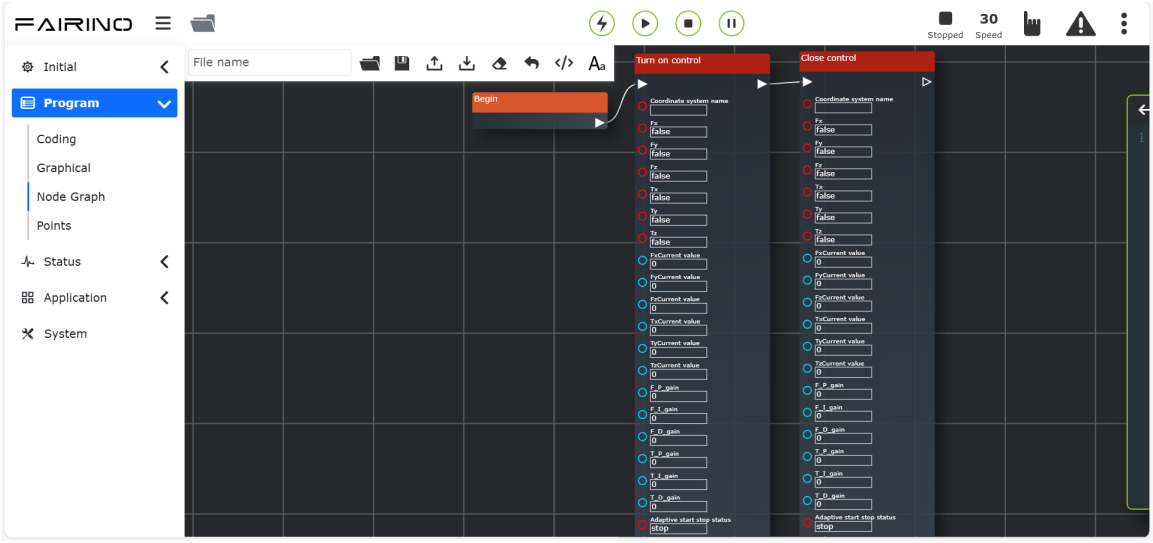


Figure 11.51-2 “Turn on/off control” command node interface

3. “FT_Compliance Start/End” command node, parameters:

- Delivery position adjustment coefficient: 0 ~ 1
- Compliant opening force threshold (N): 0 ~ 100

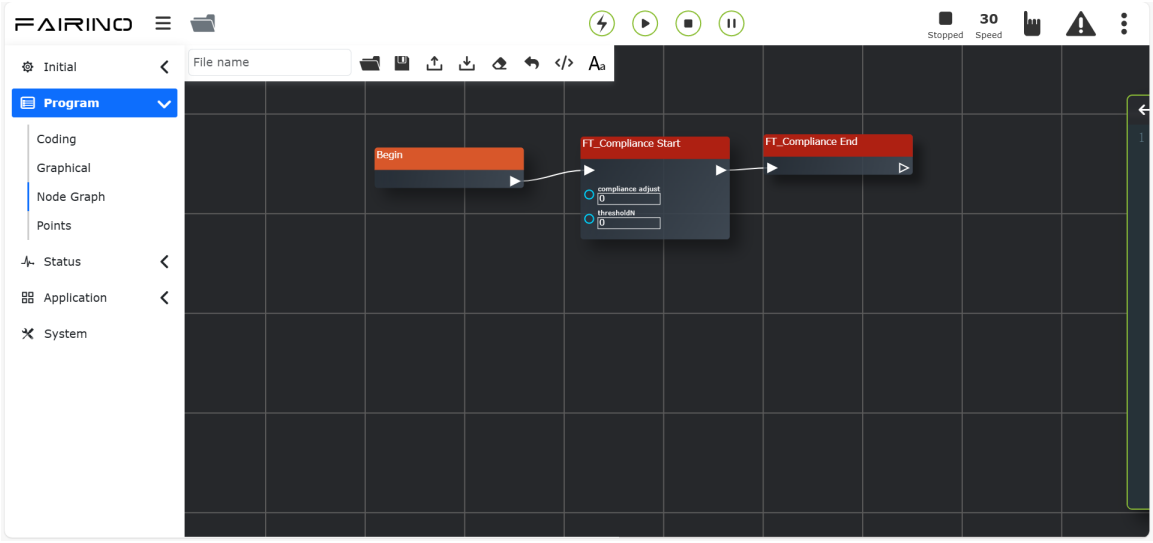


Figure 11.51-3 “FT_Compliance Start/End” command node interface

4. “FT_Spiral” command node, parameters:

- Coordinate system name: tool coordinate system/base coordinate
- Radius feed per revolution (mm): 0 ~ 100, reference value: 0.7
- Force or torque threshold (N/Nm): 0 ~ 100, reference value: 50
- Maximum exploration time (ms): 0 ~ 60000, reference value: 60000
- Maximum linear speed (mm/s): 0 ~ 100, reference value: 5

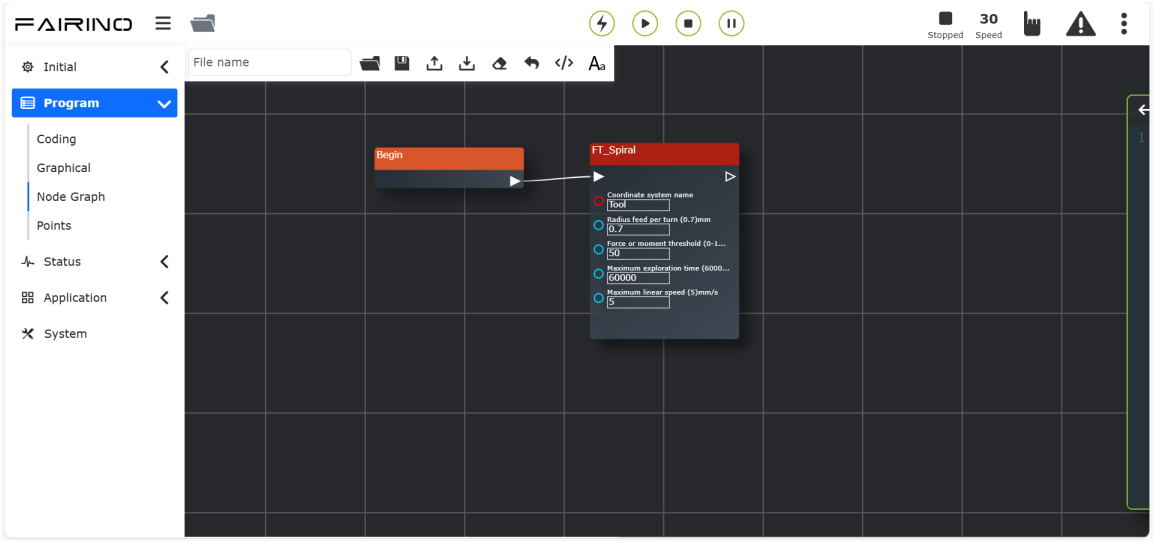


Figure 11.51-4 “FT_Spiral” command node interface

5. “FT_Rot” command node, parameters:

- Coordinate system name: tool coordinate system/base coordinate
- Rotation angular speed (°/s): 0 ~ 100, reference value: 0.7
- Trigger force or final torque (N/Nm): 0 ~ 100, reference value: 50
- Maximum rotation angle (°): 0 ~ 100, reference value: 5
- Direction of force: direction z/direction mz
- Maximum rotation angle acceleration (°/s^2): 0 ~ 100
- Insertion direction: positive/negative

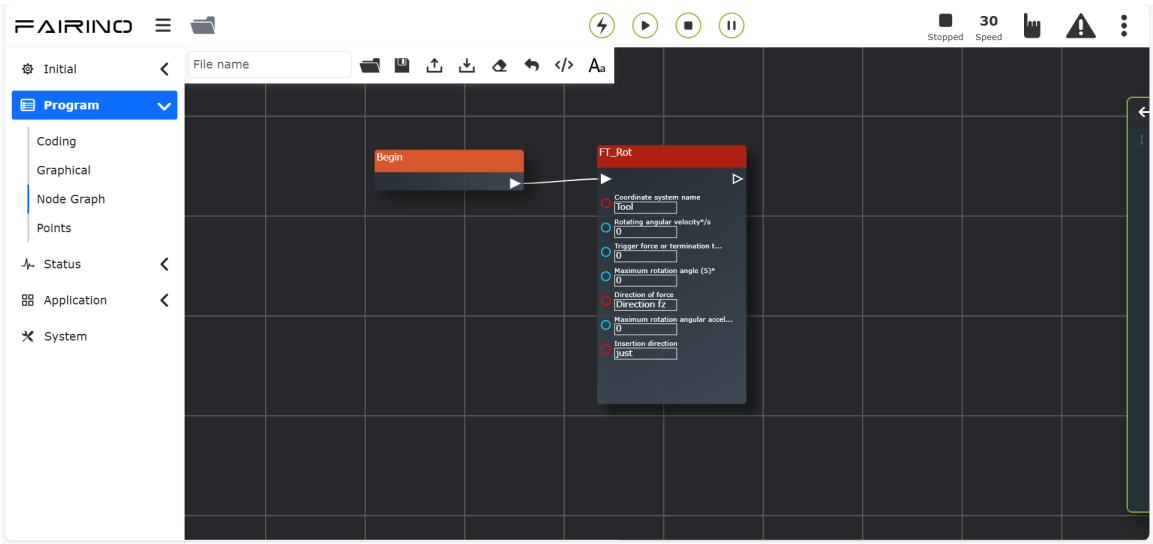


Figure 11.51-5 “FT_Rot” command node interface

6. “FT_Lin” command node, parameters:

- Coordinate system name: tool coordinate system/base coordinate
- Action termination force threshold (N): 0 ~ 100
- Linear speed (mm/s): 0 ~ 100, reference value: 1
- Linear acceleration (°/s^2): 0 ~ 100
- Maximum insertion distance (mm): 0 ~ 100
- Insertion direction: positive/negative

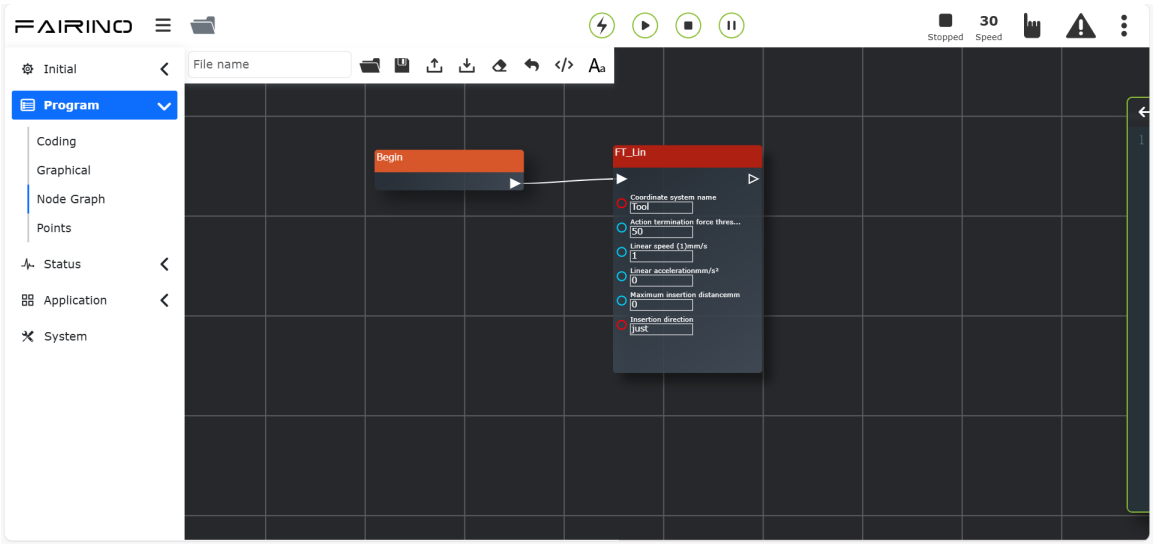


Figure 11.51-6 “FT_Lin” command node interface

7. “FT_FindSurface” command node, parameters:

- Coordinate system name: tool coordinate system/base coordinate
- Movement direction: positive/negative
- Moving axis: X/Y/Z
- Explore linear speed (mm/s): 0 ~ 100
- Exploration acceleration (mm/s^2): 0 ~ 100
- Maximum exploration distance (mm): 0 ~ 100
- Action termination force threshold (N): 0 ~ 100

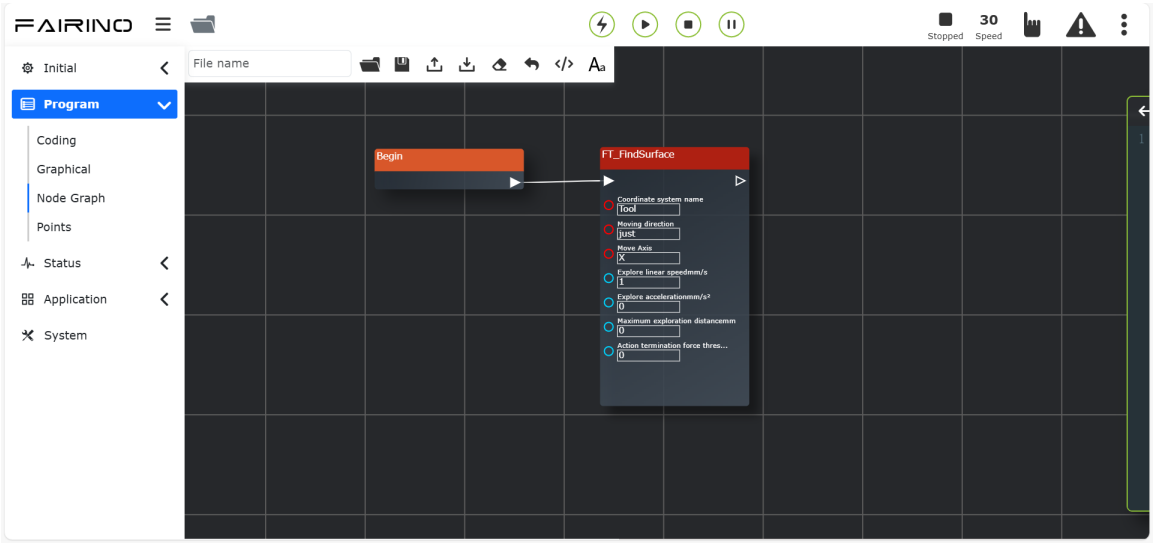


Figure 11.51-7 “FT_FindSurface” command node interface

8. “FT_CalCenter Start/End” command node

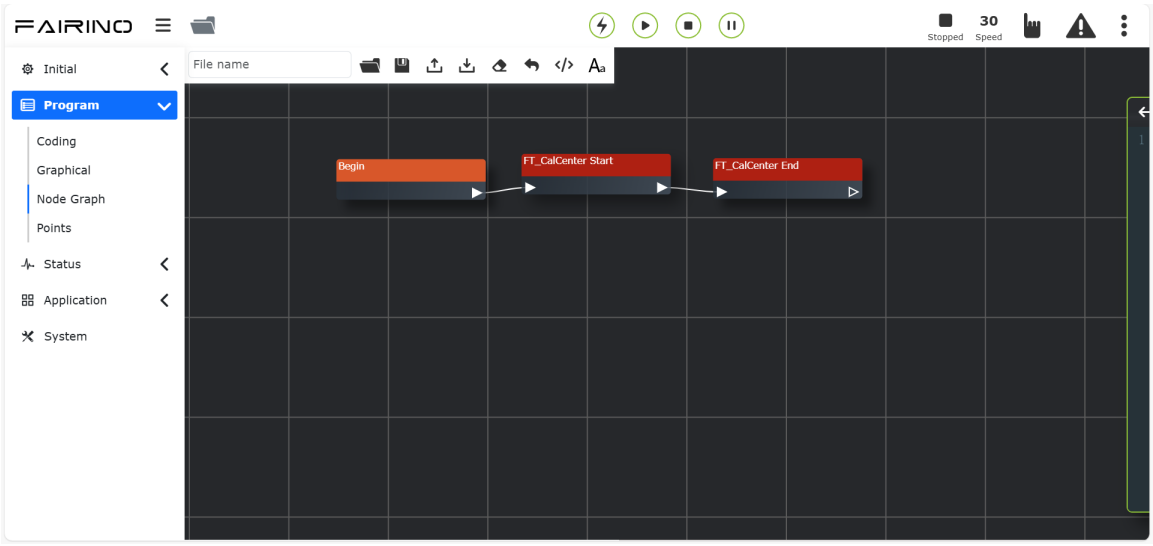


Figure 11.51-8 “FT_CalCenter Start/End” command node interface

1.3.11.52 Torque recording command

Click on the “Torque Record” related command node to enter the node graph programming interface.

This command is a torque recording command, which includes three commands: “Start Torque Recording/”Stop Torque Recording” and “Reset Torque Recording”.

Real-time torque recording and collision detection function.

Click the “Torque Record Start” button to continuously record the collision situation during the operation of the motion command. The recorded real-time torque is used as the theoretical value for collision detection judgment to reduce the probability of false alarms.

When the set threshold range is exceeded, the collision detection duration is recorded.

Click the “Torque Record Stop” button to stop recording. Click “Torque Record Reset” to restore the status to the default state.

1. “Torque recording start” command node, parameters:

- Smoothing selection: unsmoothed (original data)/smoothed (smoothed data)
- Joint negative threshold (Nm): -100 ~ 0
- Joint positive threshold (Nm): 0 ~ 100
- Joint continuous detection collision time (ms): 0 ~ 1000

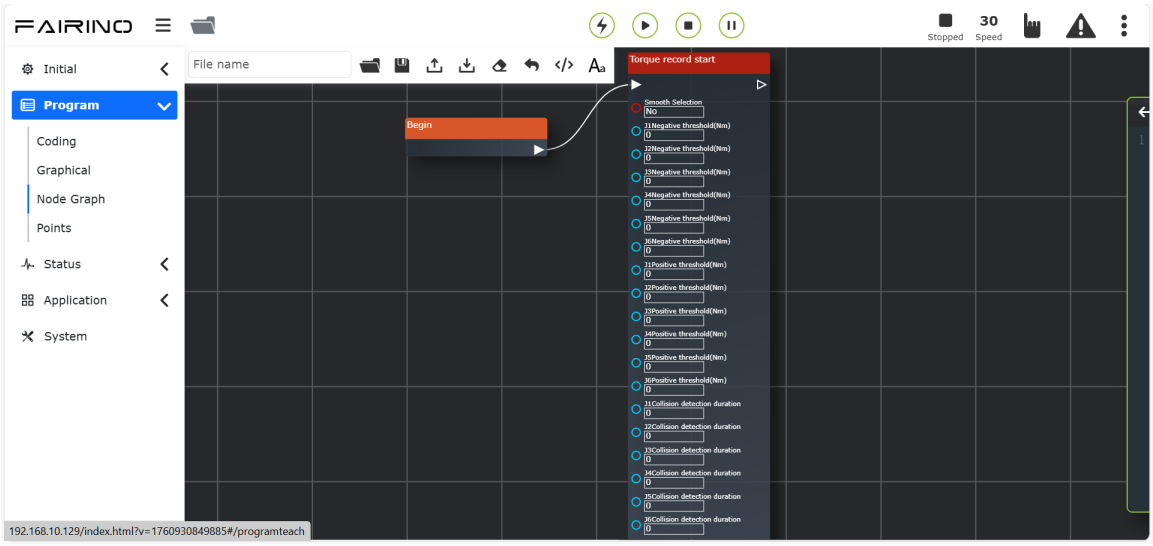


Figure 11.52-1 “Torque recording start” command node interface

2. “Torque recording end” command node

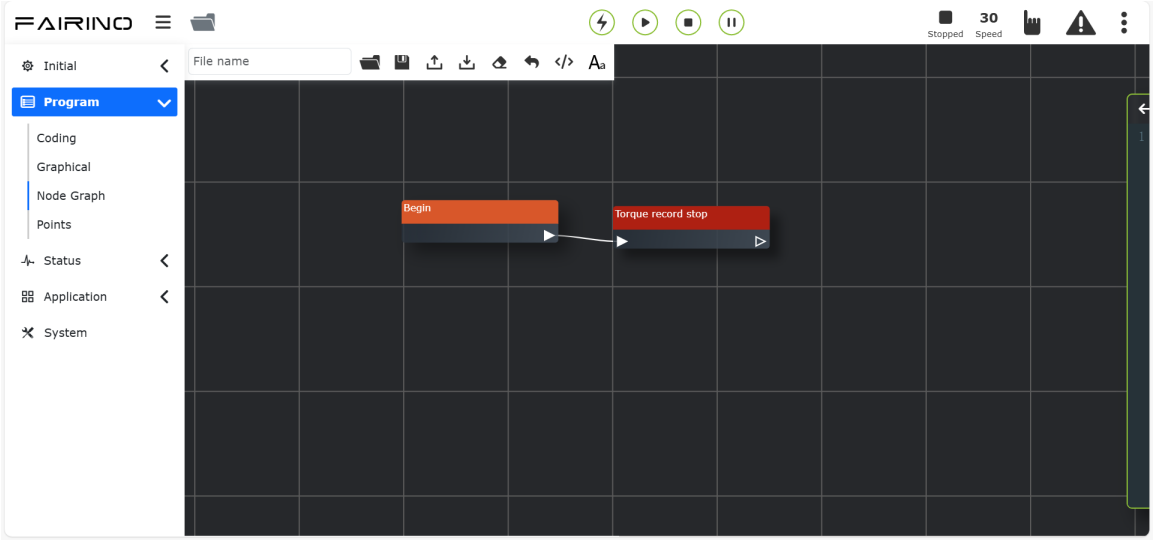


Figure 11.52-2 “Torque recording end” command node interface

3. “Torque record reset” command node

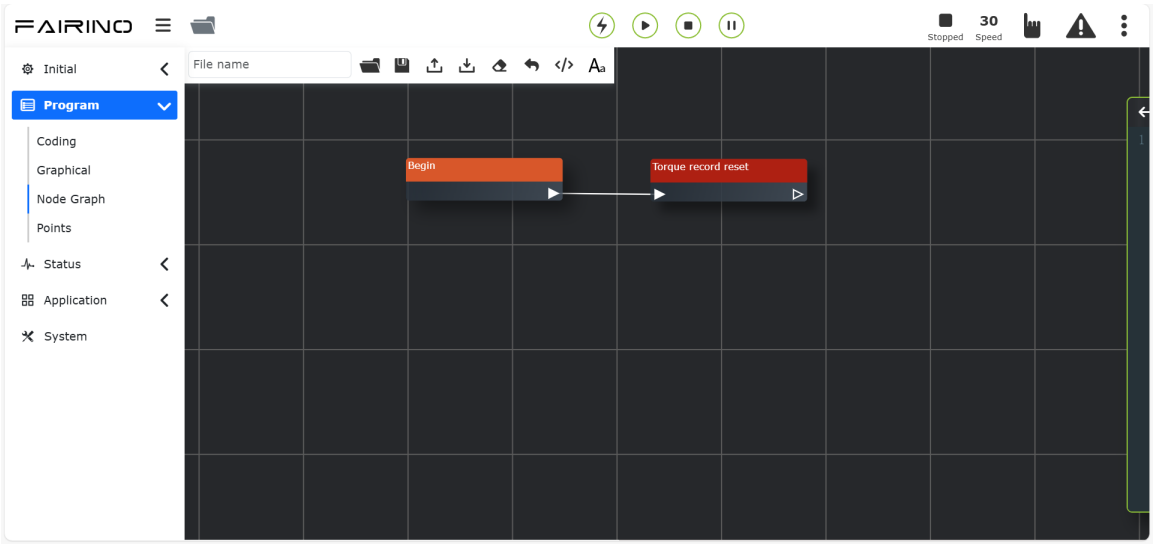


Figure 11.52-3 “Torque record reset” command node interface

1.3.11.53 Modbus instruction

Click the “Mobus” related command node to enter the node graph programming interface.

This instruction function is a bus function based on the ModbusTCP protocol. Users can control the robot to communicate with the ModbusTCP client or server (master station and slave station communication) through relevant instructions, and read and write Digital Output, Digital Input, and registers. For more operating functions of ModbusTCP, please contact us for consultation.

Master station register information: Master station register information includes type (DI, DO, AI, AO), address number, name and address value (DI and AI types cannot be entered), which can be edited and deleted.

1. Master station Digital Output settings, parameters:

- Modbus master station name: configured according to actual situation
- DO name: configured according to actual situation
- Number of registers: integer type 0 ~ 128
- Register value: Determined according to the number of registers, multiple values can be entered. For example, the quantity is 3 and the values are 1,0,1

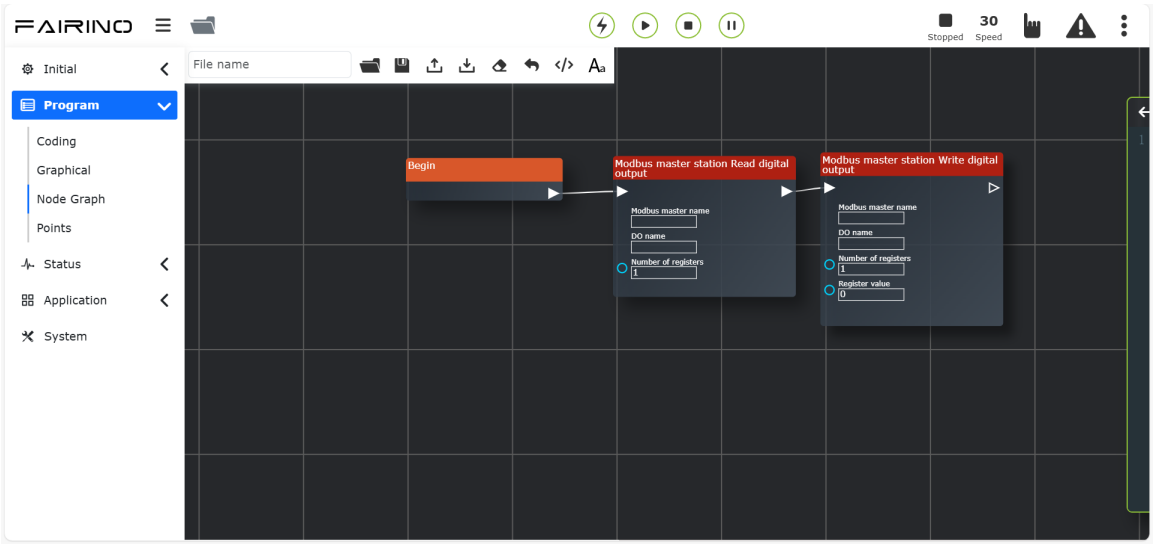


Figure 11.53-1 Master station “read/write Digital Output” command node interface

2. Master station Digital Input setting, parameters:

- Modbus master station name: configured according to actual situation
- DI name: configured according to actual situation
- Number of registers: integer type 0 ~ 128

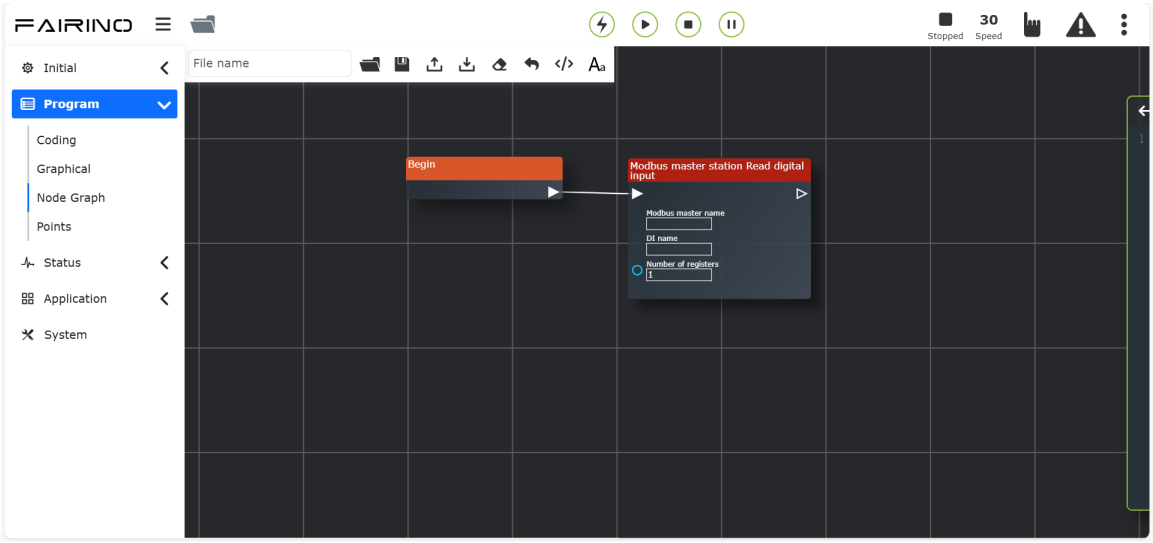


Figure 11.53-2 Master station “read Digital Input” command node interface

3. Master station analog output settings, parameters:

- Modbus master station name: configured according to actual situation
- AO name: configured according to actual situation
- Number of registers: integer type 0 ~ 128
- Register value: Determined according to the number of registers, multiple values can be entered. For example, the quantity is 3 and the values are 1,0,1

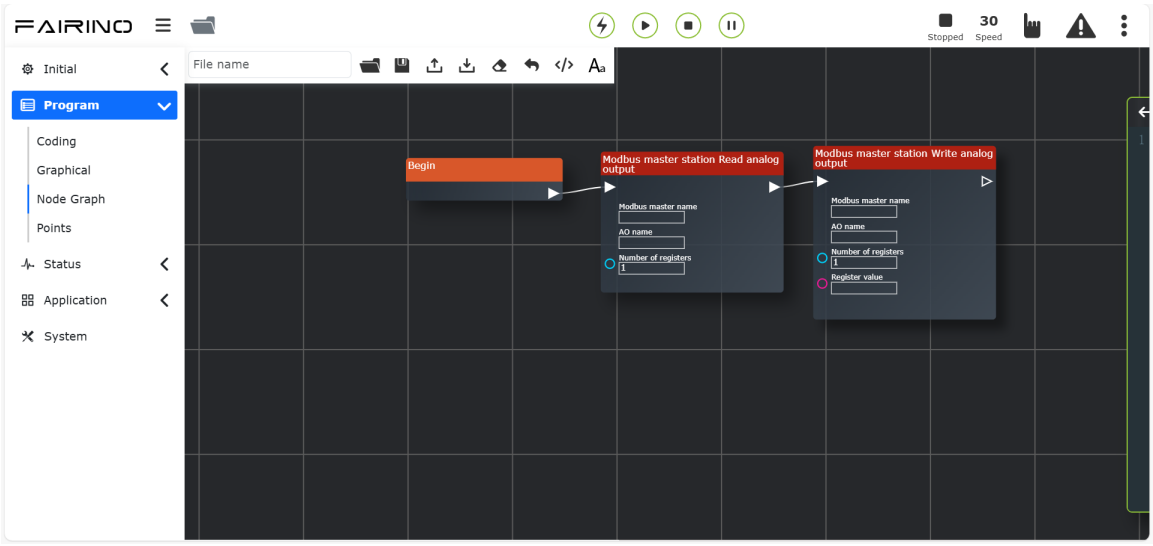


Figure 11.53-3 Master station “read/write analog output” command node interface

4. Master station analog input settings, parameters:

- Modbus master station name: configured according to actual situation
- AI name: Configure according to actual situation
- Number of registers: integer type 0 ~ 128

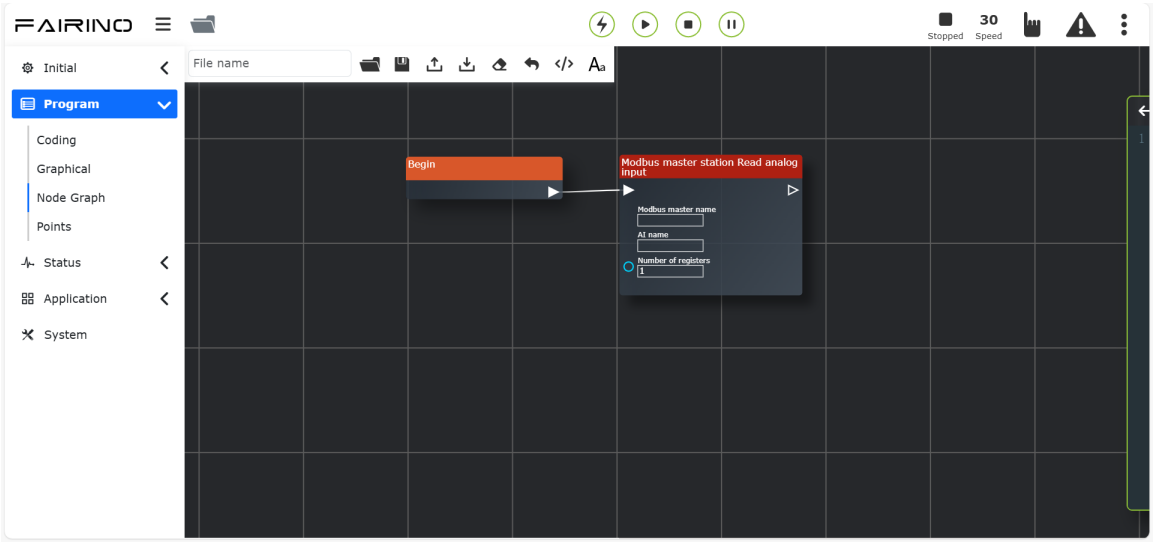


Figure 11.53-4 Master station “read analog input” command node interface

5. The master station waits for digital input settings, parameters:

- Modbus master station name: configured according to actual situation
- DI name: configured according to actual situation
- Waiting status: true/false
- Timeout (ms): integer

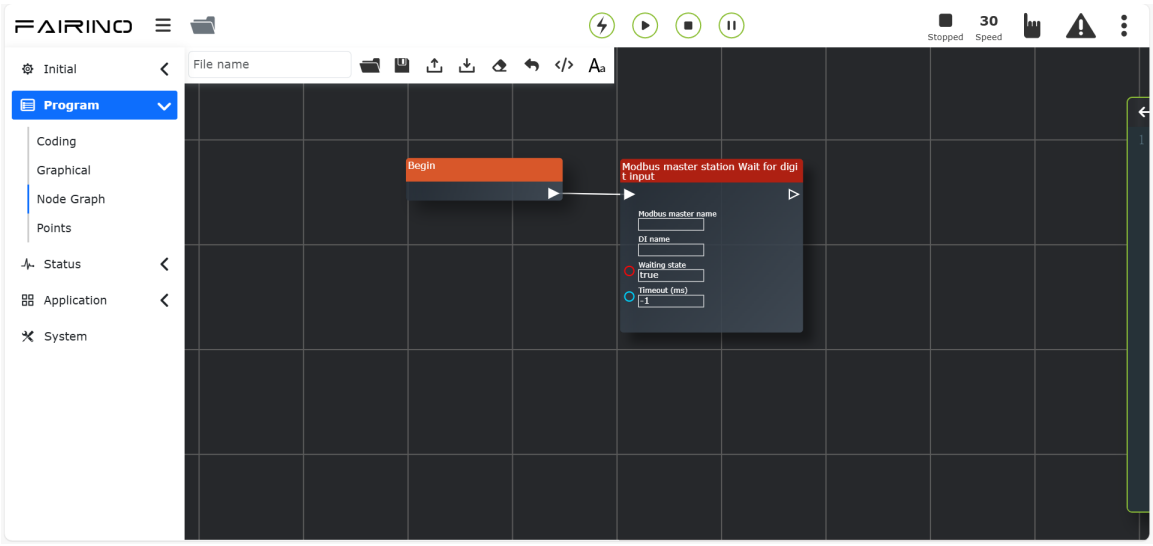


Figure 11.53-5 Master station “waiting for digital input” command node interface

6. The master station waits for analog word input settings, parameters:

- Modbus master station name: configured according to actual situation
- AI name: Configure according to actual situation
- Waiting status: greater than/less than
- Number of registers: integer type 0 ~ 128
- Register value: Determined according to the number of registers, multiple values can be entered.

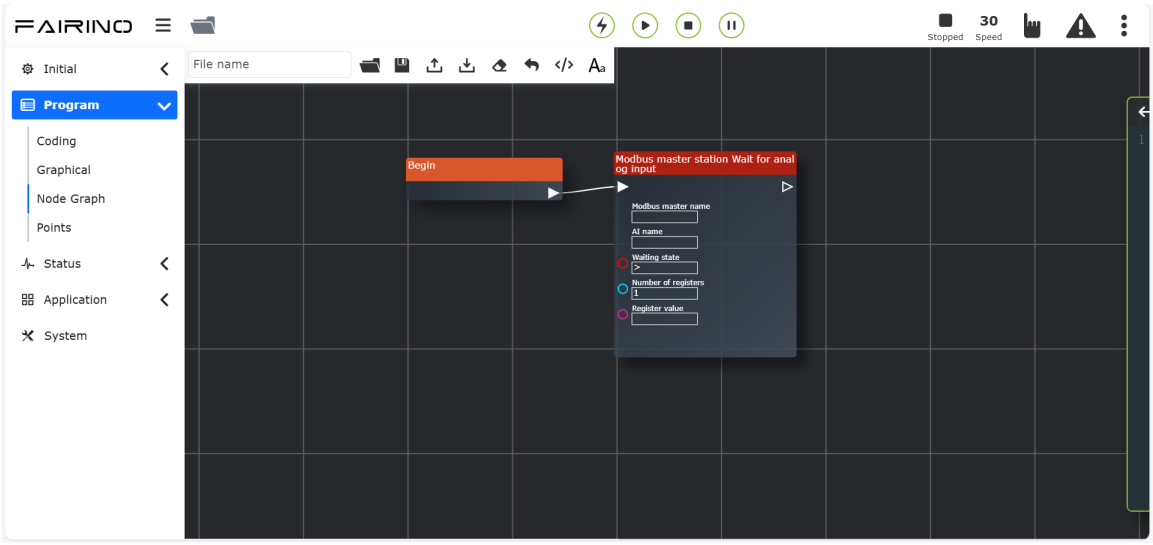


Figure 11.53-6 Master station “wait for analog input” command node interface

7. Slave Digital Output settings, parameters:

- DO name: configured according to actual situation
- Number of registers: integer type 0 ~ 128
- Register value: Determined according to the number of registers, multiple values can be entered. For example, the quantity is 3 and the values are 1,0,1

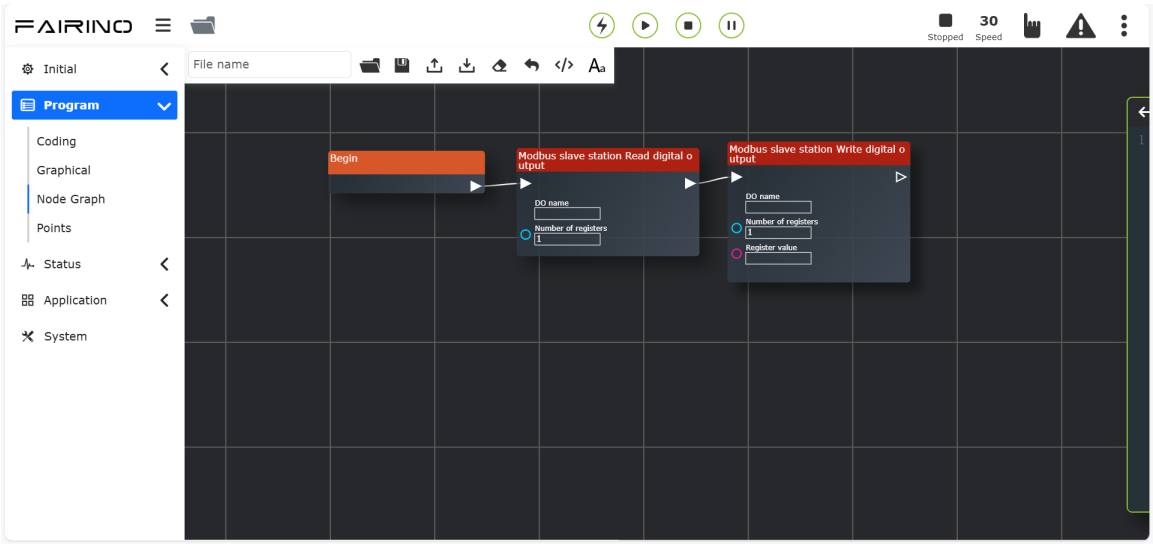


Figure 11.53-7 Slave “read/write Digital Output” command node interface

8. Slave station Digital Input setting, parameters:

- DI name: configured according to actual situation
- Number of registers: integer type 0 ~ 128

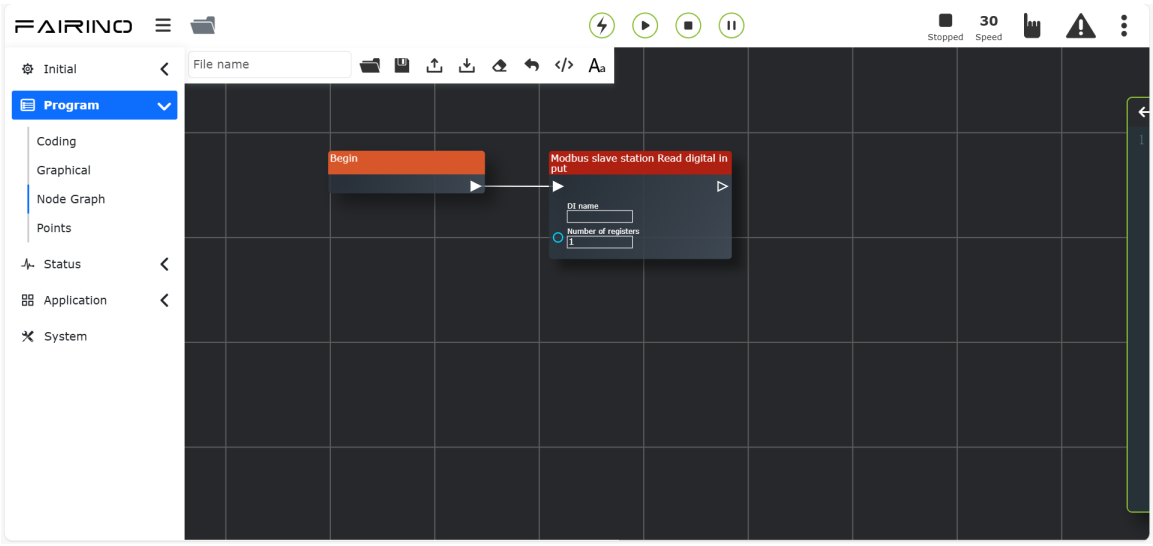


Figure 11.53-8 Slave station “read Digital Input” command node interface

9. Slave station analog output settings, parameters:

- AO name: configured according to actual situation
- Number of registers: integer type 0 ~ 128
- Register value: Determined according to the number of registers, multiple values can be entered. For example, the quantity is 3 and the values are 1,0,1

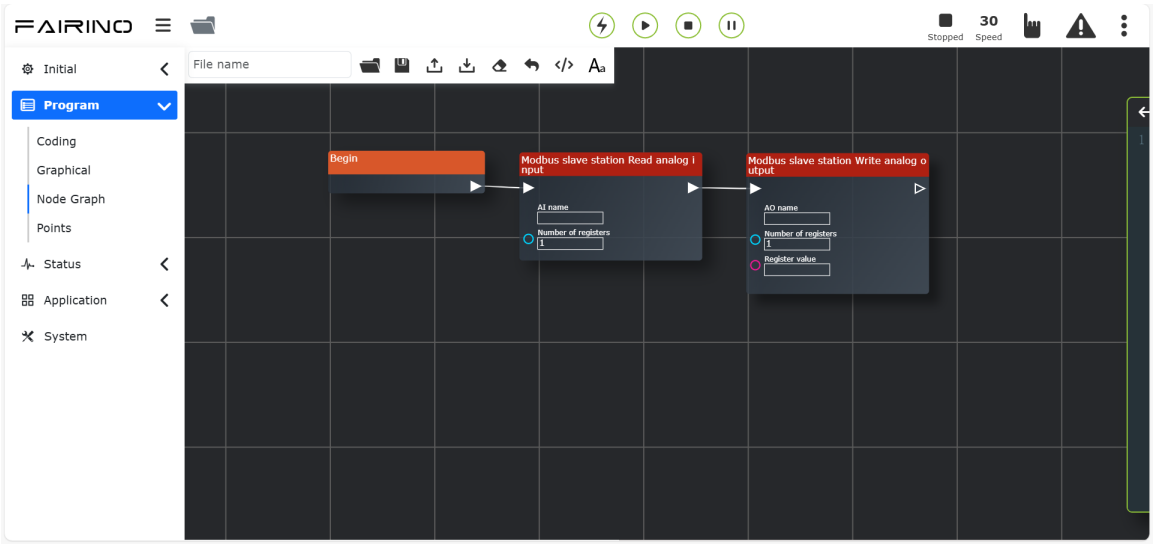


Figure 11.53-9 Slave “read/write analog output” command node interface

10. The slave station waits for digital input settings, parameters:
    - DI name: configured according to actual situation
    - Waiting status: true/false
    - Timeout (ms): integer

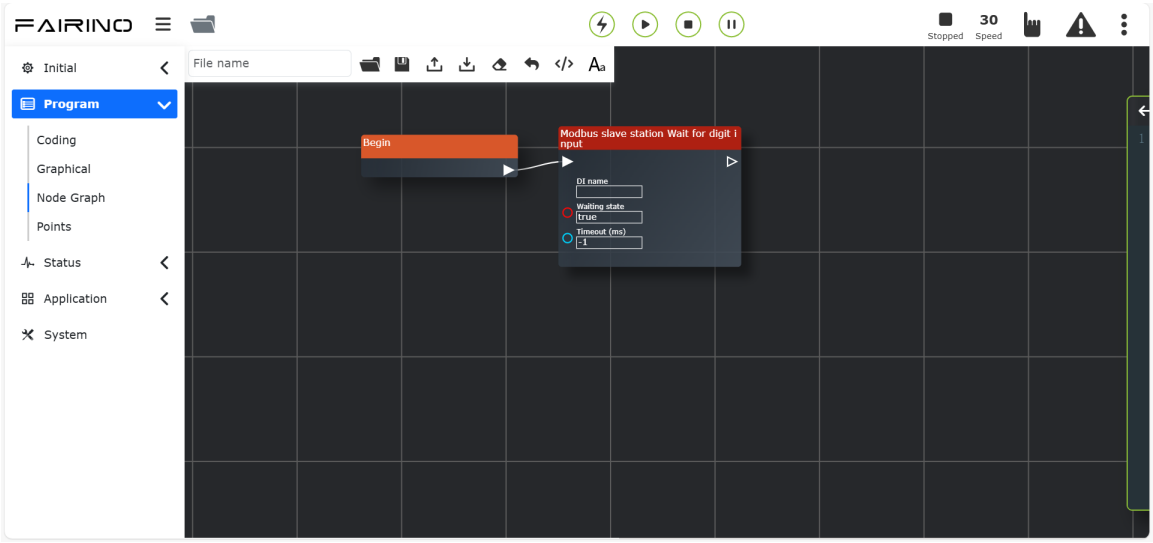


Figure 11.53-10 Slave “wait for digital input” command node interface

11. The slave station waits for analog word input settings, parameters:
    - AI name: Configure according to actual situation
    - Waiting status: greater than/less than
    - Number of registers: integer type 0 ~ 128
    - Register value: Determined according to the number of registers, multiple values can be entered.

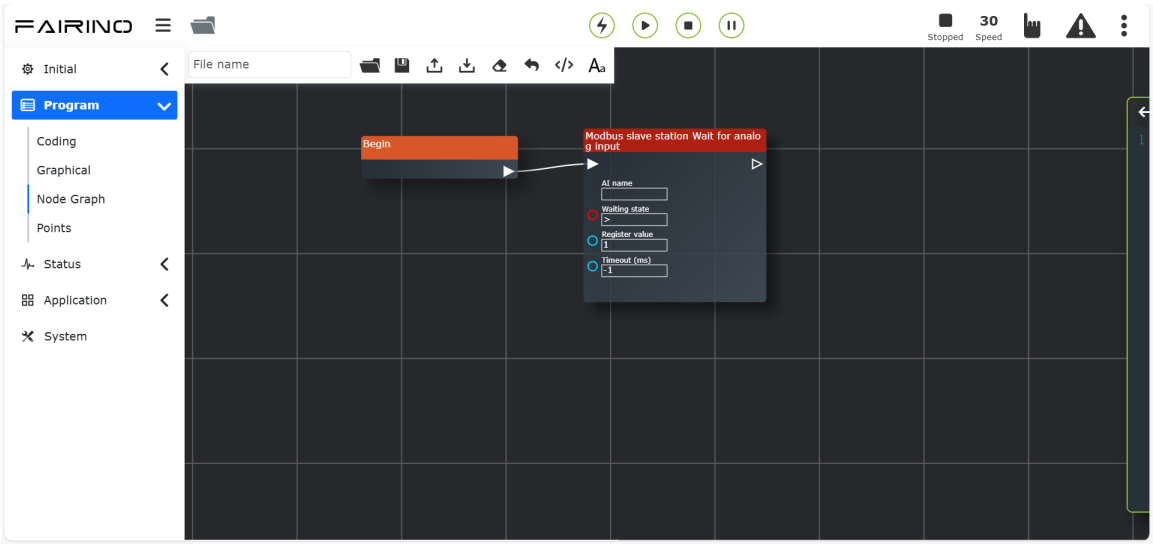


Figure 11.53-11 Master station “wait for analog input” command node interface

12. Slave station analog input settings, parameters:

- AI name: Configure according to actual situation
- Number of registers: integer type 0 ~ 128

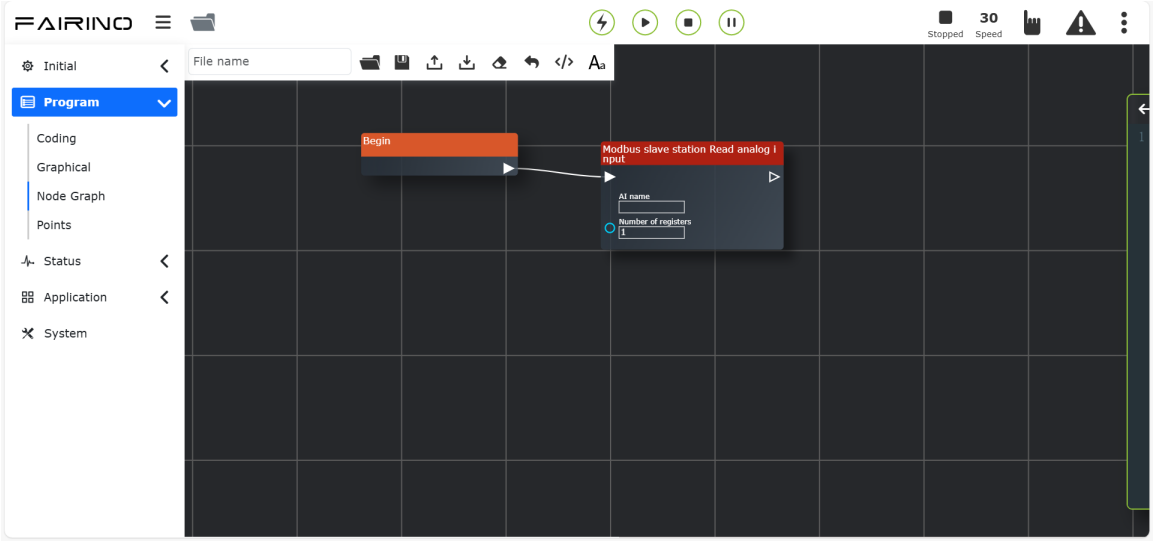


Figure 11.53-12 Slave “read analog input” command node interface

1.3.11.54 Application scenario usage examples

For example, install a tip on the end of the robot and drag it to a position near the hole of the pallet. You want to perform spiral, rotational and linear insertion operations of the force sensor.

- First, right-click the mouse button and select the “Begin”, “Start/End Control”, “Spiral Insertion”, “Rotation Insertion”, and “Linear Insertion” command nodes;
- Connect as follows and configure relevant parameters.

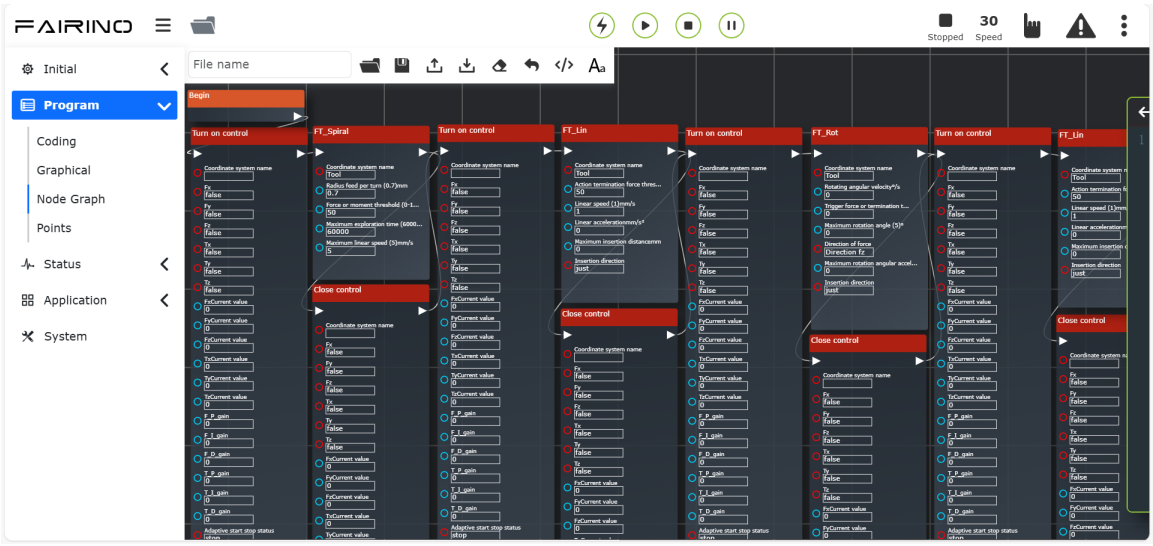


Figure 11.54-1 “Force Control” command node application configuration interface

- Enter the file name. If the correct parameters are not entered, the save will fail and a prompt will appear indicating that the command node parameter configuration is incorrect.

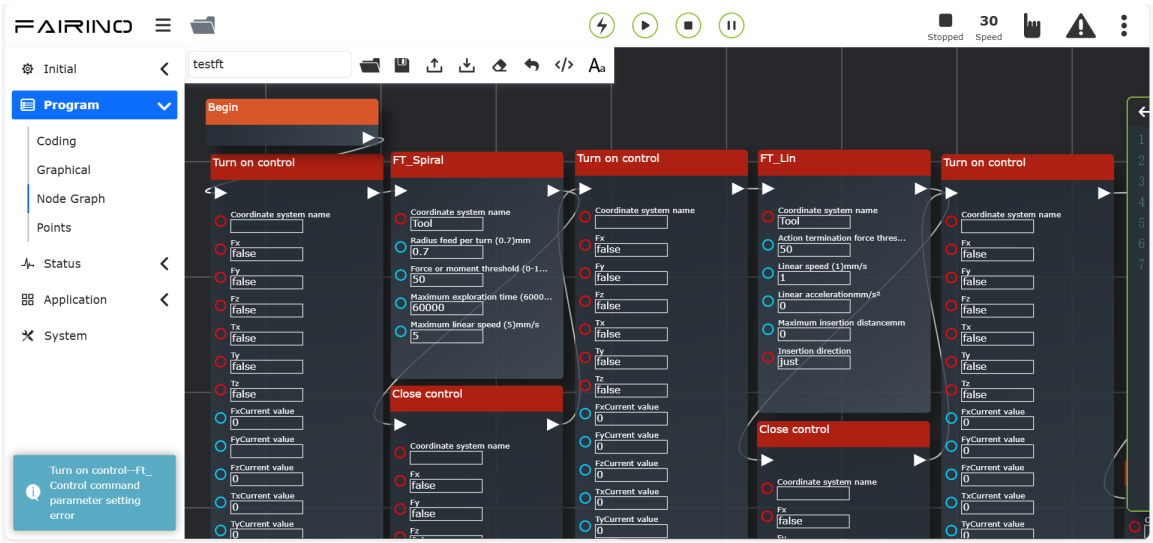


Figure 11.54-2 Command node parameter configuration error interface

- After clicking Run, the robot will explore in a spiral plus straight line motion. After exploring the correct hole position, use linear and rotational insertion movements until the hole is correctly inserted.

1.3.12 Teaching Points

Teaching management is divided into two modes: “System Mode” and “Point Table Mode”. When calling the manipulator program, different inspection schemes can be achieved by calling different point tables, fulfilling recipe requirements. For each additional device or product subsequently, the point table data package can be downloaded to the robot via the host computer. Point table data packages newly created by the robot can also be uploaded to the host computer.

System Mode: Supports “importing, exporting, deleting, renaming, modifying, overwriting, modifying, viewing” teaching point content, as well as single-step movement to teaching points.

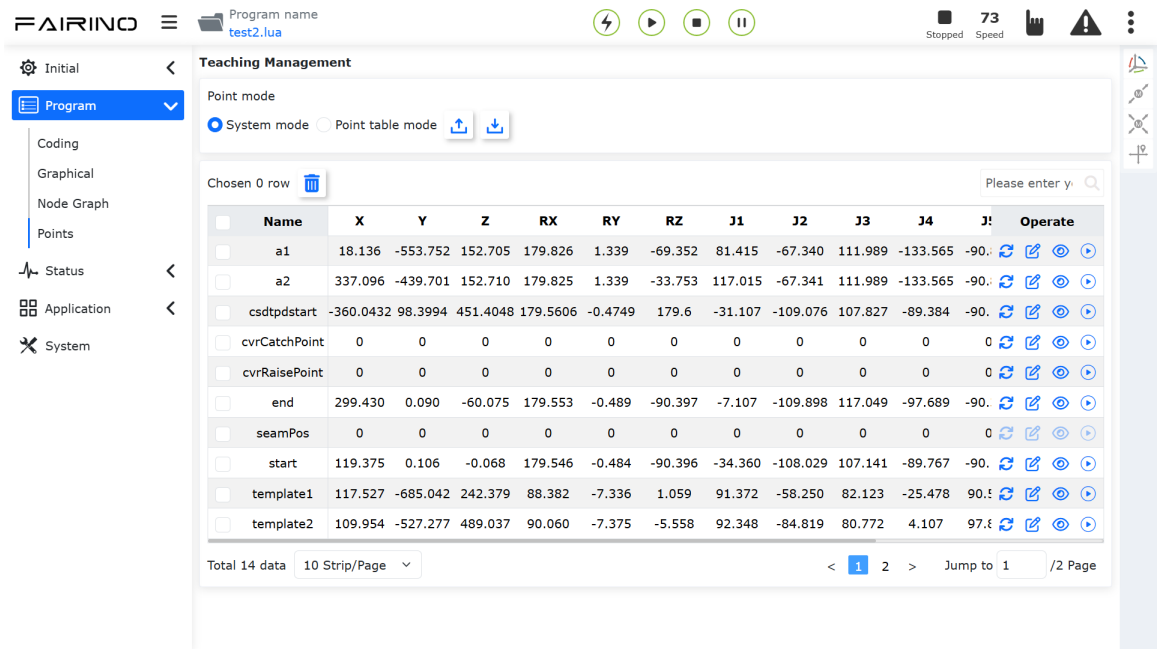


Figure 12.1-1 Teaching Management Interface - System Mode

Point Table Mode: Supports “adding, applying, renaming, deleting, importing, exporting” point tables, and “deleting, modifying, viewing, and overwriting” point content within a point table, as well as single-step movement to teaching points.

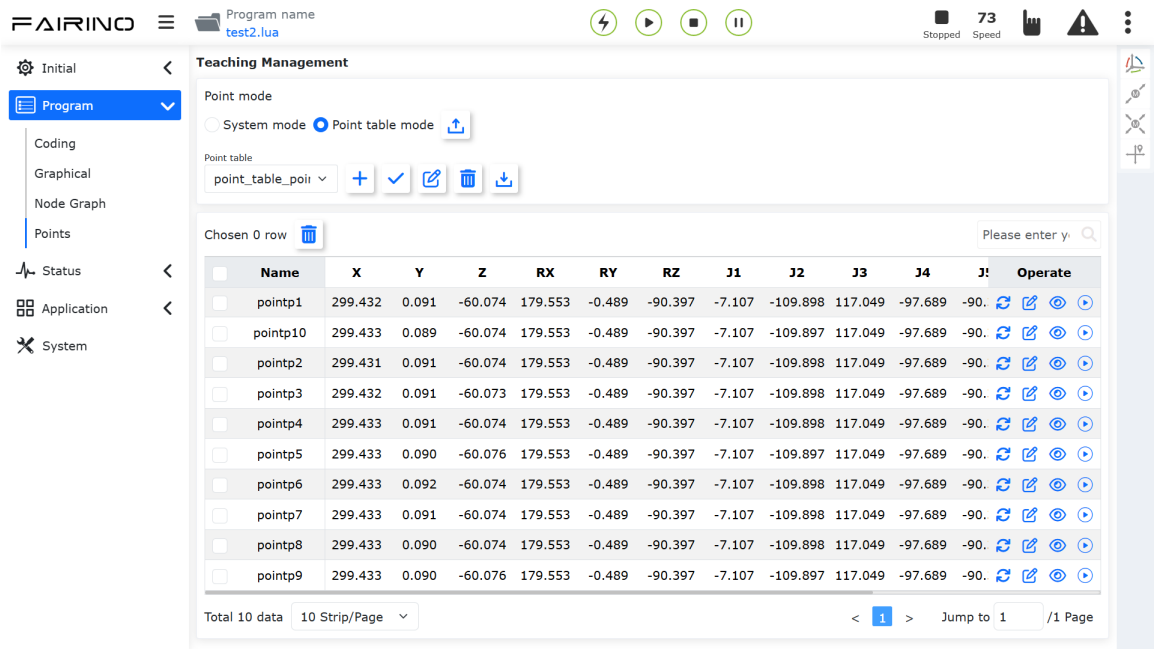


Figure 12.1-2 Teaching Management Interface - Point Table Mode

The top right corner of the Teaching Management interface displays the robot body operation bar. Users can move the robot body on this interface and then perform data overwrite operations for teaching points.

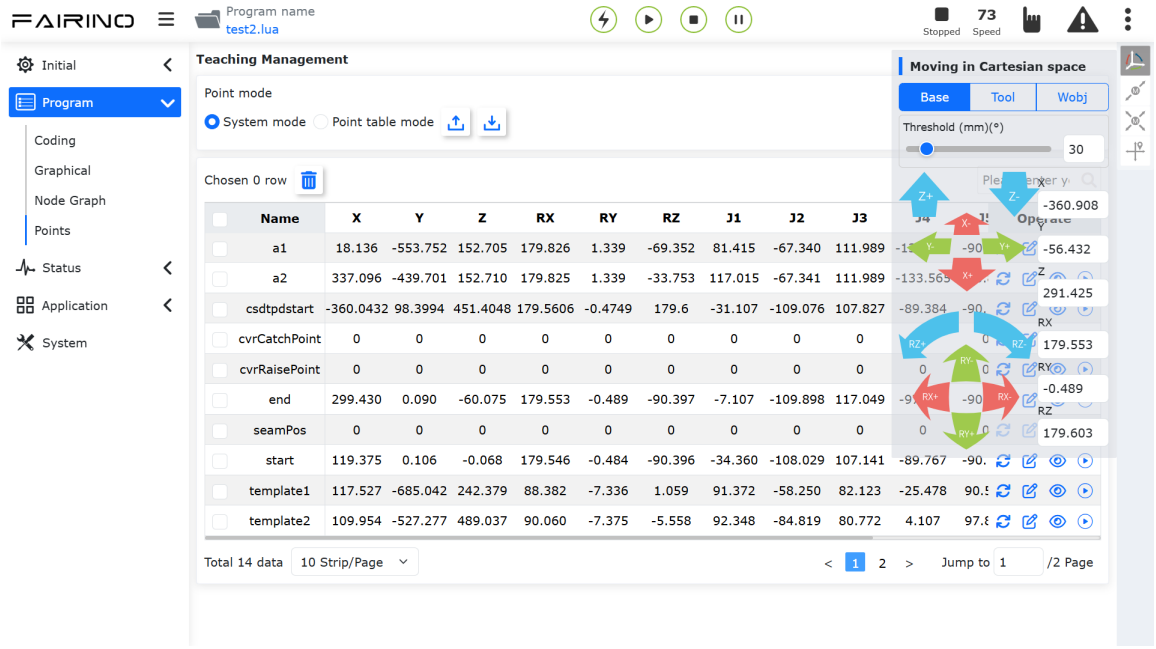


Figure 12.1-3 Teaching Management Interface - Robot Body Operation Bar

You can enter a teaching point name in the top right corner of the teaching point table data to search; Click on the teaching point name in the teaching point table data to enter edit mode, input the modified name, and click anywhere outside the teaching point name to complete the modification.

Note: 

Name: **Import Button**

Function: Import teaching point file

Note: 

Name: **Export Button**

Function: Export teaching point file

Note: 

Name: **Delete Button**

Function: After selecting one/multiple teaching points, click the “Delete” button above the table, the prompt “Please click the delete button again to confirm deletion” appears. Click again to delete the point information;

Note: 

Name: **Overwrite Point Button**

Function: Click to overwrite the teaching point with the robot’s current point data, and select “Whether to sync the teaching program” in the pop-up window

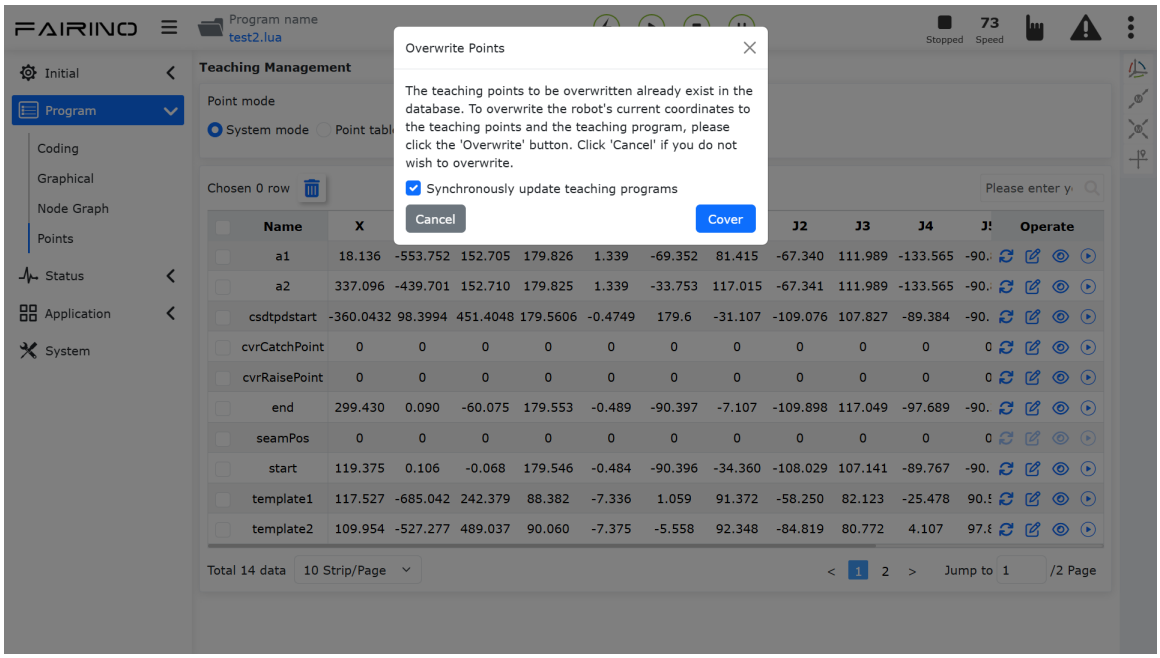


Figure 12.1-4 Overwrite Teaching Point

Note: 

Name: **Edit Button**

Function: Click to confirm modification of the teaching point's x, y, z, rx, ry, rz, and v values

Important: The modified values for the teaching point's x, y, z, rx, ry, rz should not exceed the robot's working range.

Note: 

Name: **Details Button**

Function: Click to view teaching point details

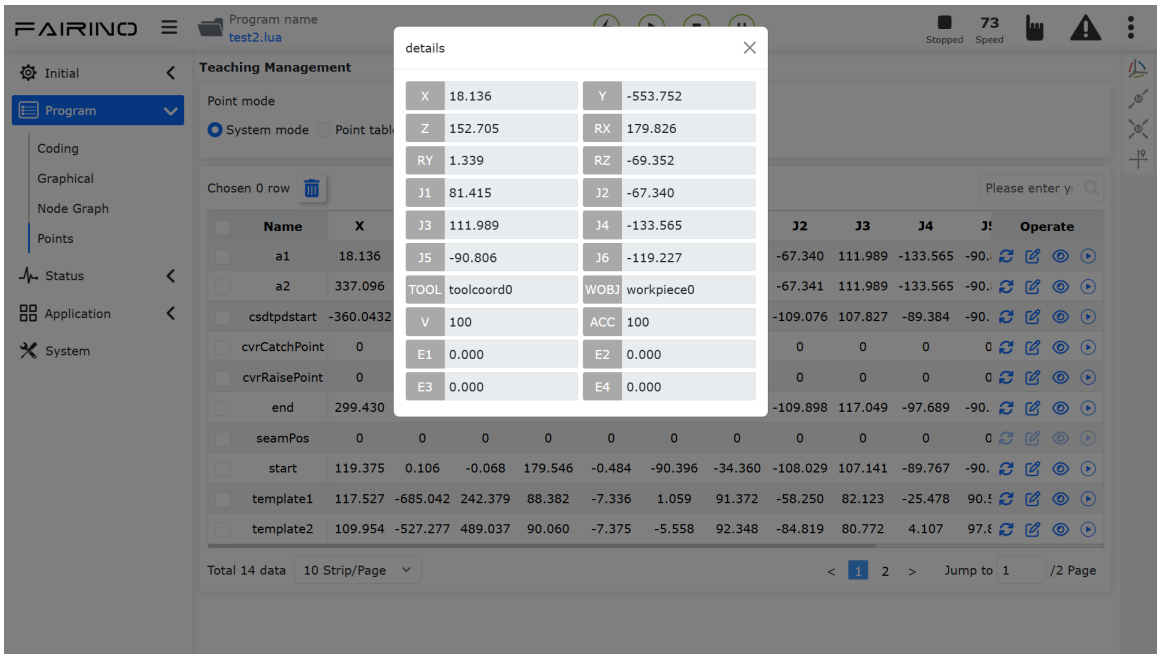


Figure 12.1-5 Teaching Point Details

Note:

Name: **Start Run Button**

Function: Click to select the method for single-point operation to move the robot to the position of this point; Select PTP for Point-to-Point motion, select Lin for Linear motion.

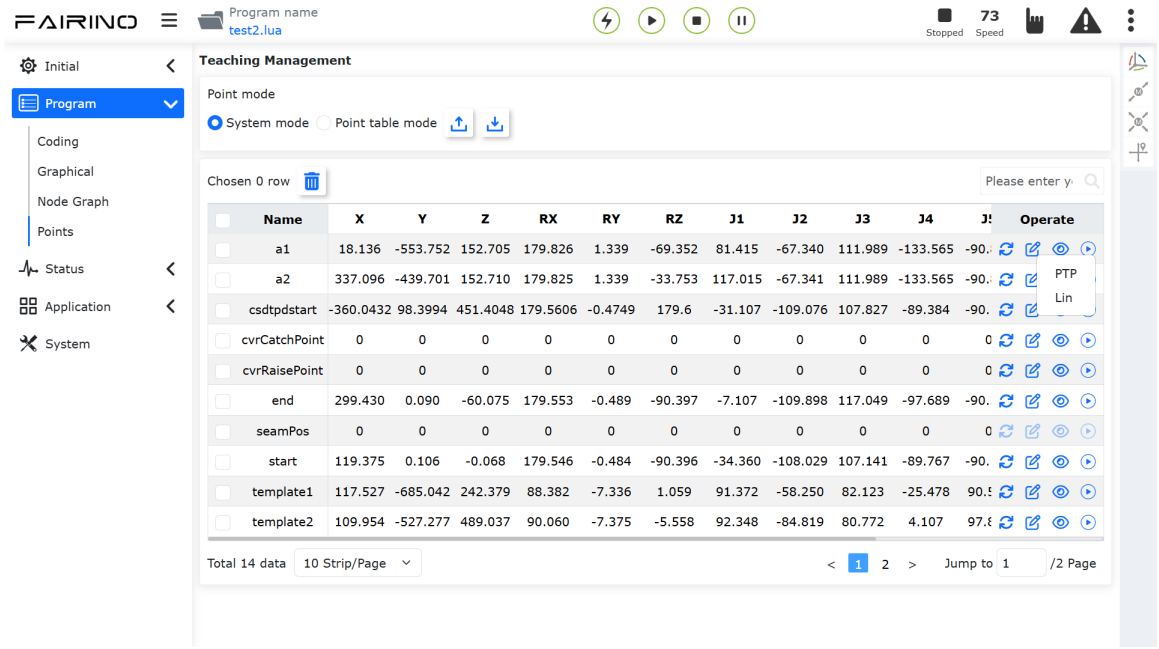


Figure 12.1-6 Run Teaching Point

1.3.13 Status

1.3.13.1 System log

When you enter the “Status Information” -> “System Log” interface for the first time, all types of log data for the day are displayed by default.

The log data is classified into levels, currently divided into: all, error warnings, basic settings, security settings, peripheral settings, main unit operations, teaching programs, tool applications, system settings, and file import and export.

There is a search input box in the upper right corner of the data table. Users can enter the filter content according to their search needs. The interface is as follows:

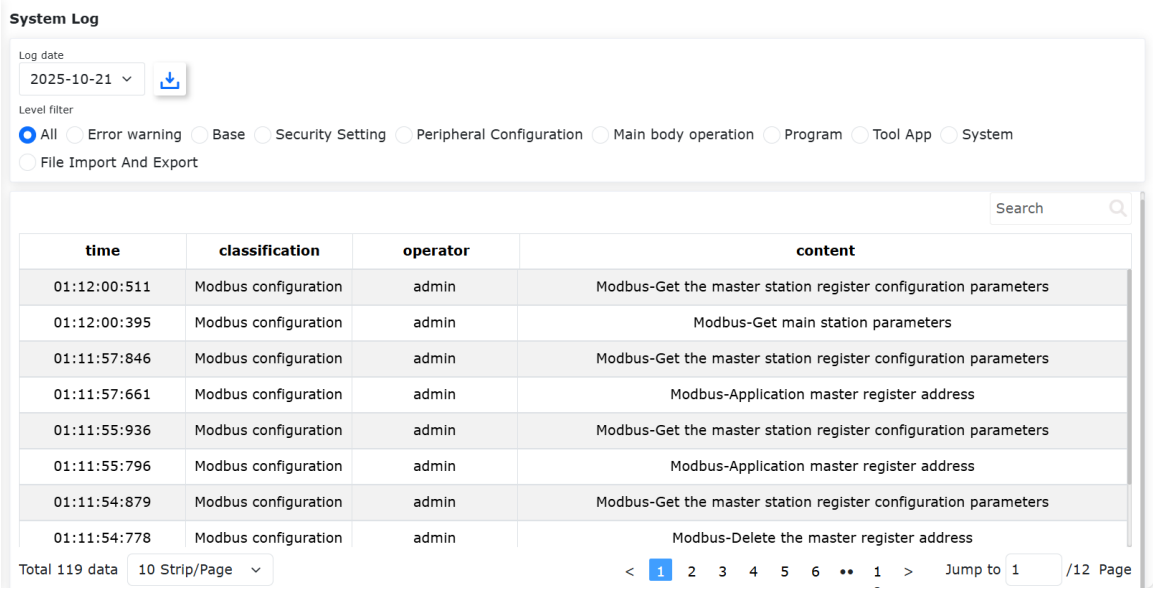


Figure 13.1-1 System log interface

1.3.13.2 Status Query

1.3.13.2.1 Function Usage

1. Turn on the control box and connect the Ethernet cable to the PC;
2. Open a browser on the PC and visit the target URL: 192.168.58.2. Log in with the account “admin” and password “123” to access the page;
3. Click the “Status Information” -> “Status Query” option in the left menu bar to enter the status query interface, as shown below;

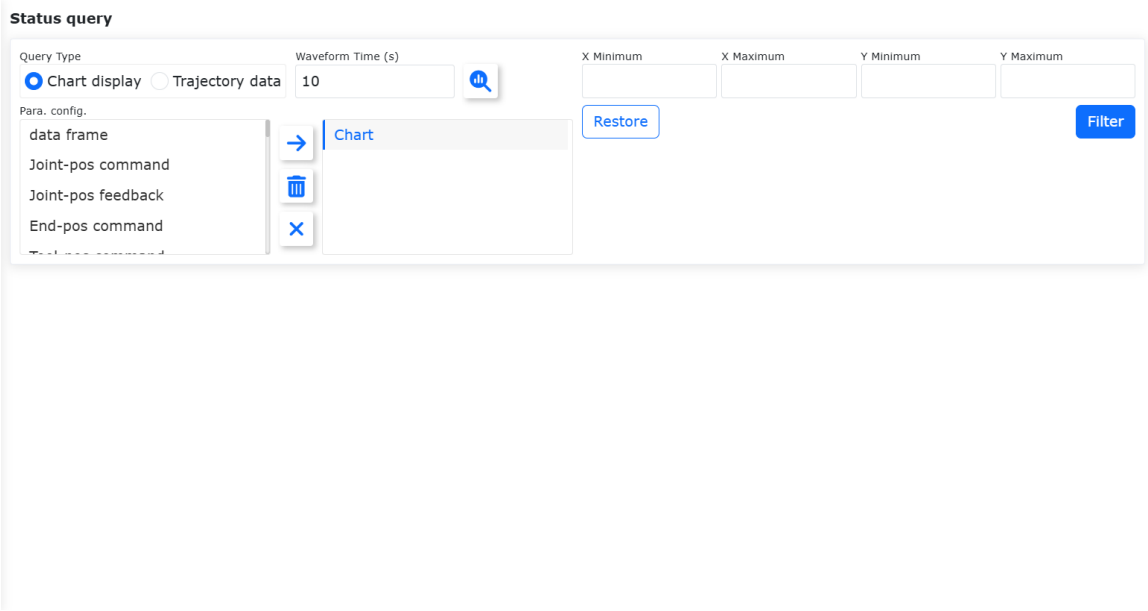


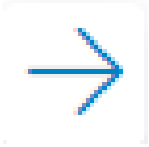
Figure 13.2-1 Status Query



Note:

Name: **Query Button**

Function: Click to send an instruction to query chart/trajectory data, indicating an unqueried state.



Note:

Name: **Right Arrow Button**

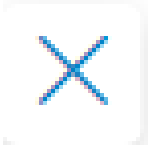
Function: Click to add the selected item on the left to the sub-items on the right.



Note:

Name: **Delete Button**

Function: Click to delete the selected sub-item on the right.



Note:

Name: **Clear Button**

Function: Click to clear all sub-items on the right.

4. Select “Chart Display,” fill in the waveform time, and choose the desired query parameters in the “Parameter Configuration” section on the left. Click the “Right Arrow” button to add the parameters to the list on the right;

Note: The waveform time can be customized (10-30s), and a maximum of 6 parameters can be selected.

5. Click the “Query” button to start the query. Based on the parameter configuration, real-time data will be displayed as a line chart, as shown below;

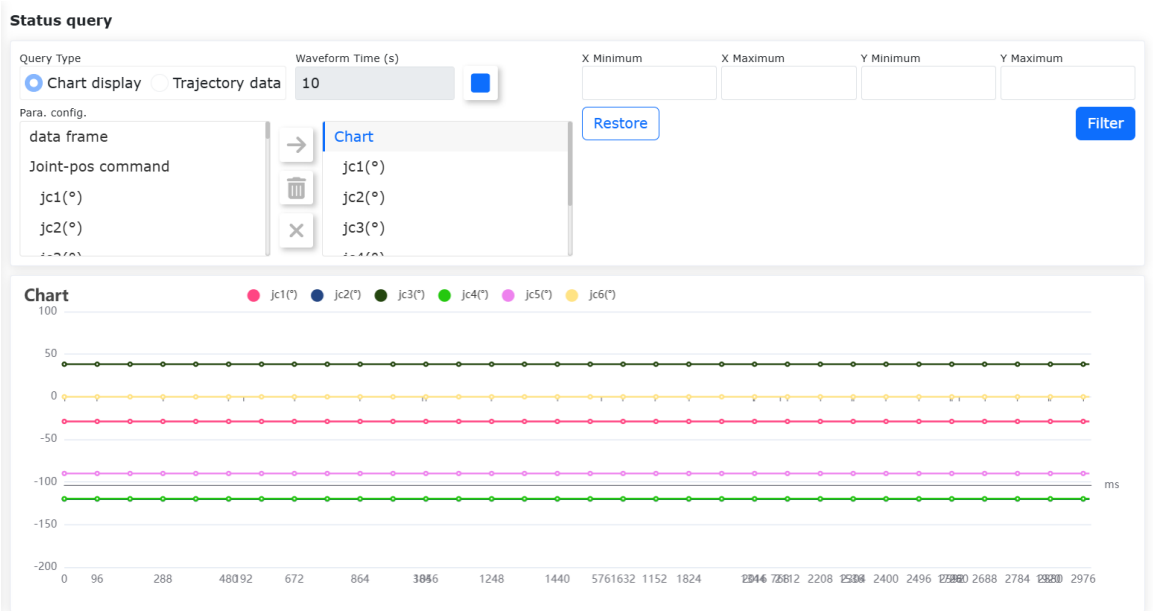


Figure 13.2-2 Chart Display

1.3.13.2.2 Chart Export

1. Click the chart title to open a dialog box where you can directly modify the title, as shown below:

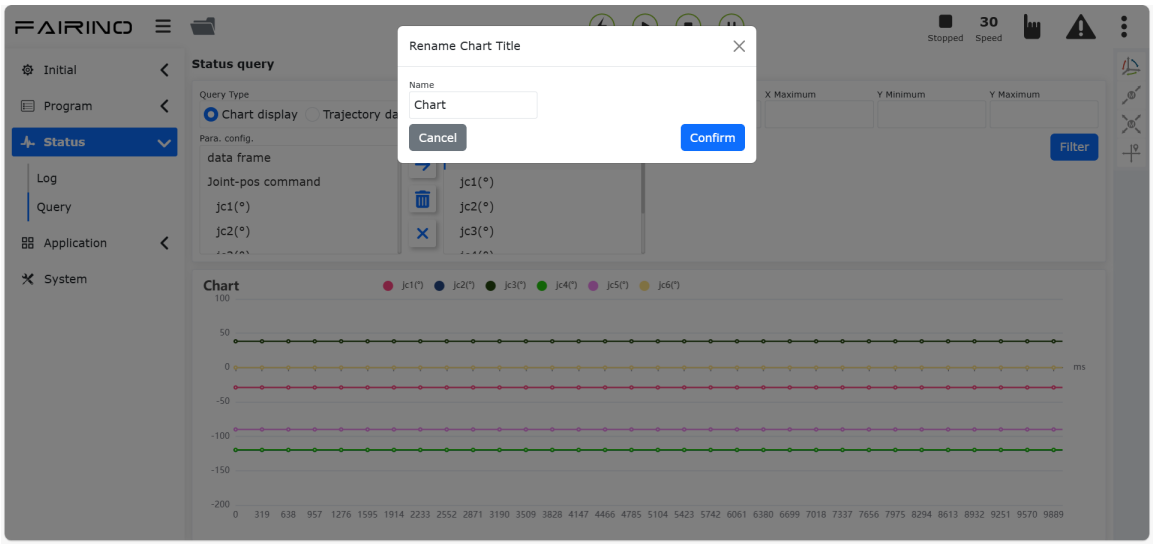


Figure 13.2-3 Rename Chart Title

2. After clicking the "Stop Query" button to successfully stop the query, a download button will appear. Click it, and the browser will download the chart file with the chart title as the filename, as shown below:



Figure 13.2-4 Chart Export

1.3.13.2.3 Data View Display

1. After stopping the query, click the “Show Data View” button in the upper-right corner of the chart, as shown below:

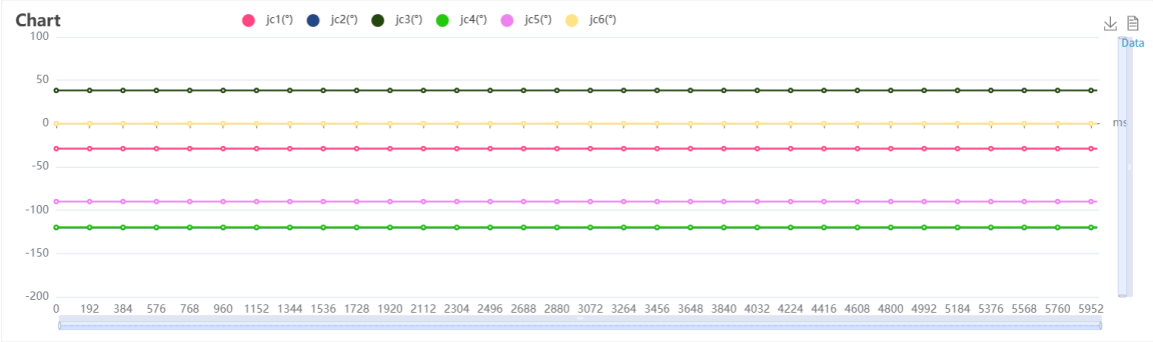


Figure 13.2-5 Data View Button

2. The data in the view is shown below, and the content supports copying.

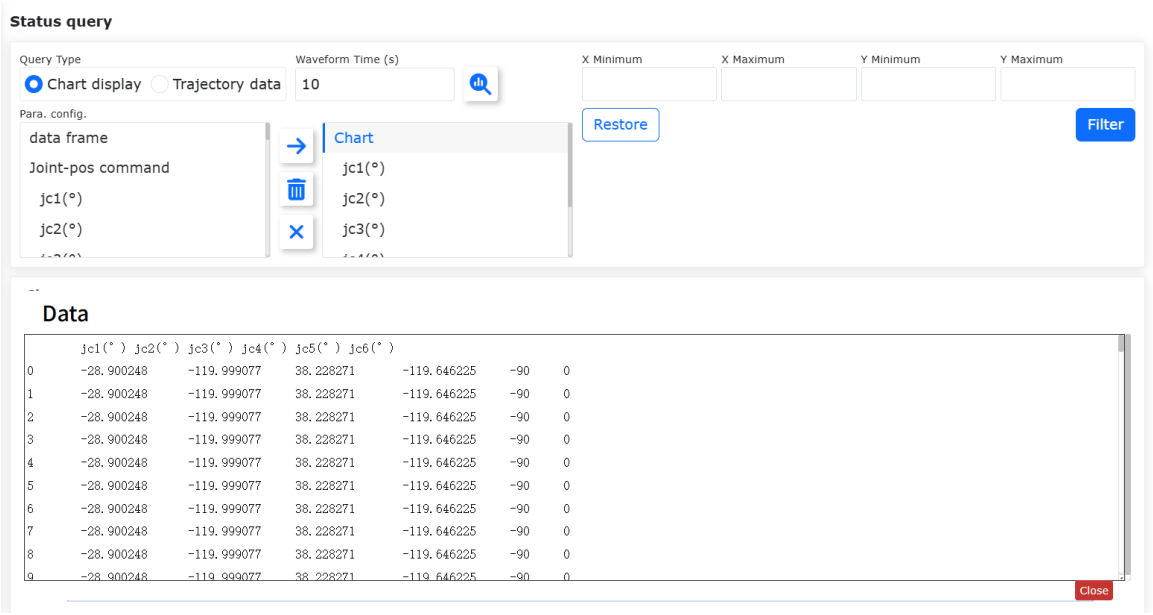


Figure 13.2-6 Data View Display

1.3.13.2.4 Data Filtering

1. After stopping the query, input the minimum/maximum values for x/y, and the chart data range will change accordingly, as shown in the figure below:

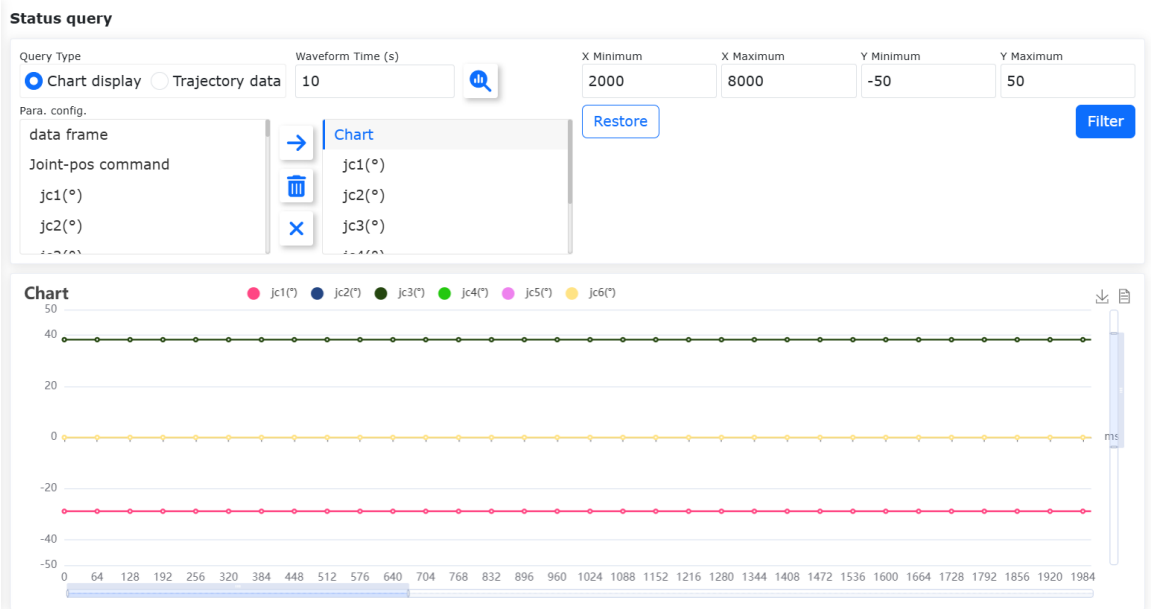


Figure 13.2-7 Data Filtering Interface

2. Click the restore button, and the chart data range will revert to the default, as shown in the figure below:

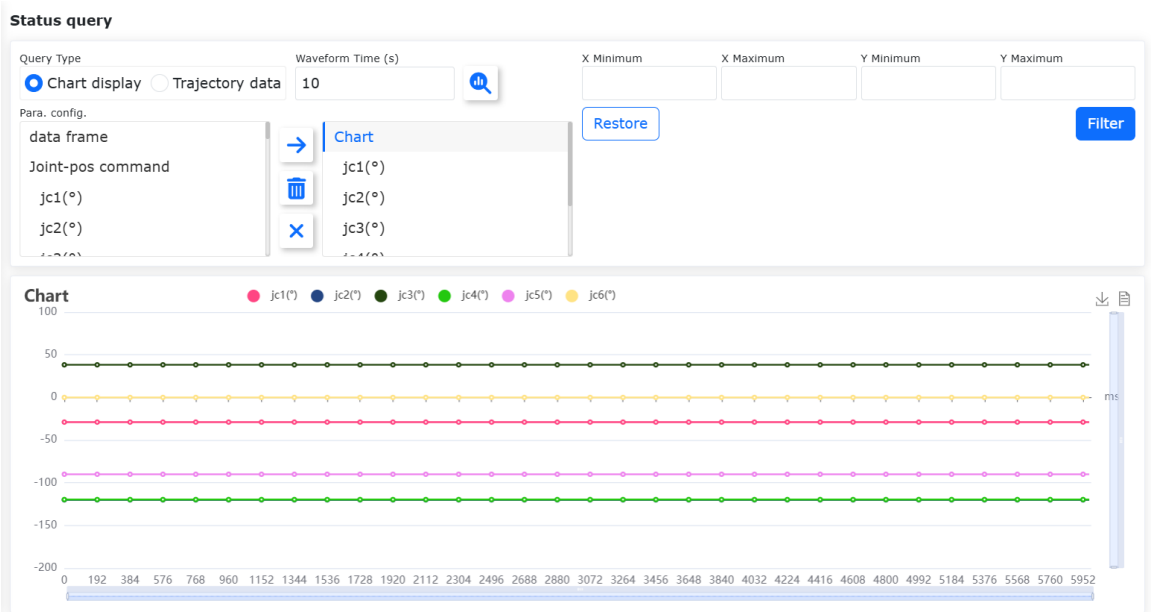


Figure 13.2-8 Data Restoration

1.3.14 Applications

1.3.14.1 Robot Packing

Under the "Applications" -> "Tool App" menu bar, click the "Robot Packing" button to enter the one-click robot packing interface.

Important: Before operating the packing function, please confirm the robot's surrounding environment and status to prevent collisions.

If leaving the factory, first go to System Settings -> General Settings and perform a factory reset before packing.

Step1: Before moving to the packing point, first move the robot to the zero point.

Step2: Click the "Move to Zero Point" button, confirm the robot's mechanical zero point is correct, and ensure the gaps in each joint are aligned as shown in the orange circles in the figure.

Step3: Click the "Move to Packing Point" button, the robot will move to the packing point according to the axis angles specified in the packaging process.

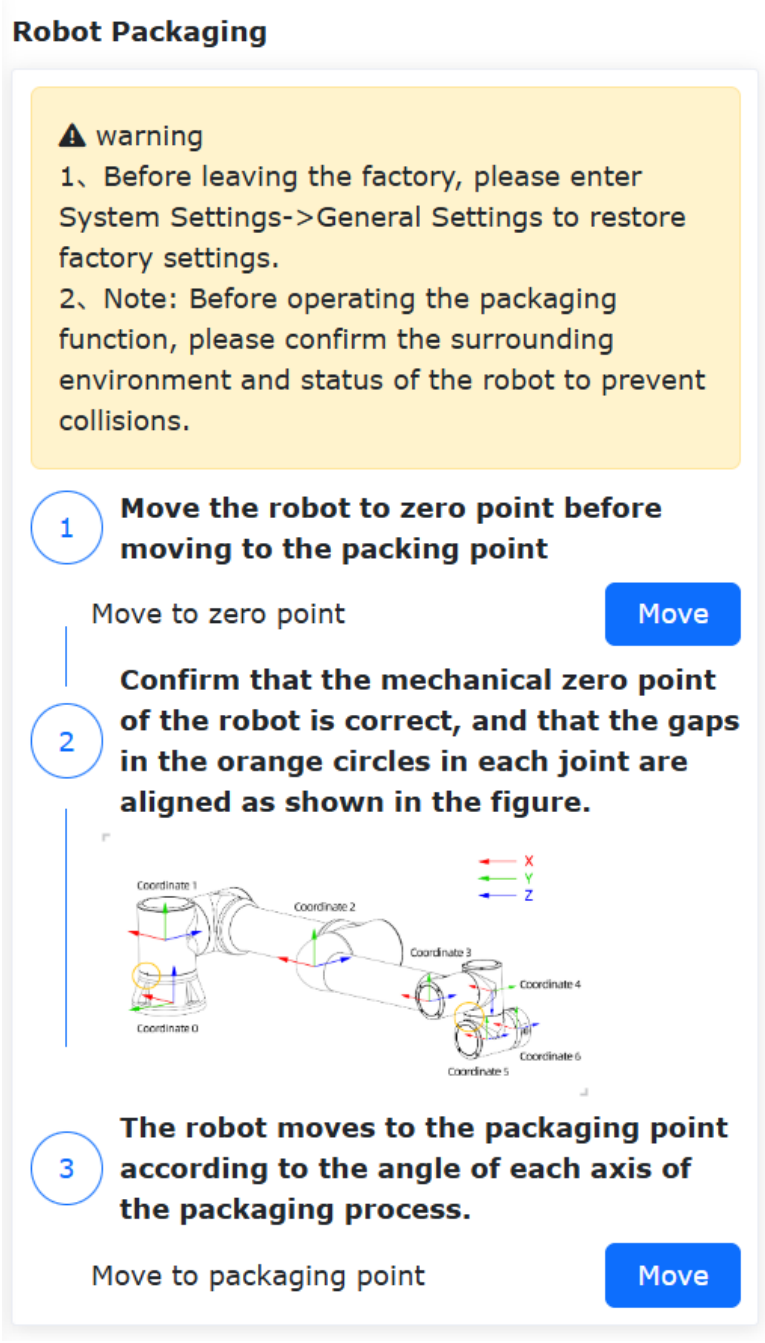


Figure 14.1-1 One-Click Robot Packing

1.3.14.2 Data Backup

1. Under the menu bar of “Auxiliary Applications” -> “Tool Applications”, click “Data Backup” to enter the data backup interface, as shown in the figure below.

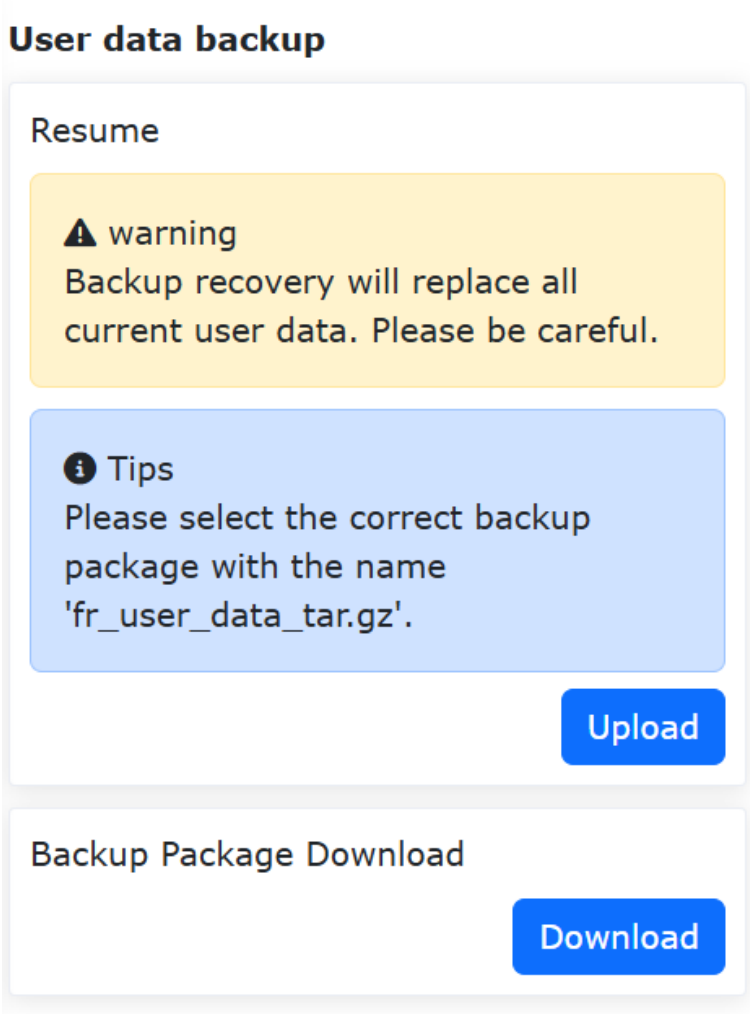


Figure 14.2-1 Data Backup Interface

2. The backup package data includes tool coordinate system data, system configuration files, teaching point data, user programs, template programs, and user configuration files. When users need to transfer data from this robot to another robot for use, this function can be used to achieve it quickly.

1.3.14.2.1 Backup Package Integrity Verification Function

To avoid potential security risks caused by inconsistent configurations such as installation methods during the import of backup packages, an MD5 verification function for backup packages and a verification function for key parameters have been added during backup package import.

1.3.14.2.1.1 Backup Package MD5 Verification Function

To ensure the integrity of the imported backup package, an MD5 verification will be performed on the backup package after upload. If the backup package is corrupted or abnormally modified, the MD5 verification will fail, and the following prompt will be displayed:

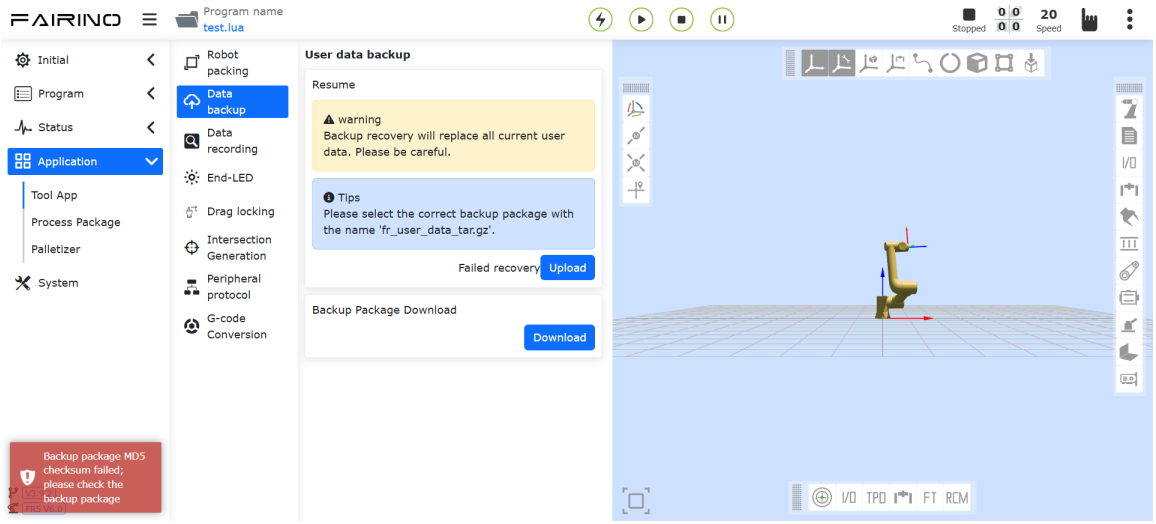


Figure 14.2-2 MD5 Verification Failed

1.3.14.2.1.2 Backup Package Key Parameter Verification Function

A verification function has been added during backup package import. The backup package must be compared with the target robot for key parameters, as detailed in the table below. Inaccurate settings of these parameters may pose certain security risks. Only when they are completely identical can the backup package be imported normally.

Table of 5 Key Parameters for Comparison:

No.	Key Parameter	Specific Meaning
1	ROBOT_TYPE	Robot Model
2	INSTALL_POS	Installation Method
3	INSTALL_YANGLE	Base Tilt Angle
4	INSTALL_ZANGLE	Base Rotation Angle
5	NEW_TEACH_ENABLE	Dynamic Configuration

If they are inconsistent, an error will be prompted. At this point, it is necessary to check whether the key parameters in the robot to be imported are consistent with those in the backup package.

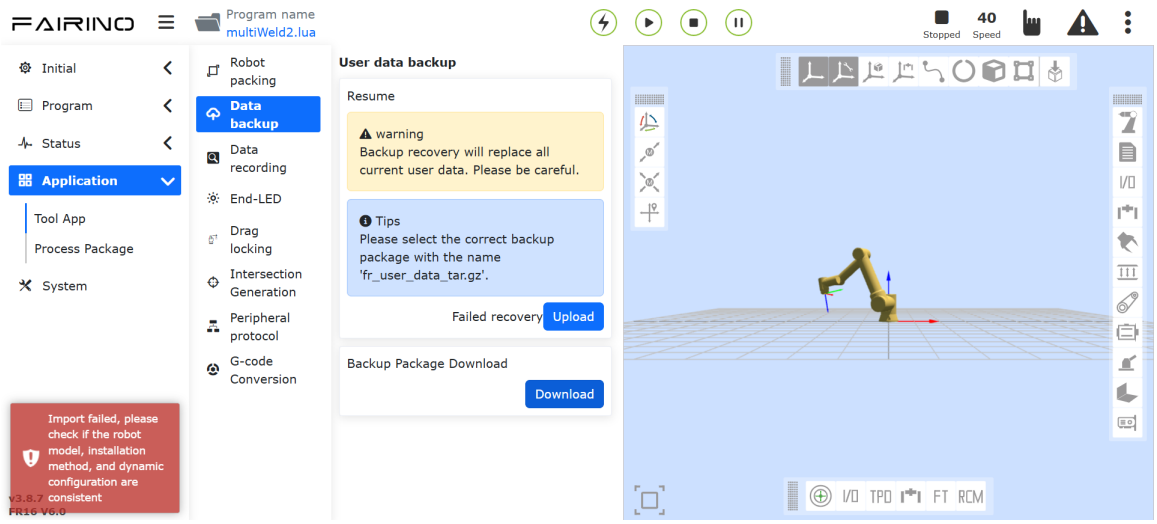


Figure 14.2-3 When Key Parameters Are Inconsistent, the Interface Will Prompt an Error

1.3.14.2.1.3 Failure/Power Failure Exception Rollback Function

If an exception occurs during the import of the backup package, or if there is an abnormal power failure during data recovery, causing the data recovery to fail to complete normally, to ensure the normal operation of the equipment, the system will automatically revert to the state before the operation after power is restored.

After power is restored, the interface will prompt, “Previous data recovery was not completed and has been automatically reverted. Please restart the control box and try again.”

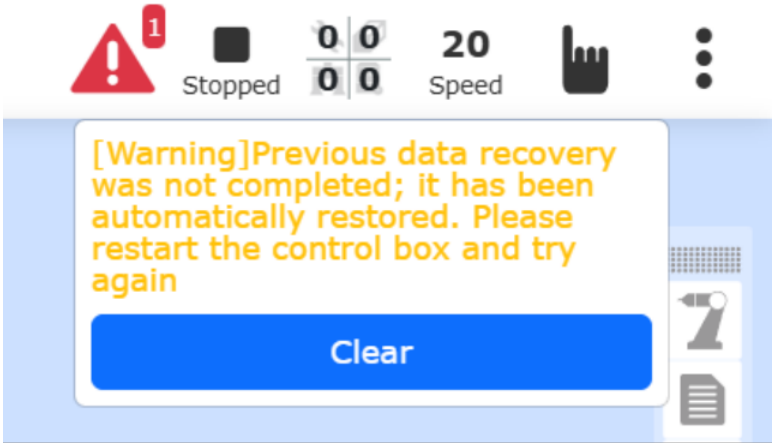


Figure 14.2-4 Power Failure Restart Restore Warning Prompt

1.3.14.3 10s Data Recording

Under the “Applications” -> “Tool App” menu bar, click “10s Data Recording” to enter the 10s data recording function interface.

First, select the recording type, which is divided into default parameter recording and self-selected parameter recording. Default parameter recording uses system-automated settings for the recorded data. For self-selected parameter recording, users can choose the parameters to record, with a maximum of 15 parameters. After selecting the parameter list, choose the parameters to record and click the “Move Right” button to add the parameters to the parameter list. Click “Start Recording” for the robot to begin recording data, click “Stop Recording” for the robot to stop recording data, and click “Download Data” to download the last 10 seconds of data.

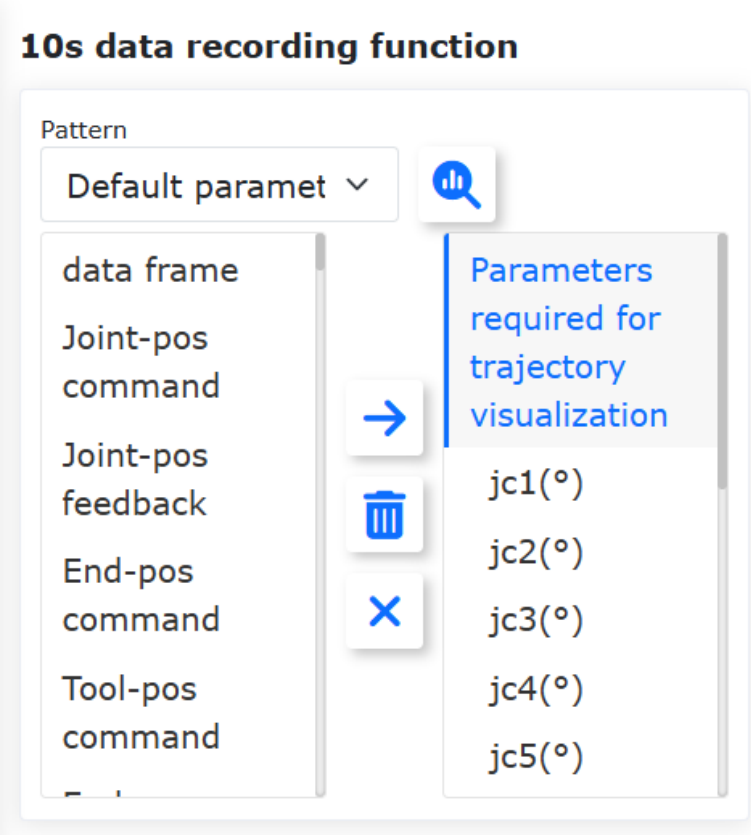


Figure 14.3-1 10s Data Recording

1.3.14.4 End-Effector LED

Under the “Applications” -> “Tool App” menu bar, click “End-Effector LED” to enter the end-effector LED color configuration function interface.

The LED color can be configured to green, blue, or white-cyan. Users can configure the LED color for Auto mode, Manual mode, and Drag mode according to their needs. Different modes cannot be configured with the same color.

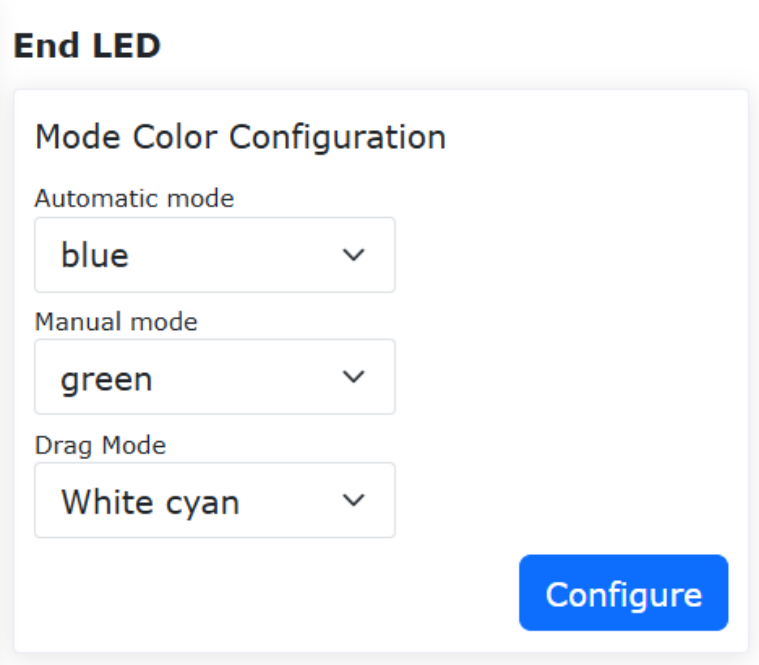


Figure 14.4-1 End-Effector LED Configuration

1.3.14.5 Drag Locking

Under the “Applications” -> “Tool App” menu bar, click “Drag Locking” to enter the drag teaching lock configuration function interface.

A degree-of-freedom locking function has been added for drag teaching. When the drag teaching function switch is set to enable, the degree-of-freedom parameters take effect when the user drags the robot. For example, when the parameters are set to X:10, Y:0, Z:10, RX:10, RY:10, RZ:10, dragging the robot in drag mode restricts movement to only the Y direction. If you need to keep the robot’s posture unchanged during dragging and only move in the X, Y, Z directions, set X, Y, Z to 0 and RX, RY, RZ to 10.

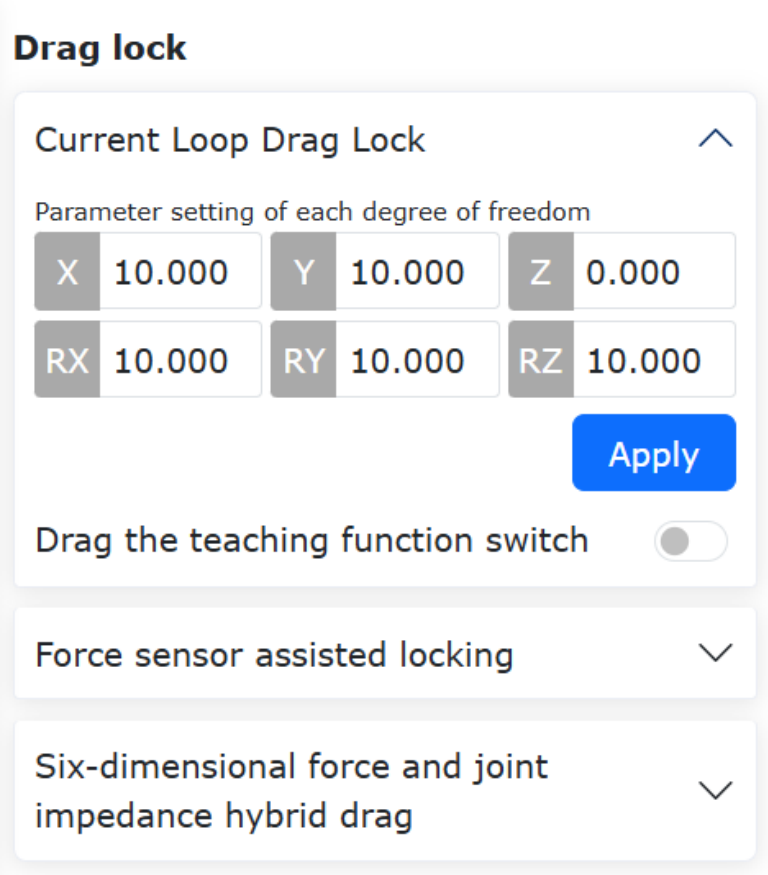


Figure 14.5-1 Drag Teaching Lock Configuration

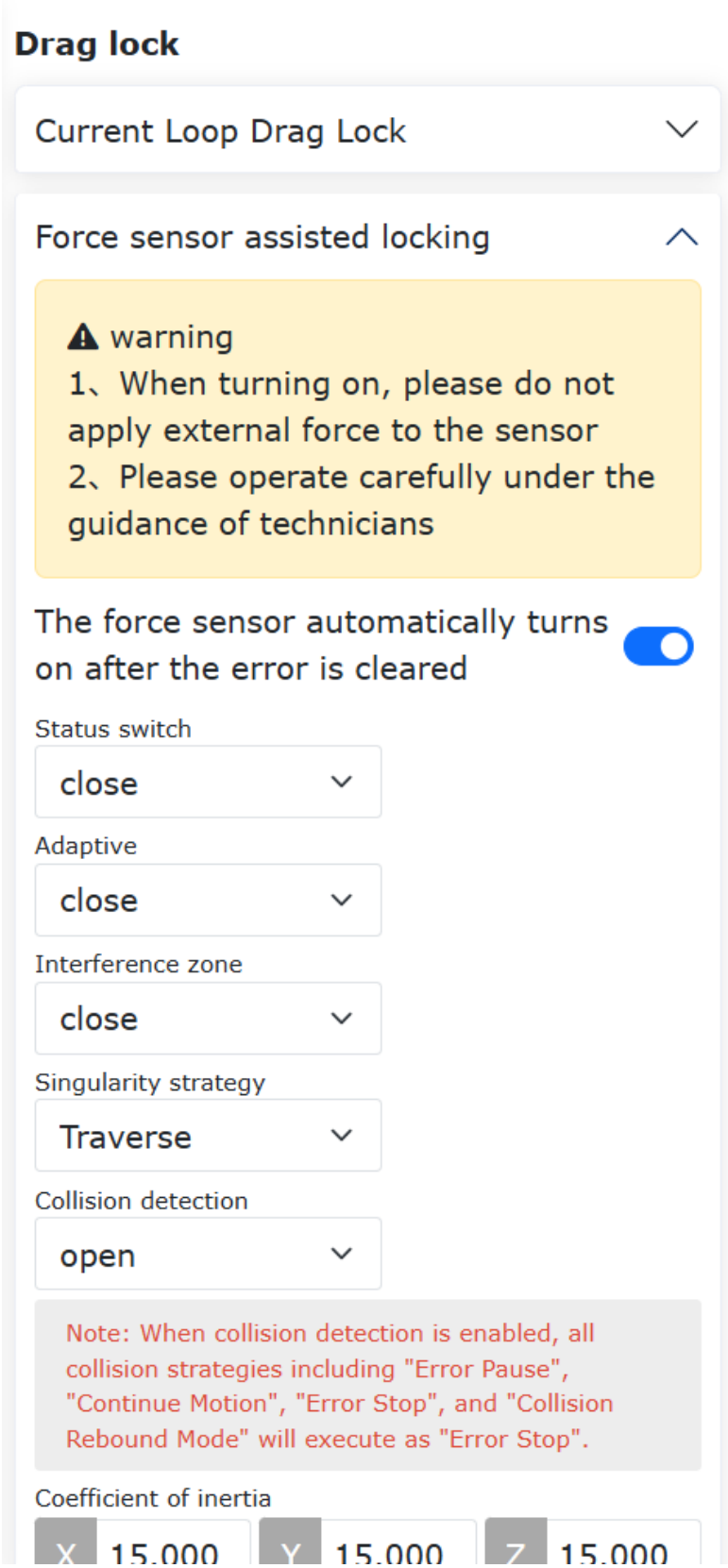
1.3.14.5.1 Normal Collision Protection Triggering Under Force Sensor Assisted Dragging

1.3.14.5.1.1 Overview

Currently, FR robots cannot trigger collision protection under force sensor assisted dragging. This enhancement adds the capability to trigger collision protection during force sensor assisted dragging, increasing robot safety and reducing operational risks.

1.3.14.5.1.2 Collision Protection

Step1: Click "Applications" -> "Tool App" -> "Drag Locking" to enter the Force Sensor Assisted Locking configuration interface. Set the "Status Switch" and "Collision Detection" to ON, as shown below.


Drag lock
Current Loop Drag Lock
Force sensor assisted locking
warning
1、When turning on, please do not apply external force to the sensor
2、Please operate carefully under the guidance of technicians
The force sensor automatically turns on after the error is cleared
Status switch
close
Adaptive
close
Interference zone
close
Singularity strategy
Traverse
Collision detection
open
Note: When collision detection is enabled, all collision strategies including "Error Pause", "Continue Motion", "Error Stop", and "Collision Rebound Mode" will execute as "Error Stop".
Coefficient of inertia
X 15.000
Y 15.000
Z 15.000

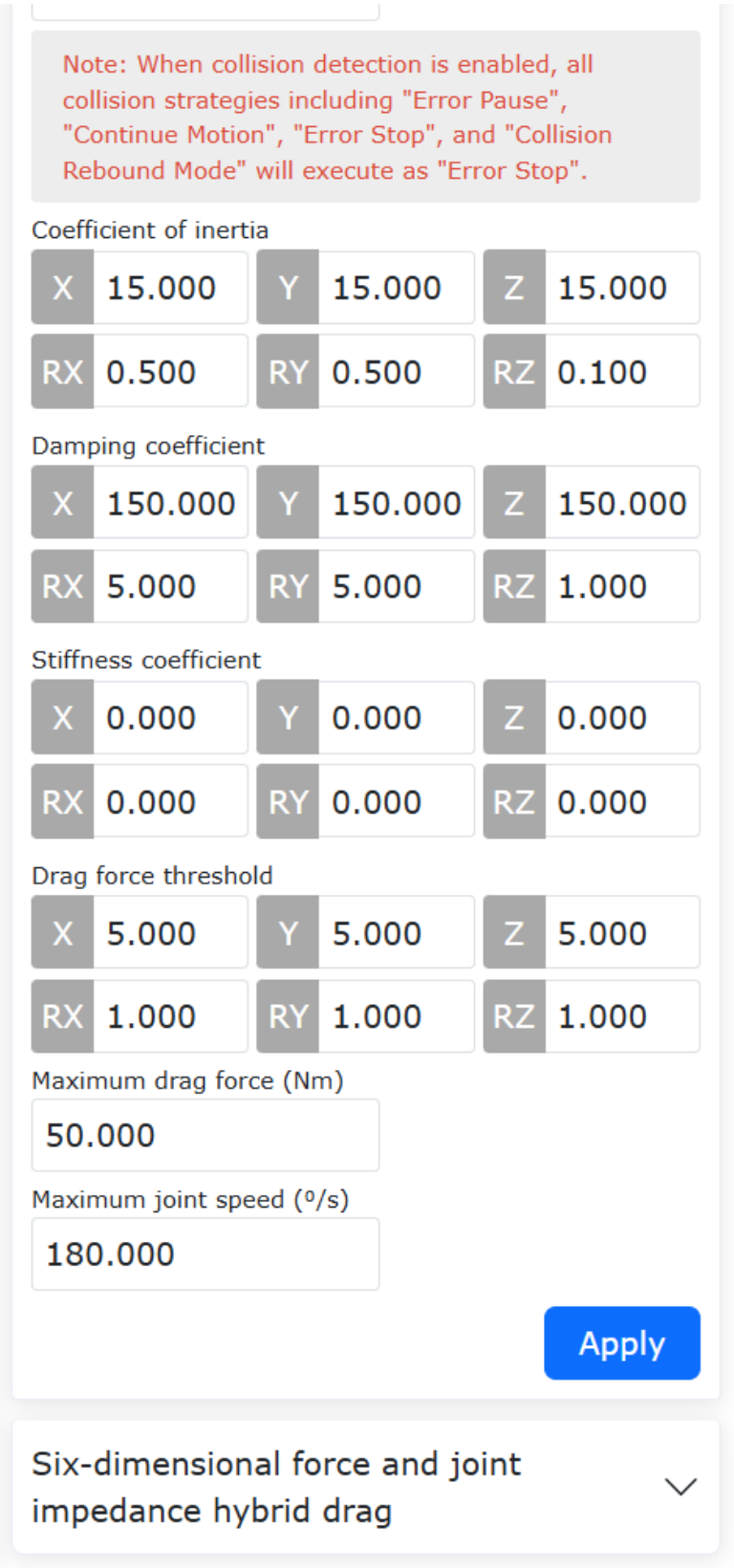


Figure 14.5-2 Configuring Force Sensor Assisted Locking

Step2: Drag the robot. During robot movement, apply external force to the joints to trigger collision protection. The web interface will report the error "Force Sensor Assisted Dragging Collision Fault", and you can quickly resume/disable force sensor assisted dragging via the web interface, as shown. Click "Resume" to clear the error and resume force sensor assisted dragging; click "Disable" to clear the error and keep force sensor assisted dragging disabled.

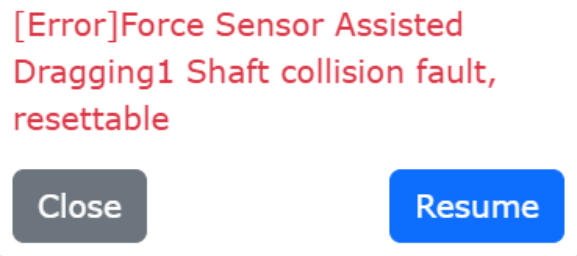


Figure 14.5-3 Collision Triggered Under Force Sensor Assisted Dragging

Note: Under force sensor assisted dragging, the robot itself is in a stopped state. During dragging, there is a difference between the joint torque command and feedback. It is recommended to set the collision level to level 7 or higher. Setting the collision level too low may cause false collision errors during dragging.

1.3.14.5.1.3 Parameters Calibration of Joint Torque Sensor on the Whole Machine

1.3.14.5.1.4 Overview

Joint torque sensor sensitivity refers to the sensor's responsiveness to torque changes, describing the proportional relationship between the sensor's output voltage and the actual joint torque being measured. Linearity measures how well a regression model fits the observed data. Hysteresis error is the maximum difference between measurements during forward stroke (from small to large) and reverse stroke (from large to small) under the same test conditions for the joint torque sensor's raw data. Repeatability is the ratio of the current test result to the previous test result, used to determine the repeatability accuracy of the joint torque sensor.

The parameter calibration method involves running the robot through a predetermined trajectory, and calculating the joint torque sensor's sensitivity, linearity, hysteresis error, and repeatability accuracy by acquiring joint gravity torque and joint torque sensor raw data under different postures.

1.3.14.5.1.5 Parameter Calibration

Step1: Set the tool coordinate system to "Tool0". Click "Auxiliary Applications" -> "Tool Applications" -> "Drag Lock". In the joint torque sensor whole-machine drag module, click "Function Enable".

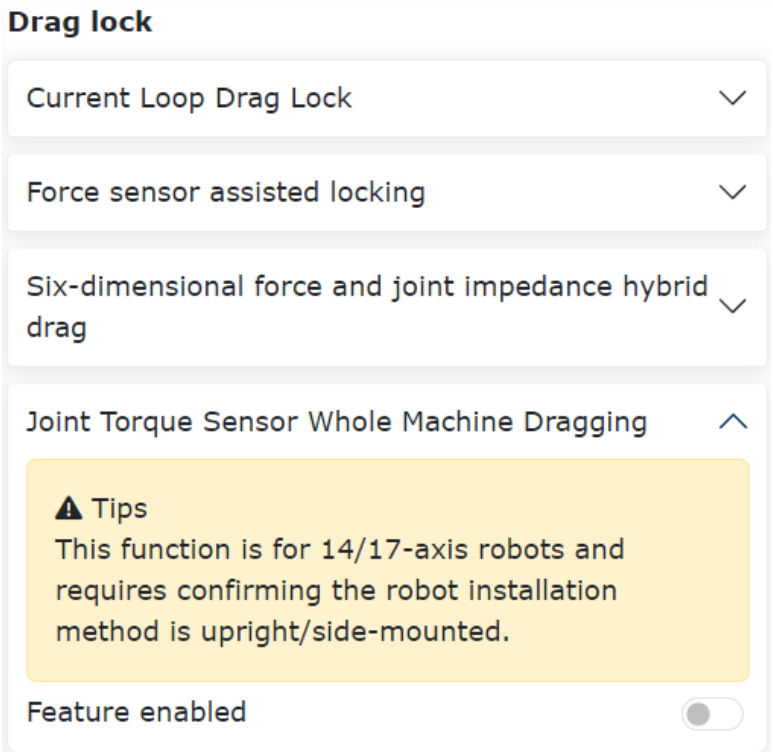


Figure 14.5-4 Function Enable

Step2: After clicking "Function Enable", proceed with sensitivity calibration. Click "Generate Program" to deploy the internal controller Lua script. Switch the robot to automatic mode and set the run speed to "10". Click "Run" and wait for the robot to move.

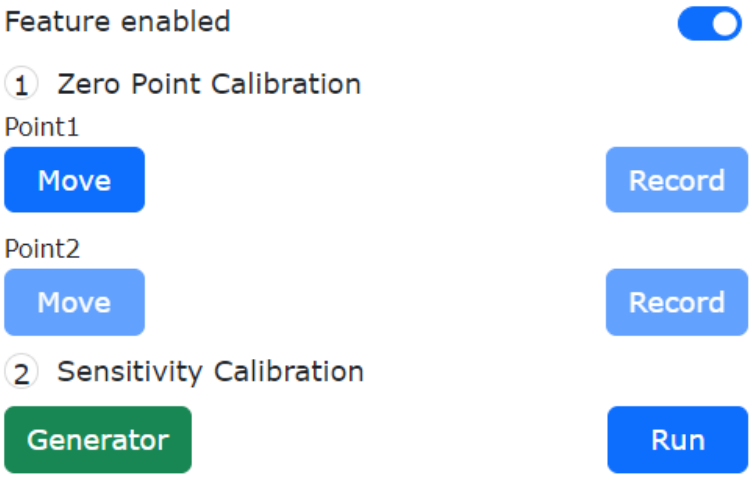


Figure 14.5-5 Sensitivity Calibration

Note: If the joint torque sensor sensitivity calibration is already completed, you can proceed directly to drag function

parameter settings.

Step3: After the robot completes running the predetermined trajectory, the sensitivity, linearity, hysteresis error, and repeatability calibration results are automatically displayed on the web interface. Click “Set” to apply.

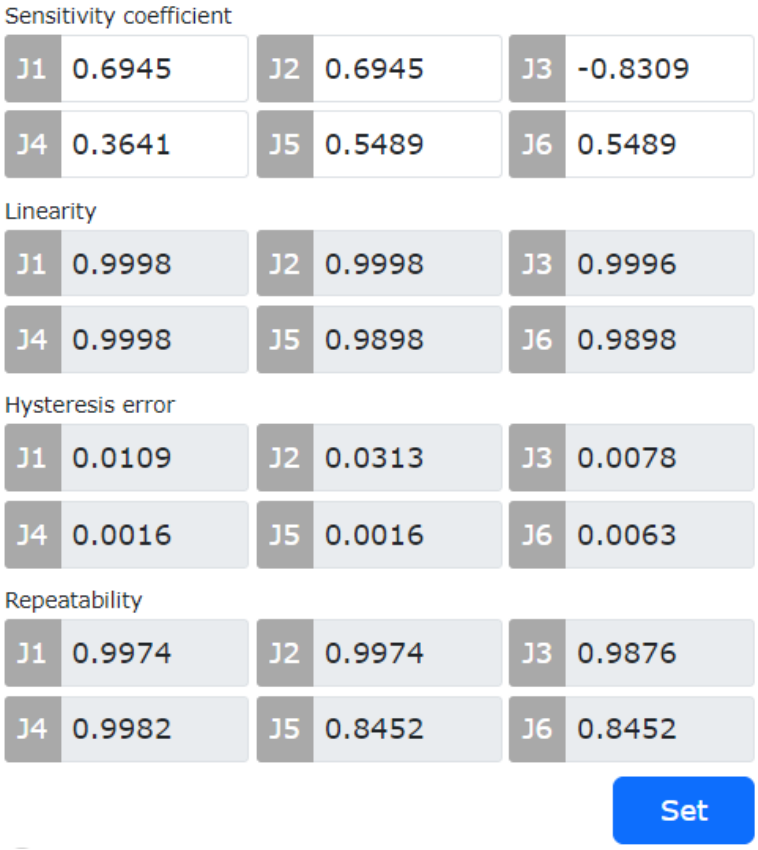


Figure 14.5-6 Parameter Calibration Results

1.3.14.5.1.6 External Force Estimation and Torque Compensation Based on Momentum Observer

1.3.14.5.1.7 Overview

After enabling the torque compensation function, the robot reduces the dragging torque during current loop dragging, improving the dragging experience.

1.3.14.5.1.8 Operation Process

Step1: Set the dynamics configuration to “Dynamics 2.0”. Click “Auxiliary Applications” -> “Tool Applications” -> “Drag Lock”. In the dual encoder torque compensation module, click the function switch to enable it.

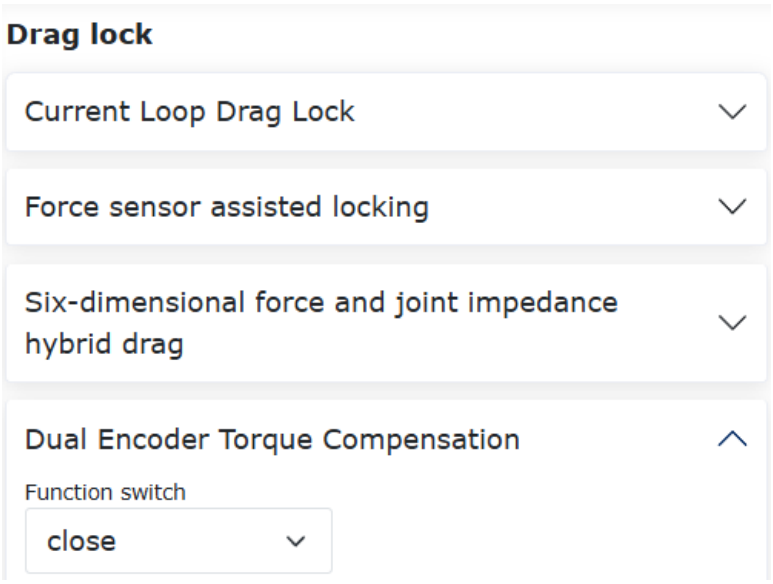


Chart 14.5-7 Enable Function

Step2: Set the "Function Switch" to "ON", and set the drag gain for each axis to 0.5. Click "Set" to apply, as shown in the figure.

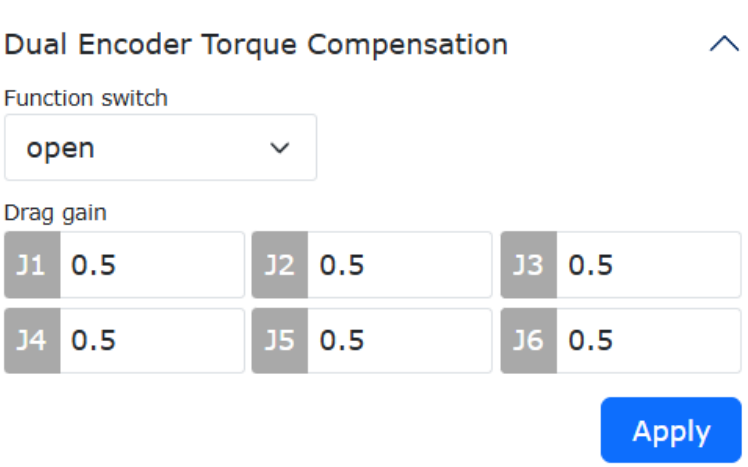


Chart 14.5-8 Gain Setting

Note: Drag gain setting range: 0-1. The larger the gain, the greater the compensation torque, and the easier the dragging under the current loop.

1.3.14.5.1.9 Assisted Drag Optimization Function Based on Joint Torque Sensor

1.3.14.5.1.10 Overview

This user manual describes the usage of the assisted drag optimization function based on the joint torque sensor. It involves three drag modes and, compared to traditional drag teaching methods, can improve drag compliance and reduce the drag force required by each joint.

1.3.14.5.1.11 Assisted Drag Optimization Function Based on Joint Torque Sensor

1.3.14.5.1.12 Zero Point Calibration and Sensitivity Calibration

Step1: If the zero point calibration and sensitivity calibration have already been completed (the indicator light before calibration is green), there is no need to perform them again. Recalibrate the zero point only when there is a floating sensation during dragging. The calibration process is described below.

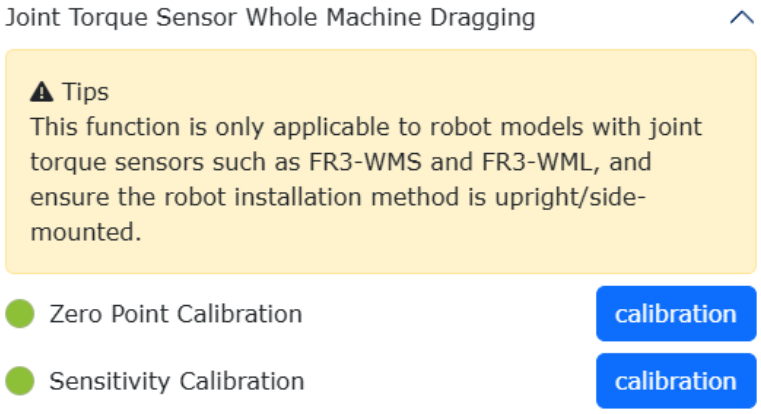


Figure 14.5-9 Indicator Light Status After Zero Point and Sensitivity Calibration of Joint Torque Sensor

Step2: Zero point calibration. Click “Auxiliary Applications” → “Tool Applications” → “Drag Lock” to enter the “Joint Torque Sensor Whole Body Drag” module. Click the “Calibrate” button for zero point calibration to calibrate the zero point data of the joint torque sensor. When the calibration is complete, a “” will appear, and the zero point calibration result will be updated.

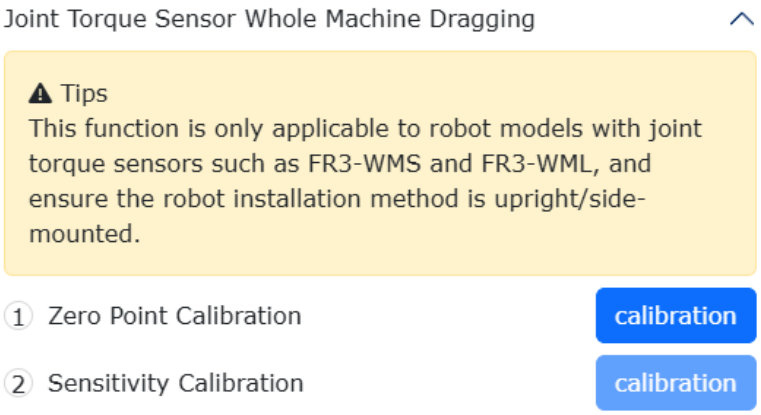


Figure 14.5-10 Zero Point Calibration of Joint Torque Sensor

Step3: Sensitivity calibration (Note: It is recommended to use only the robot body without any load during calibration). Switch the robot motion mode to "Automatic Mode" and set the operation speed to "10%". Click the "Calibrate" button for sensitivity calibration and wait for the robot to complete its motion. After the robot completes the predetermined trajectory, the sensitivity coefficient, linearity, hysteresis error, and repeatability calibration results are automatically displayed on the web interface.

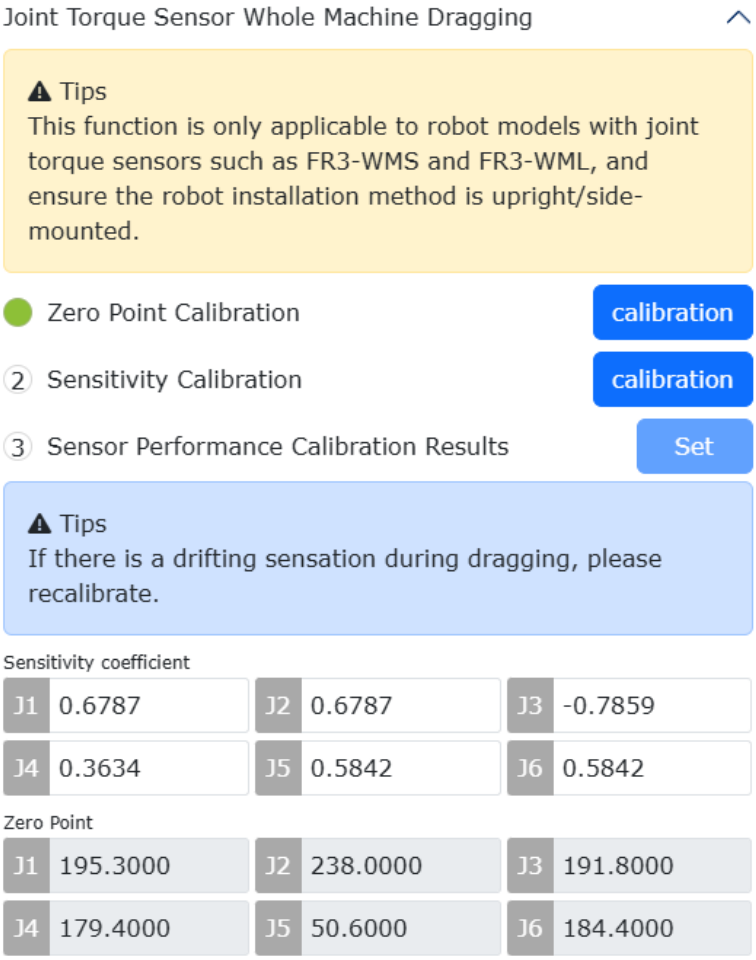


Figure 14.5-11 Sensitivity Calibration of Joint Torque Sensor

Step4: Set the assisted drag function. There are three drag modes, which can be set after completing "Zero Point Calibration" and "Sensitivity Calibration". If not set, the default is "Mode Three", meaning that drag teaching can be performed directly in drag mode after calibration.

1.3.14.5.1.13 Assisted Drag Function - Mode One

Step1: Select the drag mode as "Mode One". When the robot motion mode is "Manual Mode", set the sliding window size, gain coefficient, and joint speed, then click "Apply". At this point, holding down the end "Drag Button" or in drag mode enables drag teaching.



Figure 14.5-12 Mode One: Parameter Setting

Note:

(1) The recommended setting for the sliding window size is 30, with a maximum value of 100;

(2) The gain coefficient affects the feel during dragging. A larger coefficient makes dragging more sensitive but can easily cause instability. The recommended setting for J1-J6 is 0.5;

(3) The recommended joint speed is 6°/s, which can alleviate overshoot during point alignment.

1.3.14.5.1.14 Assisted Drag Function - Mode Two

Step1: Select the drag mode as "Mode Two". When the robot motion mode is "Manual Mode", set the mass coefficient, damping coefficient, stiffness coefficient, and force control threshold, then click "Apply". At this point, drag teaching can be performed in position mode.

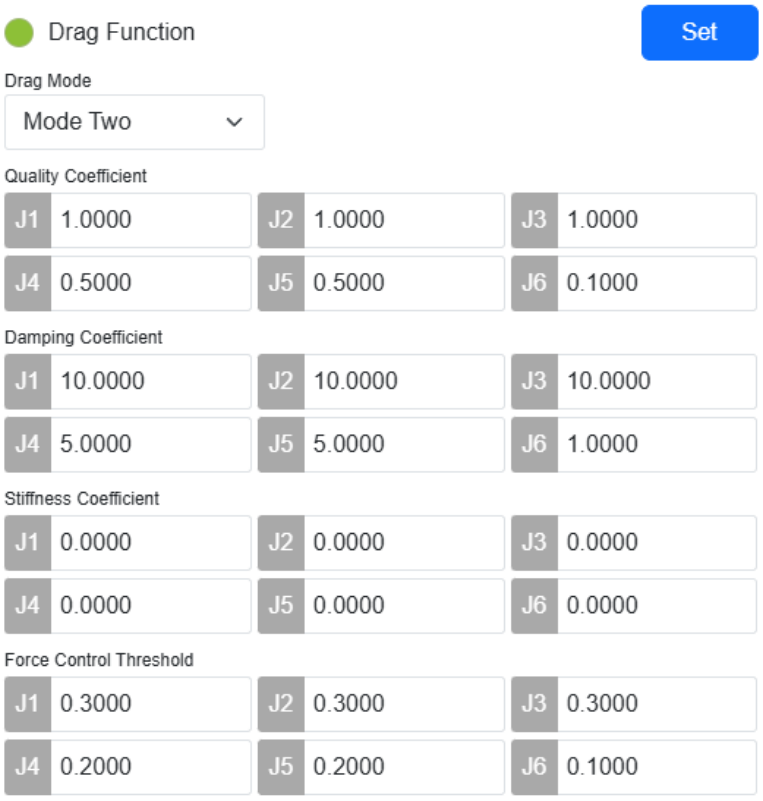


Figure 14.5-13 Mode Two: Parameter Setting

Note:

(1) The mass coefficient affects the joint inertial force during dragging. The recommended settings are: J1-J3: 1.0, J4-J5: 0.5, J6: 0.1;

(2) The damping coefficient affects the feel during dragging. Higher damping results in a heavier feel. The recommended settings are: J1-J3: 10.0, J4-J5: 5.0, J6: 1.0;

(3) The stiffness coefficient should be set to 0 for all;

(4) The force control threshold is the activation torque during dragging. The recommended settings are: J1-J3: 0.3, J4-J5: 0.2, J6: 0.1.

1.3.14.5.1.15 Assisted Drag Function - Mode Three

Step1: Select the drag mode as "Mode Three". When the robot motion mode is "Manual Mode", set the gain coefficient for each joint, then click "Apply". At this point, holding down the end "Drag Button" or in drag mode enables drag teaching.



Figure 14.5-14 Mode Three: Parameter Setting

Note:

(1) The gain coefficient affects the joint drag force during low-speed dragging. When the coefficient ranges from 0.1 to 1.0, the resistance during low-speed dragging increases as the coefficient increases. For precise point alignment tasks, it is recommended to set the J1-J6 gain coefficient to 1.0. When considering overall ease of dragging and compliance, it is recommended to set the J1-J6 gain coefficient to 0.3.

1.3.14.6 Intersection Point Generation (Laser Point Capture Motion)

During welding, the laser point capture motion can be configured with posture, enabling the robot to achieve the expected posture when reaching the position point. This easily handles special scenarios such as fillet welds and groove welds.

1.3.14.6.1 Laser Point Capture Motion Function Operation Process

Step1: Before using the laser sensor, first apply the "Welding Torch" tool coordinate system to the current tool coordinate system. Open the Teach Page, click "Initial Setup", "Basic", "Tool Coordinates" in sequence, select "Welding Torch" as the coordinate system name and apply it. The tool coordinate system in the system status bar will display as Tool1.

Figure 14.6-1 Apply Welding Torch Coordinate System

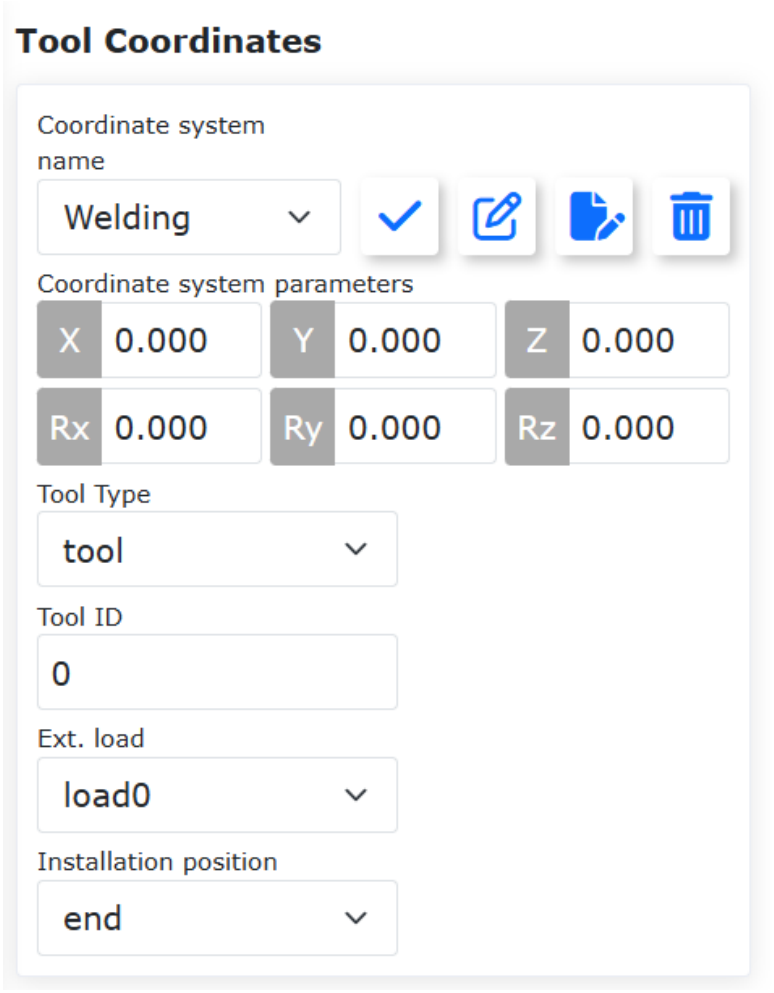
Step2: Write the laser point capture motion Lua program. Click "Teach Program" -> "Program Programming" -> "New Button" in sequence, and create a new user program "testPointRecord.lua".

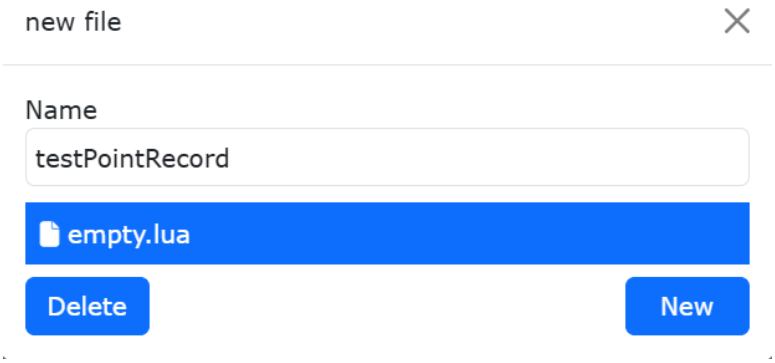
Figure 14.6-2 New Laser Point Capture Motion Program

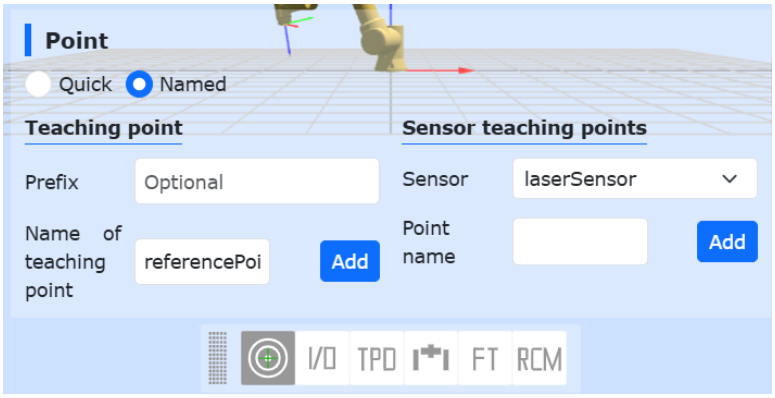
Step3: Configure the reference posture teach point (Optional). In manual mode, drag the robot to the desired welding posture. On the Teach Page, click "Teach Point Record", "Name Point" in sequence, and save the posture teach point as "referencePoint".

Figure 14.6-3 Save Posture Reference Teach Point

Step4: Generate the laser point capture motion program. Click "Teach Program" -> "Program Programming" -> "Welding Instructions", "Laser Tracking" in sequence, scroll down to the Sensor Point Capture Motion section, select


Tool Coordinates
Coordinate system name
Welding
Coordinate system parameters
X 0.000
Y 0.000
Z 0.000
Rx 0.000
Ry 0.000
Rz 0.000
Tool Type
tool
Tool ID
0
Ext. load
load0
Installation position
end


new file
Name
testPointRecord
empty.lua
Delete
New



the required "Motion Mode", "Debug Speed", and posture reference point to generate the corresponding laser point capture LUA program.

If no posture reference point is selected, the posture at the point of capture is maintained by default during motion. If a posture reference point is selected, the robot moves to the laser-captured point using the reference posture.

Figure 14.6-4 Select Posture Reference Teach Point

Execute the laser point capture motion. Drag the robot so that the laser sensor beam points to the desired weld seam point. The laser sensor will acquire the weld seam position and capture the point. After executing the laser point capture motion, the welding torch will move to the point scanned by the laser sensor using the reference posture.

Figure 14.6-5 Laser Acquires Weld Seam Position

Figure 14.6-6 Welding Torch Points to Weld Seam Position with Reference Posture

1.3.14.6.2 Finding Intersection Coordinates Using Three-Point and Four-Point Seam Finding

When it is inconvenient to directly teach the position of a fillet weld, the collaborative robot can calculate the intersection point of the two plate planes by manually teaching or finding the positions on both sides of the fillet weld, thereby generating the location of the fillet weld.

For right-angle fillet welds, the three-point seam finding method can be used to calculate the intersection coordinates; for non-right-angle fillet welds, the four-point seam finding method is used.

Both command and Lua script methods are provided to obtain the intersection coordinates for seam finding motion, and a reference posture can be configured, allowing the robot carrying the welding torch to move to the intersection point with the posture of the reference teach point.

1.3.14.6.2.1 Command-based Intersection Calculation

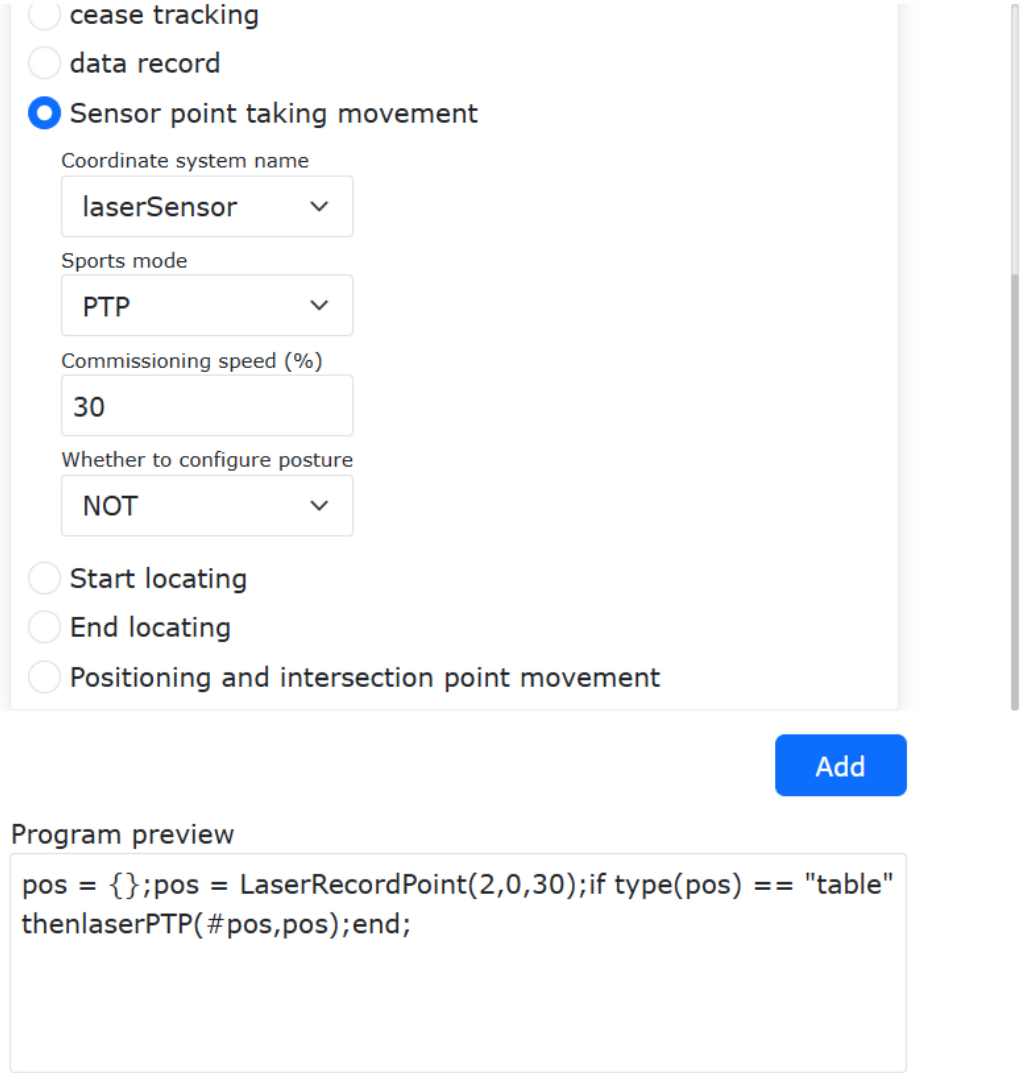
1.3.14.6.2.2 Three-Point Intersection Calculation

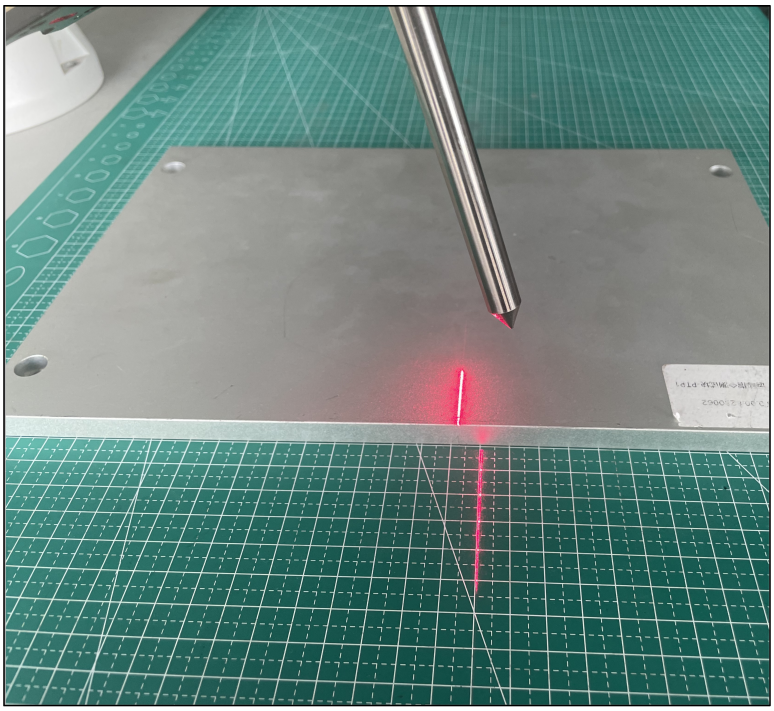
Step1: Collect three plane contact points and save them as teach points; configure a reference teach point.

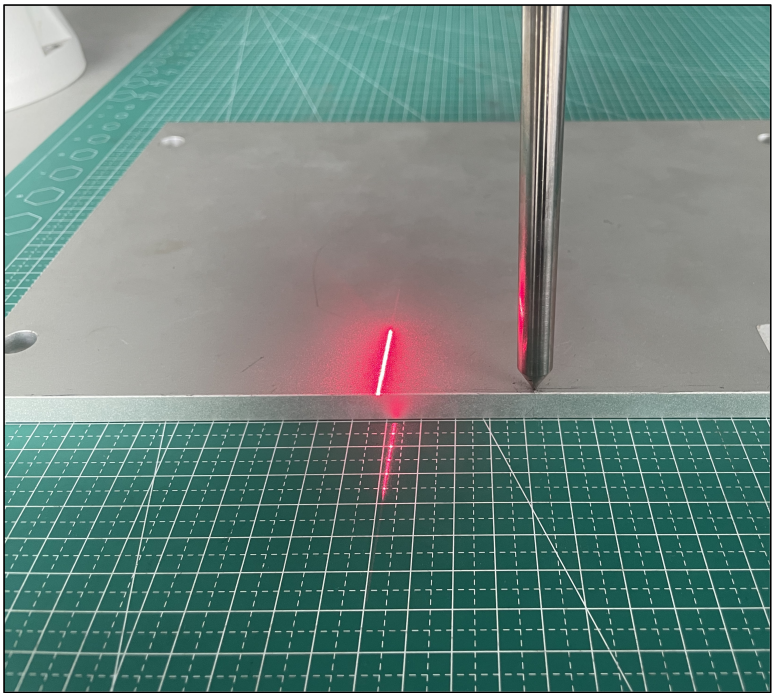
Figure 14.6-7 Select Three Seam Finding Points

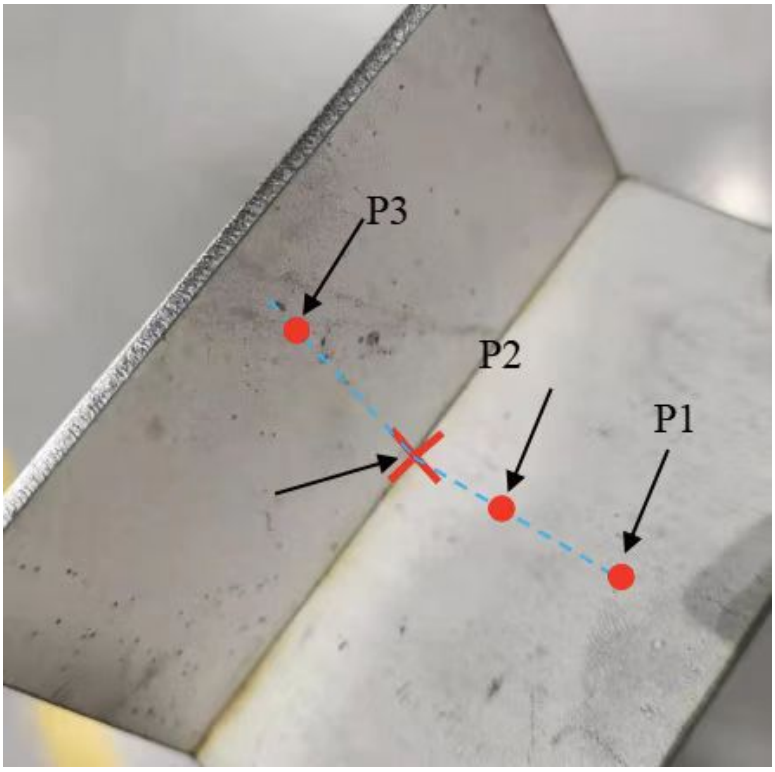
The collected contact points include three points, where two points lie on the same plane and the other point lies on the perpendicular plane.

Laser


cease tracking
data record
Sensor point taking movement
Coordinate system name
laserSensor
Sports mode
PTP
Commissioning speed (%)
30
Whether to configure posture
NOT
Start locating
End locating
Positioning and intersection point movement
Add
Program preview
pos = {};pos = LaserRecordPoint(2,0,30);if type(pos) == "table" thenlaserPTP(#pos,pos);end;







Note: If no posture reference point is selected, the generated intersection point's posture is consistent with point P3 by default. If a posture reference point is selected, it matches the posture of the reference teach point.

Step2: On the Teach Page, click "Applications" -> "Tool App" -> "Intersection Generation", and find the Three-Point and Four-Point Intersection Calculation function module.

Figure 14.6-8 Select Seam Finding Points for Intersection Calculation

Step3: Select "Three-Point Seam Finding" from the dropdown, sequentially select the three collected contact points, click Calculate, check if the displayed intersection point in the 3D model is correct, name the intersection point, and save it.

Figure 14.6-9 Calculate Intersection Coordinates and Save

Step4: Save the teach point, and teach motion can be performed.

Figure 14.6-10 Save Intersection Coordinates as a Teach Point

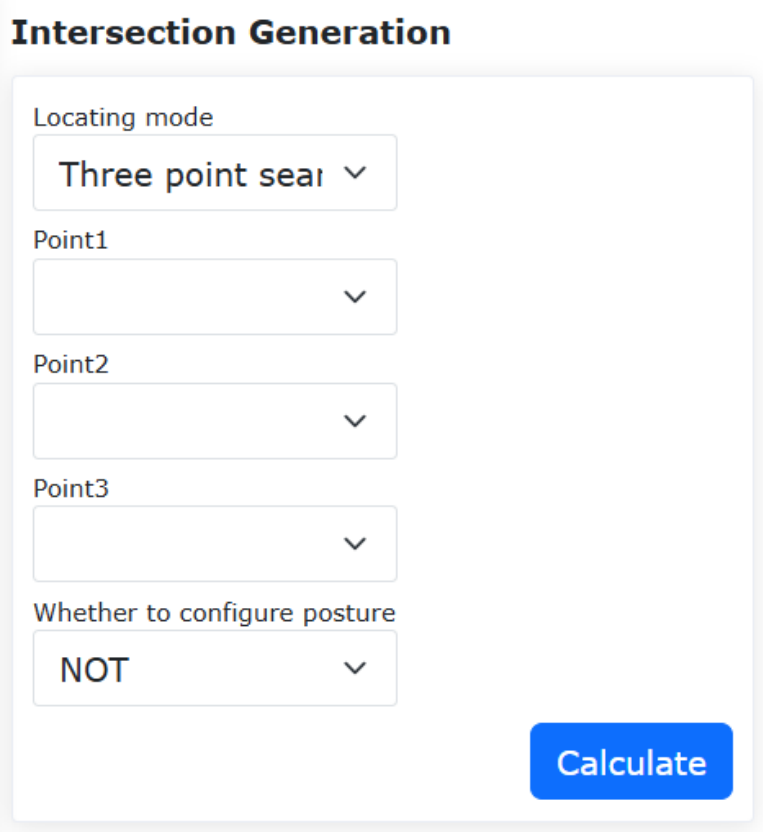
1.3.14.6.2.3 Four-Point Intersection Calculation

Step1: Collect four plane contact points and save them as teach points; configure a reference teach point.

Figure 14.6-11 Select Four Seam Finding Points

The collected contact points include four points, where the first two points lie on the same plane and the last two points lie on the perpendicular plane.

Note: If no posture reference point is selected, the generated intersection point's posture is consistent with point P4



by default. If a posture reference point is selected, it matches the posture of the reference teach point.

Step2: On the Teach Page, click “Initial Setup”, “Peripherals”, “Tracking”, “Sensors” in sequence, and find the Three-Point and Four-Point Intersection Calculation function module.

Figure 14.6-12 Select Seam Finding Points and Reference Point for Intersection Calculation

Step3: Select “Four-Point Seam Finding” from the dropdown, sequentially select the four collected contact points, click Calculate, check if the displayed intersection point in the 3D model is correct, name the intersection point, and save it.

Figure 14.6-13 Calculate Intersection Coordinates and Save

Step4: Save the teach point, and teach motion can be performed.

Figure 14.6-14 Save Intersection Coordinates as a Teach Point

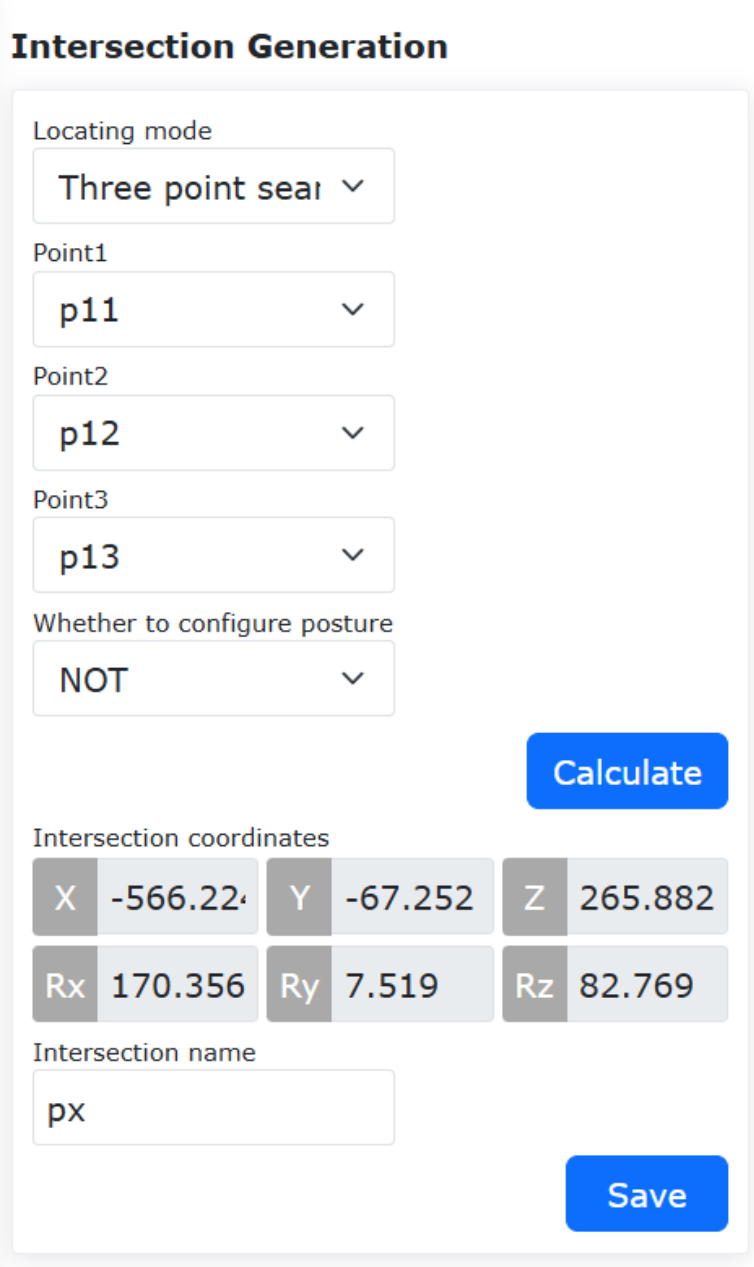
1.3.14.6.2.4 Lua Script for Intersection Seam Finding Motion

1.3.14.6.2.5 Three-Point Intersection Calculation

Step1: Collect three plane contact points and save them as teach points; configure a reference teach point.

Figure 14.6-15 Select Three Seam Finding Points

The collected contact points include three points, where two points lie on the same plane and the other point lies on the perpendicular plane.


Intersection Generation
Locating mode
Three point sear
Point1
p11
Point2
p12
Point3
p13
Whether to configure posture
NOT
Calculate
Intersection coordinates
X -566.22
Y -67.252
Z 265.882
Rx 170.356
Ry 7.519
Rz 82.769
Intersection name
px
Save

Teaching Management

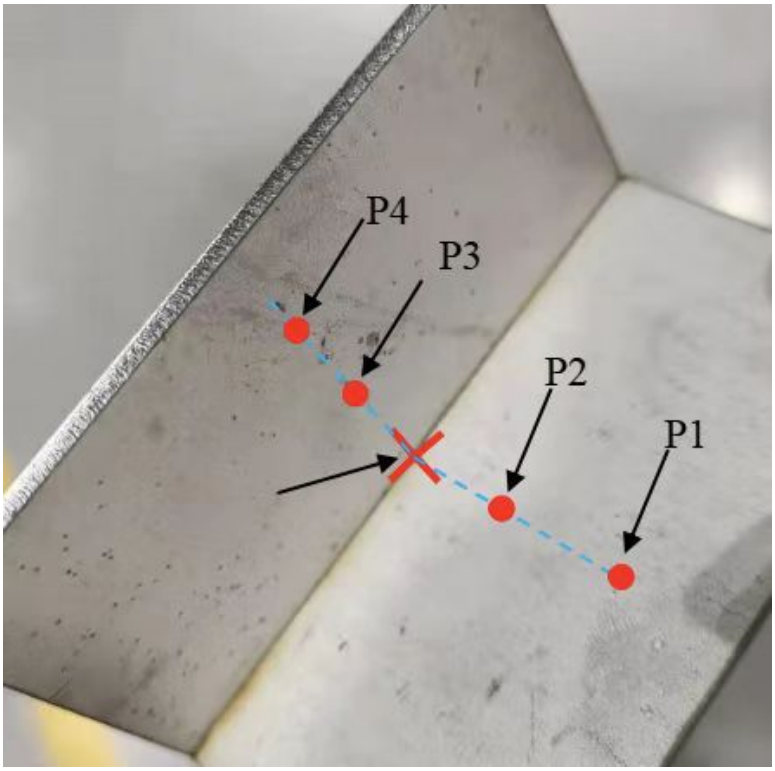
Point mode

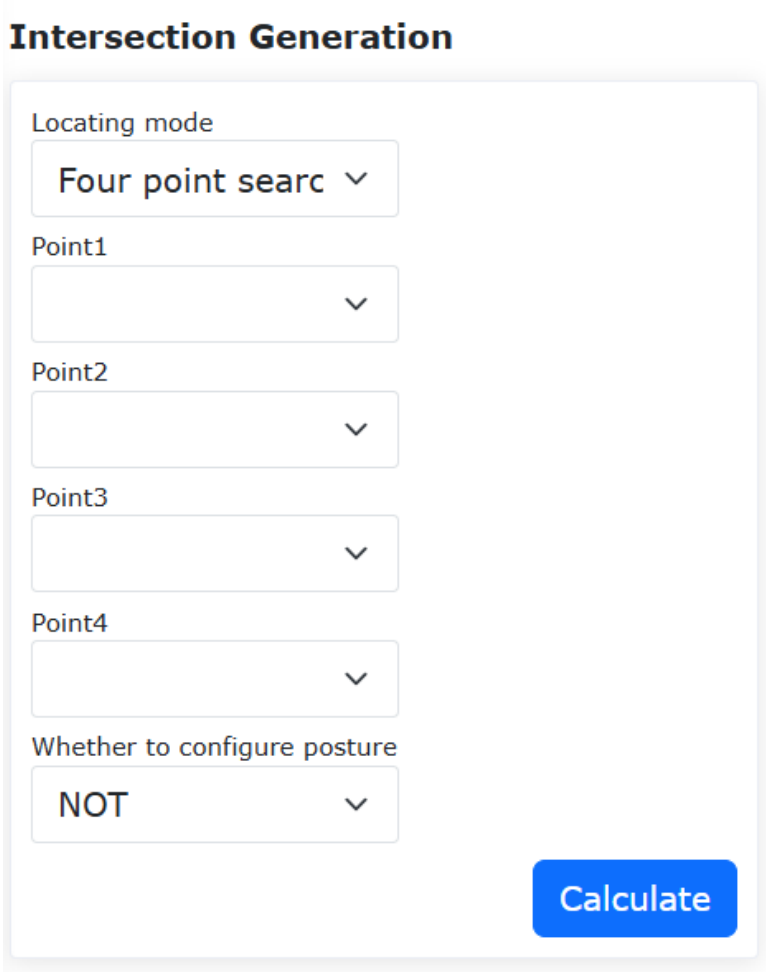
System mode Point table mode

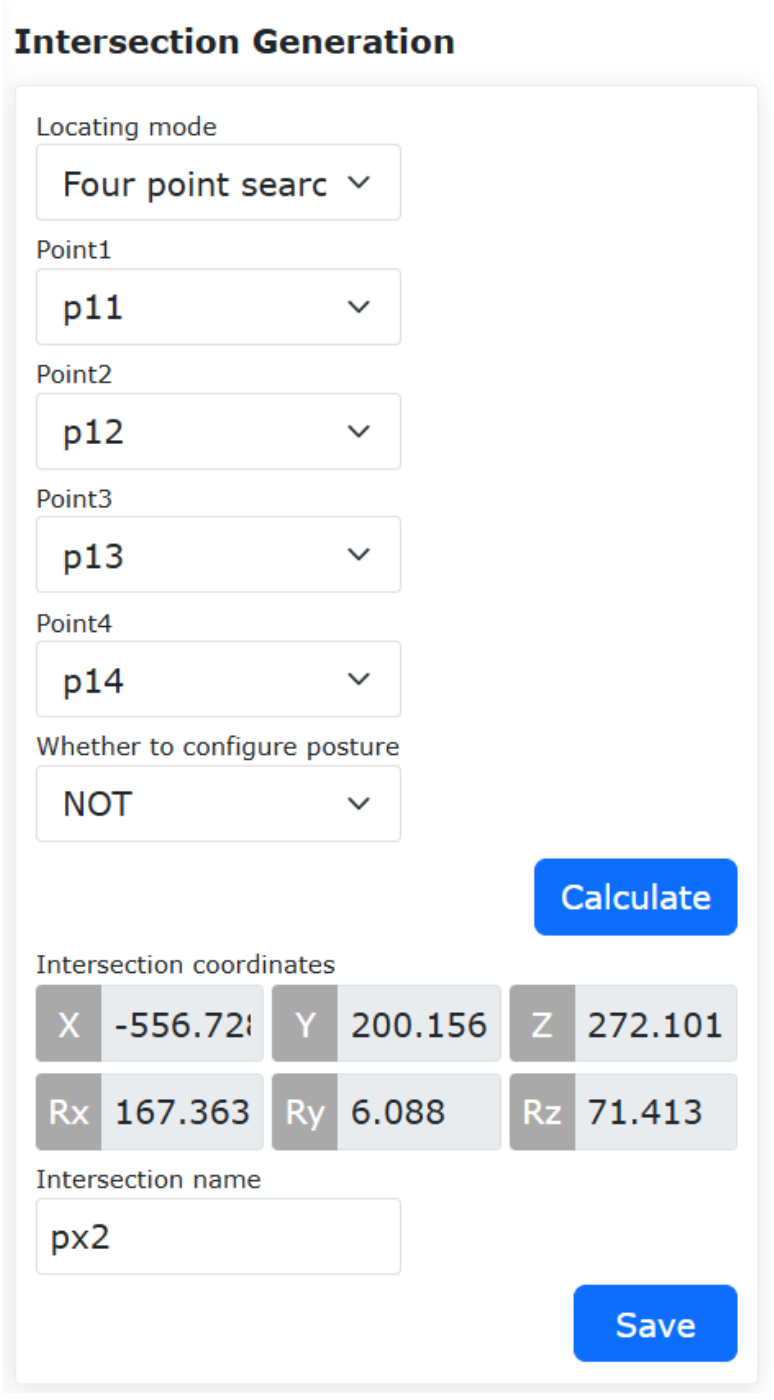
Chosen0row Please enter y

	Name	X	Y	Z	RX	RY	RZ	J1	J2	J3	J4	J5	J6	TOOL	WOBJ	V	Operate
	p14	-630.38	66.112	271.587	167.363	6.088	71.413	-21.402	-65.570	105.192	-116.645	-84.686	-4.094	toolcoord0	workpiece0	100	
	p2	-672.63	62.521	336.639	-175.87	8.819	85.921	-20.437	-67.045	100.841	-125.307	-80.387	-15.913	toolcoord0	workpiece0	100	
	palletizing_1lefttransitionpoint	-508.65	-424.71	63.186	179.895	-4.270	9136.394	26.660	-43.153	111.329	-159.522	-94.055	-19.778	toolcoord0	workpiece0	100	
	palletizing_1righttransitionpoint	-569.32	300.246	68.565	177.301	1.728	97137.013	-41.638	-45.090	113.598	-156.716	-92.657	-88.651	toolcoord0	workpiece0	100	
	px	-566.22	-67.252	65.882	170.356	7.519	82.769	-10.699	-71.114	116.389	-125.123	-83.187	-4.710	toolcoord0	workpiece0	100	
	px2	-556.72	200.156	72.101	167.363	6.088	71.413	-35.704	-69.259	110.889	-117.762	-88.022	-18.033	toolcoord0	workpiece0	100	
	recordEndPt	-444.70	555.391	97.781	-160.71	5.219	42.473	-65.819	-67.210	96.726	-136.470	-79.330	-15.810	Welding	workpiece0	100	
	recordStartPt	-444.70	555.390	97.782	-160.71	5.219	42.472	-65.819	-67.210	96.726	-136.471	-79.330	-15.810	Welding	workpiece0	100	
	referencePoint	-580.16	-174.94	72.883	166.403	2.395	94.933	1.328	-67.997	108.718	-116.996	-88.524	-4.068	toolcoord0	workpiece0	100	

Total 105 data 10 Strip/Page < 1 ••• 6 7 8 9 10 11 > Jump to 10 /11Page




Intersection Generation
Locating mode
Four point searc
Point1
Point2
Point3
Point4
Whether to configure posture
NOT
Calculate


Intersection Generation
Locating mode
Four point searc
Point1
p11
Point2
p12
Point3
p13
Point4
p14
Whether to configure posture
NOT
Calculate
Intersection coordinates
X -556.72
Y 200.156
Z 272.101
Rx 167.363
Ry 6.088
Rz 71.413
Intersection name
px2
Save

Teaching Management

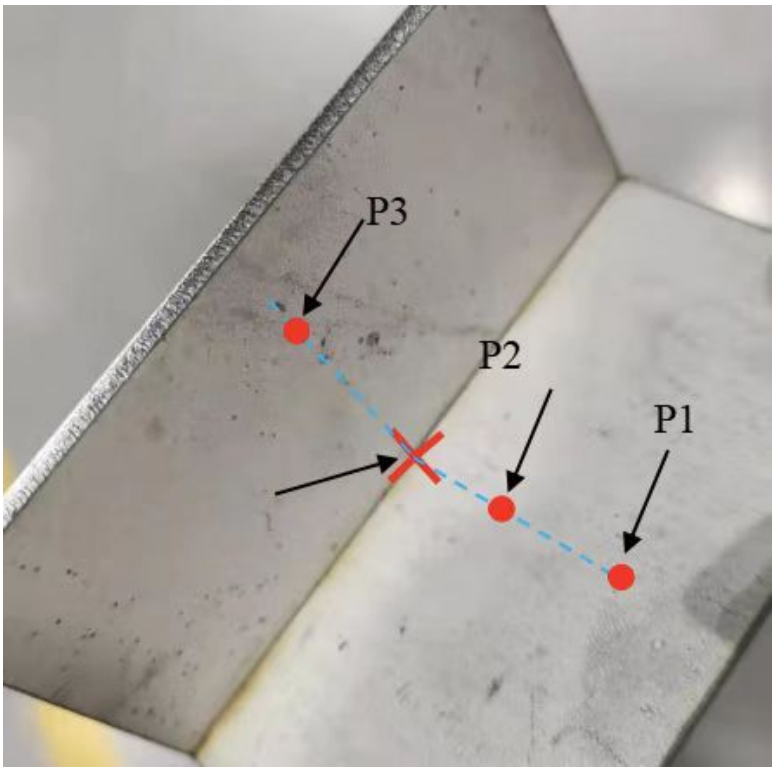
Point mode

System mode Point table mode

Chosen0row Please enter y

	Name	X	Y	Z	RX	RY	RZ	J1	J2	J3	J4	J5	J6	TOOL	WOBJ	V	Operate
	palletizing_1righttransitionpoint	-569.32	300.24	668.565	177.301	1.728	97137.013	-41.638	-45.090	113.598	-156.716	-92.657	-88.651	toolcoord 0	workpiece0	100	
	px	-566.22	-67.252	265.882	170.356	7.519	82.769	-10.699	-71.114	116.389	-125.123	-83.187	-4.710	toolcoord 0	workpiece0	100	
	px2	-556.72	200.156	272.101	167.363	6.088	71.413	-35.704	-69.259	110.889	-117.762	-88.022	-18.033	toolcoord 0	workpiece0	100	
	recordEndPt	-444.70	555.391	397.781	-160.71	5.219	42.473	-65.819	-67.210	96.726	-136.470	-79.330	-15.810	Welding	workpiece0	100	
	recordStartPt	-444.70	555.390	397.782	-160.71	5.219	42.472	-65.819	-67.210	96.726	-136.471	-79.330	-15.810	Welding	workpiece0	100	
	referencePoint	-580.16	-174.94	272.883	166.403	2.395	94.933	1.328	-67.997	108.718	-116.996	-88.524	-4.068	toolcoord 0	workpiece0	100	

Total 105 data 10 Strip/Page < 1 ••• 6 7 8 9 10 11 > Jump to 10 /11Page



Note: If no posture reference point is selected, the generated intersection point's posture is consistent with point P3 by default. If a posture reference point is selected, it matches the posture of the reference teach point.

Step2: Write the three-point intersection seam finding motion Lua program. Click "Teach Program" -> "Program Programming" -> "New Button" in sequence, and create a new user program "test3point.lua".



Figure 14.6-16 New Three-Point Intersection Seam Finding Motion Program

Step3: Generate the three-point intersection seam finding motion program. Click "Teach Program" -> "Program Programming" -> "Welding Instructions" -> "Laser Tracking" in sequence, scroll down to the Seam Finding Intersection Motion section, select the "Three-Point Seam Finding" method, use the dropdowns to sequentially select the collected contact points "Point 1", "Point 2", "Point 3", and the posture reference point, select the required "Motion Mode" and "Debug Speed", then click the "Add" and "Apply" buttons to generate the corresponding three-point intersection seam finding motion program.

Figure 14.6-17 Three-Point Intersection Seam Finding Motion

Step4: In Auto mode, click the Run button to automatically perform the three-point intersection calculation. The robot will drag the welding torch and move to the intersection position using the reference posture.

test3point.lua

```
1 pos = {}
2 pos = GetIntersectionThrough3Point(p11,...
3 if type(pos) == "table" then
4 laserPTP(#pos,pos)
5 end
```

1.3.14.6.2.6 Four-Point Intersection Calculation

Step1: Collect four plane contact points and save them as teach points; configure a reference teach point.

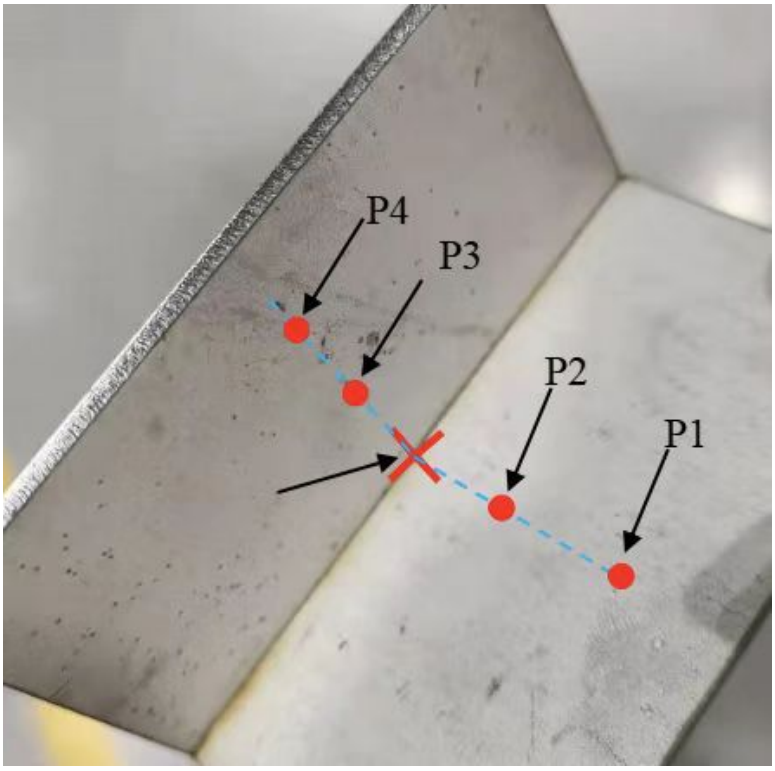


Figure 14.6-18 Select Four Seam Finding Points

The collected contact points include four points, where the first two points lie on one plane and the last two points lie on another plane.

Note: If no posture reference point is selected, the generated intersection point's posture is consistent with point P4 by default. If a posture reference point is selected, it matches the posture of the reference teach point.

Step2: Write the four-point intersection seam finding motion Lua program. Click “Teach Program” -> “Program Programming” -> “New Button” in sequence, and create a new user program “test4point.lua”.

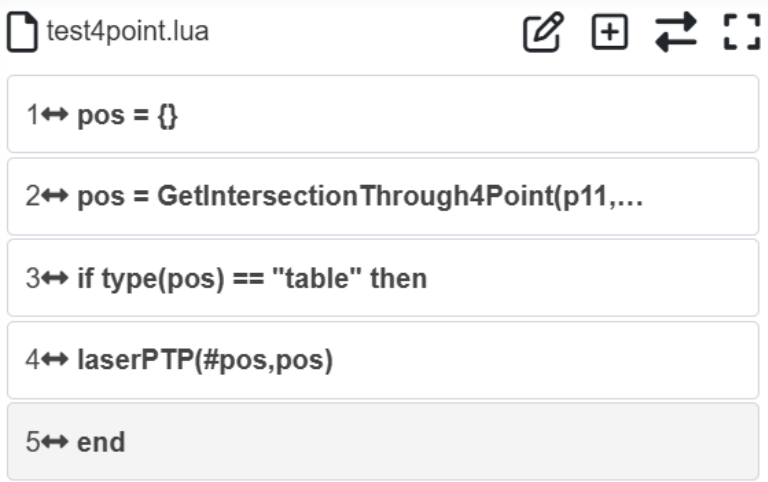


Figure 14.6-19 New Four-Point Intersection Seam Finding Motion Program

Step3: Generate the four-point intersection seam finding motion program. Click “Teach Program” -> “Program Programming” -> “Welding Instructions” -> “Laser Tracking” -> scroll down to the Seam Finding Intersection Motion section, select the “Four-Point Seam Finding” method, use the dropdowns to sequentially select the collected contact points “Point 1”, “Point 2”, “Point 3”, “Point 4”, and the posture reference point, select the required “Motion Mode” and “Debug Speed”, then click the “Add” and “Apply” buttons to generate the corresponding four-point intersection seam finding motion program.

Figure 14.6-20 Four-Point Intersection Seam Finding Motion

Step4: In Auto mode, click the Run button to automatically perform the four-point intersection calculation. The robot will drag the welding torch and move to the intersection position using the reference posture.



1.3.14.7 Peripheral Protocol

Under the “Applications” -> “Tool App” menu bar, click “Peripheral Protocol” to enter the peripheral protocol configuration function interface.

This page is for configuring peripheral protocols. Users can configure the protocol based on the currently used peripheral device.

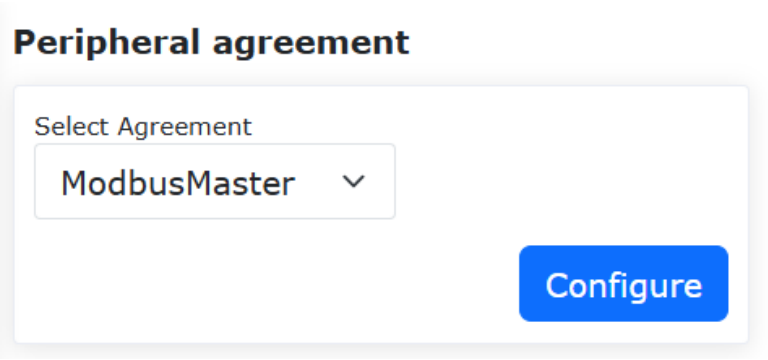


Figure 14.7-1 Peripheral Protocol Configuration

Lua interfaces for reading and writing registers based on Modbus-RTU communication have been added in program teaching. The input register address is 0x1000 with 50 registers, totaling 100 bytes of data content; the holding register address is 0x2000 with 50 registers, totaling 100 bytes of data content.

```
ModbusRegRead(fun_code, reg_add, reg_num): Read Registers;

fun_code: Function code, 0x03 - Holding Register, 0x04 - Input Register

reg_add: Register Address

reg_num: Number of Registers
```

```
ModbusRegWrite(fun_code, reg_add, reg_num, reg_value): Write Registers;

fun_code: Function code, 0x06 - Single Register, 0x10 - Multiple Registers
```

(continues on next page)

(continued from previous page)

```
reg_add: Register Address

reg_num: Number of Registers

reg_value: Byte Array
```

```
ModbusRegGetData(reg_num): Get Register Data;

reg_num: Number of Registers

Return value description:

reg_value: Array variable
```

Example program screenshot:

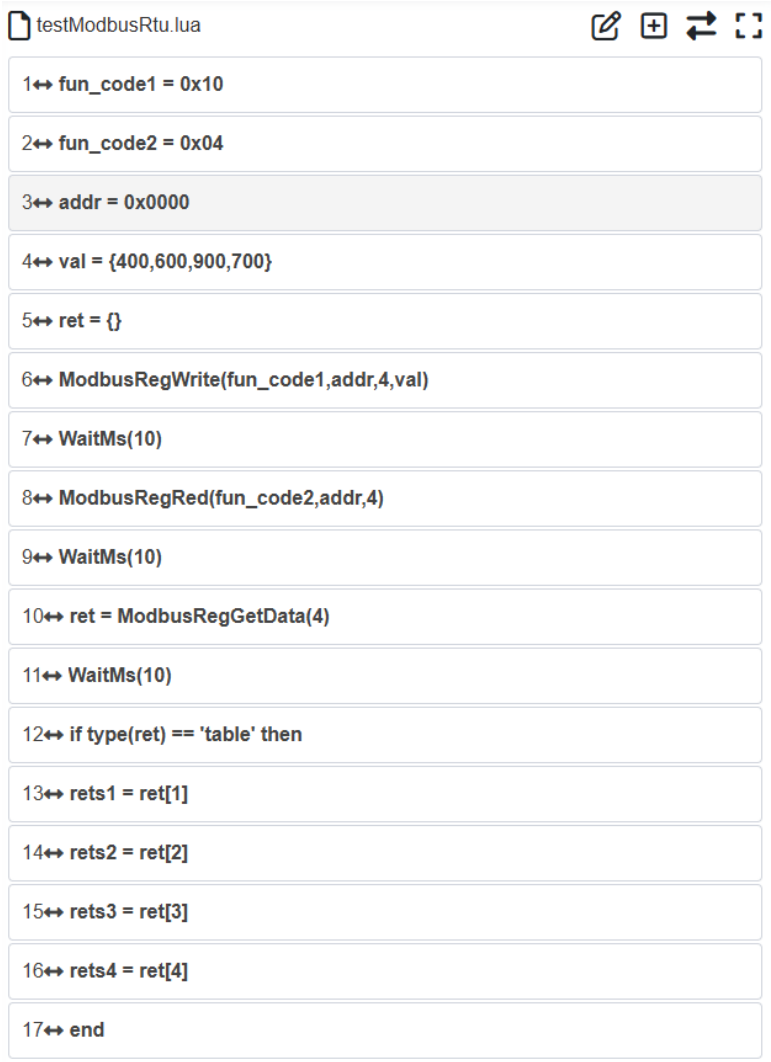


Figure 14.7-2 Modbus-RTU Communication Lua Program Example

1.3.14.8 G-code to Robot Trajectory Planning Function

1.3.14.8.1 Function Overview

The G-code to Robot Trajectory Planning function uses CAD software to convert paths (lines, arcs, full circles, splines) into G-code files with the ".gcode" extension. The G-code for spline paths consists of many small line segments. Importing the generated G-code file on the Web converts it and generates a LUA file.

G-code to Robot Trajectory Planning function description:

(1) The Web can only import G-code files with the ".gcode" extension. Upon successful conversion, a LUA file with the same name as the G-code file is generated. If a LUA file with the same name already exists before conversion, the conversion will fail.

(2) Currently, it can convert rapid move G0, linear interpolation G1, clockwise arc interpolation G2, and counter-clockwise arc interpolation G3 commands in G-code. G0 corresponds to the MoveJ instruction, G1 corresponds to MoveL, arcs (G2/G3) correspond to MoveC, and full circles (G2/G3) correspond to the Circle instruction.

(3) Currently, only G-code for arcs and circles in the XY plane can be converted.

(4) The spindle speed S set in G-code instructions corresponds to the speed in the MoveJ instruction. The unit of spindle speed is revolutions per minute (RPM), corresponding to millimeters per minute (mm/min) for move speed. The feedrate F corresponds to the speed in MoveL, MoveC, and Circle. The unit of feedrate is millimeters per minute (mm/min), consistent with the move speed unit. The speed settings in G-code must not exceed the robot's maximum moving speed.

(5) When executing the converted LUA file, the robot needs to have the speed percentage in the upper right corner of the Web interface changed to 100.

1.3.14.8.2 Operation Process

The calculation of the robot's posture along the working path is shown in the figure below.

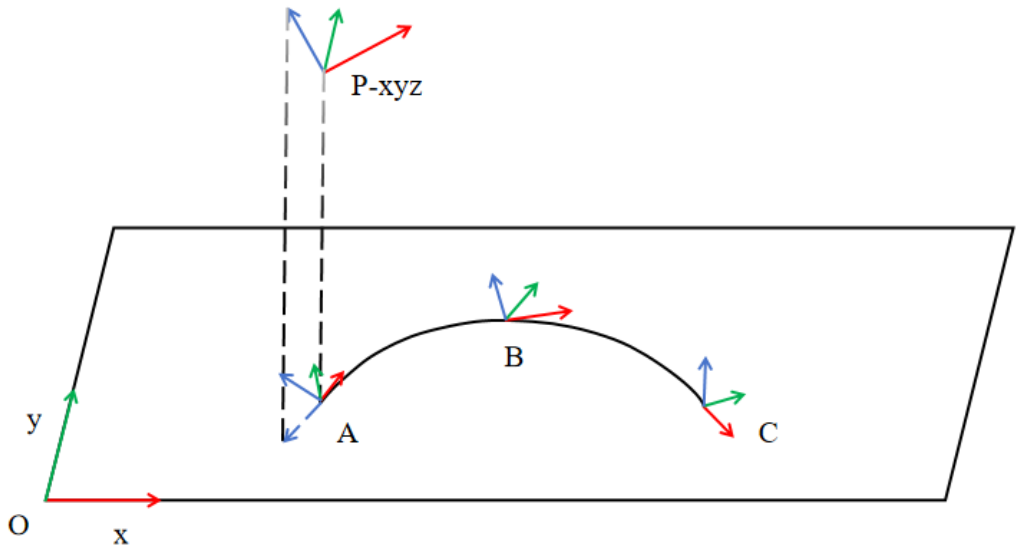


Figure 14.8-1 Robot Posture Calculation Schematic

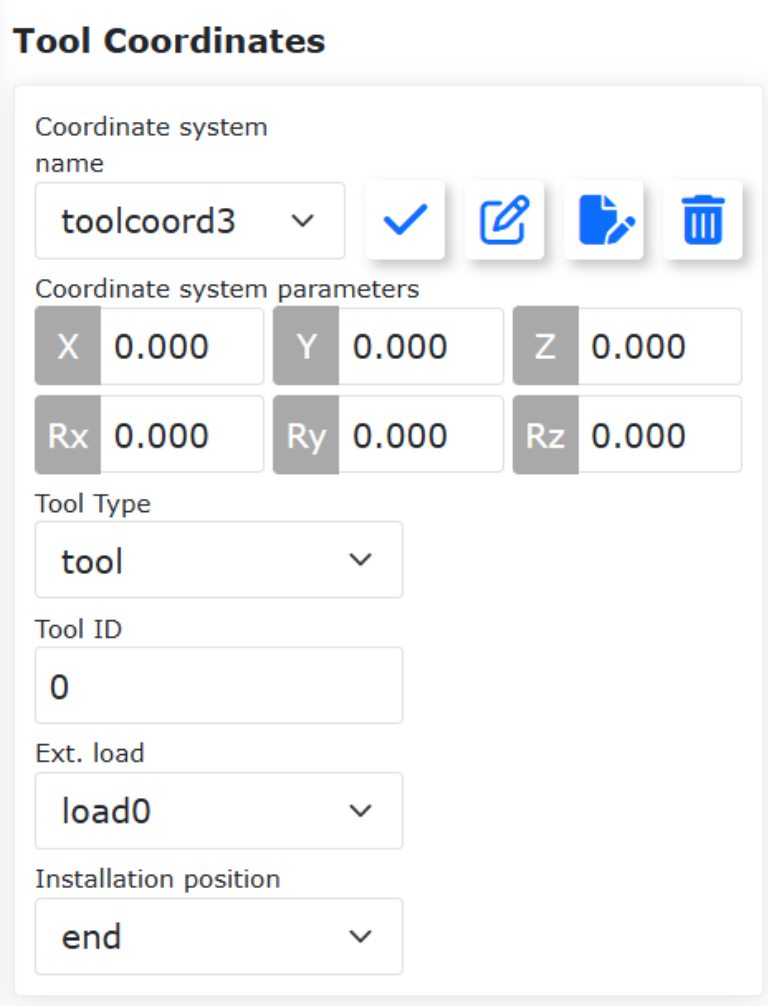
Where P-xyz is the posture of the recorded reference posture teach point, and O-xy is the coordinate system of the CAD drawing. The robot's posture at the starting point A is the reference posture. Based on the angle between the Z-axis

of the reference posture and the CAD drawing plane, and the angle between the projection of the Z-axis on the CAD drawing plane and the tangent at the path's starting point, the robot's posture at intermediate point B and endpoint C of the path is calculated.

The functional operation process is as follows:

Step 1: Use CAD software with CAM functionality to convert the machining path into a G-code file, and use a G-code simulator, such as NC Viewer, to verify the correctness of the tool path.

Step 2: Before converting G-code to robot trajectory, first calibrate the tool coordinate system and the workpiece coordinate system. Note that the workpiece coordinate system needs to coincide with the machine coordinate system in the CAD software.



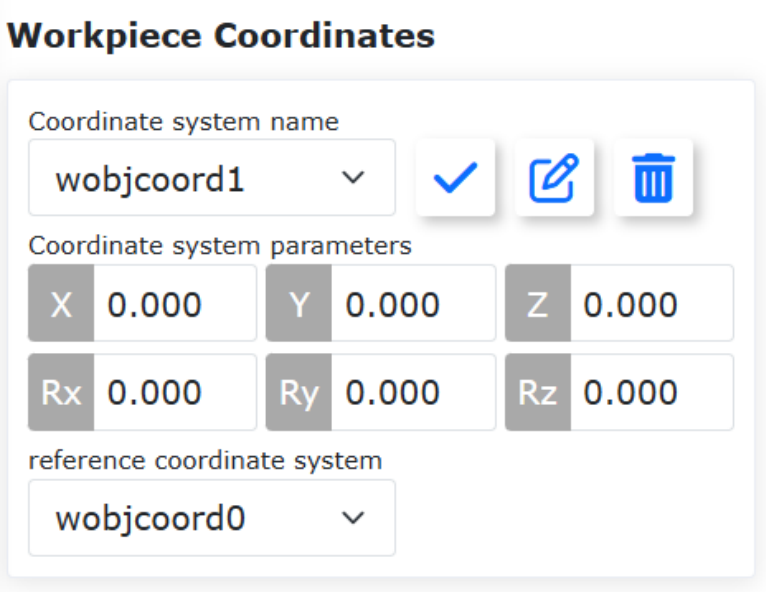


Figure 14.8-2 Tool and Workpiece Coordinate System Calibration Interface

Step 3: Under the calibrated tool coordinate system and workpiece coordinate system, record a reference posture teach point. The robot's posture along the working path will be calculated using the posture of this reference point.

Step 4: Click "Applications", "Tool App", "G-code Conversion" button in sequence to enter the G-code file to robot motion LUA file interface.

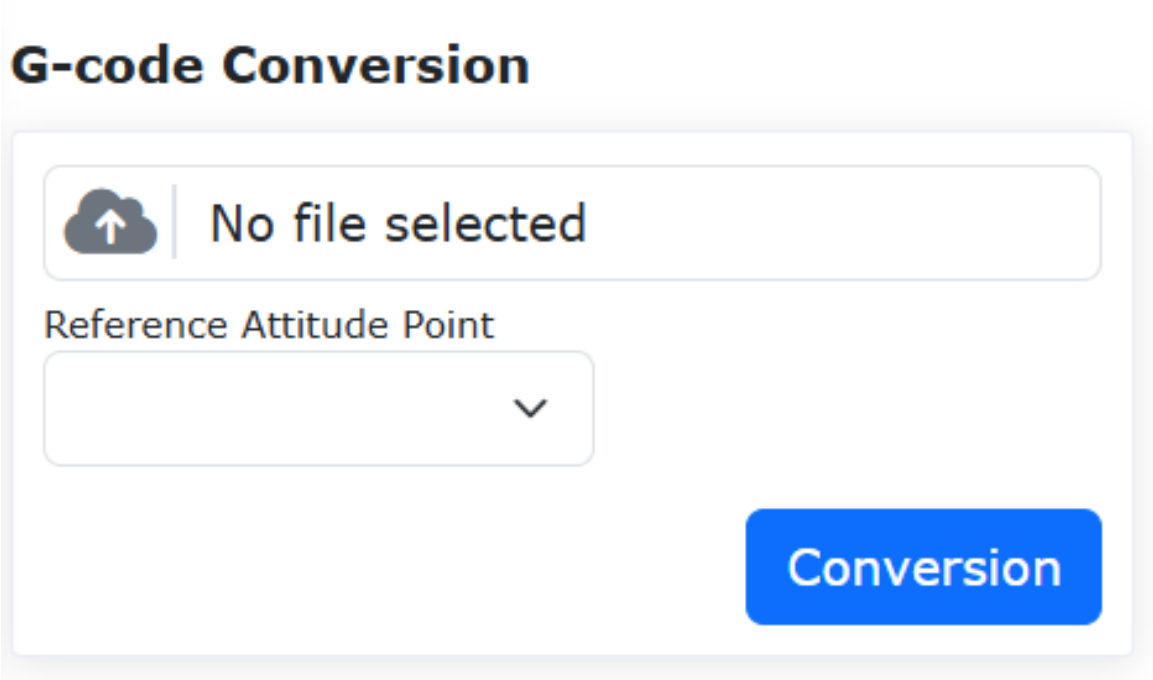


Figure 14.8-3 G-code Conversion Interface

Step 5: Click the "Select File" button to find the G-code file that needs conversion. Note that the file extension of the G-code file must be ".gcode". Select the reference posture point recorded in Step 2. Upon successful selection, the tool coordinate system and workpiece coordinate system of the currently selected teach point will be displayed below. Finally, click the "Convert" button. Upon successful conversion, a "G-code Conversion Successful" message

will appear. Additionally, if a LUA file with the same name as the G-code file already exists, clicking the "Convert" button will prompt that the filename already exists.

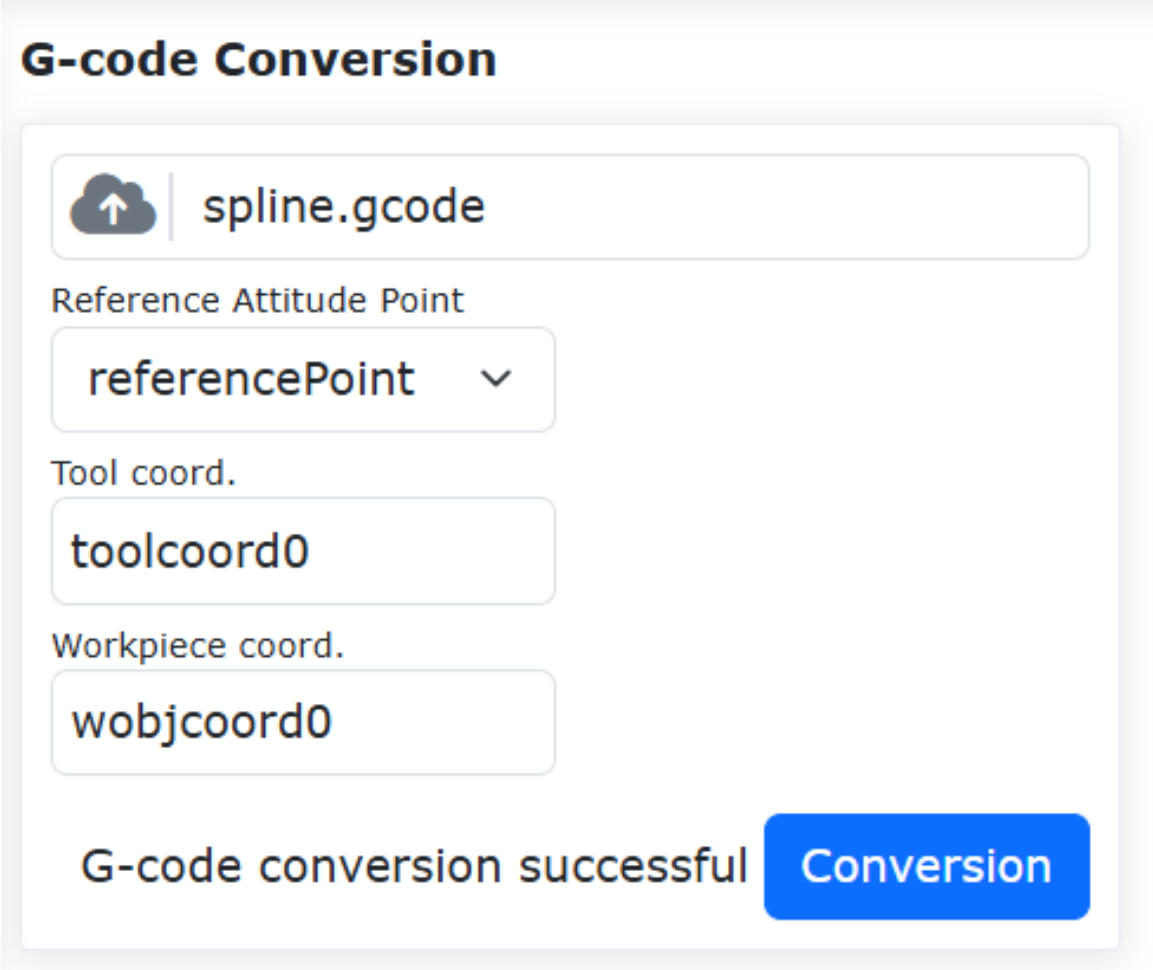


Figure 14.8-4 G-code Conversion Success Interface

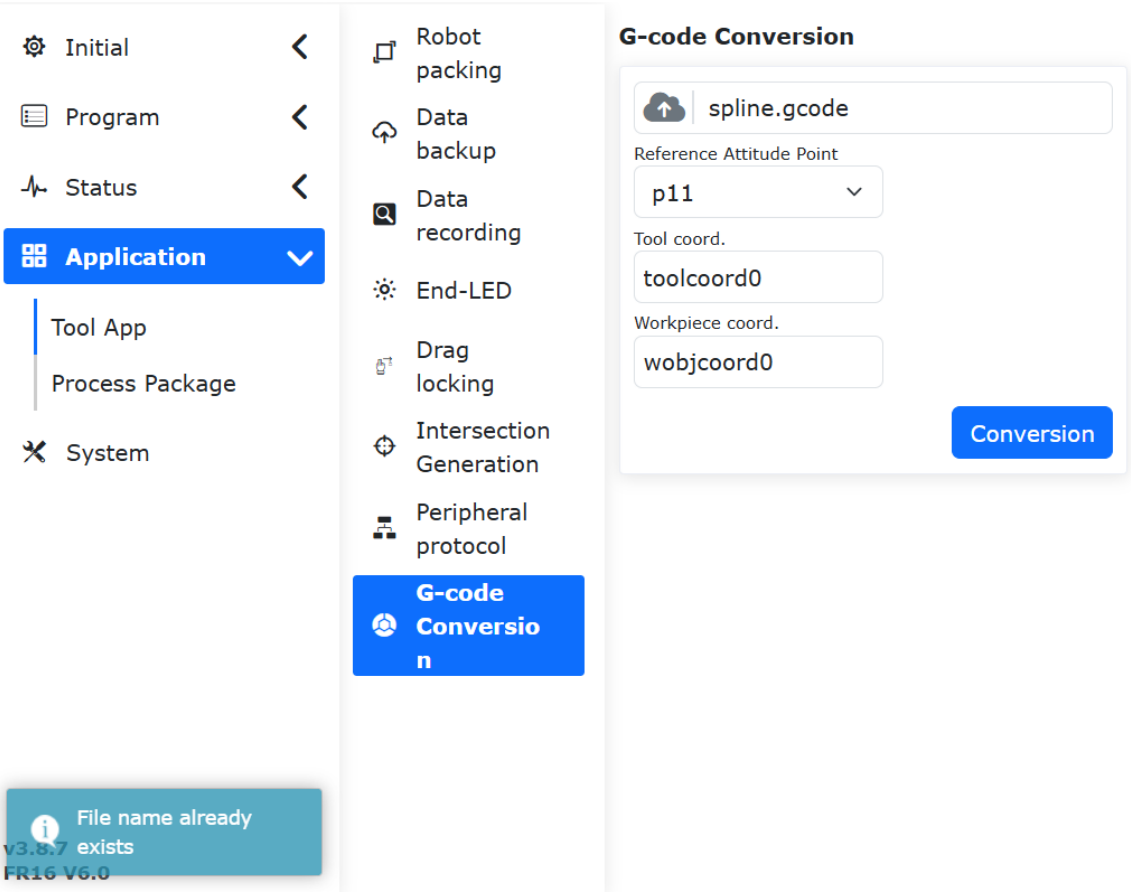


Figure 14.8-5 G-code Conversion Failure Interface

Step 6: Click “Teach Programming” -> “Program Programming” button, open the LUA file generated after G-code conversion, switch the robot to Auto mode, and click the Start button. The robot will then replicate the path from the G-code file.



Figure 14.8-6 Open the Converted LUA File

1.3.15 Process Package

1.3.15.1 Welding Expert Library

Click the “Welding Expert Library” menu item under “Auxiliary Application” -> “Process Package” to enter the Welding Expert Library function interface, which includes Straight Welding, Arc Welding, Multi-Layer Multi-Pass Welding, and Posture Adjustment.

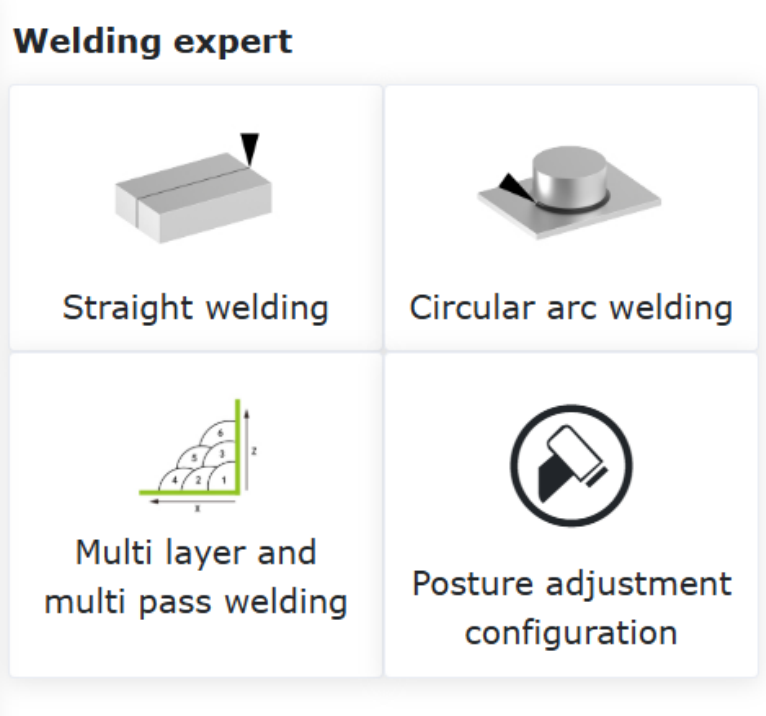


Figure 15.1-1 Extended Axis Configuration

1.3.15.1.1 Straight Welding

Click “Straight Welding” to enter the Straight Welding guidance interface. Based on the completion of basic robot settings, we can quickly generate welding teaching programs through a few simple steps. It mainly includes the following five steps. Due to functional mutual exclusion, the actual steps to generate one welding teaching program are fewer than five.

Step 1: Decide whether to use the extended axis. If using the extended axis, you need to configure the relevant coordinate system for the extended axis and enable it. Weaving function cannot be used when the extended axis is used.

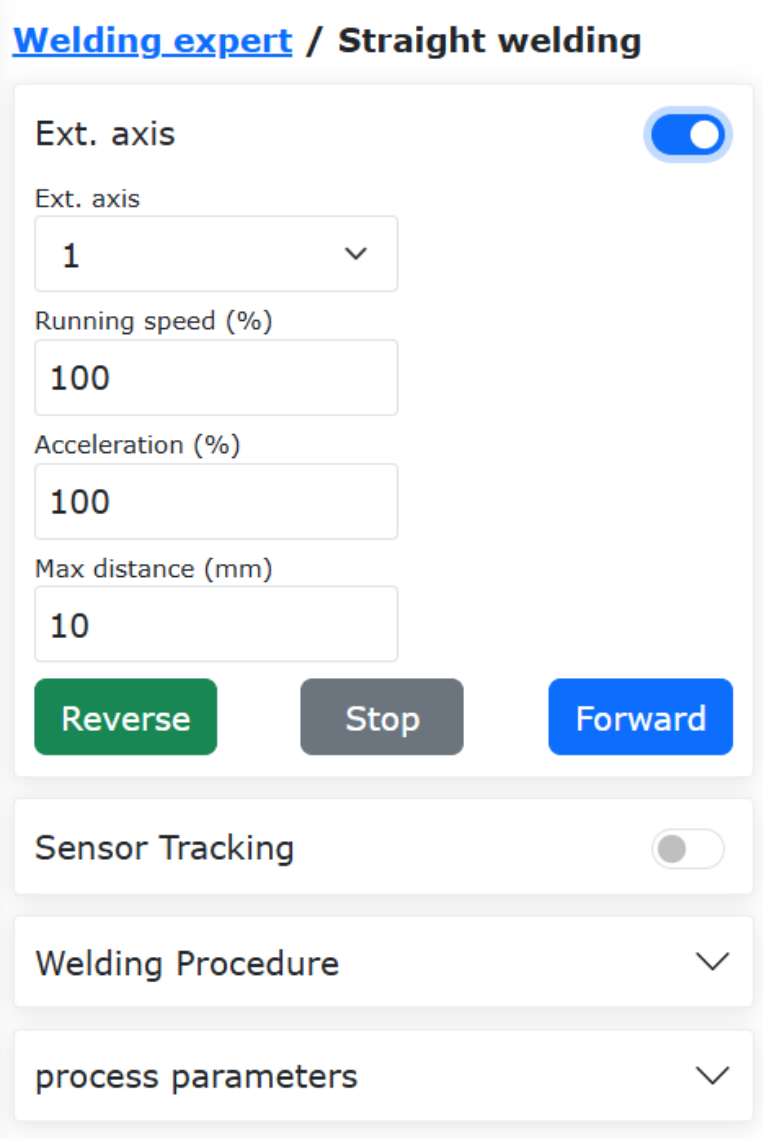


Figure 15.1-2 Extended Axis Configuration

Step 2: Choose whether sensor tracking is needed. If yes, you need to edit the parameters of the laser search command. Weaving function cannot be used when sensor tracking is used.

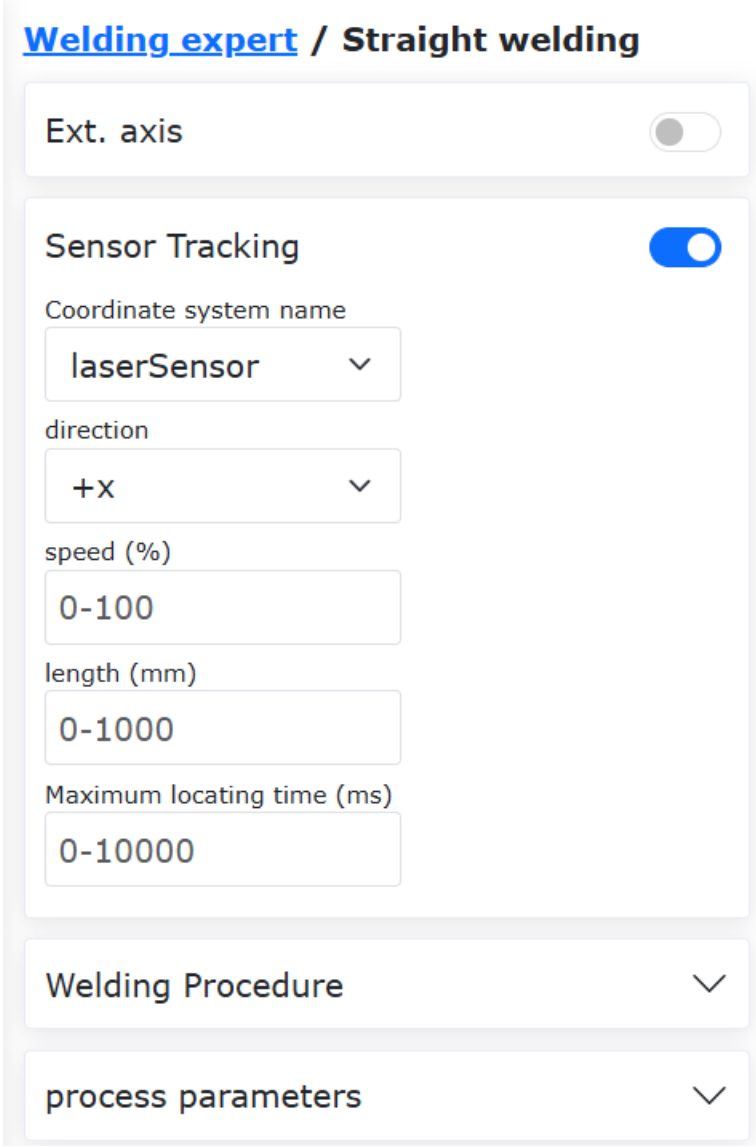


Figure 15.1-3 Laser Search Configuration

Step 3: Choose whether weaving is needed. If weaving is needed, you need to edit the relevant weaving parameters.

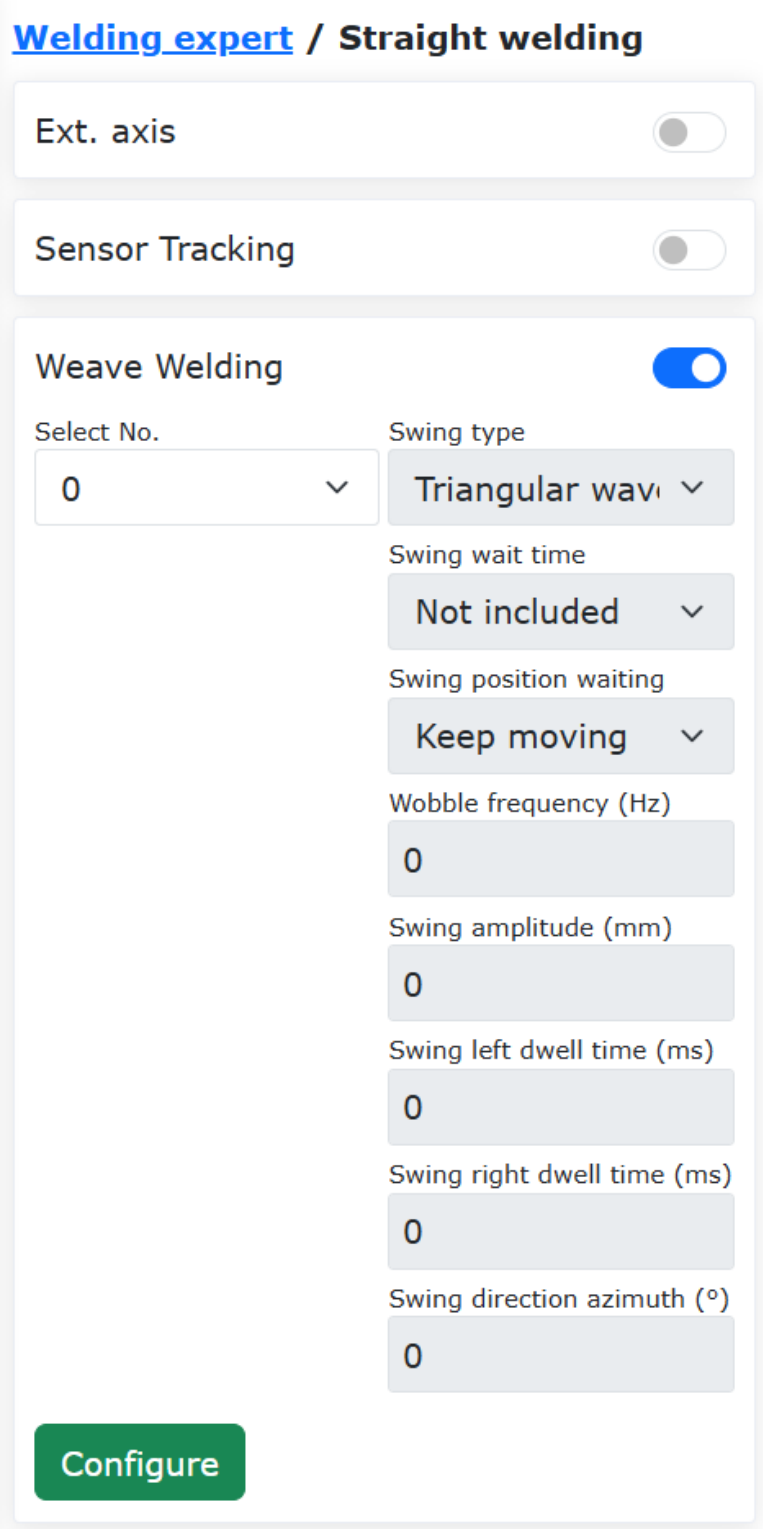


Figure 15.1-4 Weaving Configuration

Step 4: Calibrate the Start Point, Start Safe Point, End Point, and End Safe Point. If the extended axis was selected in Step 1, the extended axis movement function will be loaded to assist with the calibration of the relevant points.

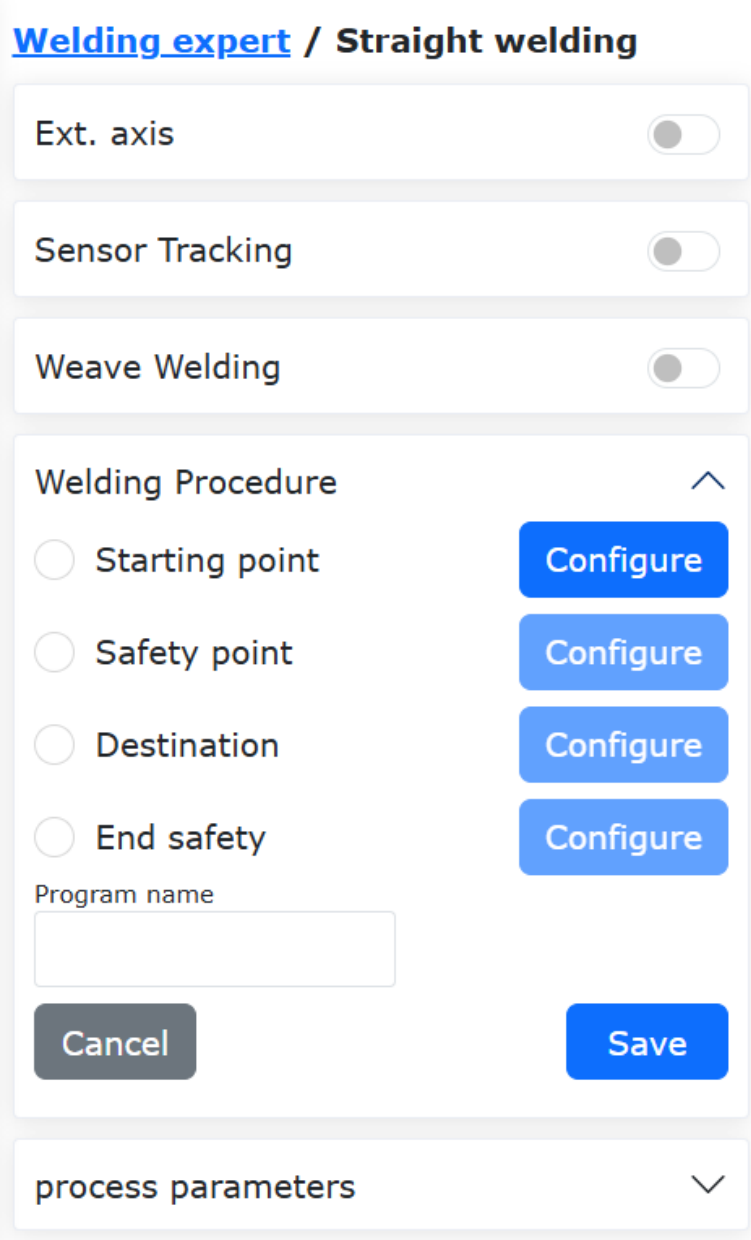


Figure 15.1-5 Calibrating Relevant Points

Step 5: Name the program, and it will automatically open in the program teaching interface.

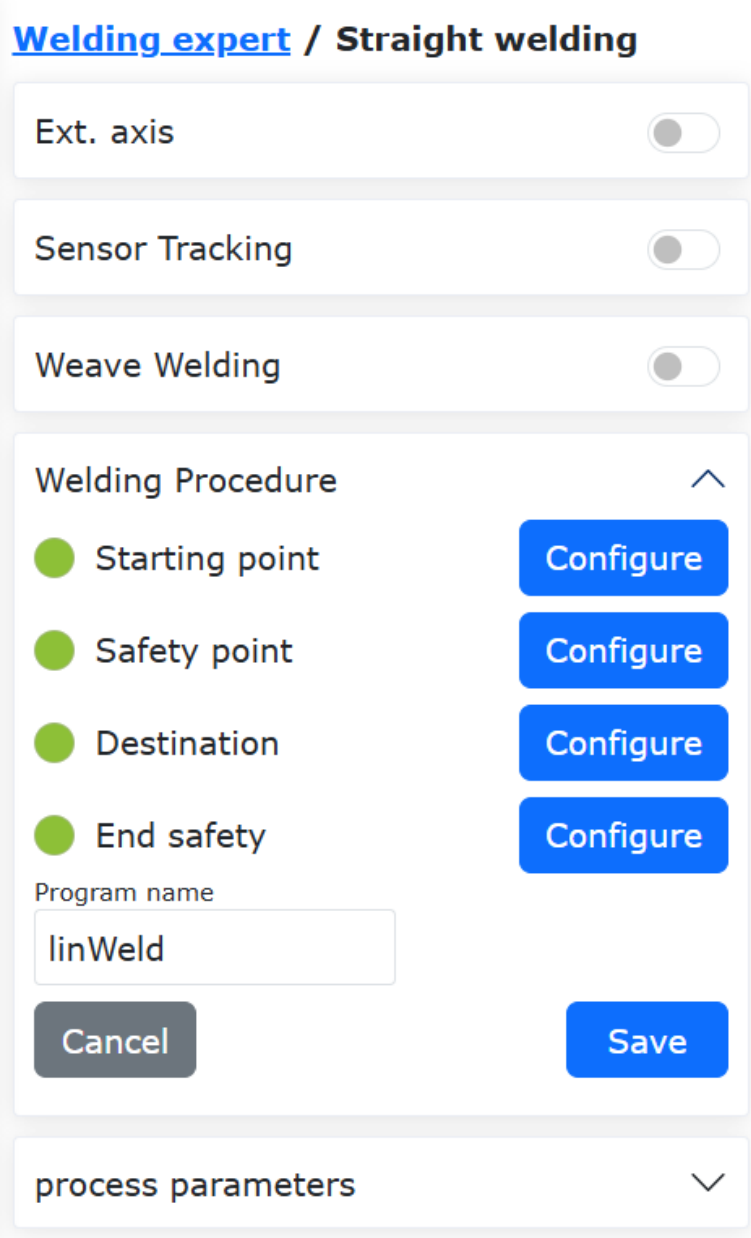


Figure 15.1-6 Saving the Program

After the program is successfully saved, you can modify the welding speed in the Process Parameters.

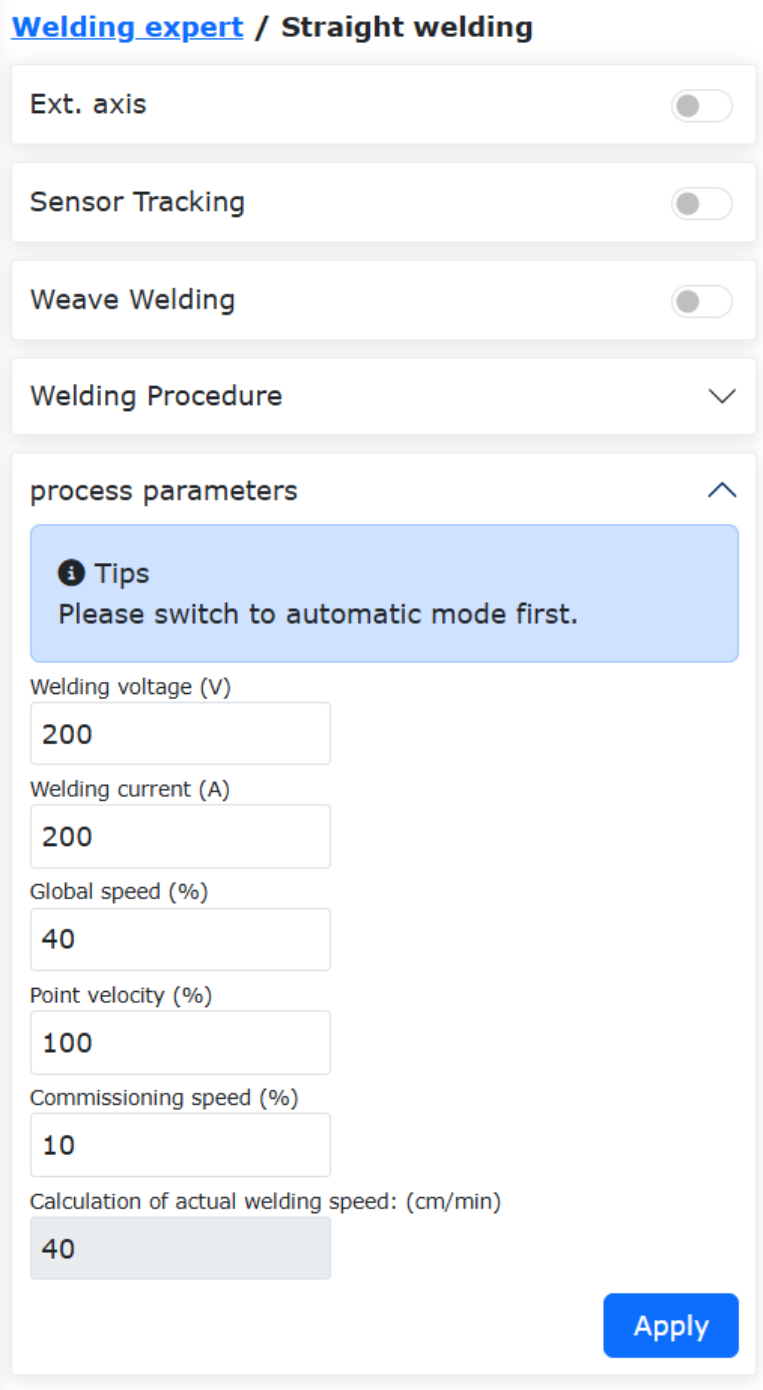


Figure 15.1-6 Process Parameters

1.3.15.1.2 Arc Welding

Click “Arc Welding” under “Weldment Shape” to enter the Arc Welding guidance interface. Based on the completion of basic robot settings, we can quickly generate welding teaching programs through two simple steps. It mainly includes the following two steps.

Step 1: Calibrate the Start Point, Start Safe Point, Arc Transition Point, End Point, and End Safe Point.

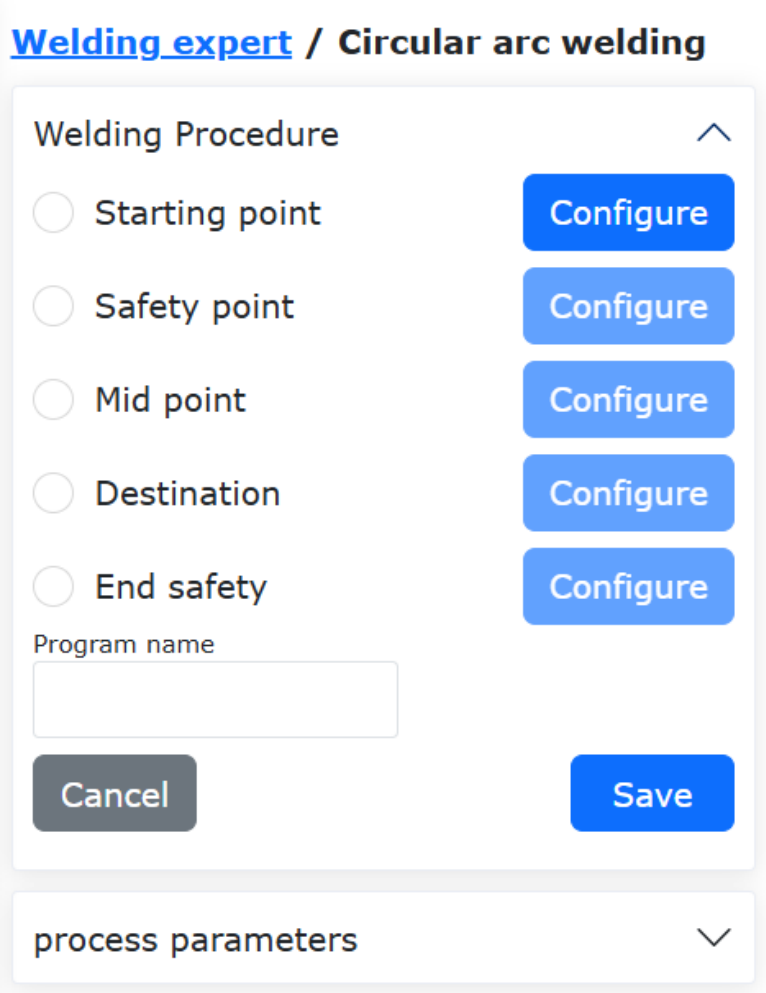


Figure 15.1-8 Calibrating Points

Step 2: Name the program, and it will automatically open in the program teaching interface.

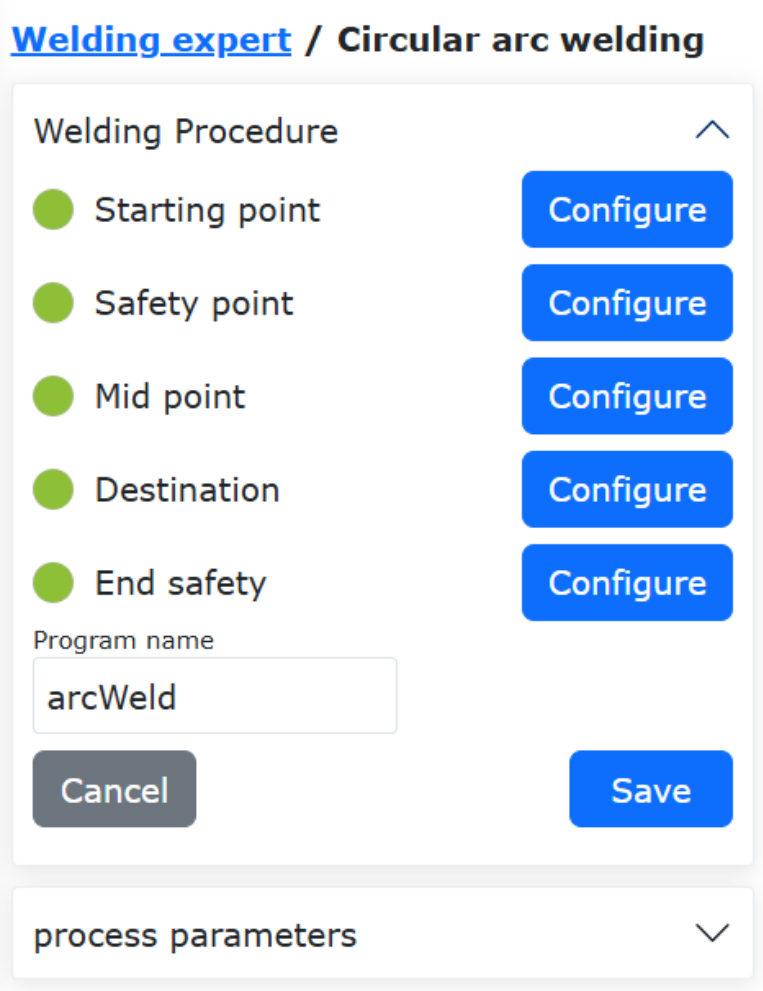


Figure 15.1-9 Saving the Program

After the program is successfully saved, you can modify the welding speed in the Process Parameters.

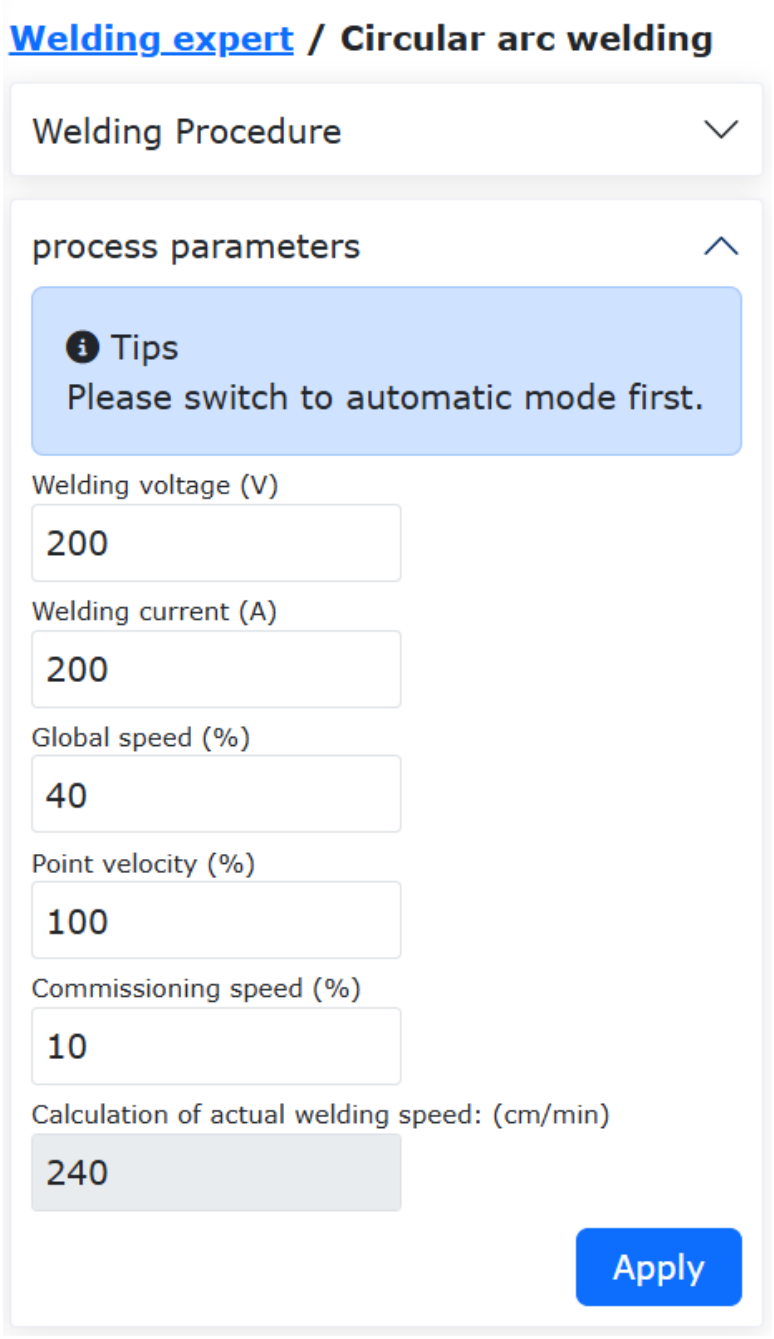


Figure 15.1-10 Process Parameters

1.3.15.1.3 Multi-Layer Multi-Pass Welding

When the weld leg size is greater than 10mm, the Multi-Layer Multi-Pass Welding function is typically used. This function enables template-based configuration of welding programs, incorporates arc tracking function during the first pass of multi-layer multi-pass welding, and corrects weld deviation in subsequent multi-pass straight welding processes, thereby improving weld quality.

The operation process for Arc Tracking Multi-Layer Multi-Pass Welding is as follows:

1) Set the Tool Coordinate System, enter the tool dimensions and posture of the welding torch.

Note: The values on the interface are for example only; use the actual tool status.

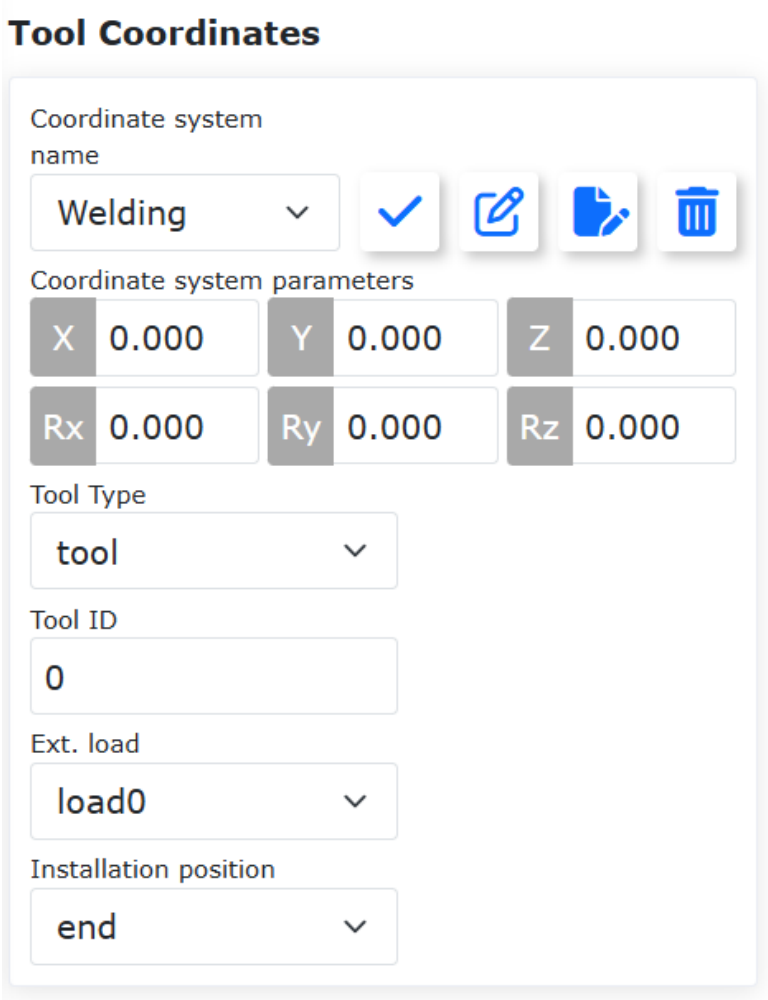


Figure 15.1-11 Setting Tool Coordinate System

2) Click “Multi-Layer Multi-Pass Welding” to enter the interface.

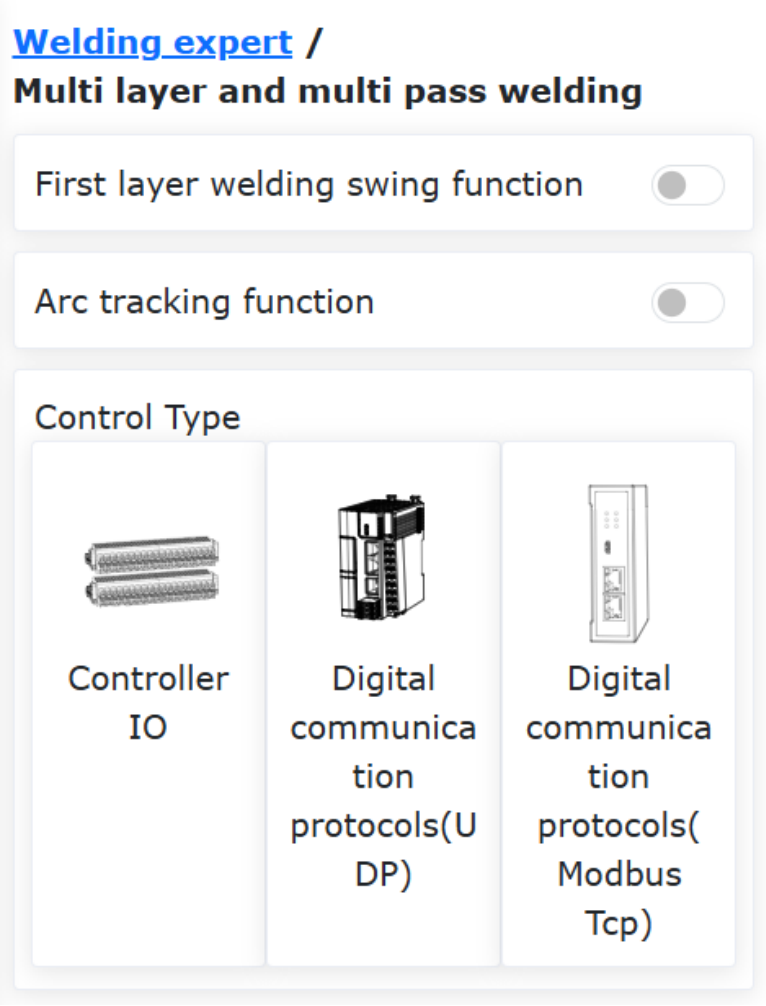


Figure 15.1-12 Opening the Multi-Layer Multi-Pass Welding Interface

3) To use the arc tracking function, be sure to turn on the “First Layer Welding Weaving Function” switch and configure the corresponding weaving parameters.

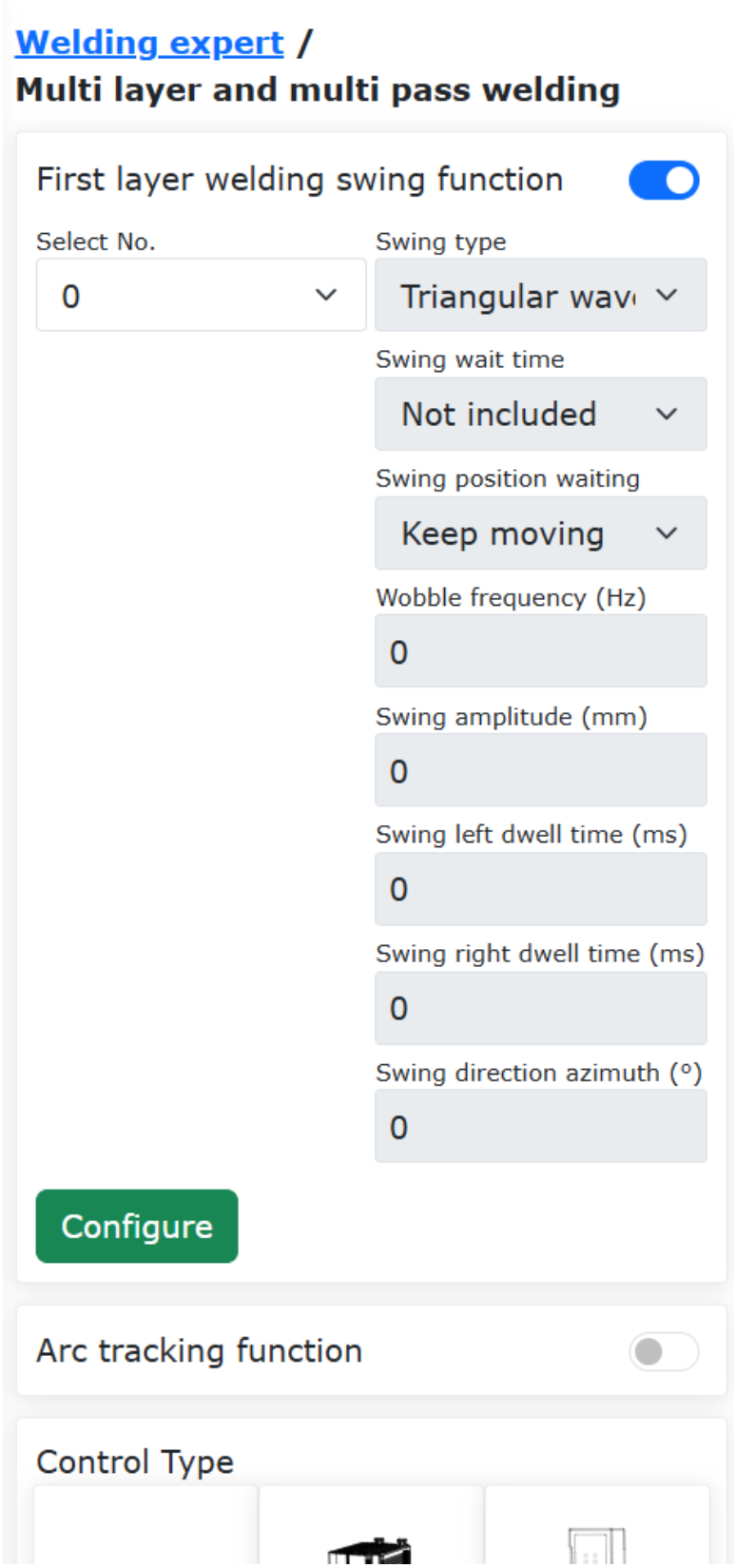


Figure 15.1-13 Enabling First Layer Welding Weaving Function

4) Click the “Configure” button, edit the weaving parameters, then click “Configure”.

Note: If arc tracking requires left/right compensation, only “Triangle Wave Weaving” and “Sine Wave Weaving” types can be selected. The weaving frequency must not be lower than 0.5Hz, the weaving amplitude must not be less than 3mm, the left and right dwell times must be the same, and the weaving azimuth angle must be 0.

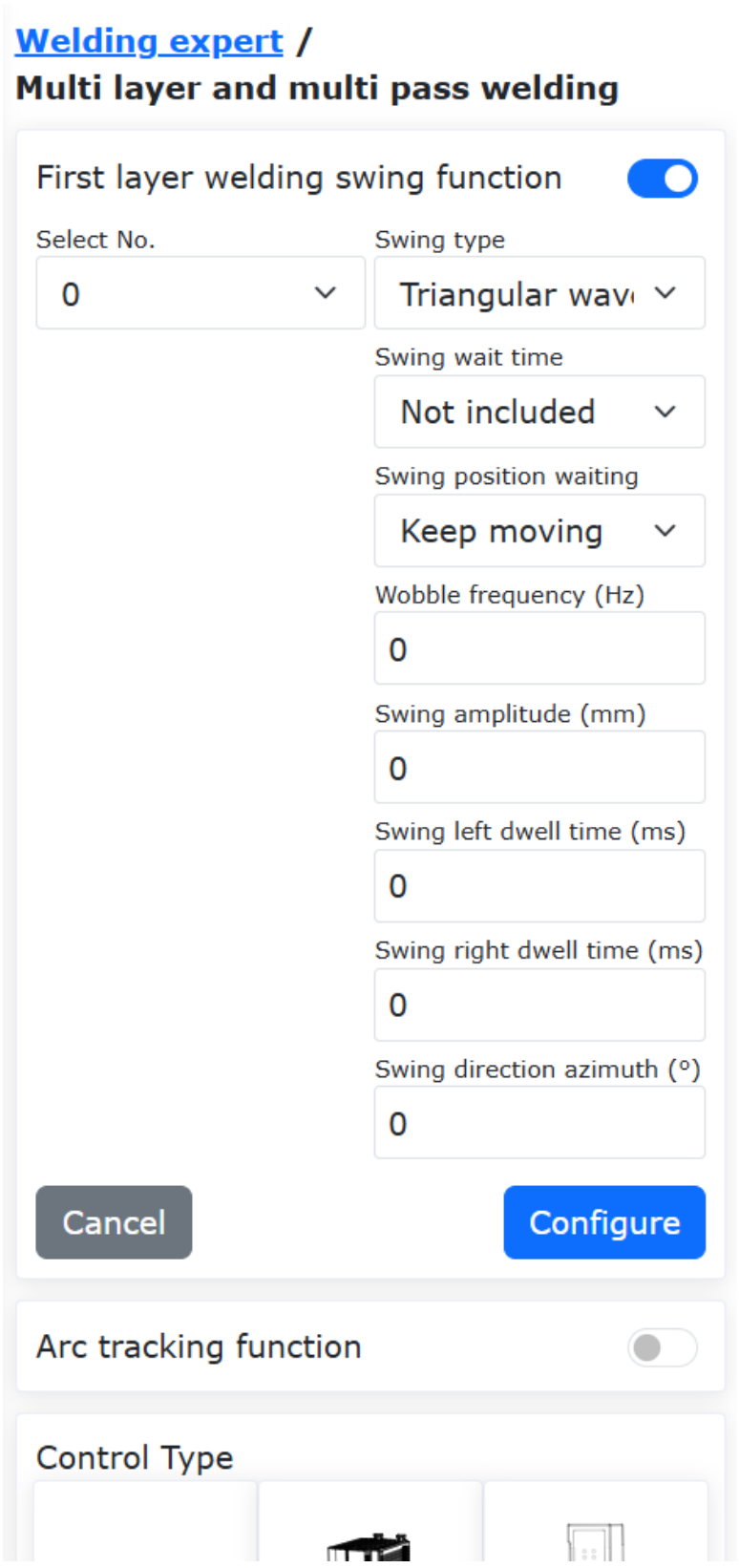


Figure 15.1-14 Configuring Weaving Parameters

5) Turn on the “Arc Tracking Function” switch and edit the corresponding up/down and left/right compensation

parameters.

Note: Configure the arc tracking parameters according to the actual welding situation, refer to the "Arc Tracking Function Operation Manual" or contact relevant technical personnel.

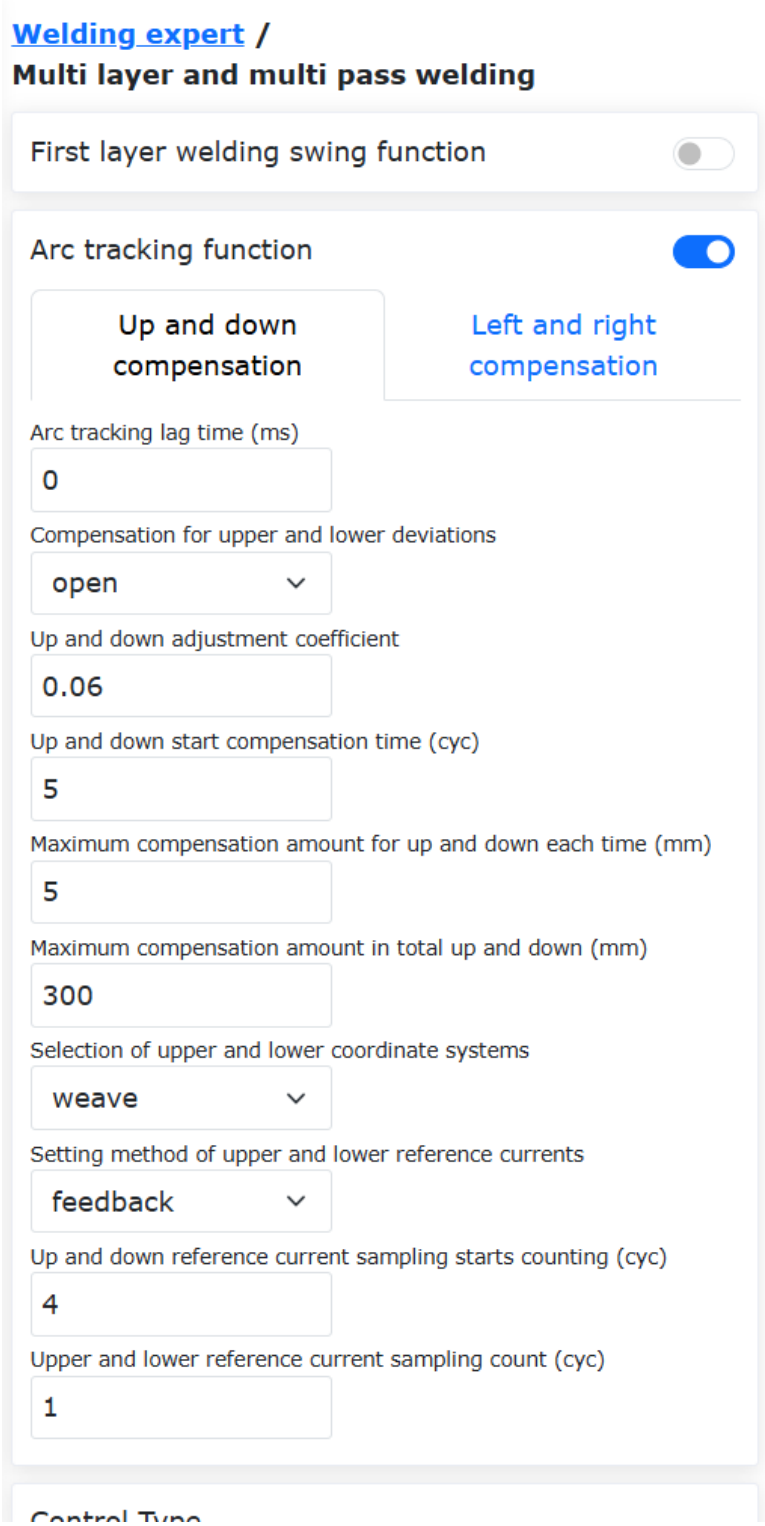


Figure 15.1-15 Configuring Arc Tracking Parameters

6) Click the corresponding type according to the control type to enter the interface. First, in the first set of points, set the "Welding Point" as the welding start position; the "X+ Point" is a point in the X+ direction of the custom offset coordinate system relative to the welding point; the "Z+ Point" is a point in the Z+ direction of the custom offset coordinate system relative to the welding point; the "Safe Point" is the transition position from the completion of the last weld to the start of the next weld. After teaching and setting, automatically proceed to the second set of points.

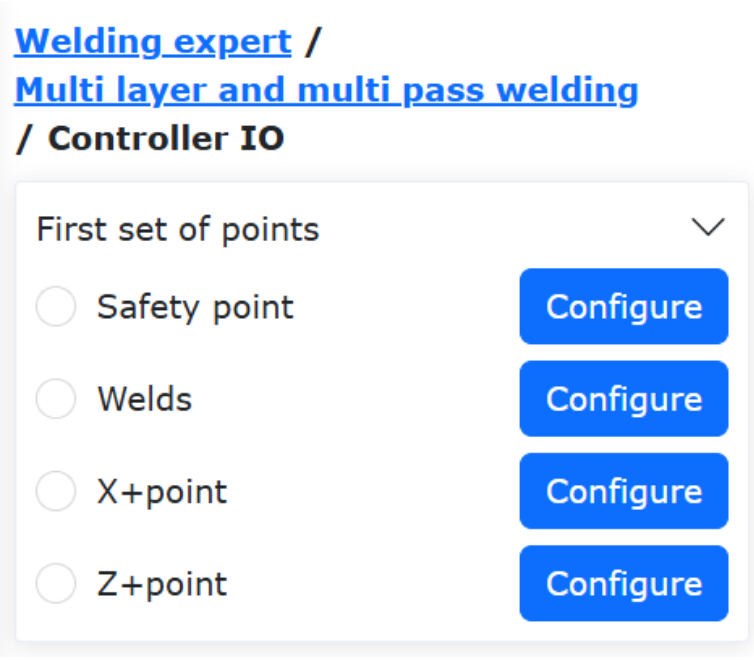


Figure 15.1-16 Multi-Layer Multi-Pass Welding Straight Start Point Position Setting

7) Select "Straight Point". Here, the "Welding Point" is the welding end position; the "X+ Point" is a point in the X+ direction of the custom offset coordinate system relative to the "Welding Point"; the "Z+ Point" is a point in the Z+ direction of the custom offset coordinate system relative to the "Welding Point". After teaching and setting, click the "Finish" button to set the multi-layer multi-pass welding parameters.

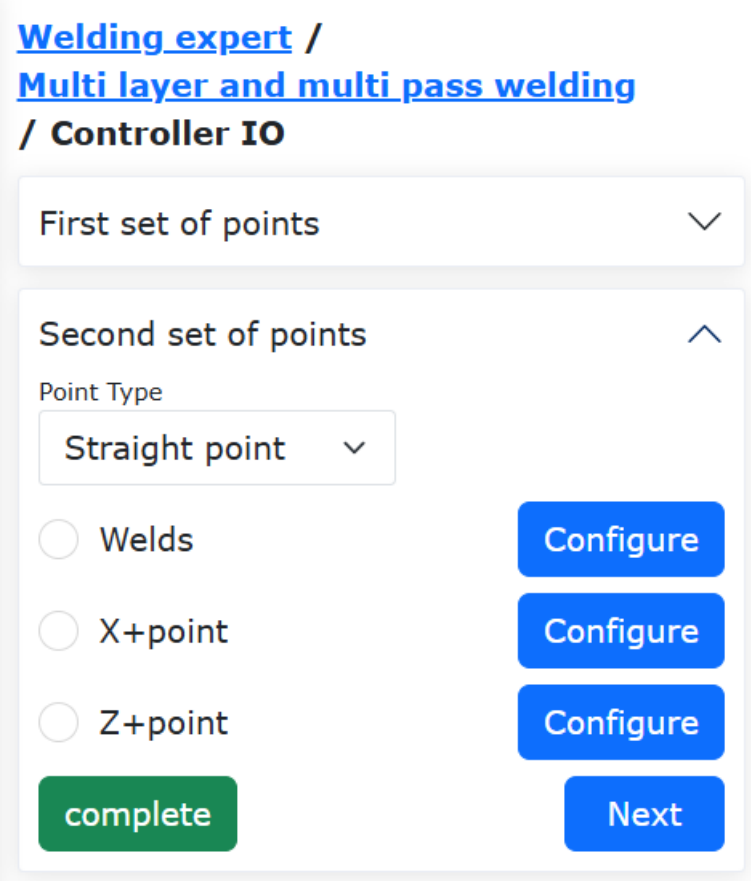


Figure 15.1-17 Multi-Layer Multi-Pass Welding Straight End Point Position Setting

8) On this page, you can set the number of multi-layer multi-pass welds and their distribution positions. Click the "On/Off" box in the parameter table to activate the corresponding offset position and angle values for the multi-layer multi-pass welding positions, and enter the desired offset positions and angles in the custom coordinate system in the "X", "Z", and "B" columns.

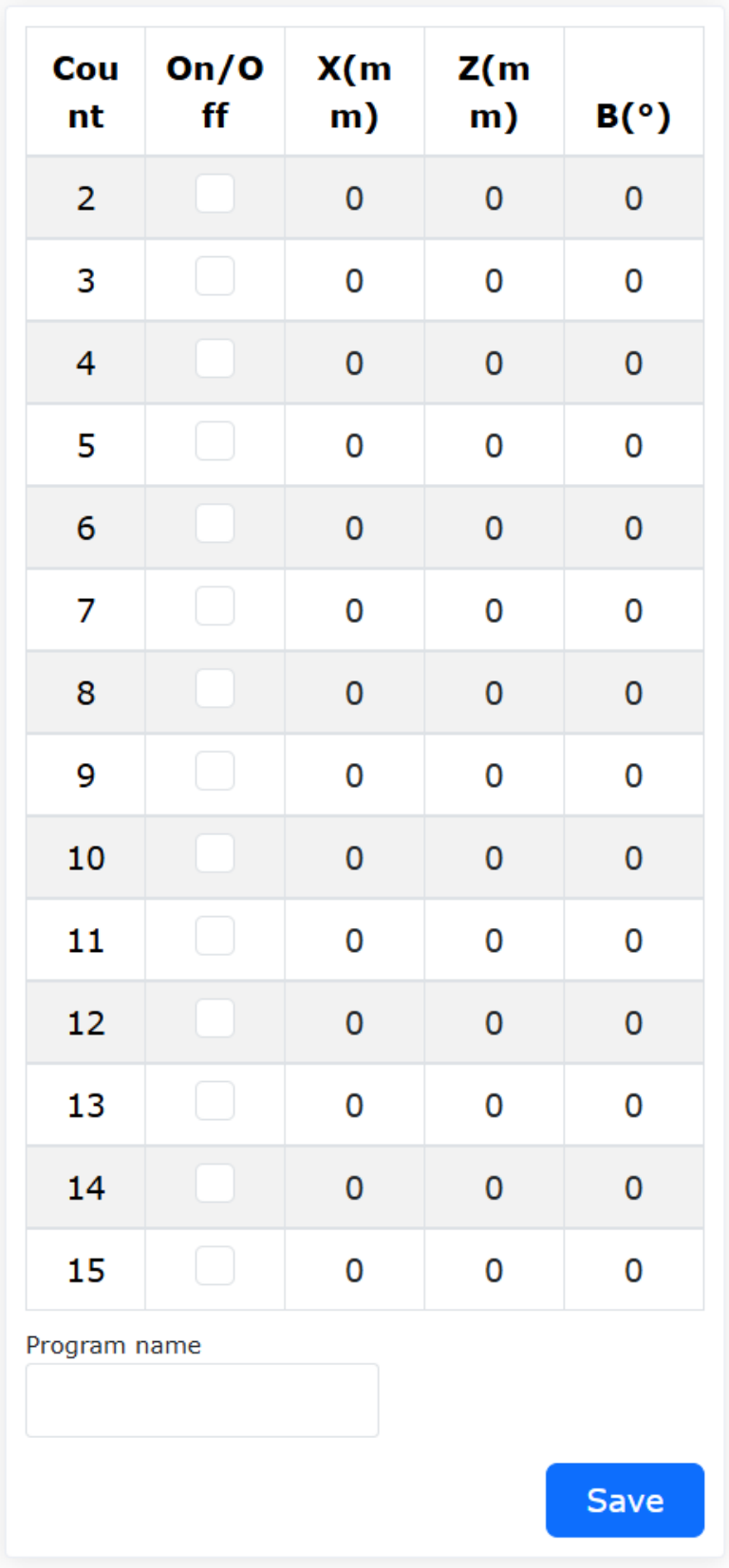


Figure 15.1-18 Multi-Layer Multi-Pass Welding Parameter Setting

9) At this point, all parameter configuration is complete. Enter the desired program name and click the "Save" button to automatically generate the corresponding multi-layer multi-pass welding program.

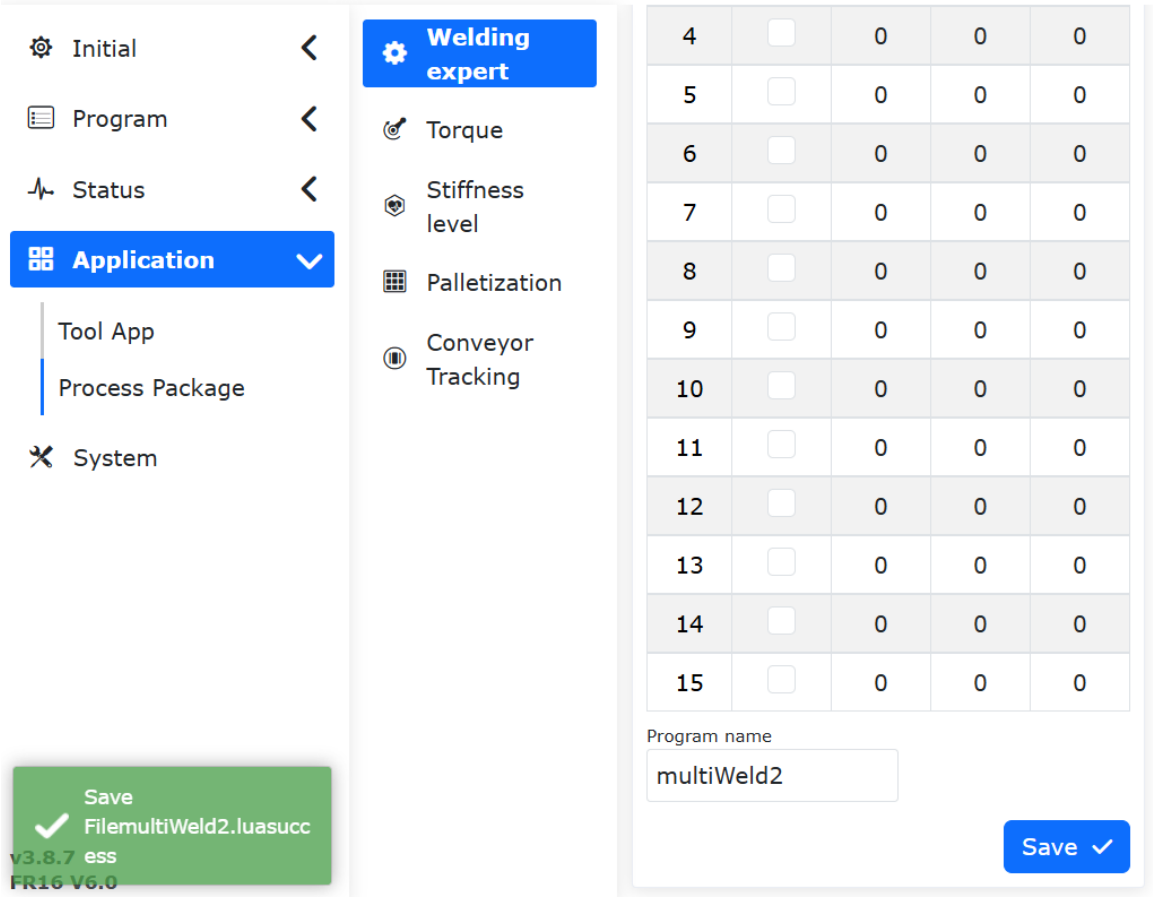


Figure 15.1-19 Multi-Layer Multi-Pass Welding Program Generation

10) Click the "Open Program" button to read the Lua program saved in the previous step. The program content is shown in the figure below.

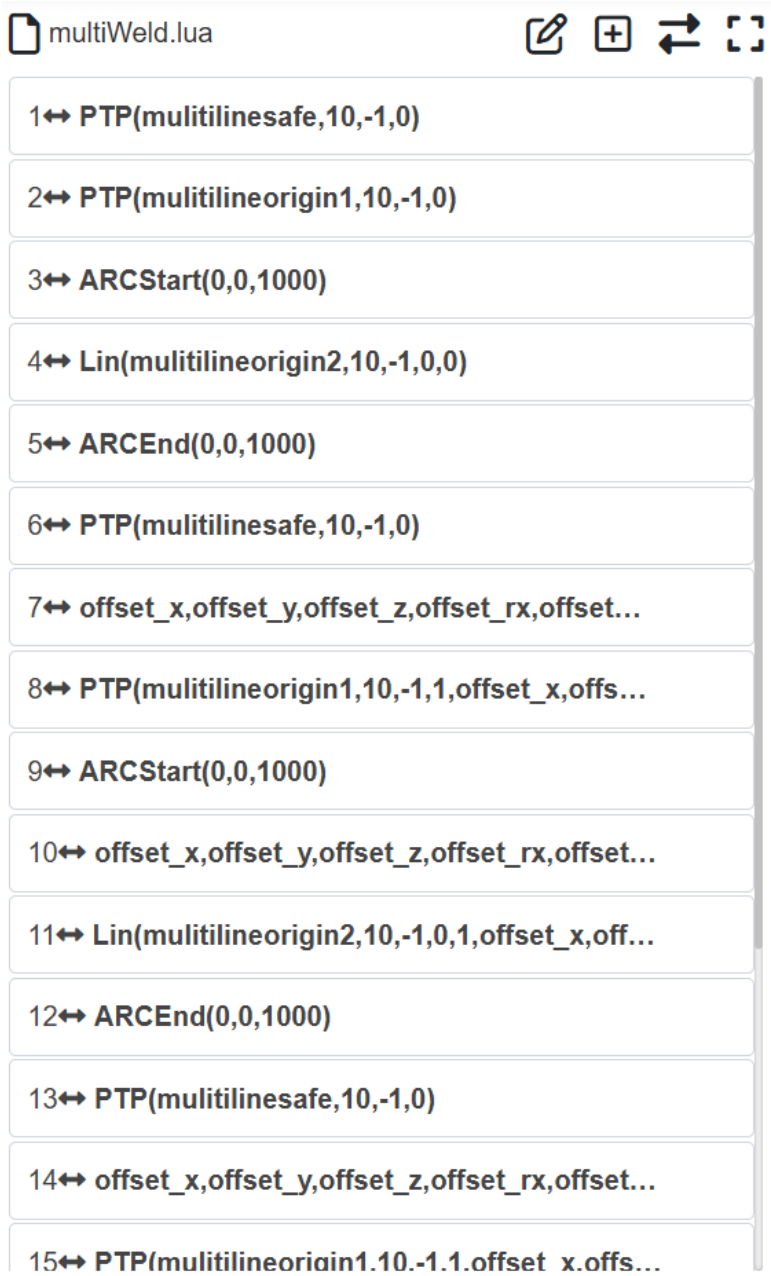


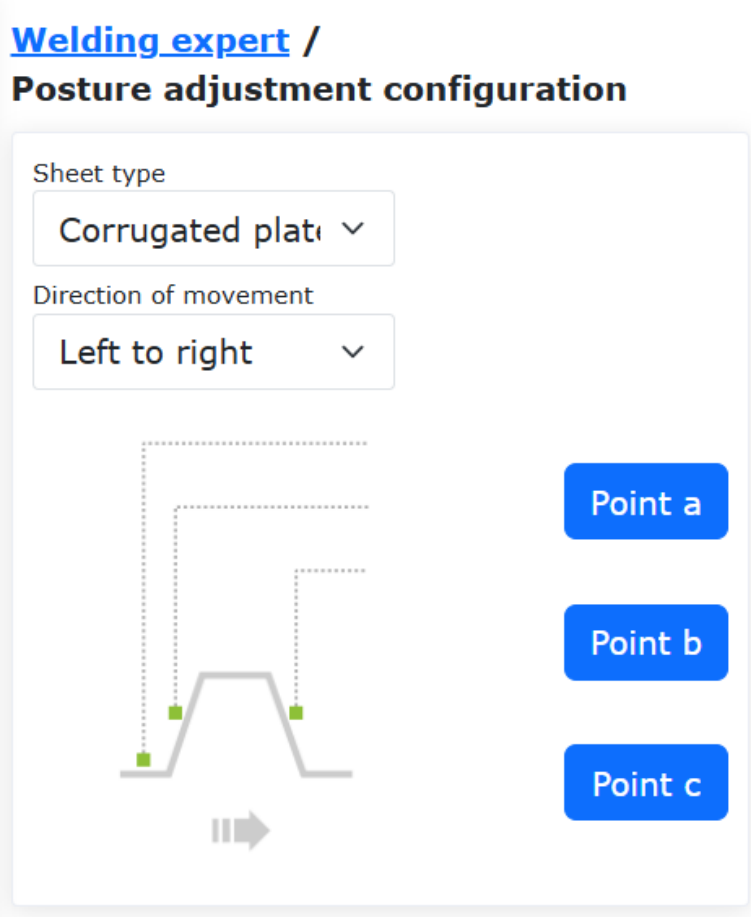
Figure 15.1-20 Arc Tracking Multi-Layer Multi-Pass Welding Program Example

1.3.15.1.4 Posture Adjustment

1.3.15.1.4.1 Posture Adaptive Configuration Steps

Step1: Enter the Posture Adjustment configuration interface, select the plate type and the actual robot working movement direction, adjust the robot posture, and set Posture Point A, Posture Point B, and Posture Point C respectively. Usually, A is the flat posture point, B is the rising edge posture point, and C is the falling edge posture point.

Figure 15.1-21 Posture Adjustment Configuration


Welding expert /
Posture adjustment configuration
Sheet type
Corrugated plat
Direction of movement
Left to right
Point a
Point b
Point c

Important: The posture change between posture A and B, and posture A and C should be as small as possible while meeting the application requirements. The posture adaptive function is an auxiliary application function, usually used in conjunction with seam tracking.

Step2: Select the "Adjust" command in the program teaching command interface. Add instructions as needed according to the specific program teaching requirements.

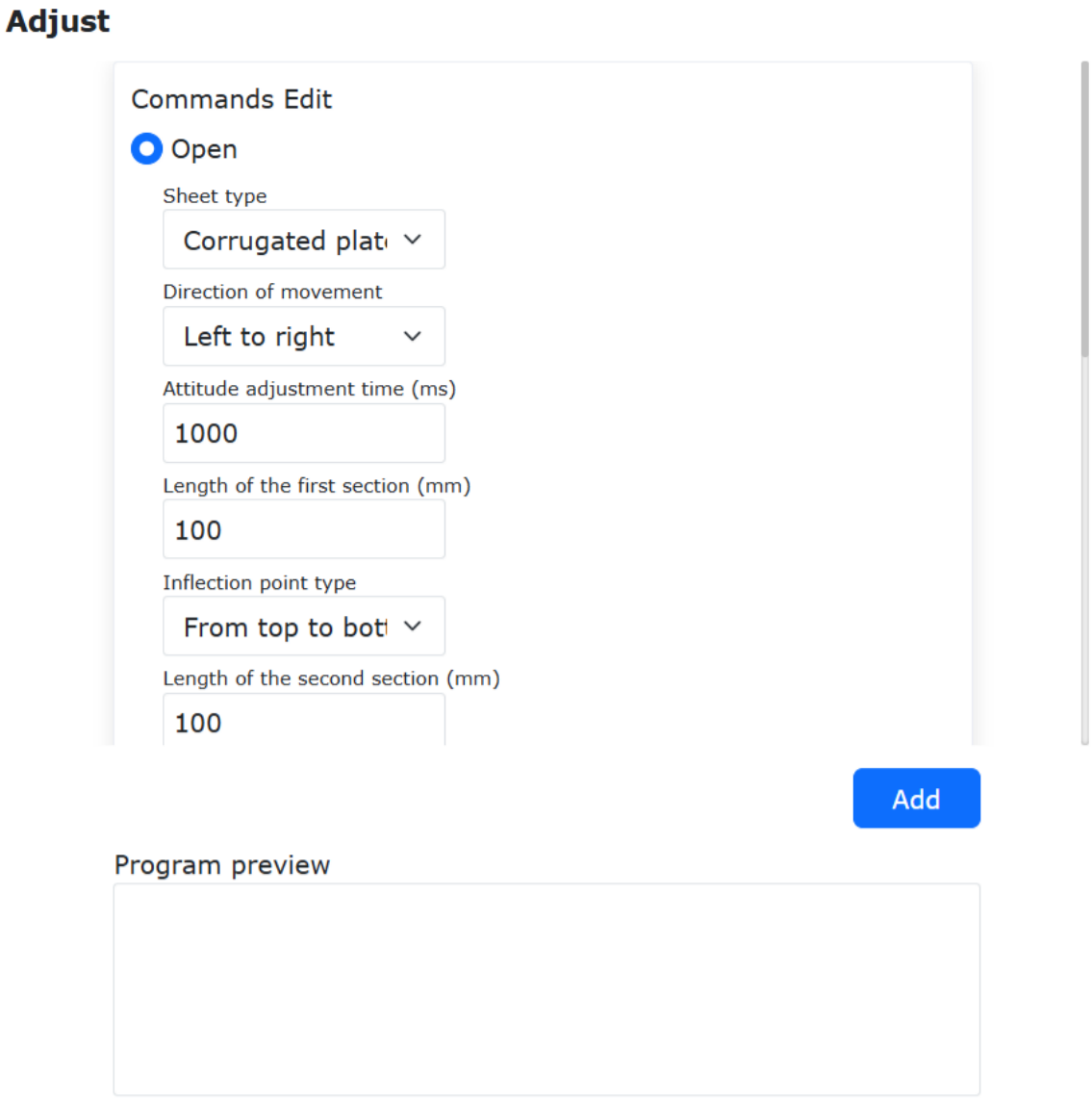


Figure 15.1-22 Posture Adjustment Command Editing

1.3.15.1.4.2 Posture Adaptive Combined with Extended Axis and Laser Tracking Welding Teaching Program

No.	Command Format	Comment
1	EXT_AXIS_PTP(1,1laserstart)	#External axis move to laser sensor start point
2	PTP(laserstart,10,-1,0)	#Robot move to laser sensor start point
3	LTSearchStart(3,20,10,10000)	#Start search
4	LTSearchStop()	#Stop search
5	EXT_AXIS_PTP(1,1,seamPos)	#External axis move to seam start point
6	Lin(seamPos,20,-1,00,0)	#Robot move to seam start point
7	LTTrackOn()	#Laser tracking on
8	ARCStart(0,10000)	#Arc start
9	PostureAdjustOn(0,PosA,PosC,PosB,1000)	#Posture adaptive adjustment on
10	EXT_AXIS_PTP(1,1,laserend)	#External axis move to seam end point
11	Lin(laserend,10,-1,0,0)	#Robot move to seam end point
12	ARCEnd(0,10000)	#Arc end
13	PostureAdjustOff(0)	#Posture adaptive adjustment off
14	LTTrackOff	#Laser tracking off

1.3.15.2 Palletizing System Configuration

1.3.15.2.1 Palletizing System Configuration Steps

Step1: Click the “Palletizing” menu item under “Auxiliary Application” -> “Process Package” to enter the Palletizing System Configuration interface.

For first-time use, you need to first create a recipe. Click “Recipe Creation”, enter the recipe name, click “Create”. After successful creation, click “Start Configuration” to enter the palletizing configuration page.

Figure 15.2-1 Palletizing Recipe Configuration

Step2: In the Workpiece Configuration section, click “Configure” to enter the workpiece configuration pop-up window. Set the workpiece’s “Length”, “Width”, “Height”, and the workpiece pickup point. Click “Confirm Configuration” to complete the workpiece information setup.

Figure 15.2-2 Palletizing Workpiece Configuration

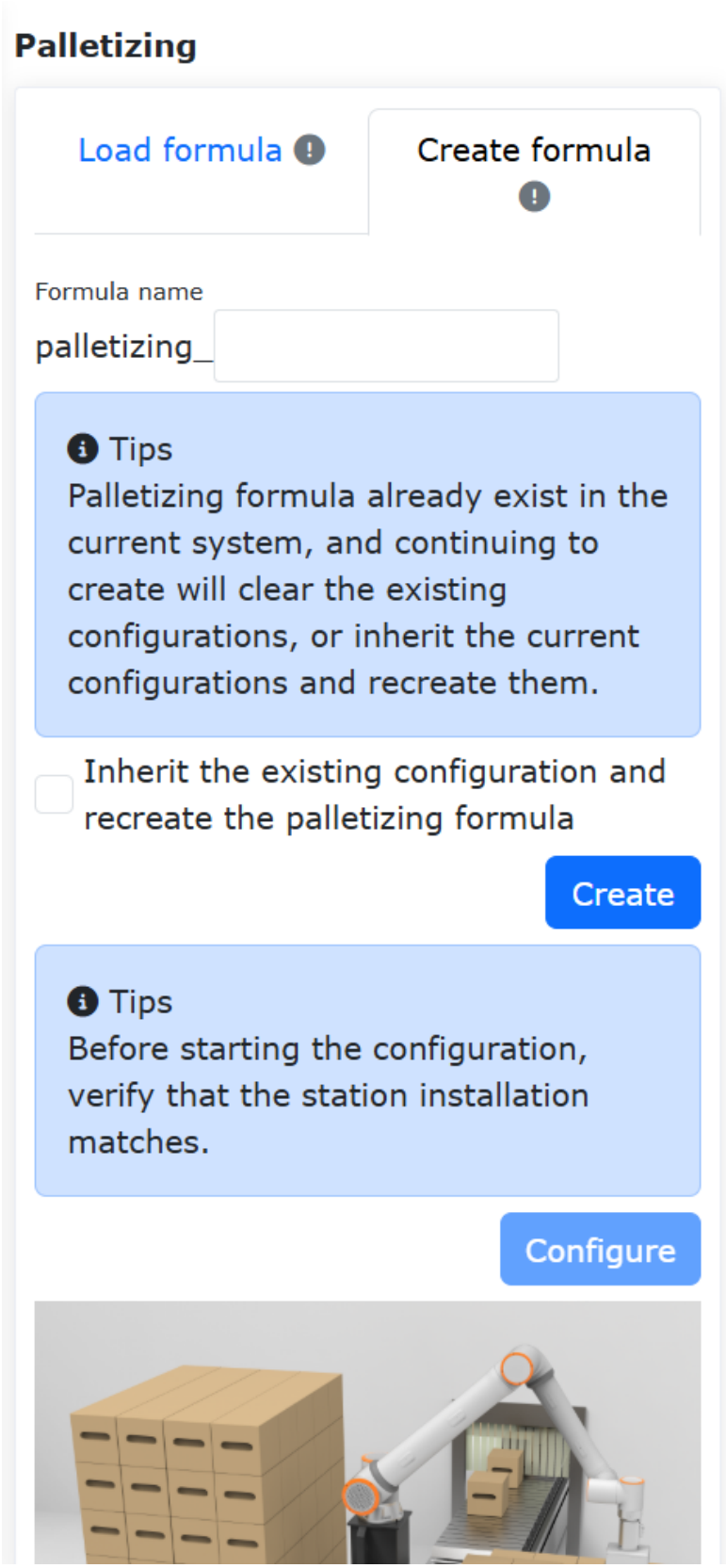
Step3: In the Pallet Configuration section, click “Configure” to enter the pallet configuration pop-up window. Set the pallet’s “Front Edge”, “Side Edge”, and “Height”, then set the station and station transition points. Click “Confirm Configuration” to complete the pallet information setup.

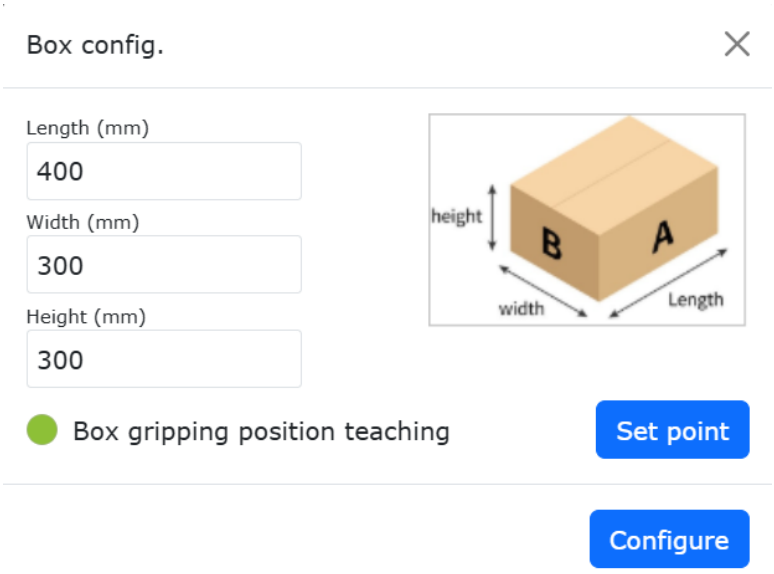
Figure 15.2-3 Palletizing Pallet Configuration

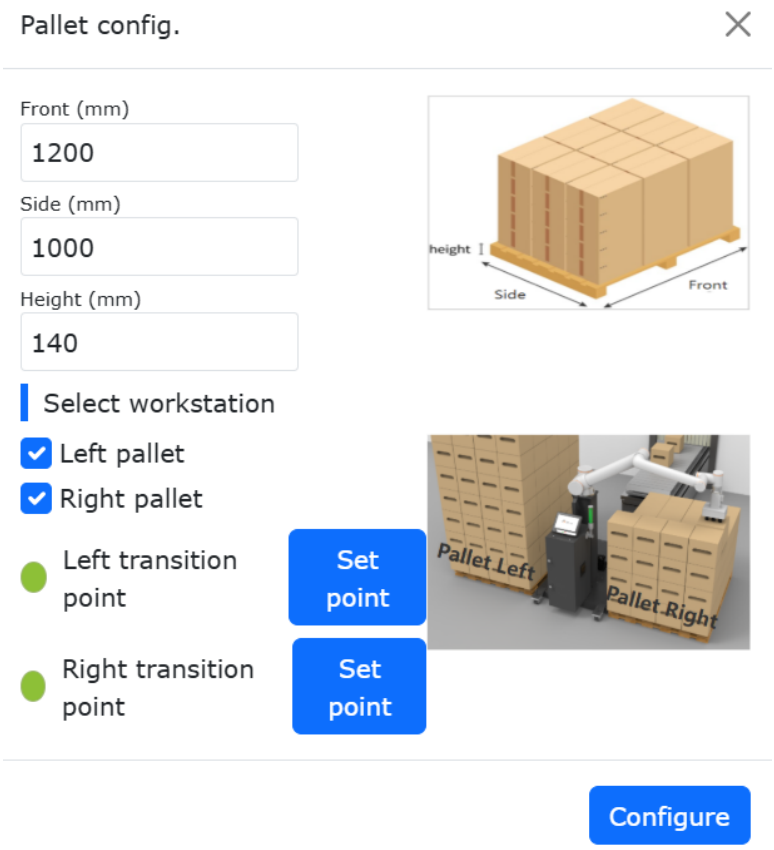
Step4: In the Palletizing Equipment Dimensions Configuration section, click “Configure” to enter the palletizing equipment dimensions configuration pop-up window. Set the equipment’s “X”, “Y”, “Z”, and “Angle”. Click “Confirm Configuration” to complete the palletizing equipment dimensions configuration.

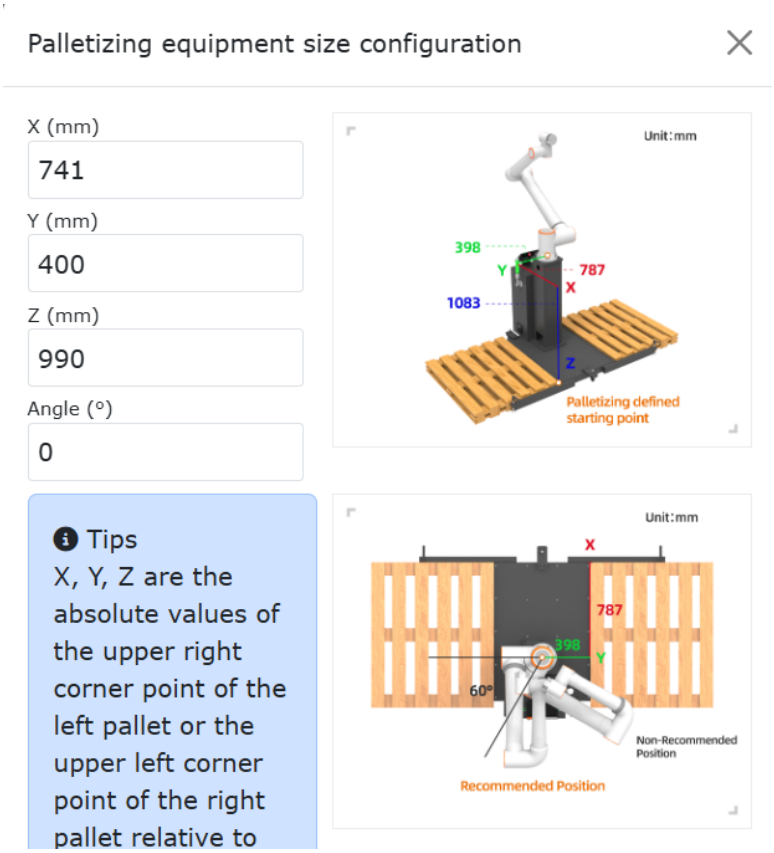
Important: X, Y, Z are the absolute values of the coordinates of the upper right corner point of the left pallet or the upper left corner point of the right pallet relative to the robot base coordinate system. Angle is the rotation angle during robot installation, recommended to be 0 during installation.

Figure 15.2-4 Palletizing Equipment Dimensions Configuration


Palletizing
Load formula
Create formula
Formula name
palletizing_
Tips
Palletizing formula already exist in the current system, and continuing to create will clear the existing configurations, or inherit the current configurations and recreate them.
Inherit the existing configuration and recreate the palletizing formula
Create
Tips
Before starting the configuration, verify that the station installation matches.
Configure


Box config.
Length (mm)
400
Width (mm)
300
Height (mm)
300
height
B
A
width
Length
Box gripping position teaching
Set point
Configure


Pallet config.
Front (mm)
1200
Side (mm)
1000
Height (mm)
140
height
Side
Front
Select workstation
Left pallet
Right pallet
Left transition point
Set point
Pallet Left
Pallet Right
Right transition point
Set point
Configure



Step5: In the Mode Configuration section, click "Configure" to enter the mode configuration pop-up window.

Mode B On/Off: On: Can switch between Mode A/B, configure Mode B for each palletizing layer; Off: Cannot switch to Mode B, cannot configure Mode B for each palletizing layer;

Mode A/B Switch: Select Mode A: Added workpieces are Mode A, workpiece numbers are A1, A2…, cannot adjust workpiece transparency; Select Mode B: Added workpieces are Mode B, workpiece numbers are B1, B2…, can then turn on/off "Show Mode A Configuration" to display Mode A workpieces;

Show Mode A On/Off: On: Adjust Mode B workpiece transparency to check if the A/B mode configuration effect is reasonable. At this time, only operations like selecting, adding, batch adding, deleting, and deleting all can be performed on Mode B workpieces; Off: Cannot set Mode B workpiece transparency;

Important: When configuring workpieces, if workpieces collide, the workpiece background color turns red, and the above operations cannot be performed. If operation is needed, please configure the workpieces to have no collision.

When configuring workpieces, first set the workpiece spacing. The right box simulates the placement of workpieces on the right pallet; you can add individually or in batches. Then set the number of palletizing layers and the mode for each layer. Click "Confirm Configuration" to complete the mode information setup.

Important: Palletizing direction: Taking the right pallet as an example, the lower right corner is the farthest point. Place a row of workpieces vertically or horizontally from the lower right corner, then place the next row of workpieces horizontally or vertically above it, and so on (The web page has marked the palletizing direction, please check).

The left pallet places workpieces mirrored based on the right pallet mode.

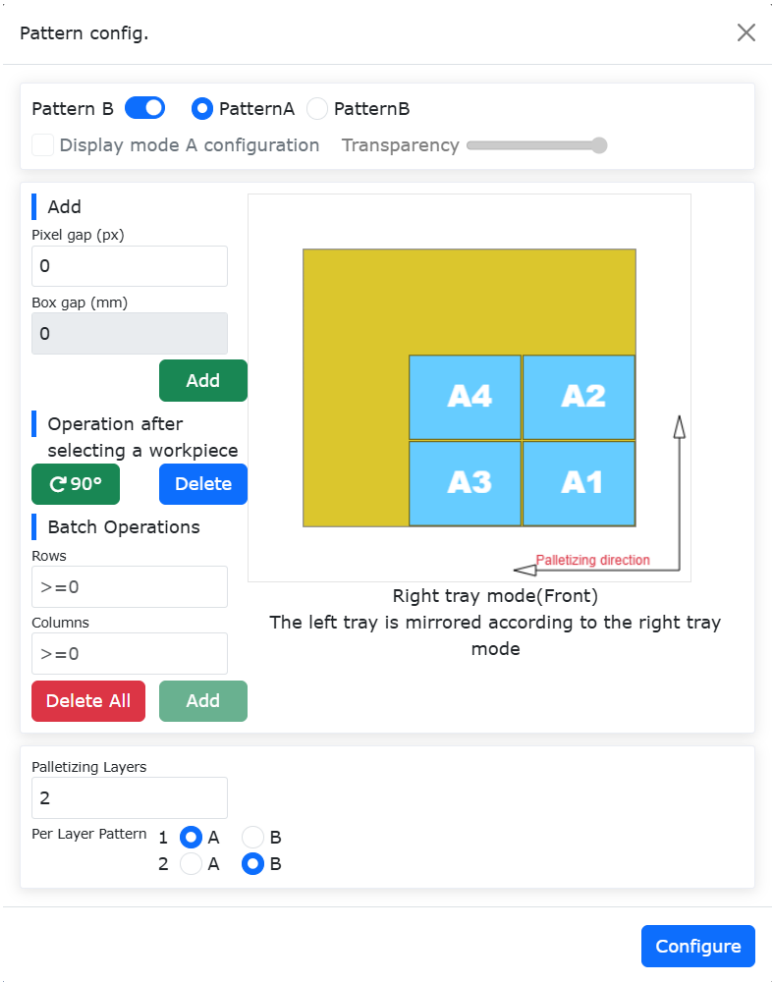

Pattern config.
Pattern B
PatternA
PatternB
Display mode A configuration
Transparency
Add
Pixel gap (px)
0
Box gap (mm)
0
Add
Operation after selecting a workpiece
90°
Delete
Batch Operations
Rows
>=0
Columns
>=0
Delete All
Add
A4
A2
A3
A1
Palletizing direction
Right tray mode(Front)
The left tray is mirrored according to the right tray mode
Palletizing Layers
2
Per Layer Pattern
1 A B
2 A B
Configure

Figure 15.2-5 Palletizing Mode A Configuration

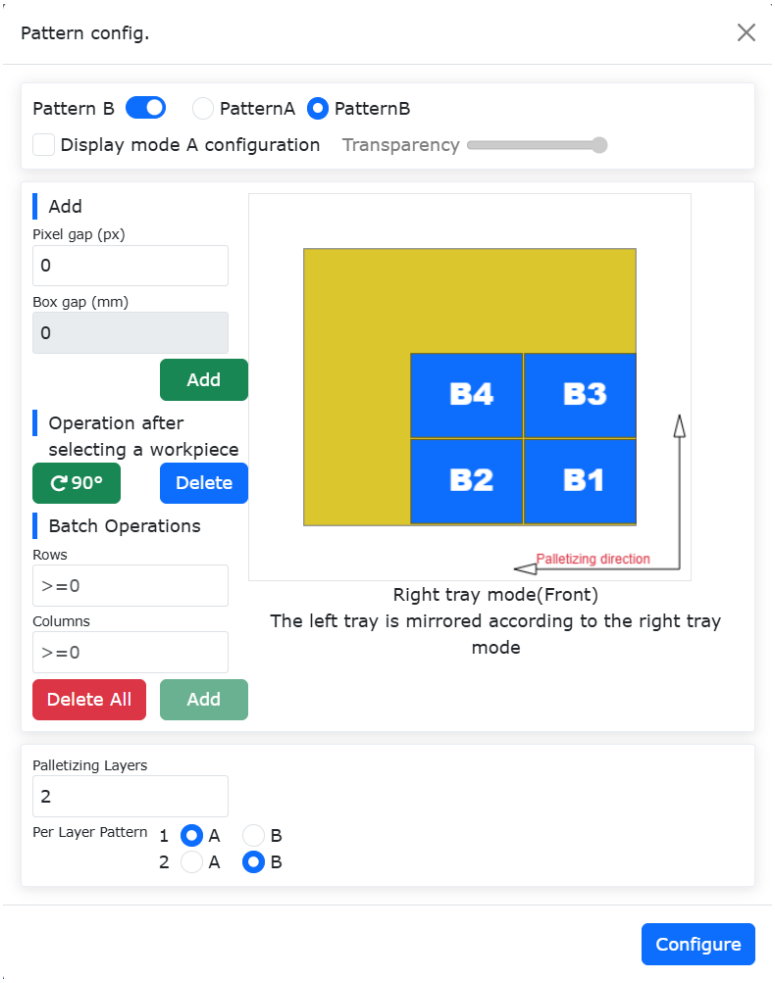


Figure 15.2-6 Palletizing Mode B Configuration

Step6: In the Teaching Program Generation section, click "Advanced Configuration" to enter the advanced configuration pop-up window. Configure the "Pick Lift Height", "First Offset Distance", "Second Offset Distance", and "Suction Wait Time".

Pick Lift Height: User-defined height lifted after successful pickup from the pickup point;

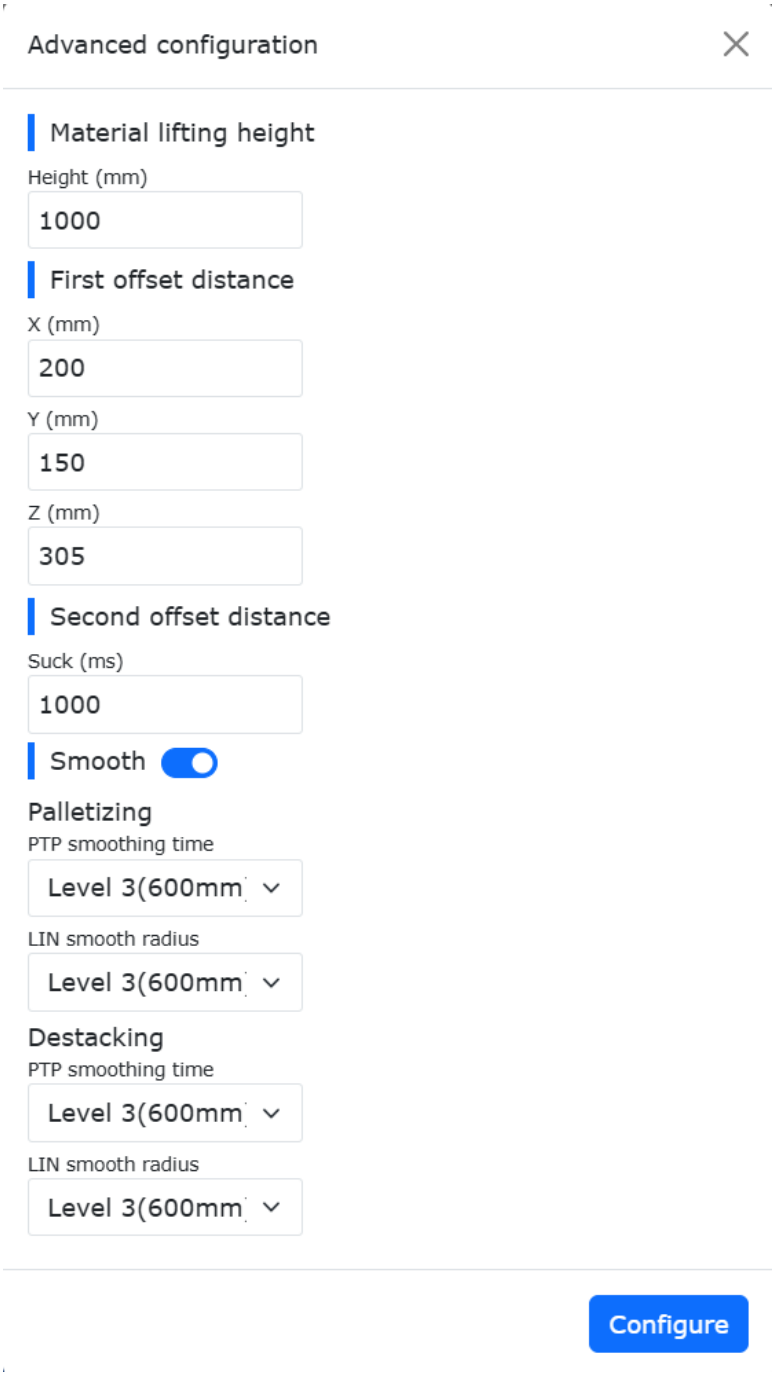
First/Second Offset Distance: User-defined offset distance for the robot to tilt and stack to the target point;

Suction Wait Time: User-defined suction wait time, monitoring the negative pressure signal after suction, repeating the suction action if not;

Smooth Transition: Turn on the Smooth Transition button to configure parameters related to Palletizing/Depalletizing PTP smooth time and LIN smooth radius.

- PTP Smooth Time: No smooth transition time / Level 1 (200ms) / Level 2 (400ms) / Level 3 (600ms) / Level 4 (800ms) / Level 5 (1000ms)
- LIN Smooth Radius: No smooth transition radius / Level 1 (200mm) / Level 2 (400mm) / Level 3 (600mm) / Level 4 (800mm) / Level 5 (1000mm)

Figure 15.2-7 Palletizing Advanced Configuration


Advanced configuration
Material lifting height
Height (mm)
1000
First offset distance
X (mm)
200
Y (mm)
150
Z (mm)
305
Second offset distance
Suck (ms)
1000
Smooth
Palletizing
PTP smoothing time
Level 3(600mm
LIN smooth radius
Level 3(600mm
Destacking
PTP smoothing time
Level 3(600mm
LIN smooth radius
Level 3(600mm
Configure

Step7: In the Teaching Program Generation section, select “Method Selection”, click “Generate Program”, open the “Palletizing Monitor Page”. On this page, you can view and check the “Generation Information”, “Alarm Information”, and “Palletizing Program”.

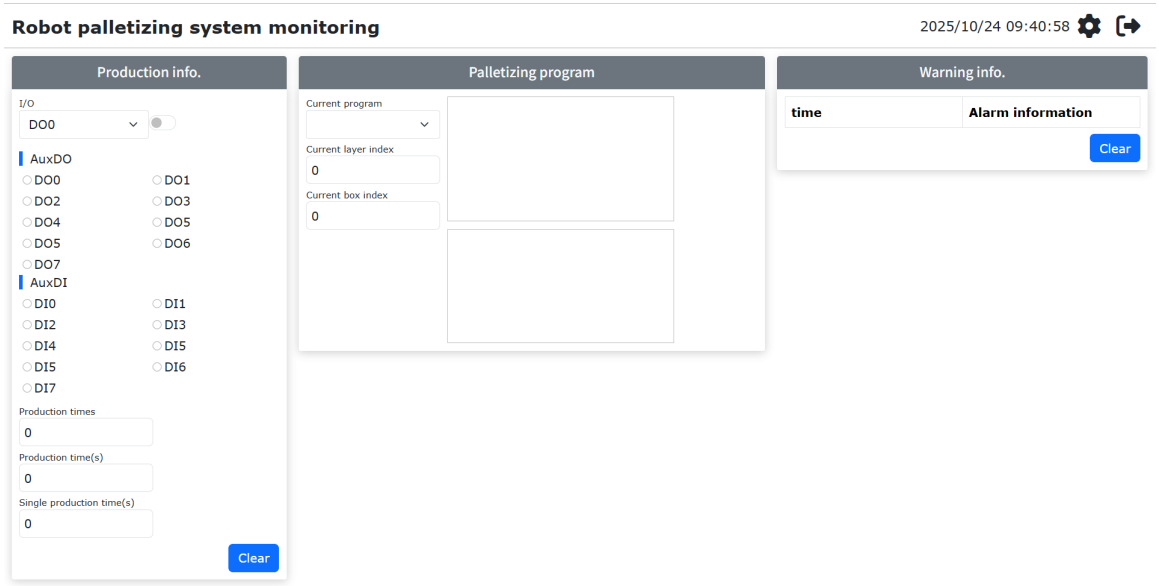


Figure 15.2-8 Palletizing System Monitoring

Step8: If the palletizing running program reports an error midway and stops, the user should first clear the error, then select the palletizing program to run again. At this time, a “Last Program Interrupted” pop-up box will appear. Click the “Resume” button to continue running, or click the “Restart” button to restart the program.

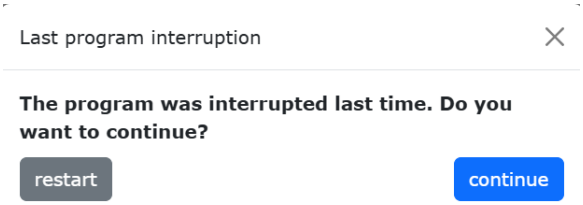


Figure 15.2-9 Palletizing Program Resume

1.3.15.3 Conveyor Tracking

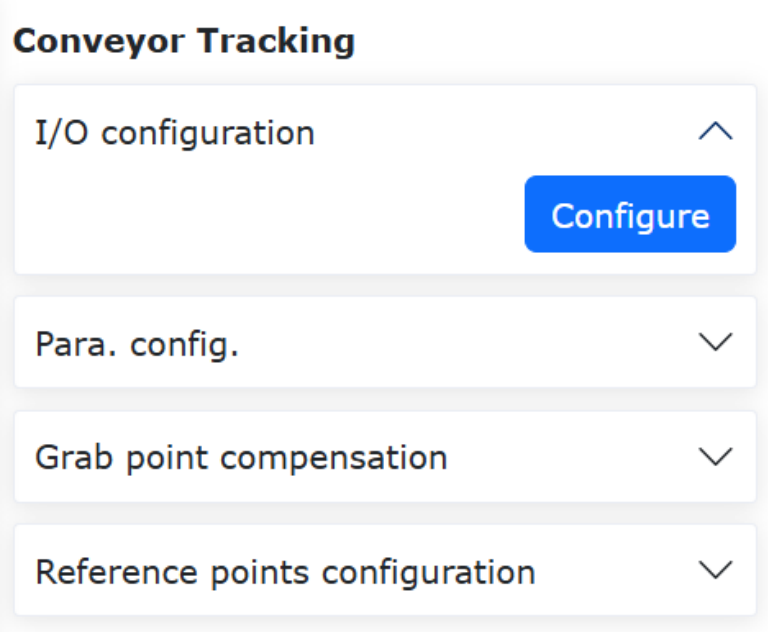
1.3.15.3.1 Conveyor Tracking Configuration Steps

Step1: Select the “Conveyor” menu item under “Auxiliary Application” -> “Process Package” to enter the Conveyor Tracking Configuration interface. Click the “Configure Conveyor IO” button to quickly configure the IO required for the conveyor function. Then, configure the corresponding parameters according to the actual function usage. Here, taking non-vision tracking picking function as an example, you need to configure the conveyor encoder channel, resolution, lead, and select “No” for vision pairing, then click Configure.

Figure 15.3-1 Conveyor Configuration

Step2: Next, set the pickup point compensation values, which are the compensation distances in the X, Y, and Z directions. These can be set during debugging based on the actual situation.

Figure 15.3-2 Conveyor Pickup Point Compensation Configuration



Step3: Start the conveyor, move the calibrated object to the defined Point A position, then stop the conveyor. Move the robot to align the tip of the calibration rod at the robot end with the tip of the calibrated object. Click the Start Point A button. A dialog box will pop up, displaying the current encoder value and robot pose. Click Calibrate to complete the Start Point A calibration.

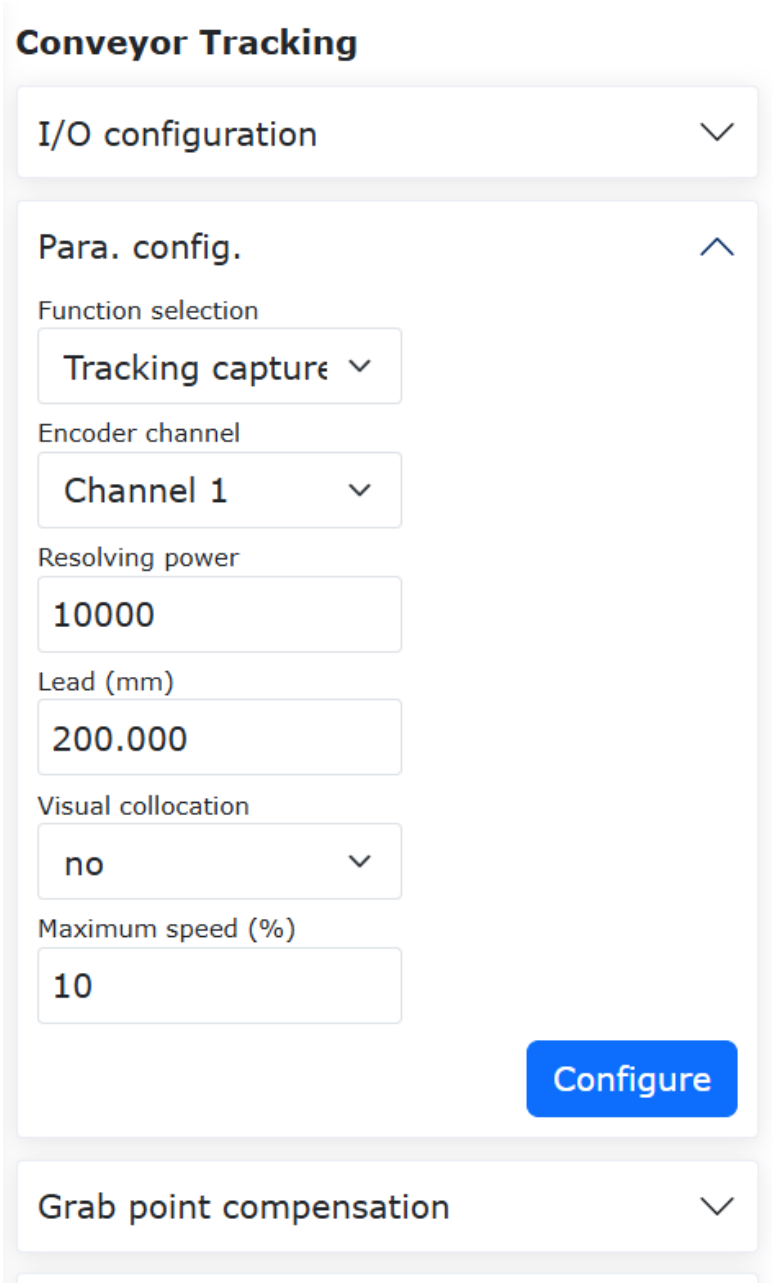
Figure 15.3-3 Start Point A Configuration

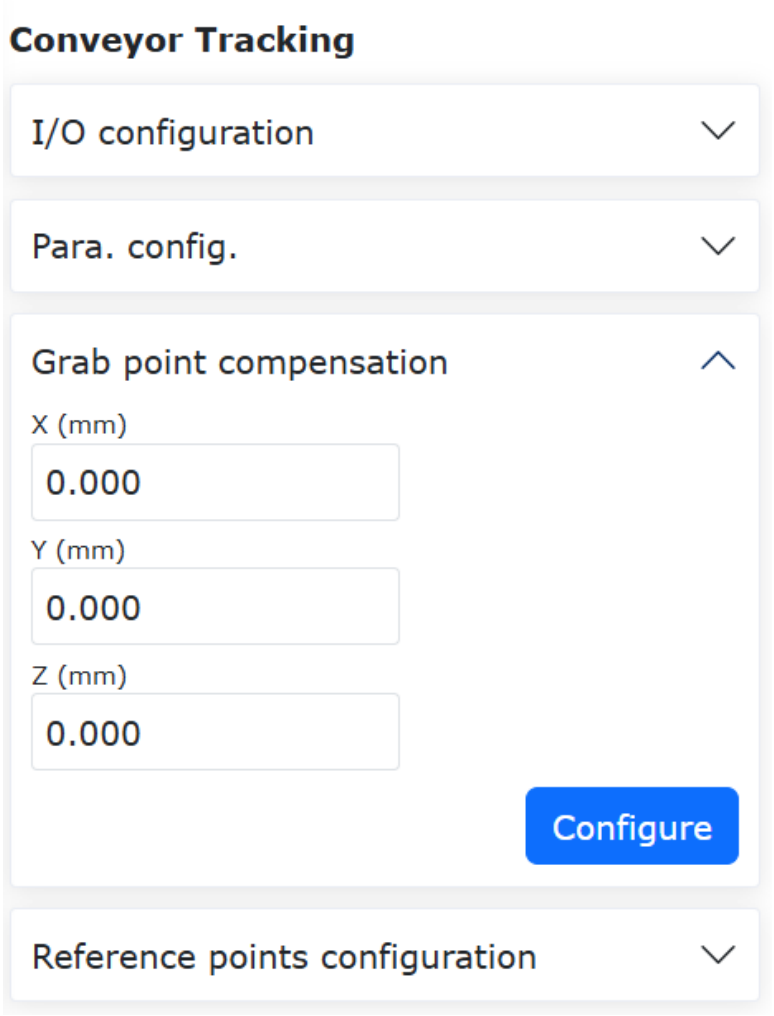
Step4: Click the Reference Point button to enter reference point calibration. When recording the reference point, record the robot's height and posture during picking. Each tracking will use the recorded reference point height and posture for tracking and picking. It can be at a different height from Points A and B. Click Calibrate to complete the reference point calibration.

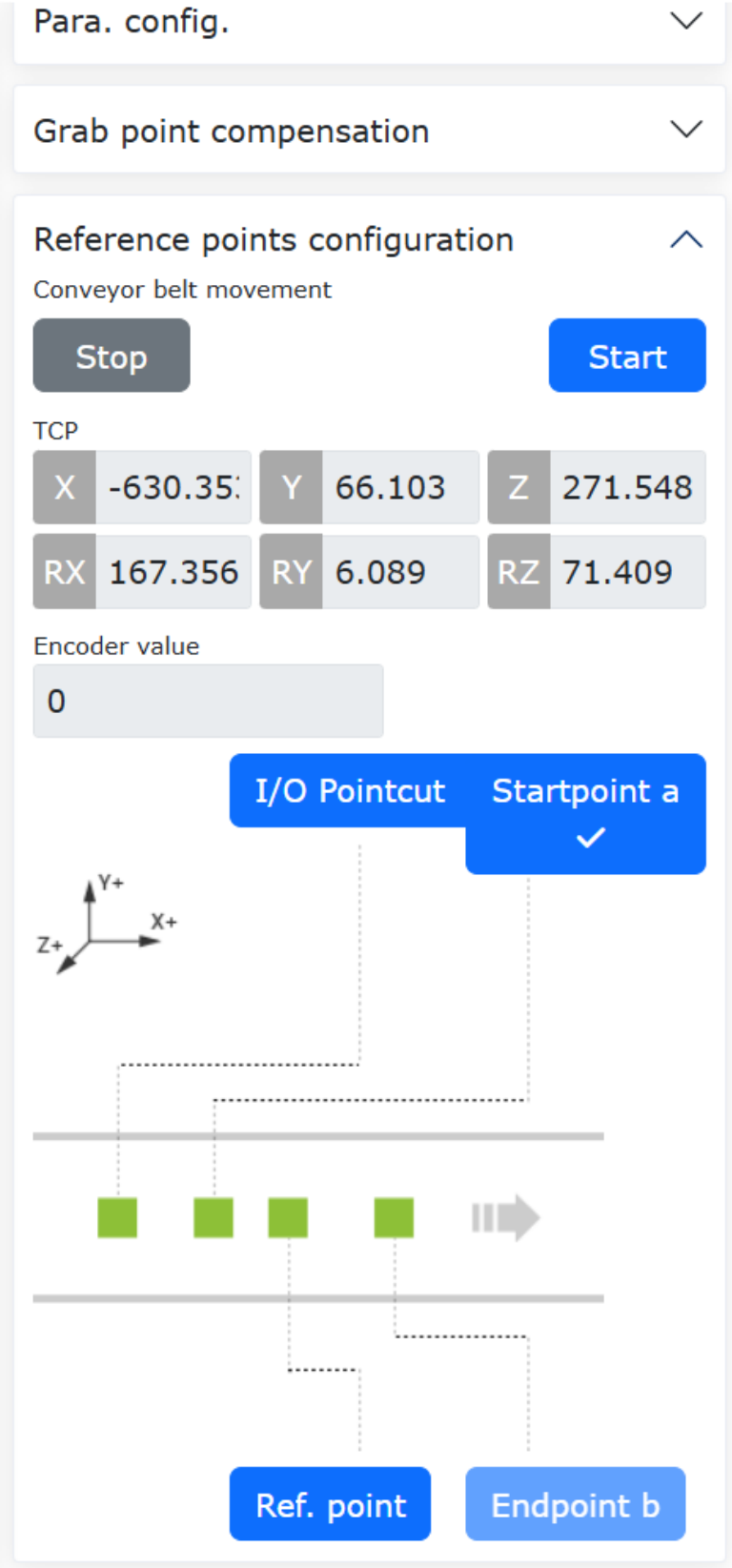
Figure 15.3-4 Reference Point Configuration

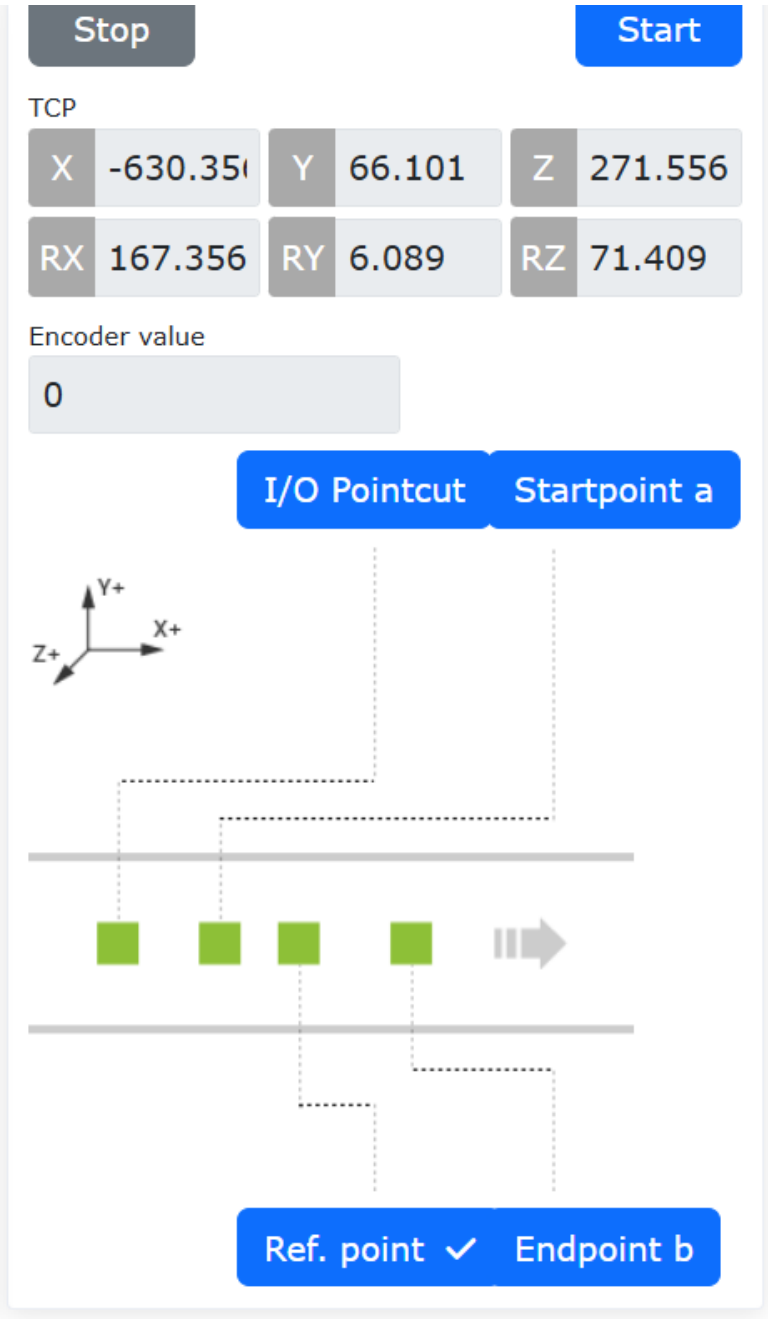
Step5: Start the conveyor, move the calibrated object to the defined Point B position, then stop the conveyor. Move the robot to align the tip of the calibration rod at the robot end with the tip of the calibrated object. Click the End Point B button. A dialog box will pop up, displaying the current encoder value and robot pose. Click Calibrate to complete the End Point B calibration.

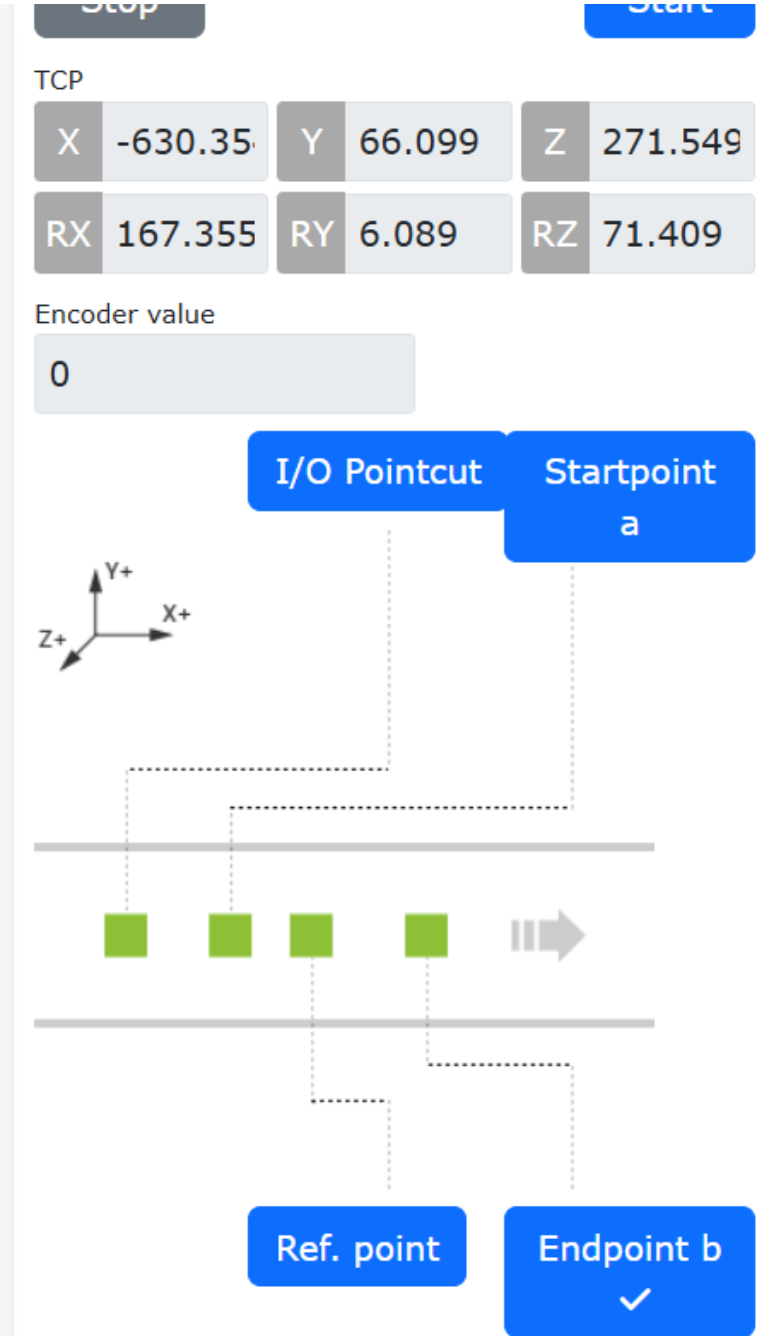
Figure 15.3-5 End Point B Configuration


Conveyor Tracking
I/O configuration
Para. config.
Function selection
Tracking capture
Encoder channel
Channel 1
Resolving power
10000
Lead (mm)
200.000
Visual collocation
no
Maximum speed (%)
10
Configure
Grab point compensation


Conveyor Tracking
I/O configuration
Para. config.
Grab point compensation
X (mm)
0.000
Y (mm)
0.000
Z (mm)
0.000
Configure
Reference points configuration


Para. config.
Grab point compensation
Reference points configuration
Conveyor belt movement
Stop
Start
TCP
X -630.35
Y 66.103
Z 271.548
RX 167.356
RY 6.089
RZ 71.409
Encoder value
0
I/O Pointcut
Startpoint a
Y+
X+
Z+
Ref. point
Endpoint b


Stop
Start
TCP
X
-630.35
Y
66.101
Z
271.556
RX
167.356
RY
6.089
RZ
71.409
Encoder value
0
I/O Pointcut
Startpoint a
Y+
X+
Z+
Ref. point
Endpoint b


TCP
X -630.35
Y 66.099
Z 271.549
RX 167.355
RY 6.089
RZ 71.409
Encoder value
0
I/O Pointcut
Startpoint a
Y+
X+
Z+
Ref. point
Endpoint b

1.3.15.3.2 Conveyor Tracking Teaching Program

No.	Command Format	Comment
1	PTP(conveyorstart,30,-1,0)	#Robot move to pick start point
2	While(1) do	#Loop picking
3	ConveyorIODetect(10000)	#IO real-time object detection
4	ConveyorGetTrackData(1)	#Object position acquisition
5	ConveyorTrackStart(1)	#Conveyor tracking start
6	Lin(cvrCatchPoint,10,-1,0,0)	#Robot reach pickup point
7	MoveGripper(1,255,255,0,10000)	#Gripper pick object
8	Lin(cvrRaisePoint,10,-1,0,0)	#Robot lift up
9	ConveyorTrackEnd()	#Conveyor tracking end
10	PTP(conveyorraise,30,-1,0)	#Robot reach wait point
11	PTP(conveyorend,30,-1,0)	#Robot reach place point
12	MoveGripper(1,0,255,0,10000)	#Gripper release object
13	PTP(conveyorstart,50,-1,0)	#Robot return to pick start point, wait for next pick
14	end	#End

1.3.15.3.3 Robot Conveyor Tracking System Composition

1.3.15.3.3.1 Conveyor Encoder Data Communication Connection Method

For machine tool processing, to achieve automated loading and unloading processes, a CNC function package based on FOCAS communication has been developed, enabling communication interaction and coordinated motion between collaborative robots and CNC machine tools.

As shown in the figure, FOCAS communication is based on Ethernet. By connecting the robot control box network port with the machine tool's embedded network port via an Ethernet cable, FOCAS communication between the robot and the machine tool can be established, enabling CNC control and machine tool status monitoring on the robot side.

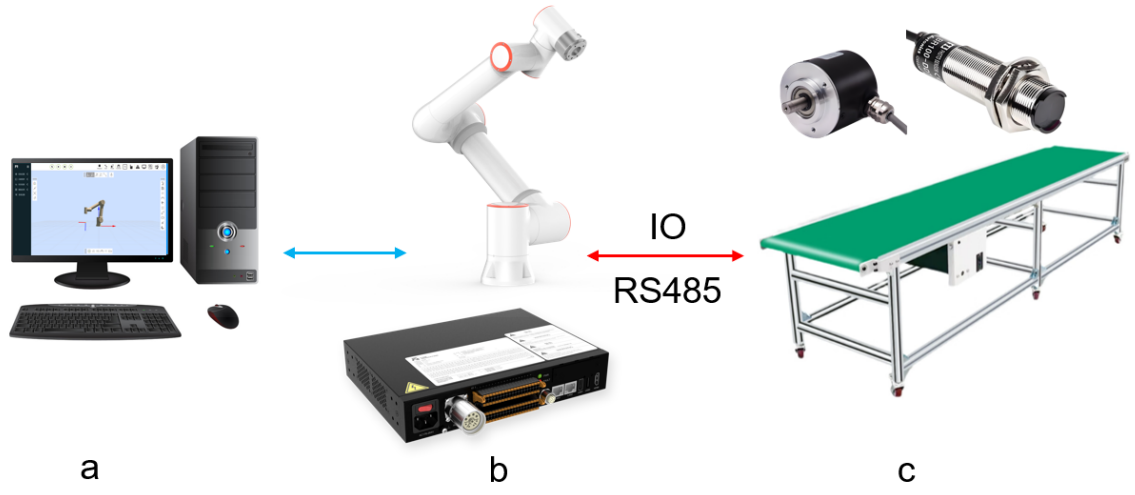


Figure 15.3-6 Robot Conveyor Tracking System Composition Topology

In the system, (a) is the computer, (b) is the robot and its control box, (c) is the conveyor system consisting of the conveyor belt, photoelectric sensor, and encoder. The robot control box is connected to the photoelectric sensor and

conveyor belt via digital IO communication, and connected to the conveyor encoder via RS485.

1.3.15.3.3.2 Conveyor Configuration

Enter the Conveyor Tracking function configuration interface under “Basic Settings”, “Peripherals”, “Tracking” on the robot Web page to configure the conveyor tracking function properties.

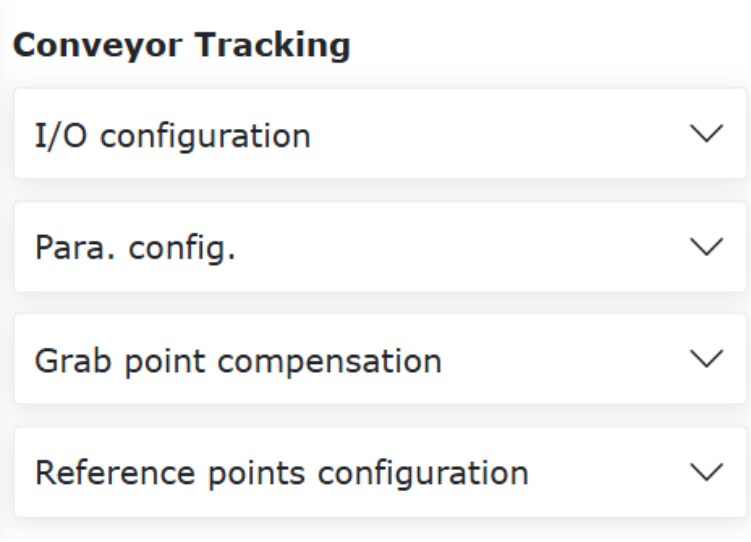


Figure 15.3-7 Conveyor Tracking Configuration Page

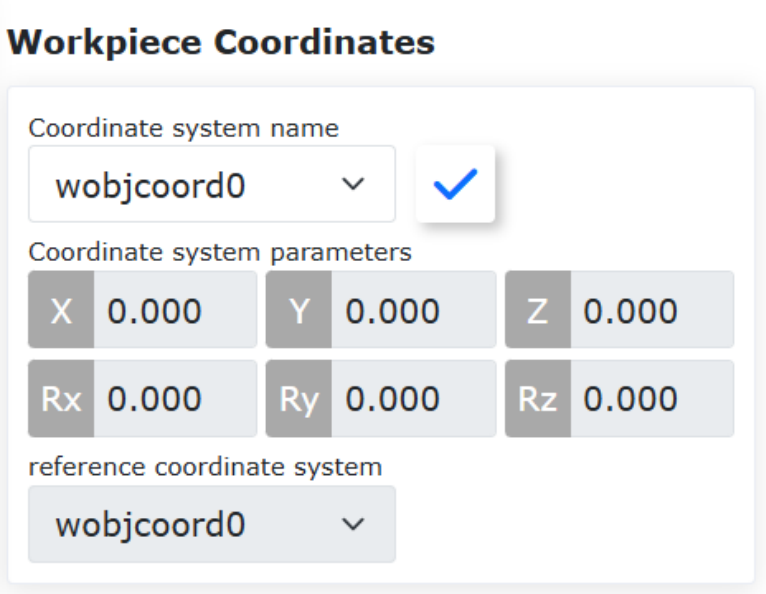
On the conveyor tracking configuration page, click the “Conveyor I/O One-Key Configuration” button to configure the conveyor physical connection with one click. Then, in the “Function Selection” dropdown under “Parameter Configuration”, select “Tracking Motion”. Subsequently, configure the encoder properties, tracking workpiece coordinate system workpiece axis, vision pairing, and select “Chase Motion” in the “Tracking Type” dropdown. Then you can input the tracking start distance and tracking end distance. Tracking Start Distance: After the tracking signal is triggered, the robot starts action after the conveyor runs the set distance. When set to -1, it triggers automatically. Tracking End Distance: The maximum distance the robot follows the conveyor in synchronous motion after starting action.

1.3.15.3.3.3 Tracking Coordinate System Configuration

Tracking motion uses the workpiece coordinate system as the conveyor coordinate system, so the workpiece coordinate system needs to be set.

Click “Initial Setup”, “Basic”, select “Workpiece Coordinate System” under “Coordinate System”, click to select a workpiece coordinate system other than “wobjcoord0” for calibration. The calibration method is not detailed here.

Figure 15.3-8 Tracking Coordinate System Setup



1.3.15.3.3.4 Conveyor Tracking Chase Motion Function

Chase motion is a type of conveyor tracking motion. Compared to tracking motion, the teaching of motion points for chase motion does not need to be done above the workpiece coordinate system; it can be taught at any position in the workpiece coordinate system. Then, through the "Tracking Start Distance" parameter, the end effector synchronizes motion with the conveyor, making it a more flexible tracking method.

1.3.15.3.3.5 Conveyor Tracking Chase Motion Function Introduction

The following gives a chase motion example to introduce its motion characteristics.

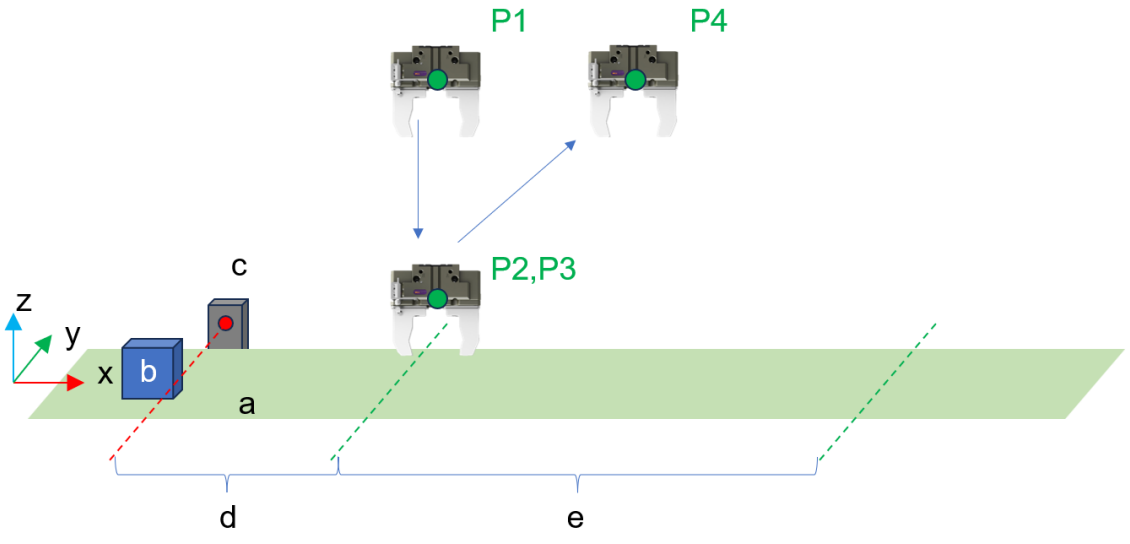


Figure 15.3-9 Conveyor Tracking Chase Motion Teaching Example

Here, x is the direction of conveyor motion in the workpiece coordinate system, a is the conveyor plane, b is the target workpiece to be picked, c is the photoelectric sensor, d is the tracking start distance, e is the tracking end distance. P1

to P4 are the taught waypoints in sequential order, P2 to P3 are the same waypoint, including gripper motion.

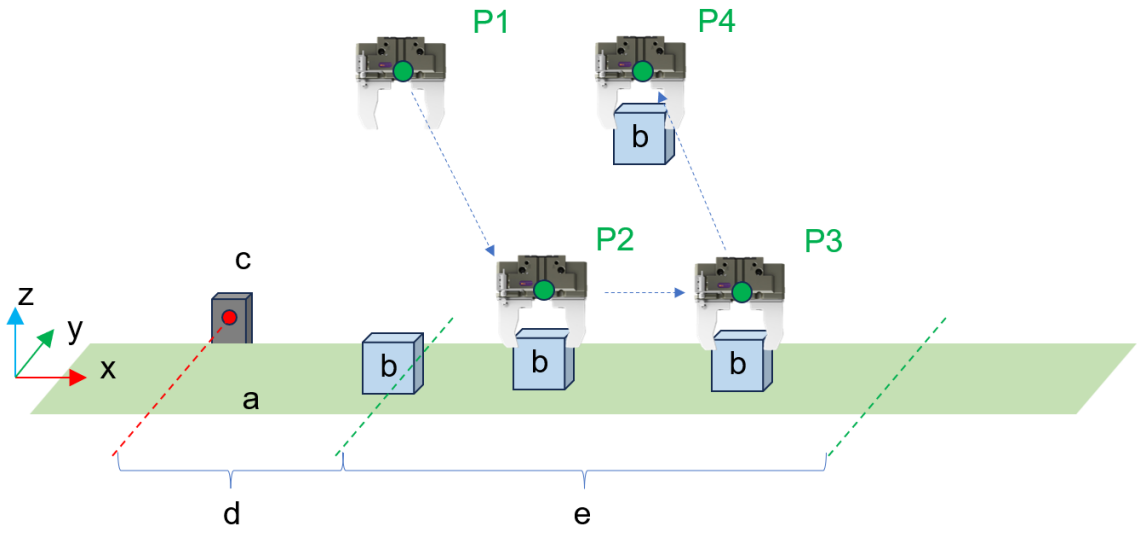


Figure 15.3-10 Conveyor Tracking Chase Motion Execution Example After Teaching

When the above taught program starts running and the workpiece triggers the photoelectric switch signal, the robot will wait for the target to move under P1 before starting the tracking motion. The robot gripper will move along the trajectory shown in the figure above.

1.3.15.3.3.6 Chase Motion Program Teaching

The chase motion program logic is basically the same as the tracking motion logic, including acquiring the trigger signal, acquiring conveyor data, and starting the tracking motion.

Step 1: Click "Teaching Program", "Program Programming", select and click the "Conveyor" button under "Peripheral Instructions" to enter the conveyor instruction configuration page.

Figure 15.3-11 I/O Real-time Monitoring Instruction

Step 2: Click "I/O Real-time Monitoring" and set the "Max Wait Time (ms)" to detect the tracking trigger signal in real-time. Click "Add" and "Apply" to add the instruction to the program.

Figure 15.3-12 Position Real-time Detection Instruction

Step 3: Click "Position Real-time Detection" and select "Tracking Motion" for the working mode. Click "Add" and "Apply" to add the instruction to the program.

Figure 15.3-13 Tracking Start Instruction

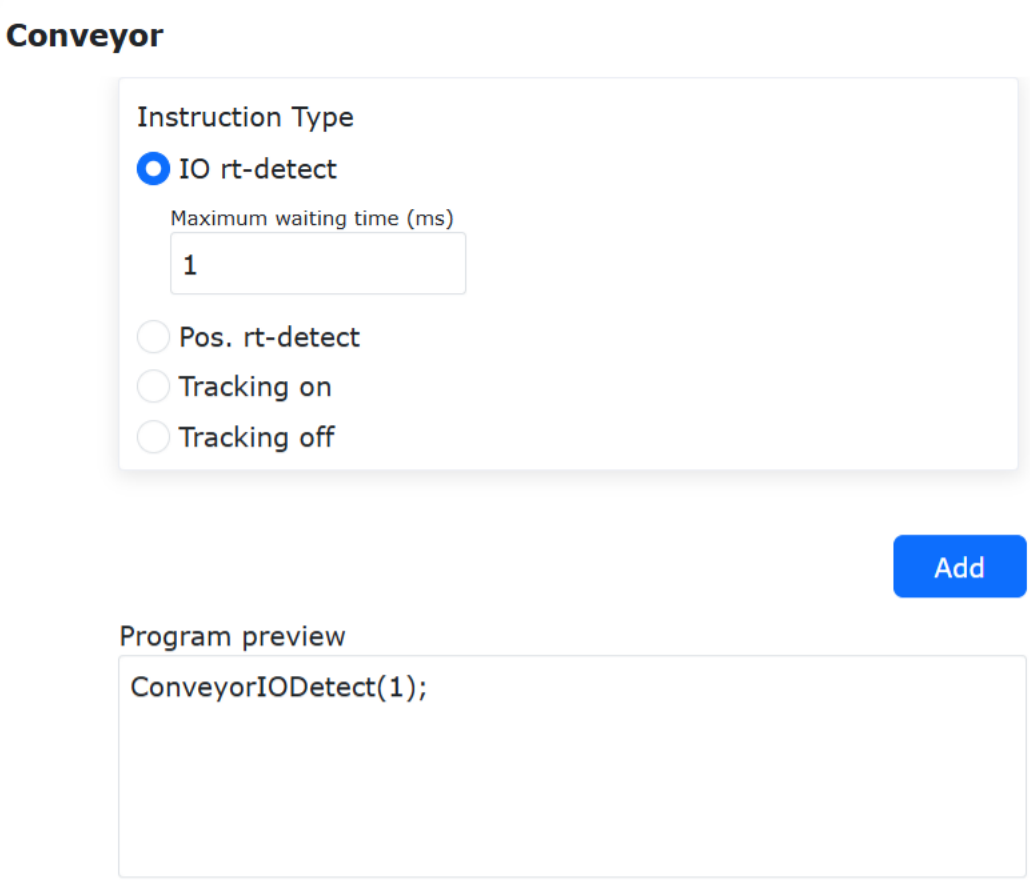
Step 4: Click "Tracking Start" and select "Tracking Motion" for the working mode. Click "Add" and "Apply" to add the instruction to the program.

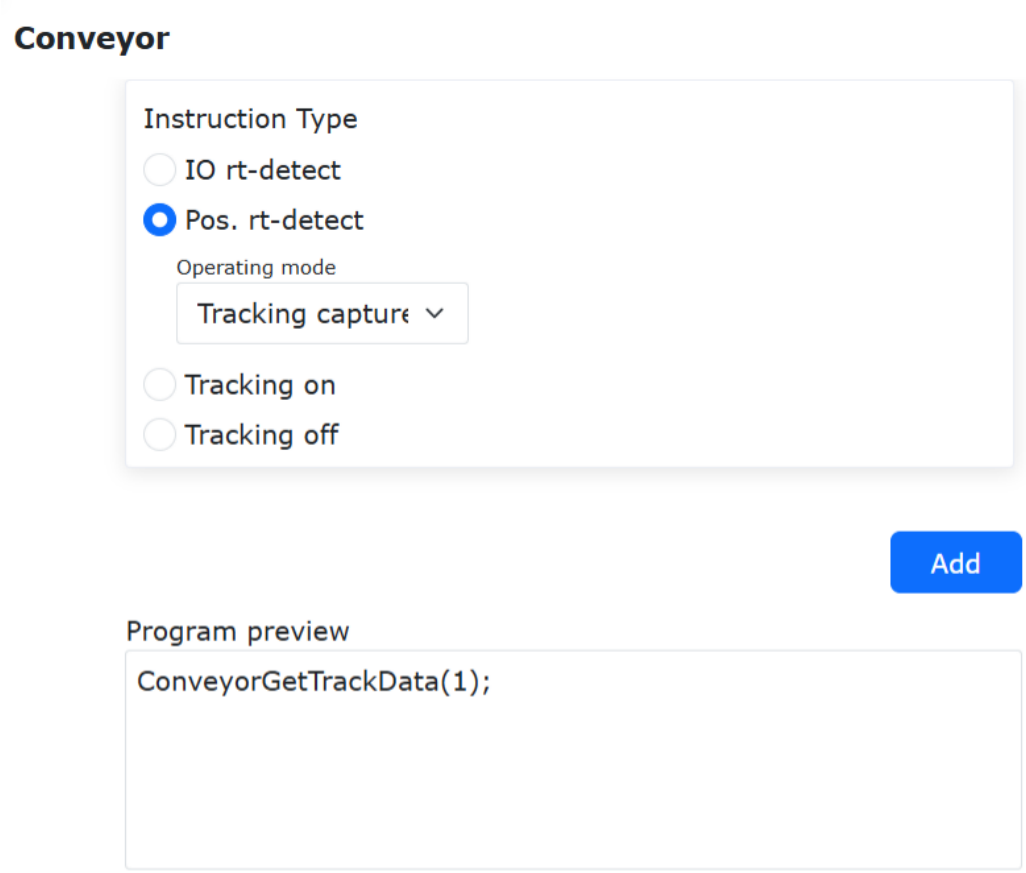
Step 5: Teach the Cartesian space motion after tracking starts and the gripper peripheral motion. During the motion process, it will maintain synchronous tracking motion with the conveyor.

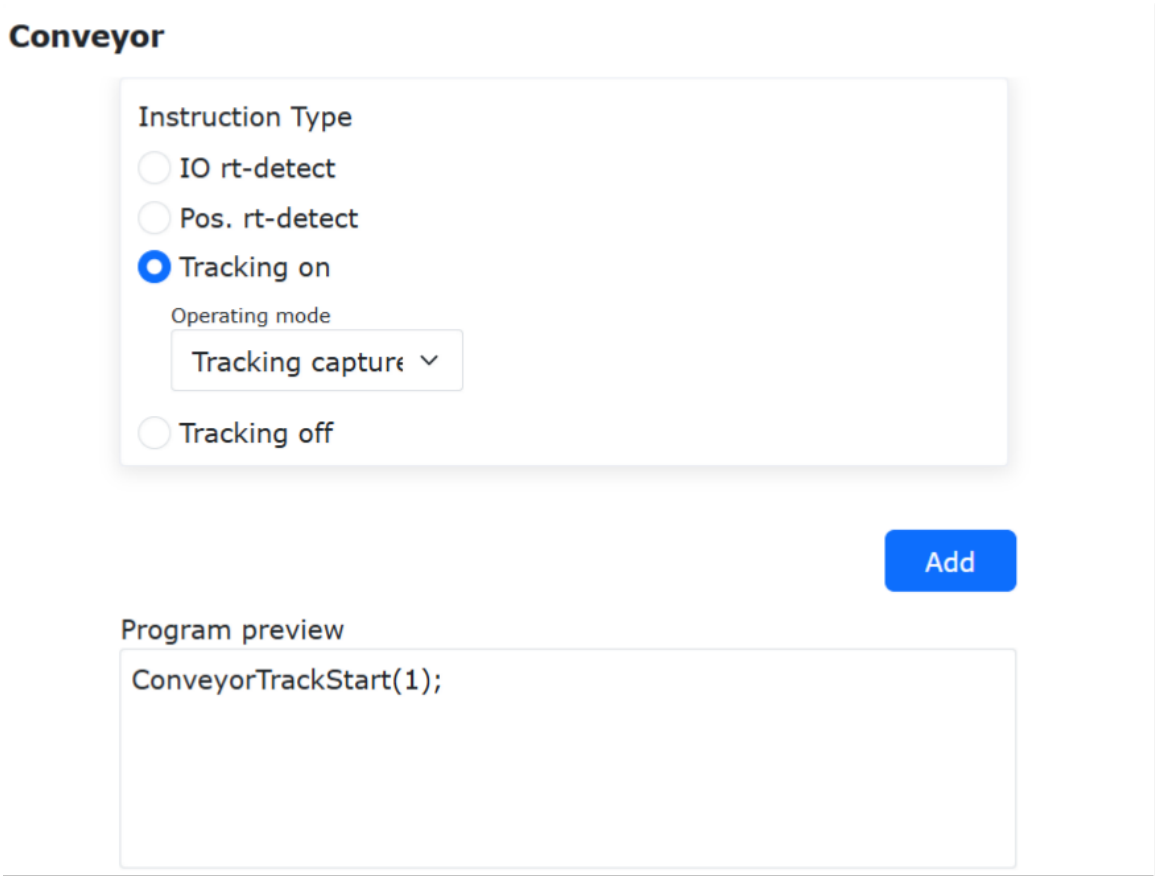
Figure 15.3-14 Tracking Stop Instruction

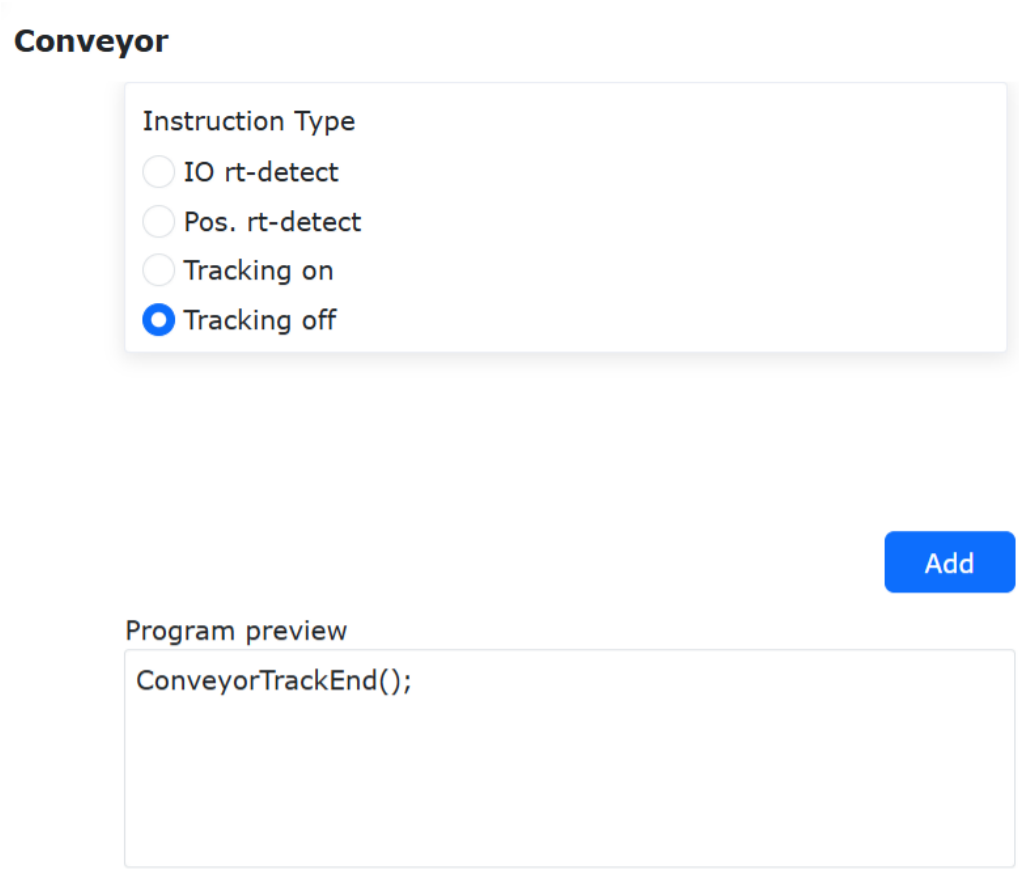
Step 6: Click "Tracking Stop" and click "Add" and "Apply" to add the instruction to the program.

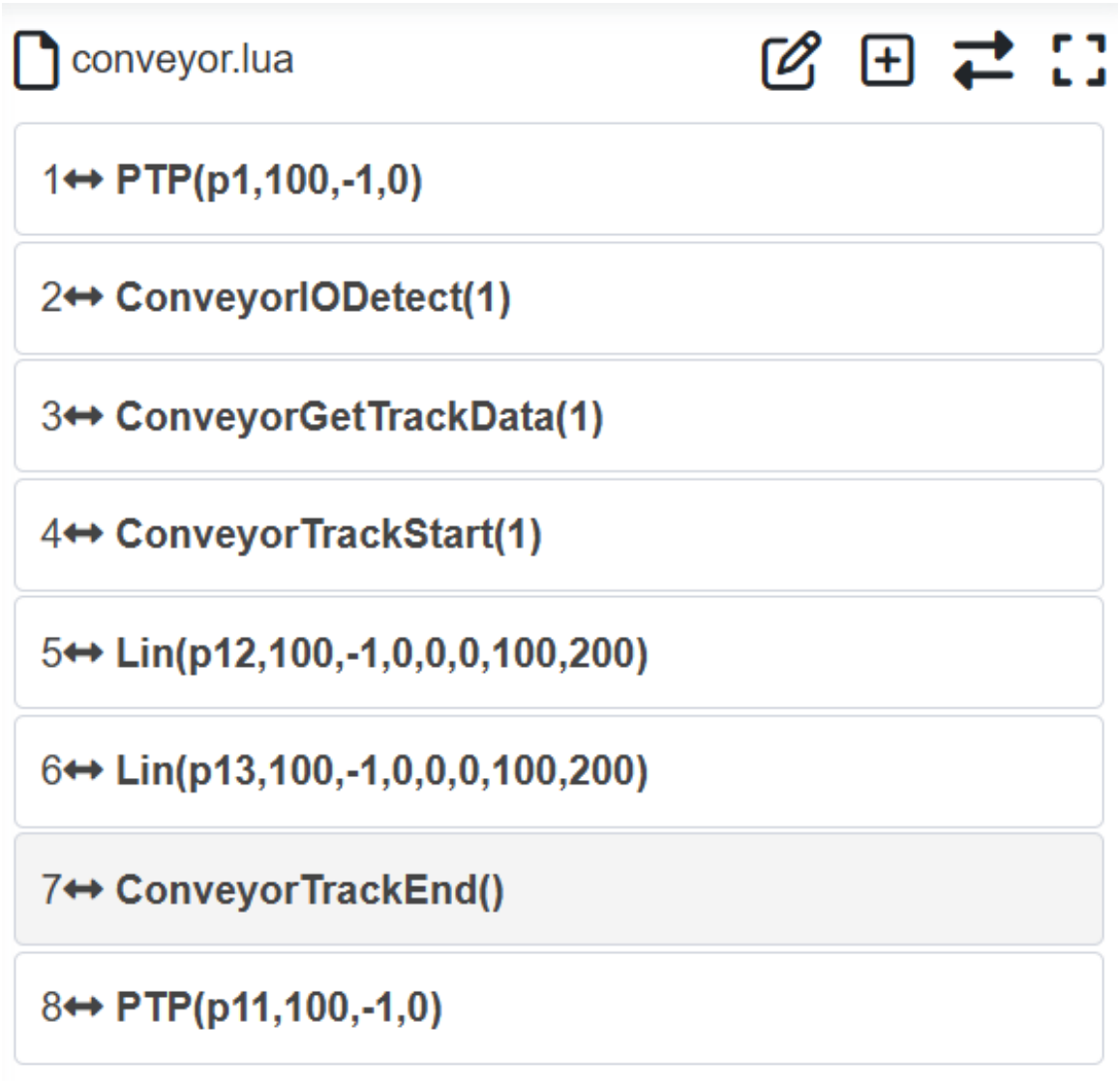
Figure 15.3-15 A Typical Conveyor Program Tracking Motion Program


Conveyor
Instruction Type
IO rt-detect
Maximum waiting time (ms)
1
Pos. rt-detect
Tracking on
Tracking off
Add
Program preview
ConveyorIODetect(1);


Conveyor
Instruction Type
IO rt-detect
Pos. rt-detect
Operating mode
Tracking capture
Tracking on
Tracking off
Add
Program preview
ConveyorGetTrackData(1);


Conveyor
Instruction Type
IO rt-detect
Pos. rt-detect
Tracking on
Operating mode
Tracking capture
Tracking off
Add
Program preview
ConveyorTrackStart(1);


Conveyor
Instruction Type
IO rt-detect
Pos. rt-detect
Tracking on
Tracking off
Add
Program preview
ConveyorTrackEnd();


conveyor.lua
1↔ PTP(p1,100,-1,0)
2↔ ConveyorIODetect(1)
3↔ ConveyorGetTrackData(1)
4↔ ConveyorTrackStart(1)
5↔ Lin(p12,100,-1,0,0,0,100,200)
6↔ Lin(p13,100,-1,0,0,0,100,200)
7↔ ConveyorTrackEnd()
8↔ PTP(p11,100,-1,0)

When two identical tracking motion targets are taught consecutively (may include offset distance), the robot motion will block at this target position, achieving continuous synchronous tracking until the tracking distance reaches the stop tracking distance.

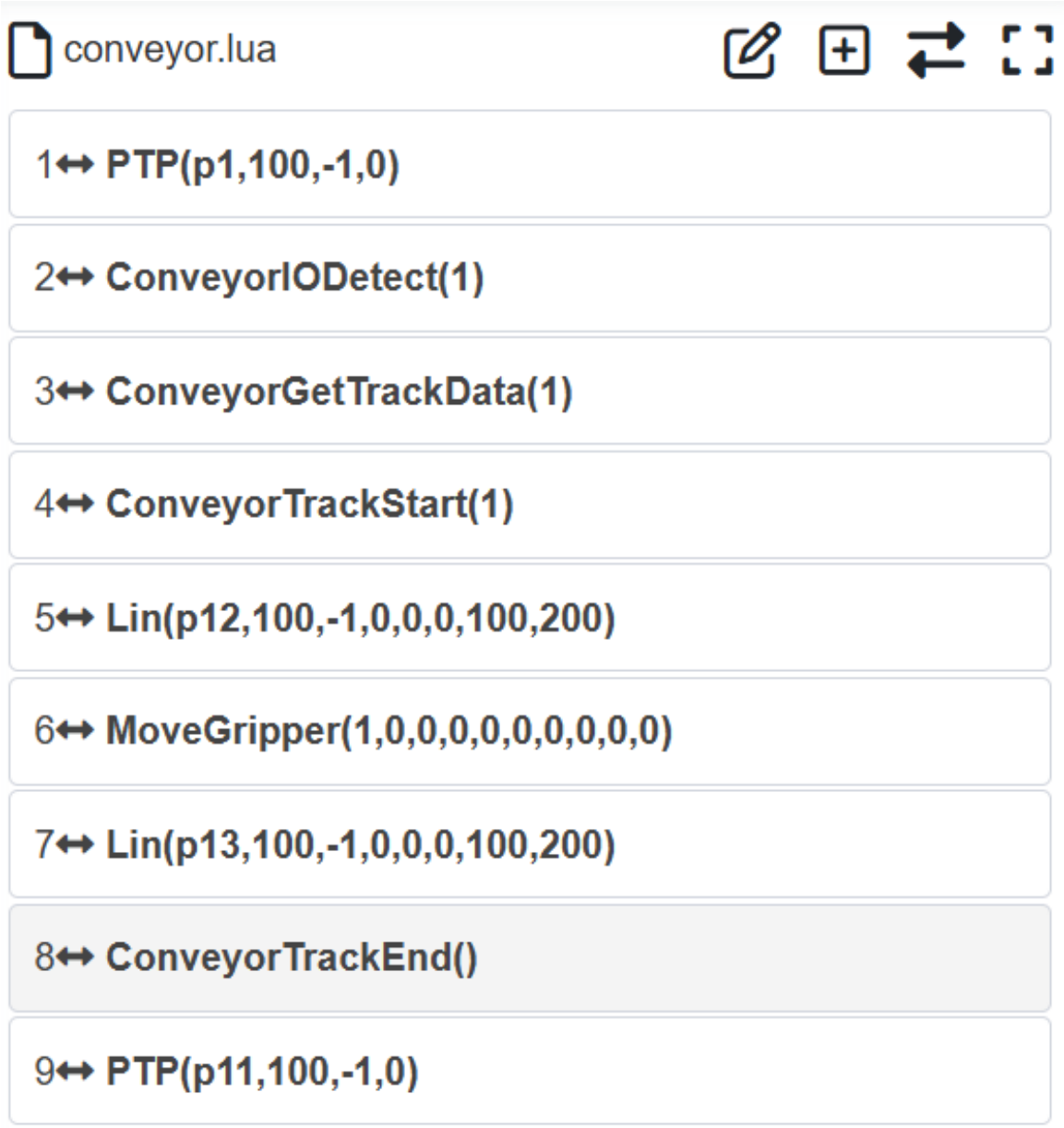


Figure 15.3-16 A Typical Conveyor Blocking Tracking Picking Motion Program

When two identical tracking motion targets are taught consecutively (may include offset distance), and gripper motion is inserted in between, the robot will continuously track the conveyor at this target position until the gripper motion is completed, achieving blocking tracking picking.

1.3.16 System

1.3.16.1 General settings

Click “System Settings” on the left menu bar, and then click “General Settings” on the secondary menu bar to enter the general settings interface. General settings can update the robot system time according to the current computer time so that the time of the log content can be recorded accurately.

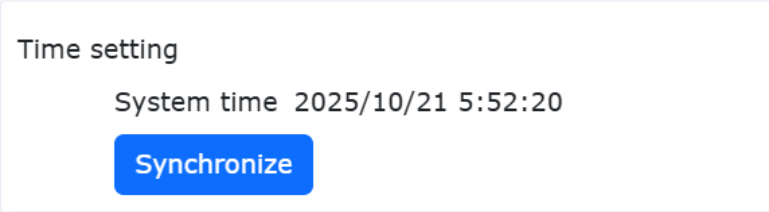


Figure 15.1-1 Time Settings

Network settings can set the controller IP, subnet mask, default gateway, DNS server and teach pendant IP (this IP is valid when using our FR-HMI teach pendant, and the teach pendant enable status needs to be configured to enable when using the FR-HMI teach pendant), which is convenient for customers to use the scene.

1.3.16.1.1 Network settings

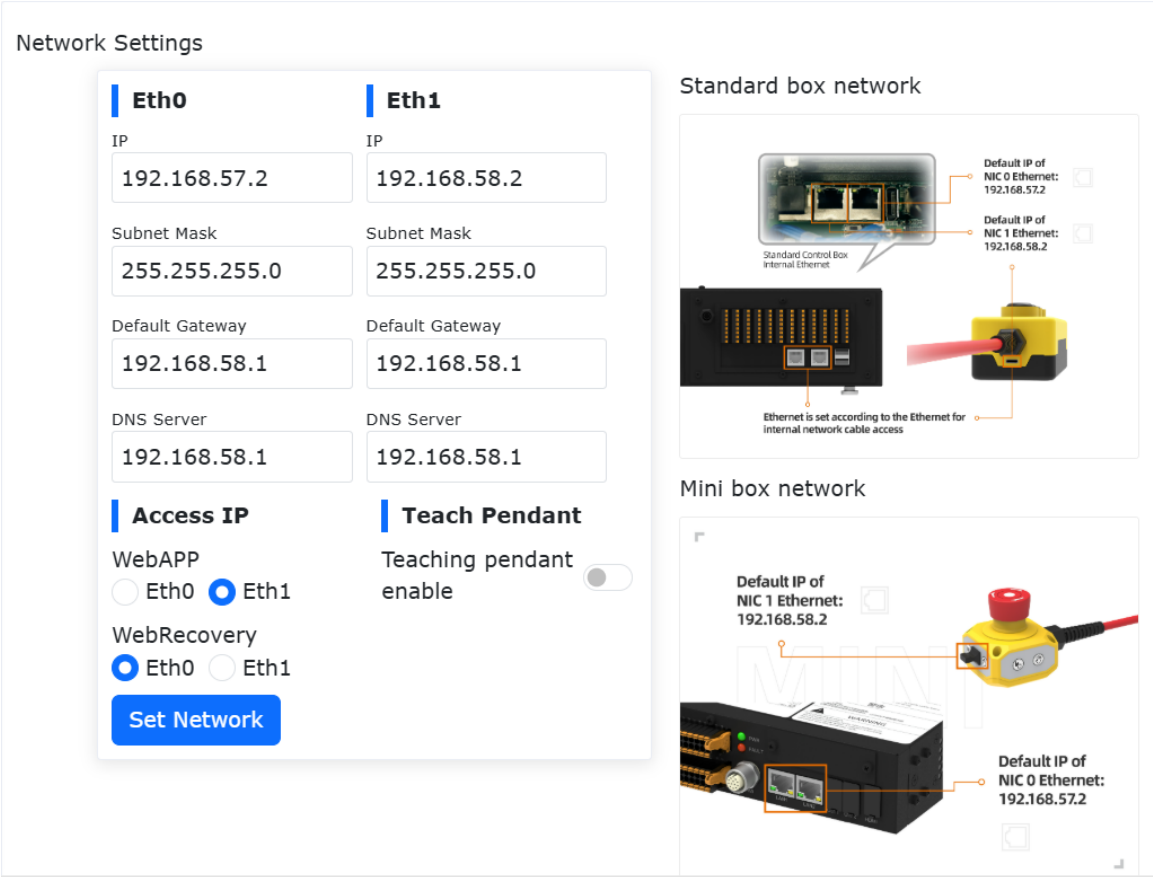


Figure 15.1-2 Network settings diagram

- **Set network card**: Enter the network card IP, subnet mask (linked with IP, automatically filled in), default gateway, and DNS server that need to communicate. The factory default IP of network card 0 network port is: 192.168.57.2, and the factory default IP of network card 1 network port is: 192.168.58.2.
- **Teach pendant enable**: Control whether to enable the teach pendant. The teach pendant is turned off by default, and the device cannot be operated using the teach pendant. Click the slide switch button to enable the teach pendant to operate the device.
- **Access IP**: Select the network card associated with WebAPP and WebRecovery. When the teach pendant is enabled, WebAPP selects network card 1 by default, and network card 0 is not selectable.
- **Set up network**: Click the "Set up network" button, and it will prompt that the configuration is in progress. After the configuration is complete, you need to restart the device.

1.3.16.1.1.1 No-Login Operation

1.3.16.1.1.2 Functional Overview

After enabling the No-Login Operation feature on the physical teach pendant, the following functions can be achieved:

- When no user is logged into the teach interface, rotating the physical key switch allows the robot to switch between manual/automatic modes, with the end-effector light color changing accordingly.
- When no user is logged into the teach interface, in automatic mode, pressing the physical start switch allows the robot to begin running the currently loaded program.
- When no user is logged into the teach interface, in automatic mode, pressing the physical stop switch allows the robot to stop running.

1.3.16.1.1.3 Instructions for Use

Log in to the webapp page, click "System Settings," then click "General Settings." In the Teach Pendant section under the Network module, turn on the "Teach Pendant Enable" switch and the "No-Login Operation" switch. Once the feature is enabled, you can control the robot's manual/automatic mode switching and program start/stop using the physical buttons without being logged into the teach pendant page. This configuration is retained after a power restart.

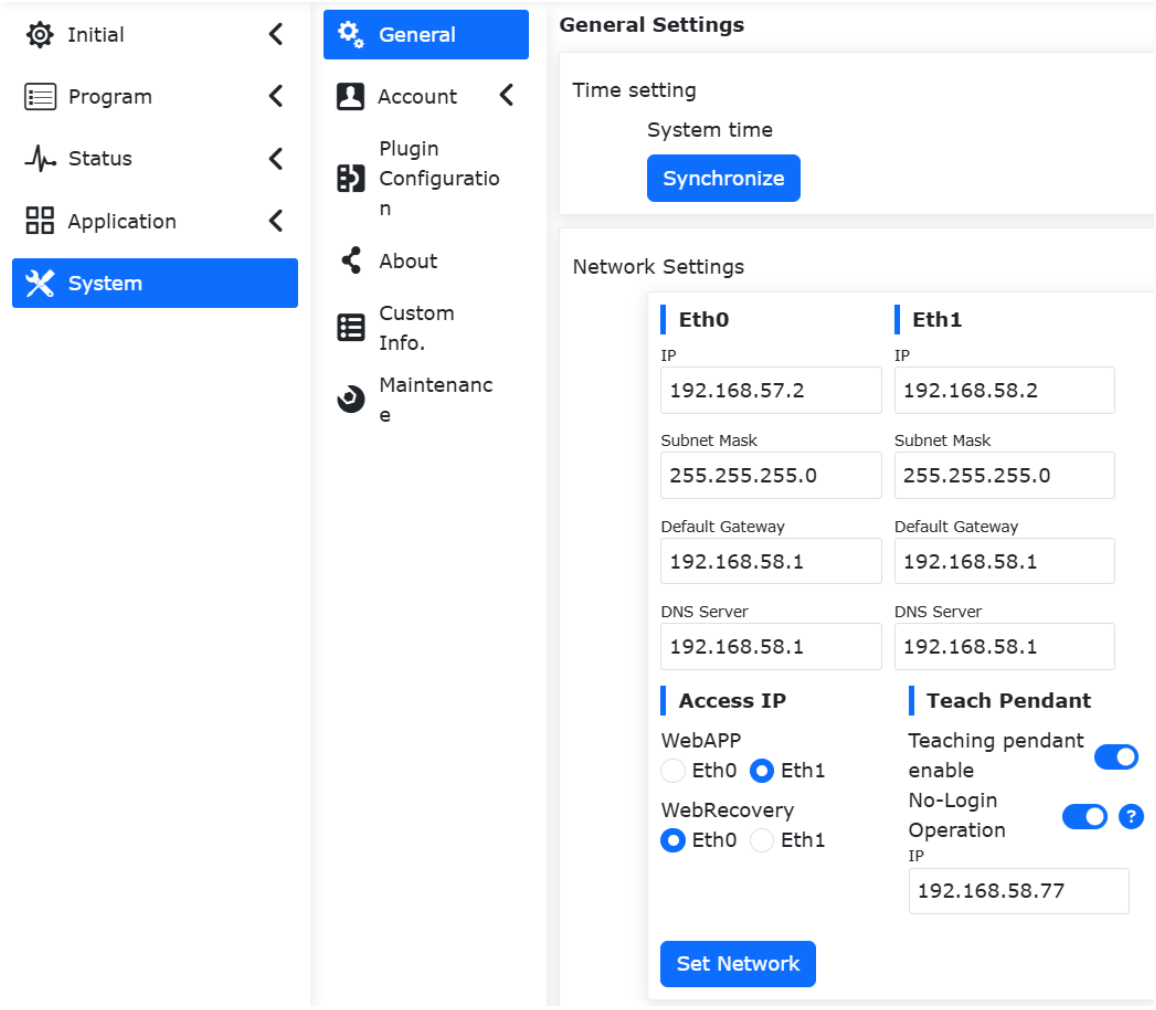


Figure 15.1-2-1 Enabling the No-Login Operation Function

1.3.16.1.2 Teach pendant touch screen calibration

After enabling the teach pendant, you can calibrate the teach pendant.

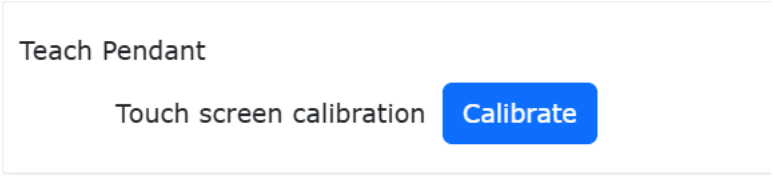


Figure 15.1-3 Teach pendant touch screen calibration

1.3.16.1.3 Peripheral industrial computer configuration

To enable the peripheral industrial computer, you need to enter the IP address. After successful configuration, you need to restart the control box and industrial computer.

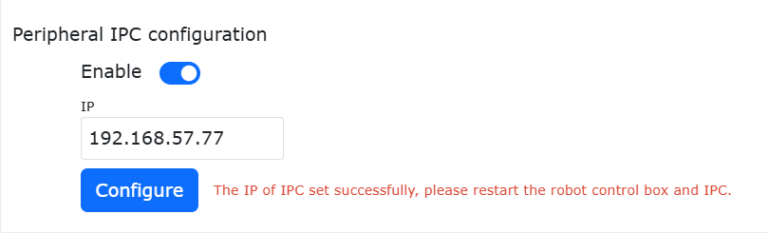


Figure 15.1-4 Peripheral IPC Configuration

1.3.16.1.4 System Language

1.3.16.1.4.1 Language Import

Select a language pack to import (Note: the import file format is [language code].json). If the import is successful and the language pack is not an existing language in the system, a new imported language pack data will be added to the system language.

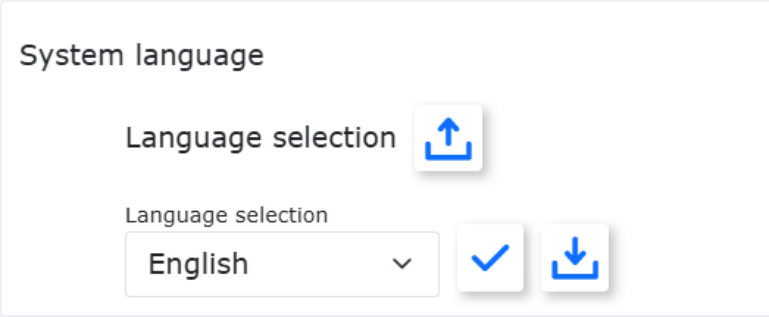


Figure 15.1-5 System language interface

1.3.16.1.4.2 Language Export

Select the system language, taking English as an example, click the “Export” button, and the exported download file will pop up on the page.

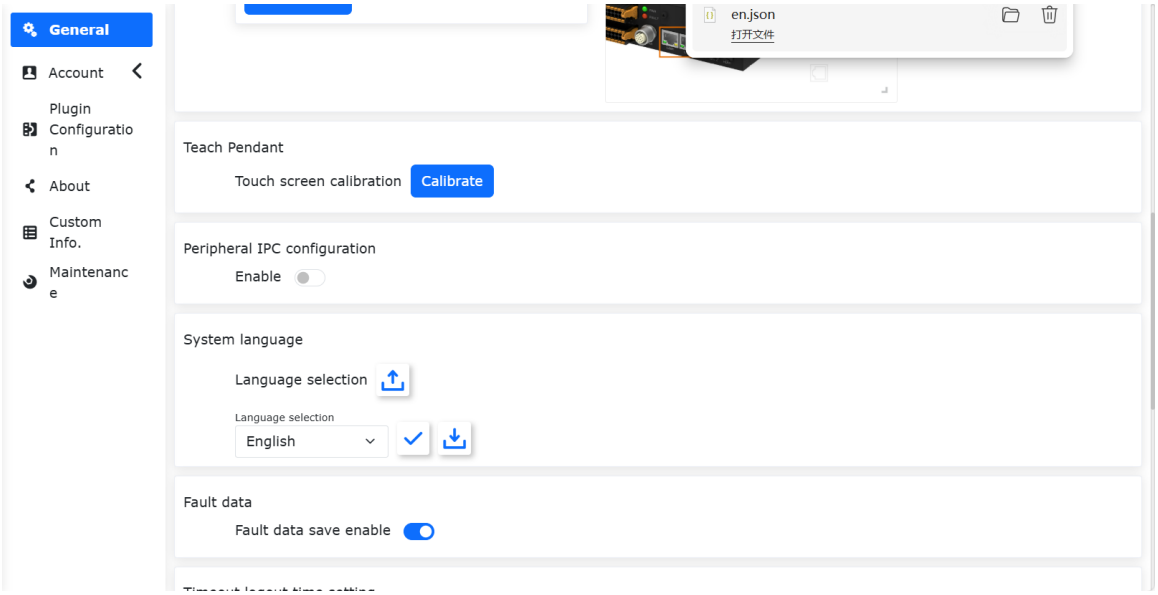


Figure 15.1-6 Export system language

1.3.16.1.4.3 Language Application

Select the system language and click the “Apply” button to switch the system language. After the language is applied successfully, the system will automatically log out to the login page, and the system language will be switched to the current language. Take English as an example:

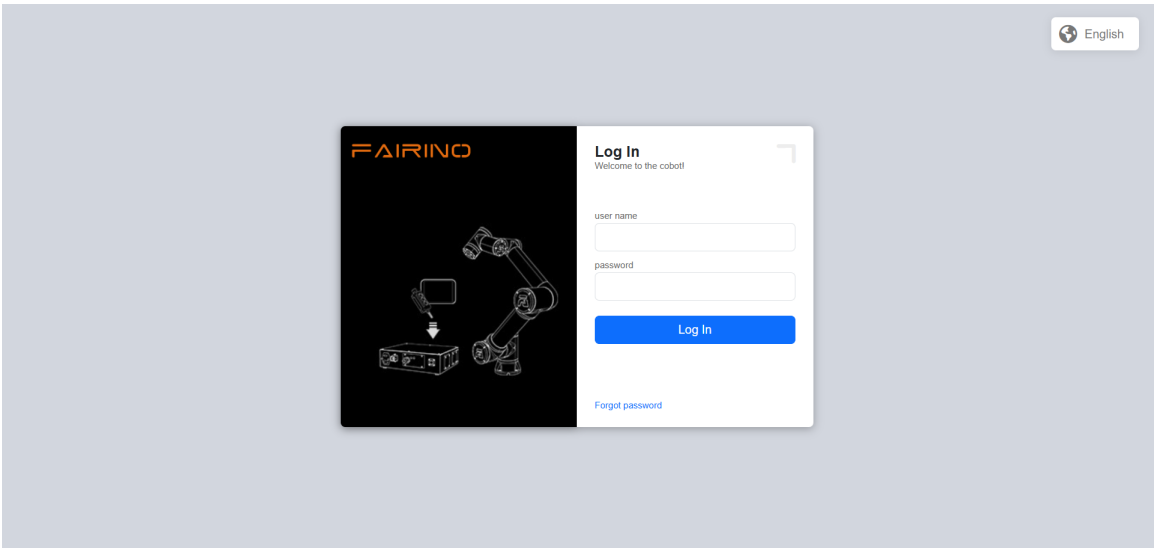


Figure 15.1-7 Language system successful interface

1.3.16.1.4.4 System safe mode recovery

When the system needs to upgrade or downgrade the version, or the system language pack import fails and the system cannot be entered normally, you need to enter the "System Security Mode Recovery" interface. The specific operations are as follows: 1. Enter the System Settings -> General Settings -> Network Settings interface, adjust the WebRecovery access IP to the network card 0 position, and click "Set Network".

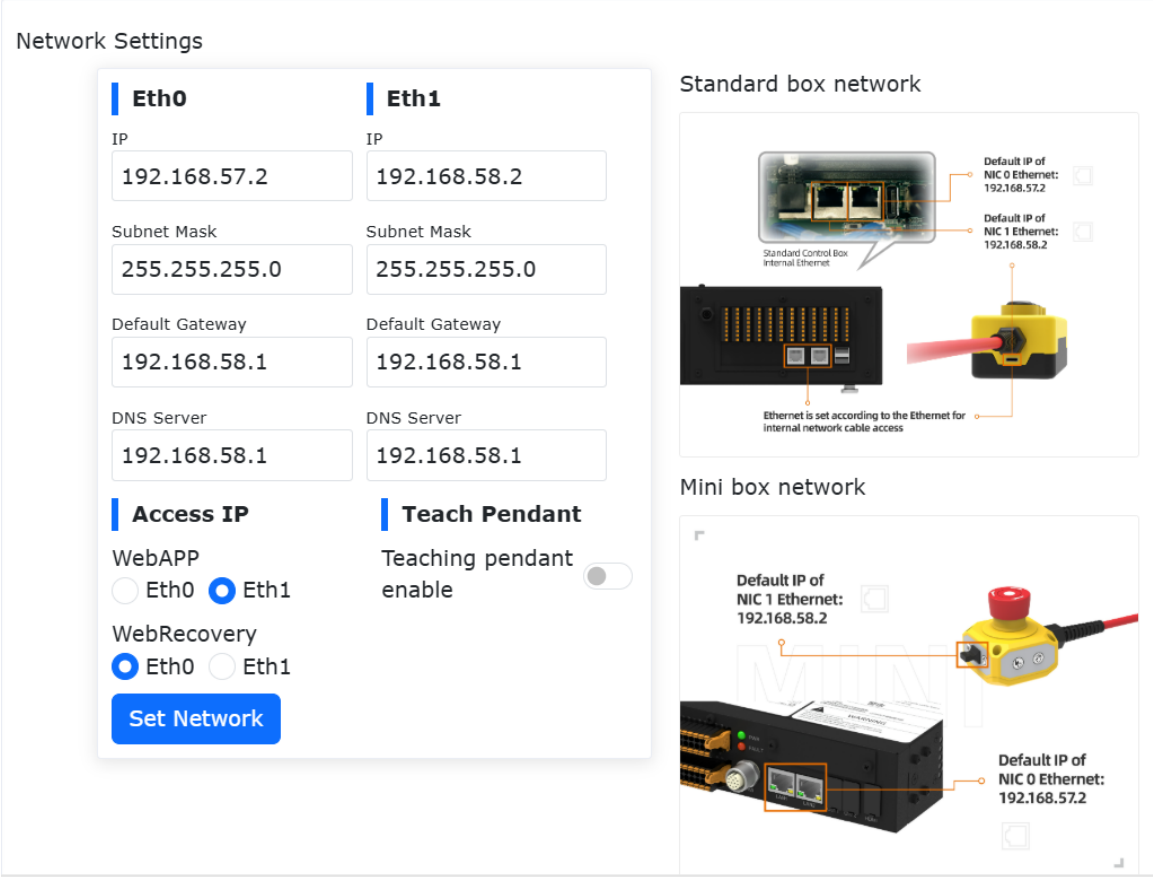


Figure 15.1-8 WebRecovery set up the network card interface

2. After the network settings are successful, restart the control box, switch the IP address to 192.168.57.xxx, and connect the network cable to the control box network card 0.
3. Log in to the URL "192.168.57.2:8050" and enter the "System Security Mode Recovery" interface.

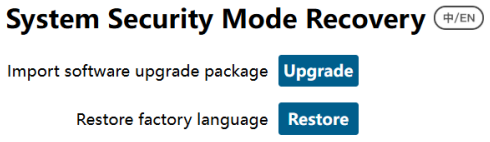


Figure 15.1-9 System security mode recovery interface

- Software upgrade package import: system software package upgrade or downgrade;
- Restore factory language: clear the imported application language package data, restore the factory language package data, and set the default language to English;

1.3.16.1.5 Fault data

Click the “Fault Data Save Enable” button to generate a fault data file when the controller fails, saving data 15 seconds before and after the fault.

After saving, you can select all data sources to export in the system settings, and unzip error_data.tar.gz to view the fault data file.



Figure 15.1-10 Fault data

1.3.16.1.6 Timeout Logout Time Setting

Users can set the timeout logout time. If the time is met, the robot will automatically log out. The unit is min.

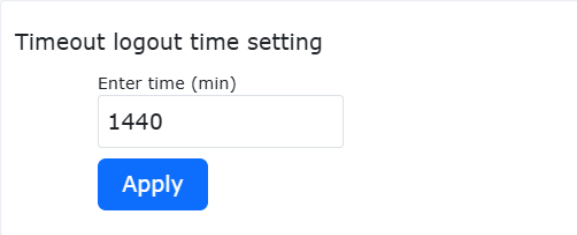


Figure 15.1-11 Timeout Logout Time Setting

1.3.16.1.7 System Settings

Restoring to factory settings under system recovery can clear user data and restore the robot to factory configuration.

The slave log generation and controller log export functions are to download some important status or error record files of the controller, which is convenient for troubleshooting robot problems.

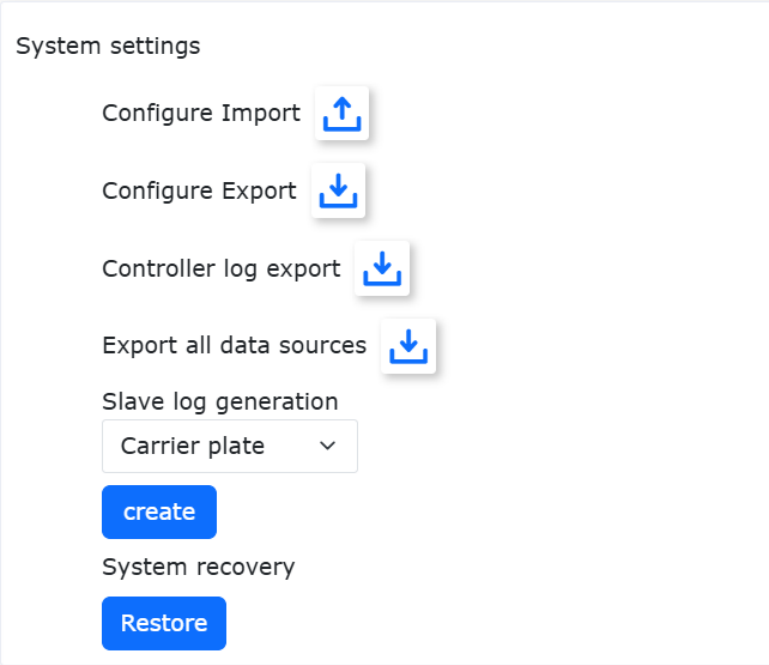


Figure 15.1-12 System Settings

1.3.16.2 Account settings account settings

Click Account Settings on the secondary menu bar to enter the Account Settings interface. Account management functions are only available to administrators. The function is divided into the following three modules:

1.3.16.2.1 User Management

User management page, used to save user information, you can add user ID, function, etc. The user can log in by entering the existing user name and password in the user list.

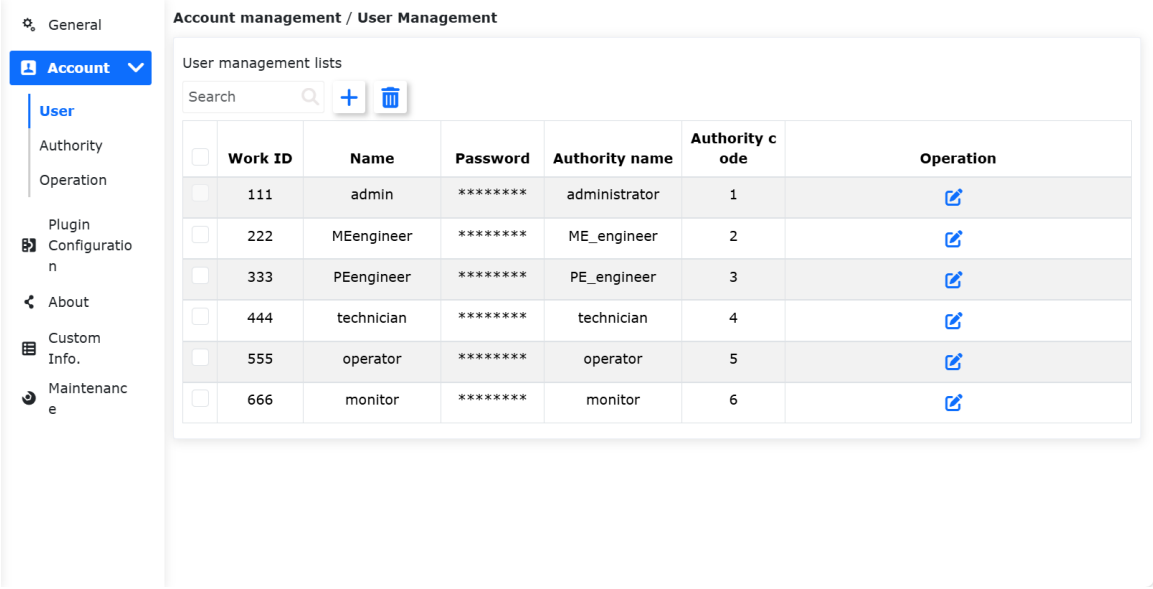


Figure 15.2-1 User Management

- **Add users**:Click the “Add” button, enter the job number, name, password and select the function.

Important: The job number can be up to 10-digit integer, and the job number and password are uniquely checked, and the password is displayed in Braille. After the user is added successfully, you can enter the name and password to log in again.

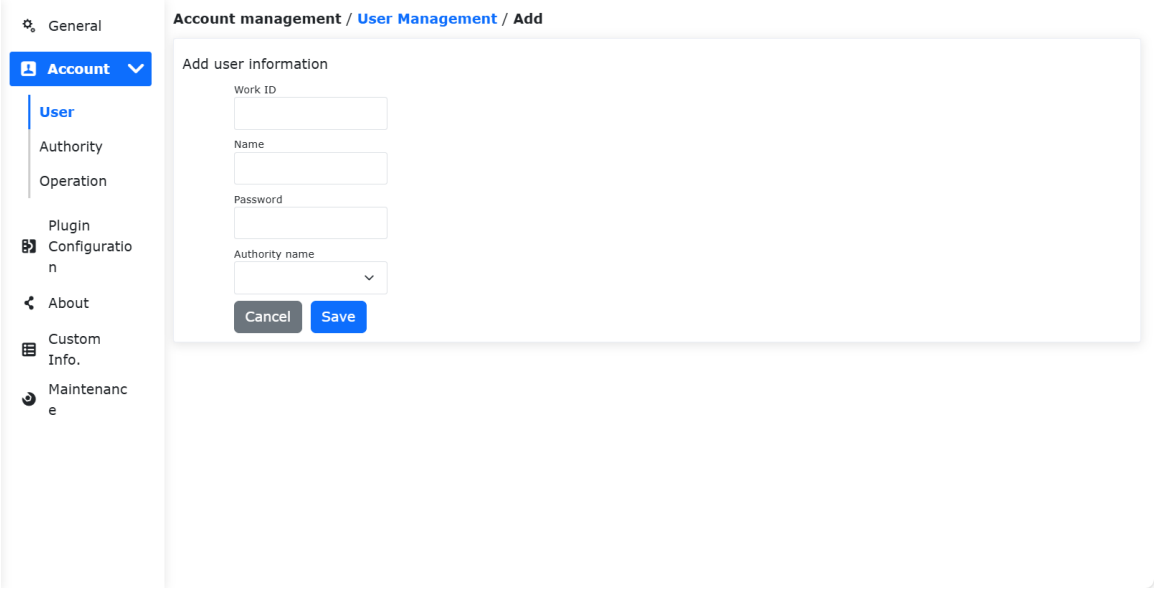


Figure 15.2-2 Add users

- **Edit users**:When there is a user list, click the “Edit” button on the right, the job number and name cannot be modified, but the password and function can be modified, and the password also needs to be uniquely verified.

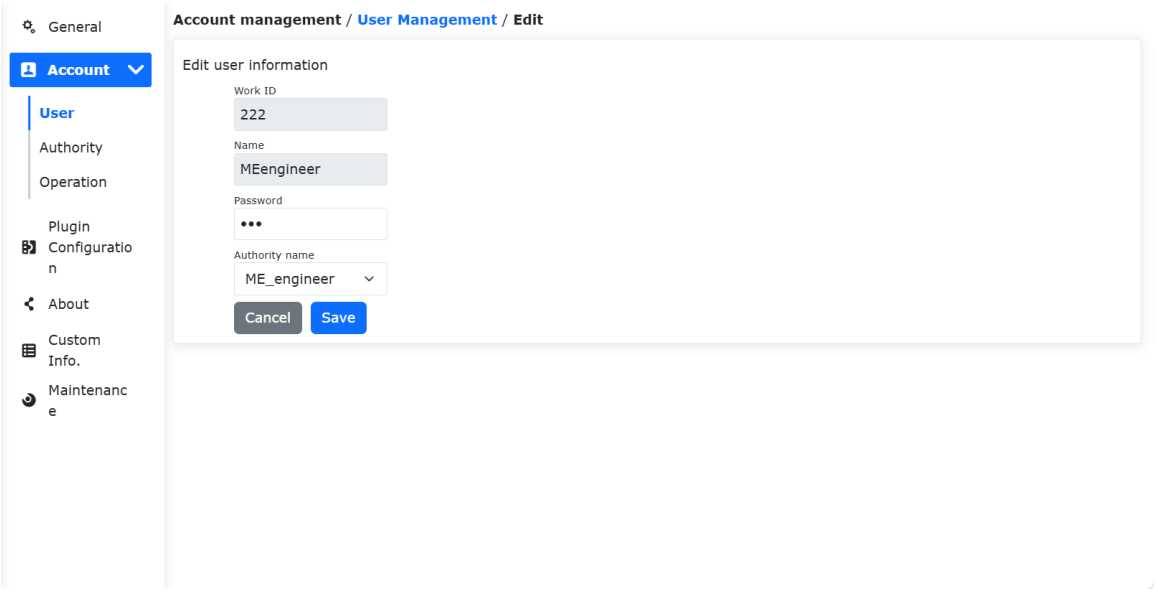


Figure 15.2-3 Edit users

- **Delete users**:The deletion methods are divided into single deletion and batch deletion.

1. Click the single “Delete” button on the right side of the list, and it will prompt “Please click the delete button again to confirm deletion”, and click the list again to delete successfully.
2. Click the check box on the left, select the users to be deleted, and then click the batch “Delete” button at the top of the list twice to delete.

Important: The initial user 111 and the current login user cannot be deleted.

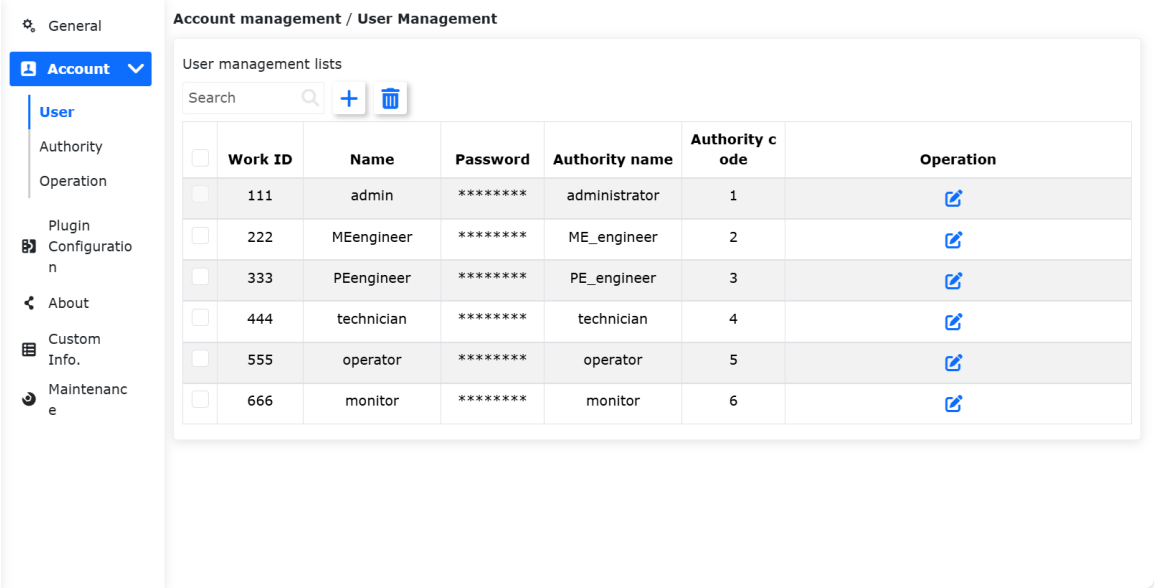


Figure 15.2-4 Delete users

1.3.16.2.2 Authority management

Important: The default function data (function code 1-6) cannot be deleted, and the function code cannot be modified, but the function name and function description can be modified and the authority of the function can be set.

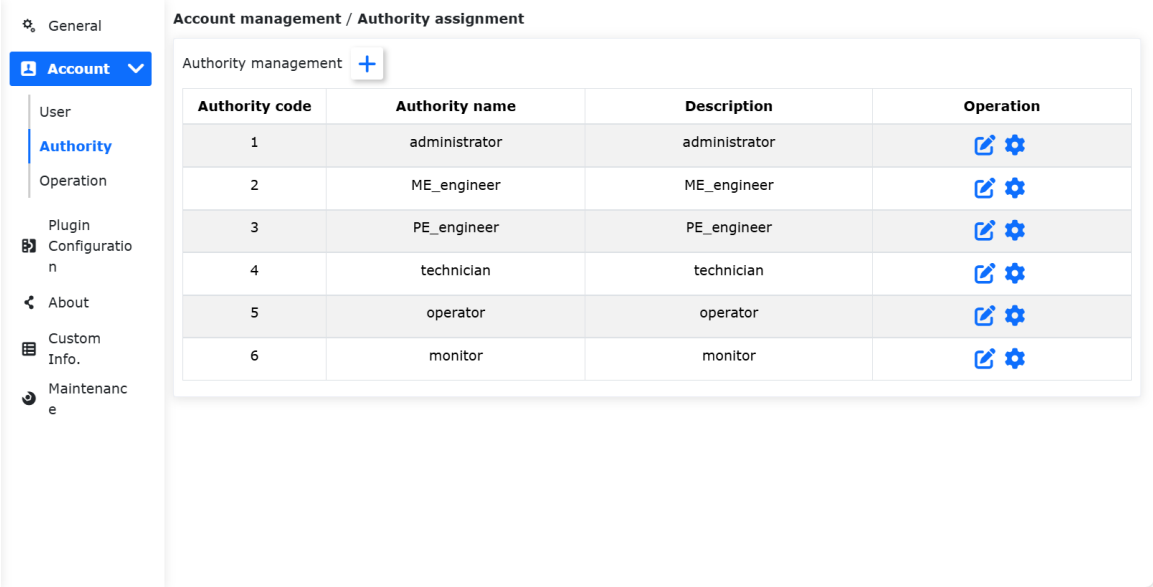


Figure 15.2-5 Authority management

There are six functions by default, administrators have no function restrictions, operators and monitors can use a small number of functions, ME engineers, PE&PQE engineers and technicians & team leaders have some function restrictions, administrators have no function restrictions, the specific default permissions are shown in the following table:

Important: Default permissions can be modified

Table 15.2-1 Permission details

Category	Permission item	Administrator	Engineer 1(ME)	Engineer2(PE,PQE)	Technician & Team Leader	Operator	Monitor
Accessible view	Robot Setting	YES	YES	YES	YES	YES	NO
	Peripheral Configuration	YES	YES	YES	YES	NO	NO
	Teach Programming	YES	YES	YES	YES	YES	YES
	Graphical Programming	YES	YES	YES	YES	YES	YES
	Node Programming	YES	YES	YES	YES	YES	YES
	Teaching Management	YES	YES	YES	YES	YES	YES
	System Log	YES	YES	YES	YES	YES	YES
	Status Query	YES	YES	YES	YES	NO	NO
	Auxiliary Application	YES	YES	YES	YES	NO	NO
	Process Package	YES	YES	YES	NO	NO	NO
	Security Setting	YES	YES	YES	NO	NO	NO
	General Setting	YES	YES	YES	YES	NO	NO
	Plugin Setting	YES	YES	YES	NO	NO	NO
	Custom Information	YES	YES	YES	NO	NO	NO
	Maintenance Mode	YES	NO	NO	NO	NO	NO
Operable function	Start/Stop/Pause & Resume	YES	YES	YES	YES	YES	NO
	Speed Scaling	YES	YES	YES	YES	NO	NO
	Manual/Auto Switch	YES	YES	YES	YES	YES	NO
	Drag Mode Switch	YES	YES	YES	YES	NO	NO
	Local/Remote Mode Switch	YES	YES	YES	YES	YES	NO
	Joint	YES	YES	YES	YES	YES	NO
	Base	YES	YES	YES	YES	YES	NO
	Tool	YES	YES	YES	YES	YES	NO
	Wobj	YES	YES	YES	YES	YES	NO
	Move	YES	YES	YES	YES	YES	NO
	EAxis	YES	YES	YES	YES	NO	NO
	I/O	YES	YES	YES	YES	YES	NO
	TPD	YES	YES	YES	YES	YES	NO
	FT	YES	YES	YES	YES	NO	NO
	RCM	YES	YES	YES	YES	NO	NO
	Teaching Point Recording	YES	YES	YES	YES	YES	NO
	Sensor Point Recording	YES	YES	YES	YES	YES	NO
	Base Coordinate System 3D Display	YES	YES	YES	YES	YES	YES
	Tool Coordinate System 3D Display	YES	YES	YES	YES	YES	YES
	Workpiece Coordinate System 3D Display	YES	YES	YES	YES	YES	YES
	Extended Axis Coordinate System 3D Display	YES	YES	YES	YES	YES	YES
	Trajectory Drawing	YES	YES	YES	YES	YES	NO
	Import Tool Model	YES	YES	YES	YES	NO	NO
	Mount	YES	YES	YES	NO	NO	NO
	World Coordinate System	YES	NO	NO	NO	NO	NO
	Tool Coordinate System	YES	YES	YES	YES	YES	NO
	External Tool Coordinate System	YES	YES	YES	YES	YES	NO
	Workpiece Coordinate System	YES	YES	YES	YES	YES	NO
	End Load	YES	YES	YES	YES	YES	NO
	Soft Limit	YES	YES	YES	NO	NO	NO
	Collision Level	YES	YES	YES	YES	YES	NO
	Friction Compensation	YES	YES	YES	YES	NO	NO
	DI Configuration	YES	YES	YES	YES	NO	NO
	DO Configuration	YES	YES	YES	YES	NO	NO
	Ai Configuration	YES	YES	YES	YES	NO	NO
	I/O Alias Configuration	YES	YES	YES	YES	NO	NO
	I/O Filtering	YES	YES	YES	YES	NO	NO
	Output Reset	YES	YES	YES	YES	NO	NO
	Starting Point	YES	YES	YES	YES	NO	NO
	File Import And Export	YES	YES	YES	YES	NO	NO
	Safe Stop Setting	YES	YES	YES	NO	NO	NO
	Safe Speed Setting	YES	YES	YES	NO	NO	NO
	DIO safety	YES	YES	YES	NO	NO	NO
	Emergency stop	YES	YES	YES	NO	NO	NO
	Protective stop	YES	YES	YES	NO	NO	NO
	Safety Plane	YES	YES	YES	NO	NO	NO
	Single Interference Region	YES	YES	YES	NO	NO	NO
	Multiple Interference Regions	YES	YES	YES	NO	NO	NO
	Reduced Mode	YES	YES	YES	NO	NO	NO
	Security daemon	YES	YES	YES	NO	NO	NO
	Tool orientation restrictions	YES	YES	YES	NO	NO	NO
	Bot restrictions	YES	YES	YES	NO	NO	NO
	Power detection configuration	YES	YES	YES	NO	NO	NO
	Motion Protection	YES	YES	YES	YES	NO	NO
	Sports Configuration	YES	YES	YES	YES	NO	NO
	Force Sensor	YES	YES	YES	YES	NO	NO
	Welding Handle	YES	YES	YES	YES	NO	NO
	Spray Gun Configuration	YES	YES	YES	YES	NO	NO
	Welder Configuration	YES	YES	YES	YES	NO	NO
	Extended Axis	YES	YES	YES	YES	NO	NO
	Line Laser Sensor	YES	YES	YES	YES	NO	NO
	Polishing Device	YES	YES	YES	YES	NO	NO
	Computer Numerical Control	YES	YES	YES	YES	NO	NO
	Auxiliary Sensor	YES	YES	YES	YES	NO	NO
	Combined Equipment	YES	YES	YES	YES	NO	NO
	Board Communication	YES	YES	YES	YES	NO	NO
	Array-type Suction Cup	YES	YES	YES	YES	NO	NO
	Export	YES	YES	YES	YES	NO	NO
	Import	YES	YES	YES	YES	NO	NO
	Modify	YES	YES	YES	NO	NO	NO
	Delete	YES	YES	YES	YES	NO	NO
	Export Log	YES	YES	YES	YES	NO	NO
	Robot Packaging	YES	YES	YES	YES	NO	NO
	Data Backup	YES	YES	YES	YES	NO	NO
	IOS Data Recording	YES	YES	YES	YES	NO	NO
	End Led Configuration	YES	YES	YES	YES	NO	NO
	Drag Lock	YES	YES	YES	YES	NO	NO
	Intersection Generation	YES	YES	YES	YES	NO	NO
	Peripheral Protocol	YES	YES	YES	NO	NO	NO
	G-Code Conversion	YES	YES	YES	YES	NO	NO
	Welding Expert Database	YES	YES	YES	YES	NO	NO
	Torque System	YES	YES	YES	YES	NO	NO
	Stiffness grade	YES	YES	YES	YES	NO	NO
	Palletizing System	YES	YES	YES	YES	NO	NO
	Conveyor Tracking	YES	YES	YES	YES	NO	NO
	Time Setting	YES	YES	YES	YES	NO	NO
	Network Setting	YES	YES	YES	YES	NO	NO
	Teach Pendant Setting	YES	YES	YES	YES	NO	NO
	System Language	YES	YES	YES	YES	NO	NO
	Fault data	YES	YES	YES	YES	NO	NO
	Logout Timeout	YES	YES	YES	YES	NO	NO
	System File Export	YES	YES	YES	YES	NO	NO
	Upload Information Package	YES	NO	NO	NO	NO	NO
	Modify Robot Type Name	YES	NO	NO	NO	NO	NO
	Teaching Program Encryption	YES	YES	YES	NO	NO	NO

- **Add function**: Click the "Add" button, enter the function code, function name and function description, click the "Save" button, and return to the list page after success. Among them, the function code can only be an integer greater than 0 and cannot be the same as the existing function code, and all input items are required.

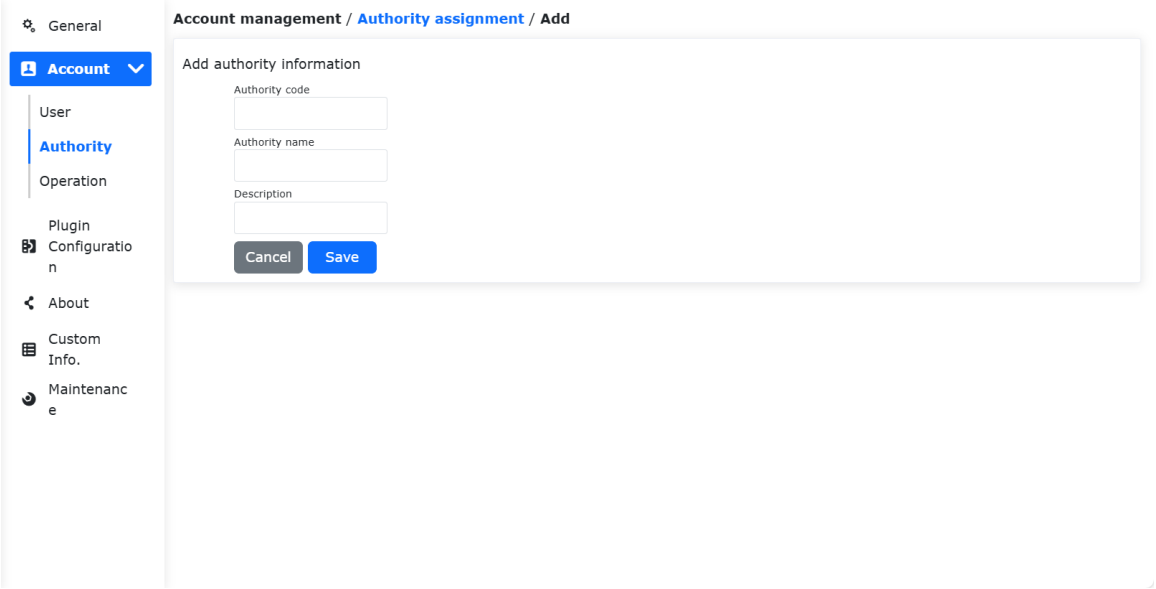


Figure 15.2-6 Add function

- **Edit function name and description**: Click the "Edit" icon in the table operation bar to modify the function name and function description of the current function. After the modification is completed, click the "Save" button below to confirm the modification.

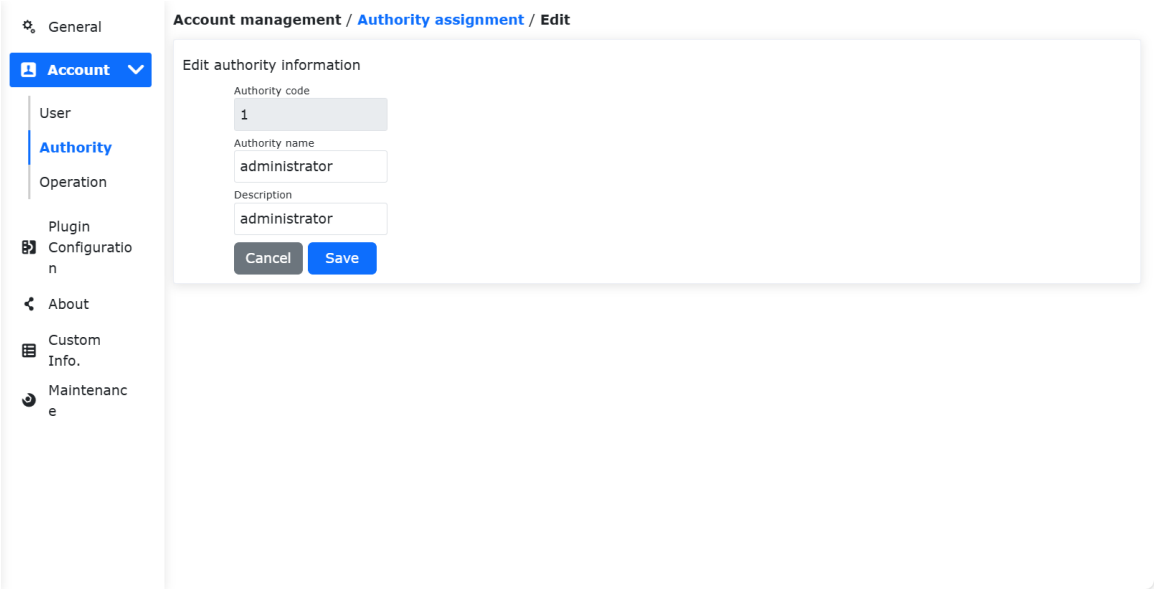


Figure 15.2-7 Edit function name and description

- **Set function permissions**: Click the "Settings" icon in the table operation bar to set the permissions of the current function. After setting, click the "Save" button below to confirm the settings.

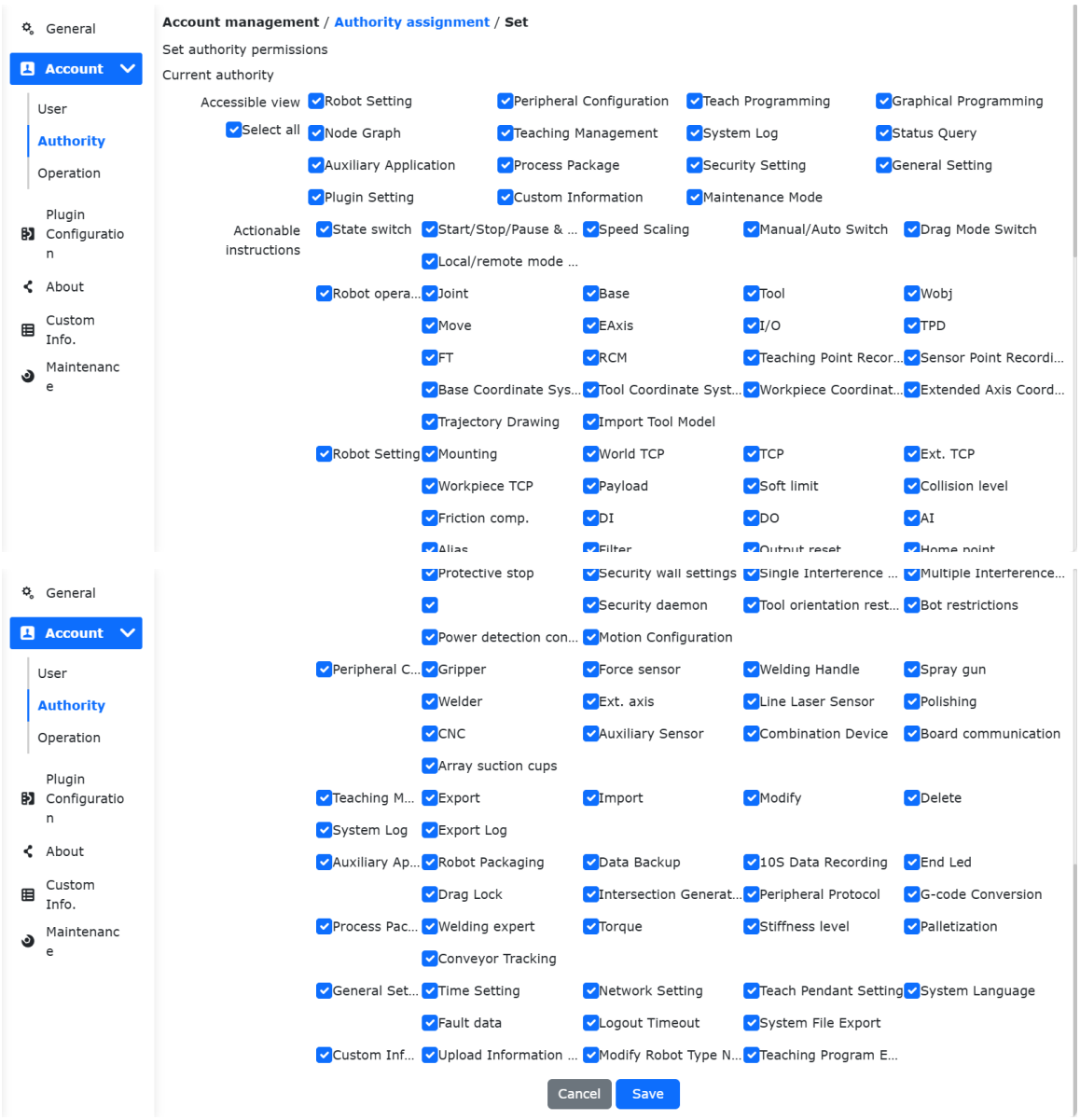


Figure 15.2-8 Set function permissions

- **Delete function**: Click the “Delete” icon in the table operation bar, firstly, it will check whether the current function is used by a user, if no user uses it, the current function can be deleted, otherwise it cannot be deleted.

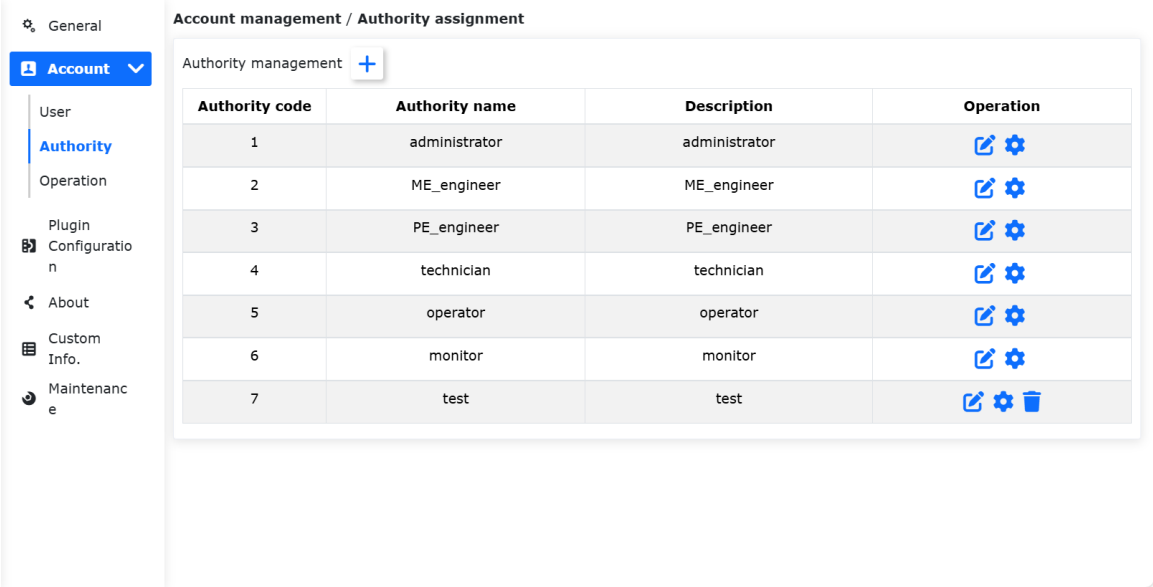


Figure 15.2-9 Delete function

1.3.16.2.3 Import/Export

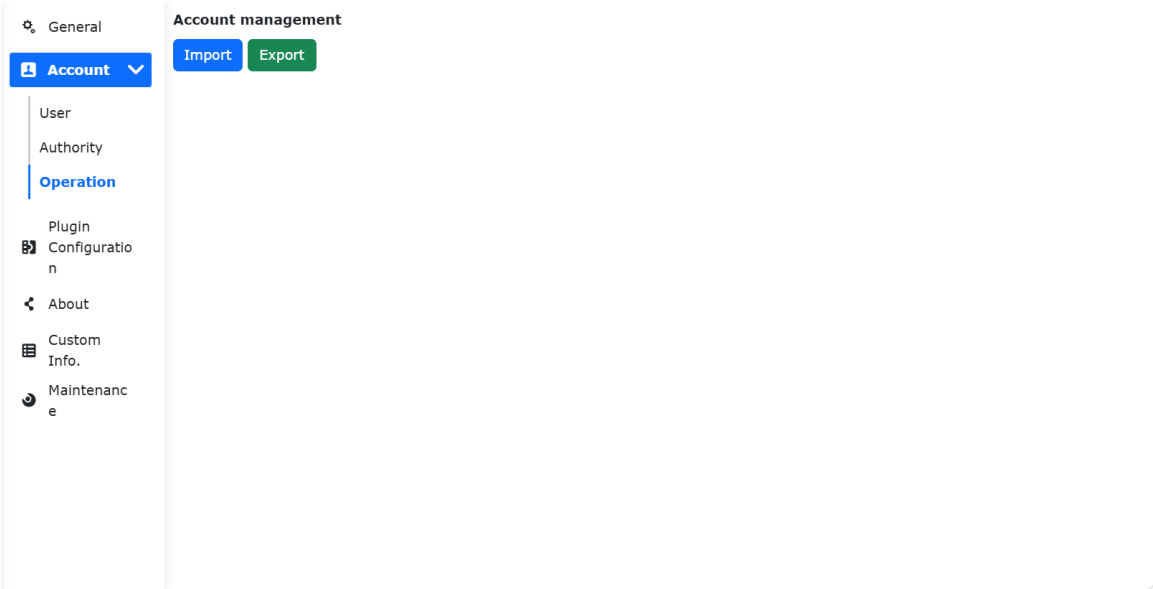


Figure 15.2-10 Account settings import/export

- **Import**: Click the “Import” button to import user management and rights management data in batches.
- **Export**: Click the “Export” button to export the data of user management and rights management in batches.

1.3.16.3 About

Click About on the secondary menu bar to enter the About interface. This page shows the model and serial number of the robot, the web version and control box version used by the robot, hardware version and firmware version.

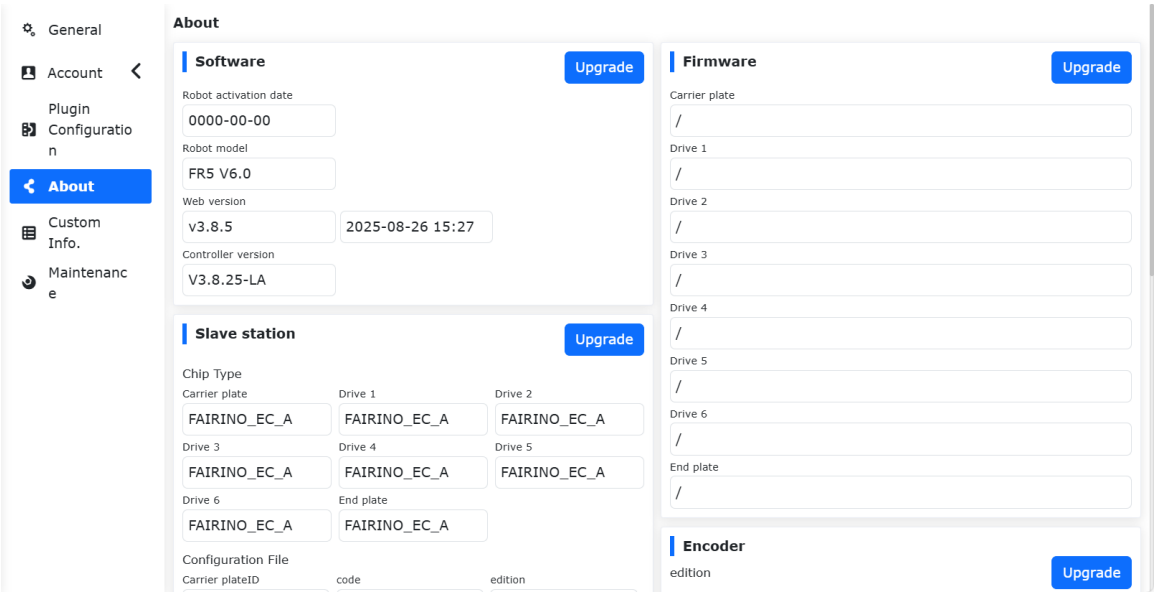


Figure 15.3-1 About Schematics

1.3.16.3.1 Software Upgrade

1.3.16.3.1.1 Operation Preparation

1. Before upgrading, check and confirm the current software version in “System Settings - About”.
2. Software upgrade package: Download from the corresponding version’s FARRDOC “Resources Download - Robot Software Download”. After extraction, it contains the software upgrade package “software.tar.gz” for the corresponding version.

1.3.16.3.1.2 Important Notes

1. **Data Backup**: It is recommended to perform a backup before upgrading (refer to section 3.2.1) to avoid data loss due to upgrade anomalies.
2. **Version Restrictions**:

Table 15.3-1 Version Upgrade Restrictions

Current Version	Maximum Upgradeable Version
< v3.6.1	v3.6.1
v3.6.1 - v3.6.4	v3.6.5
v3.6.5 - v3.6.8	v3.6.9
v3.6.9 - v3.7.4	v3.7.5
v3.7.5	v3.7.6
v3.7.6	No restrictions

3. **Cache Clearance**: After each upgrade (especially for cross-version upgrades), it is recommended to clear the browser cache to ensure normal system operation.

1.3.16.3.1.3 Operation Steps

Software Upgrade:

1. Under the “System Settings” -> “About” menu, click the “Upgrade” button to enter the software upgrade interface;

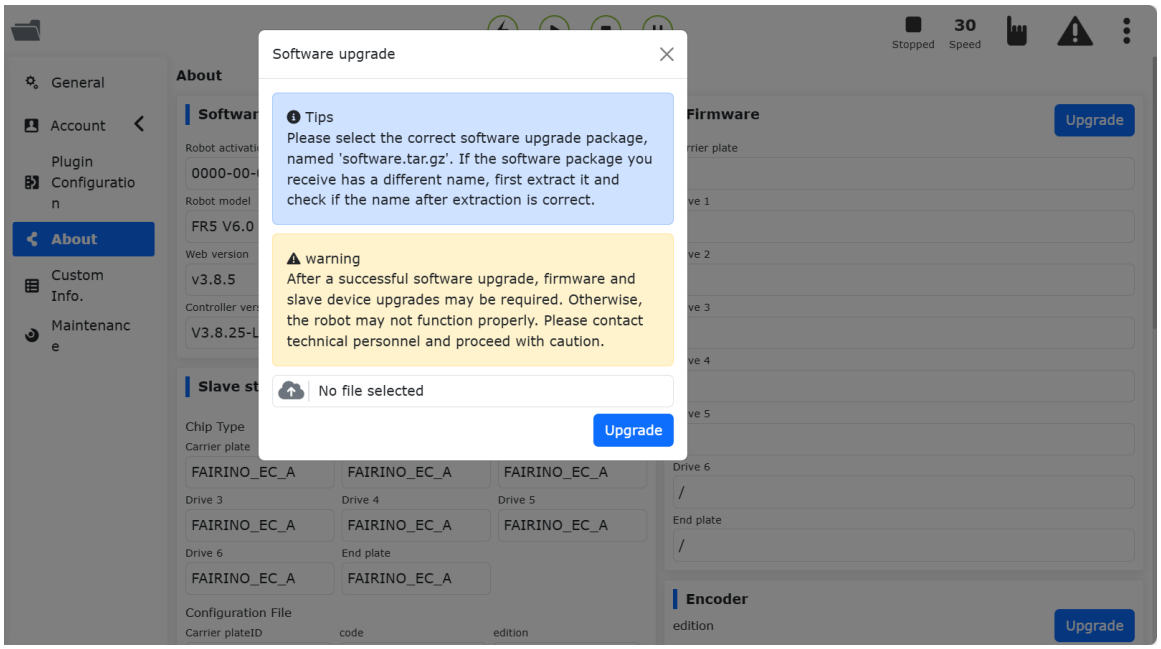


Figure 15.3-2 System Upgrade Interface

2. Click “Choose File” and select the software package “software.tar.gz” downloaded from the official website;

Important: The software upgrade package name must be exactly “software.tar.gz”. If the upgrade package name differs, the upgrade will fail. Rename it to the correct package name.

---

3. Click “Upload Upgrade Package” to start the upgrade. The progress bar will be displayed during the upgrade process;
4. When the upgrade progress reaches 100%, the interface will prompt “Upgrade successful, please restart the control box”;

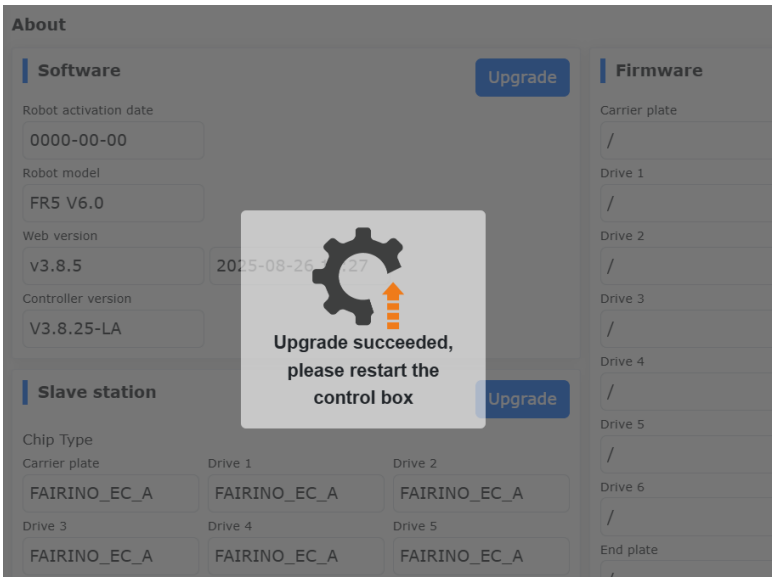


Figure 15.3-3 Software Upgrade Successful

5. After restarting the control box, the upgrade is complete. Confirm the version information in “About”.

Firmware Upgrade: After the robot enters BOOT mode, upload the upgrade compressed package, select the required slave stations (control box slave station, body drive slave stations 1~6, end effector slave station) to perform the upgrade operation, and display the upgrade status.

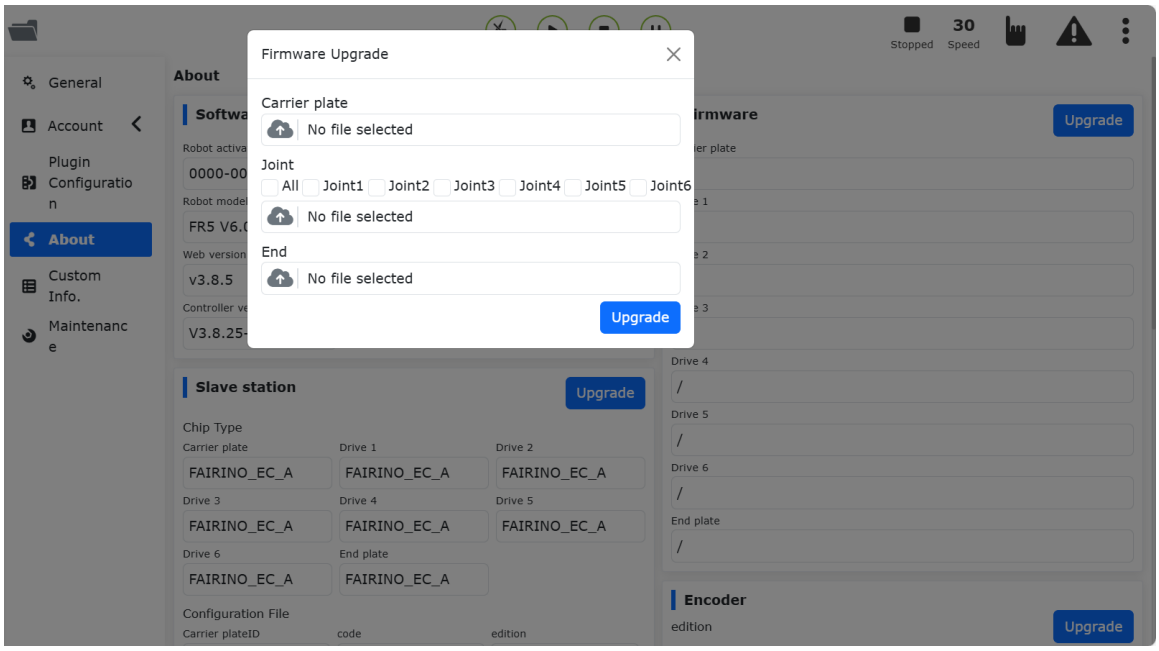


Figure 15.3-4 Firmware Upgrade

Slave Station Configuration File Upgrade: After the robot is disabled, upload the upgrade file, select the required slave stations (control box slave station, body drive slave stations 1~6, end effector slave station) to perform the upgrade operation, and display the upgrade status.

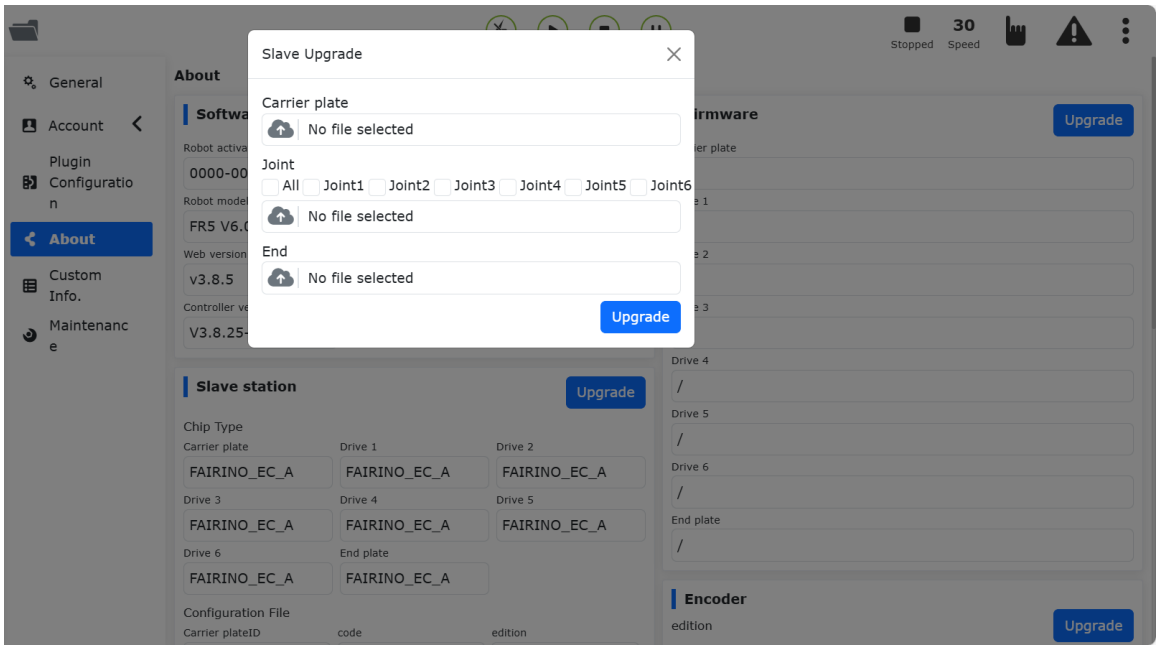


Figure 15.3-5 Slave Station Configuration File Upgrade

Encoder Upgrade: After the robot is disabled, upload the upgrade file, select the required joints (Joint1~Joint6) to upgrade, and configure the encoder mode.

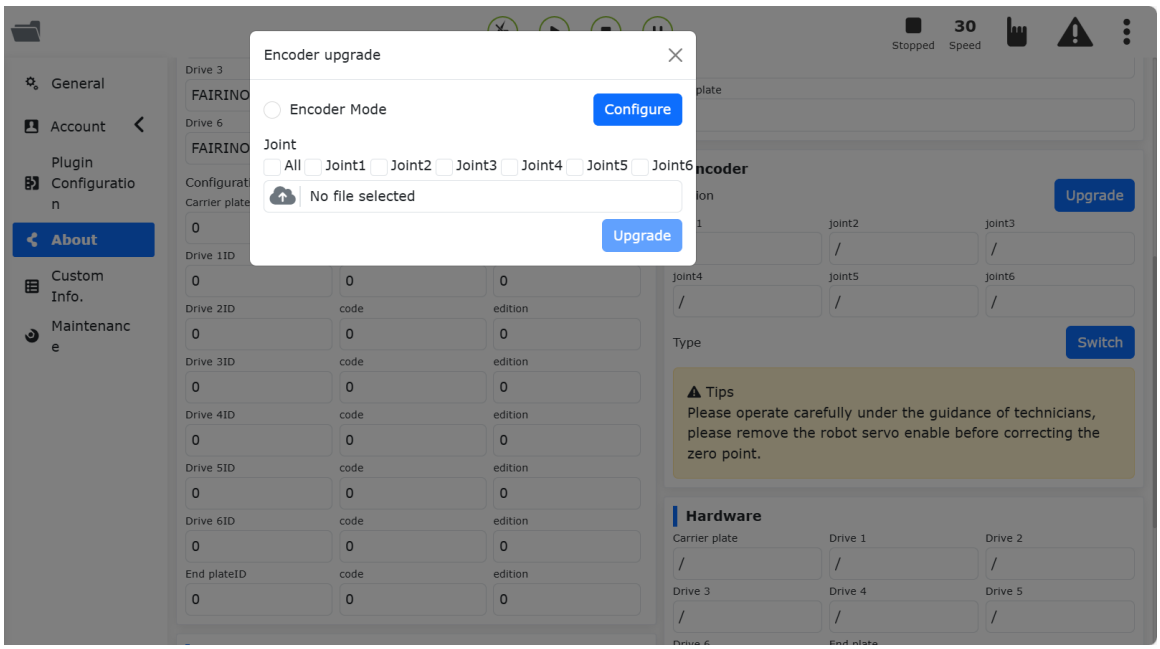


Figure 15.3-6 Encoder Upgrade

1.3.16.4 Custom information

Click the custom information of the secondary menu bar to enter the custom information interface. Custom information functions can only be used by administrators. This page can upload user information packages, robot models, and setting teaching program encryption status.

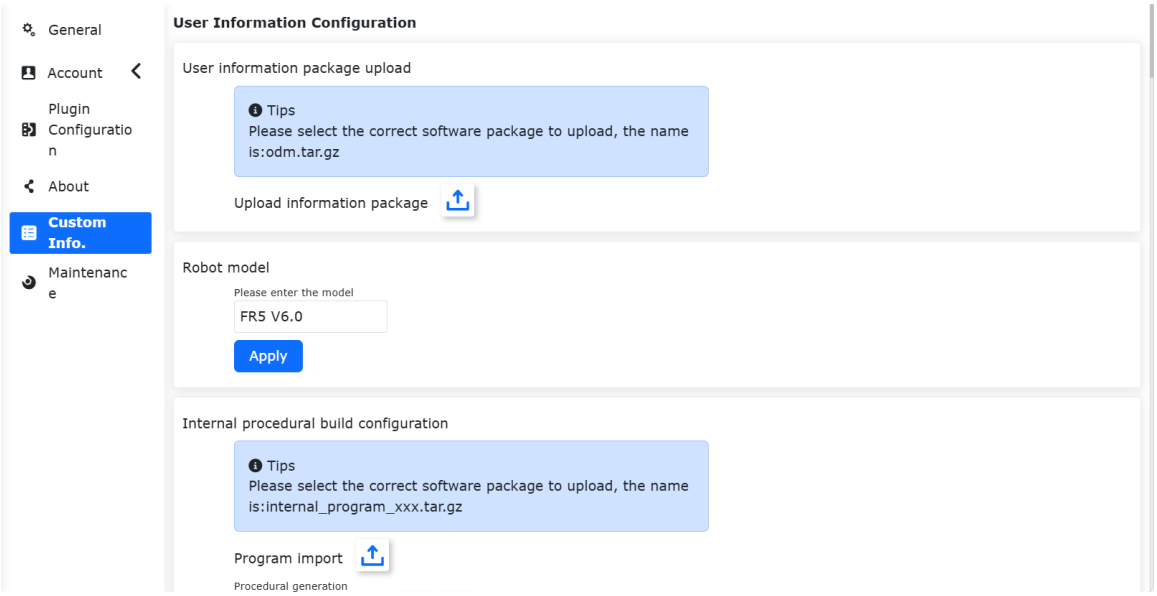


Figure 15.4-1 Custom information schematic diagram

1.3.16.4.1 Robot Model

Important:

1. The robot model configured here is a custom robot model name, which is inconsistent with the robot model function configured in “System” -> “Maintenance” -> “Controller Compatible”;
2. It is not recommended to use names starting with “FR” and “ART”. If you enter a custom robot model starting with “FR” and “ART”, the model name you enter must be consistent with the “Model Abbreviation” in the robot model catalog table (for details of the robot model catalog table, see the “Robot Model Configuration” section).

1.3.16.4.2 Parameter range configuration

Parameter range configuration, only the administrator can adjust the parameter range, and the parameters of other authorized members can only be set within the parameter range set by the administrator.

There are two ways to set parameters: slider dragging and manual input.

Important: The maximum value of the parameter range must be greater than the minimum value. 3 seconds after the parameter range is successfully configured, it will automatically jump to the login page, and you need to log in again.

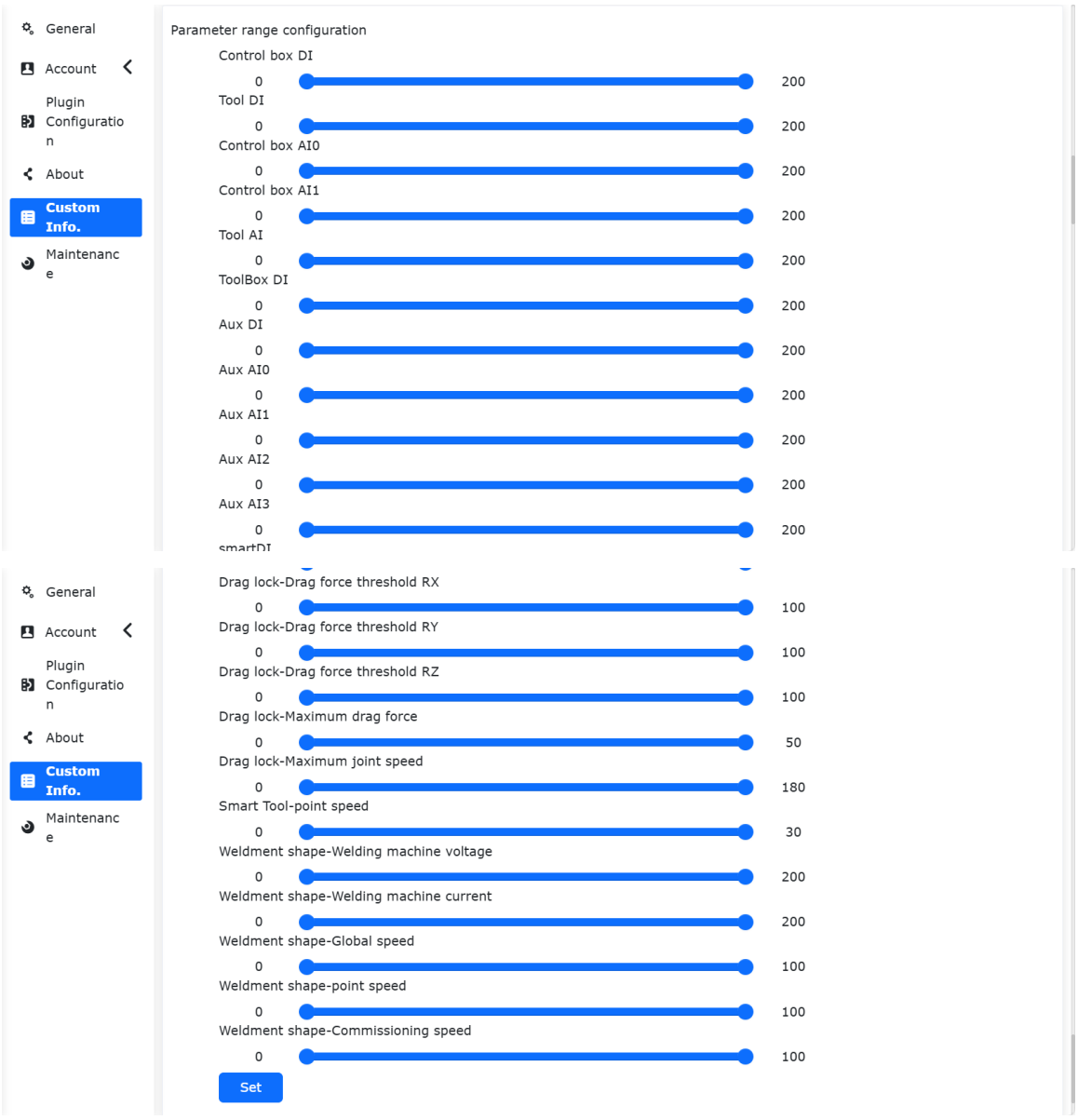


Figure 15.4-2 Schematic diagram of parameter range configuration

1.3.16.4.3 Robot License Usage Time

1. License Usage Time settings

Check the web interface lock screen settings in “Custom” and set whether this function is turned on. When choosing to turn on this function, select the usage period. If not selected, it will prompt “The usage period cannot be empty”.

Note: If the Robot License Usage Time is turned on, secondary settings cannot be made, and the system time cannot be updated.) After selecting the usage period, click the “Configure” button.

After selecting the usage period, click the “Configure” button.

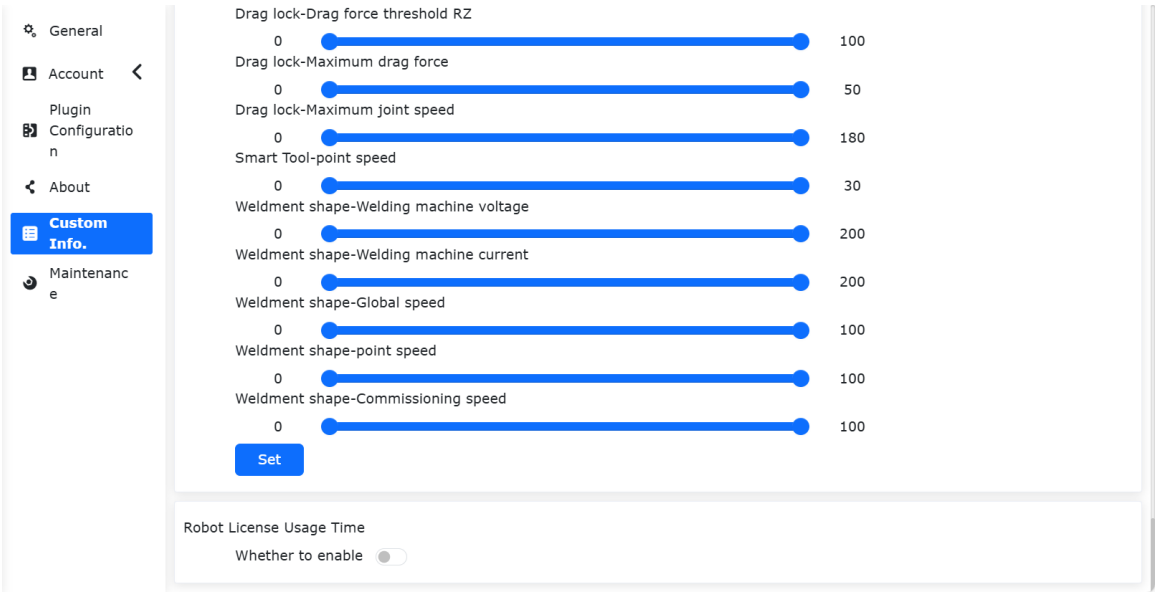


Figure 15.4-3 WEB interface lock screen shutdown settings



Figure 15.4-4 WEB interface lock screen enable settings

2. Expiration reminder

When the web interface Robot License Usage Time is turned on, the following prompt will appear after logging in to the interface:

1)5 days before the expiration of the device, if you power on and log in successfully, a pop-up window will prompt the remaining days of the use period, which can be eliminated by resetting.

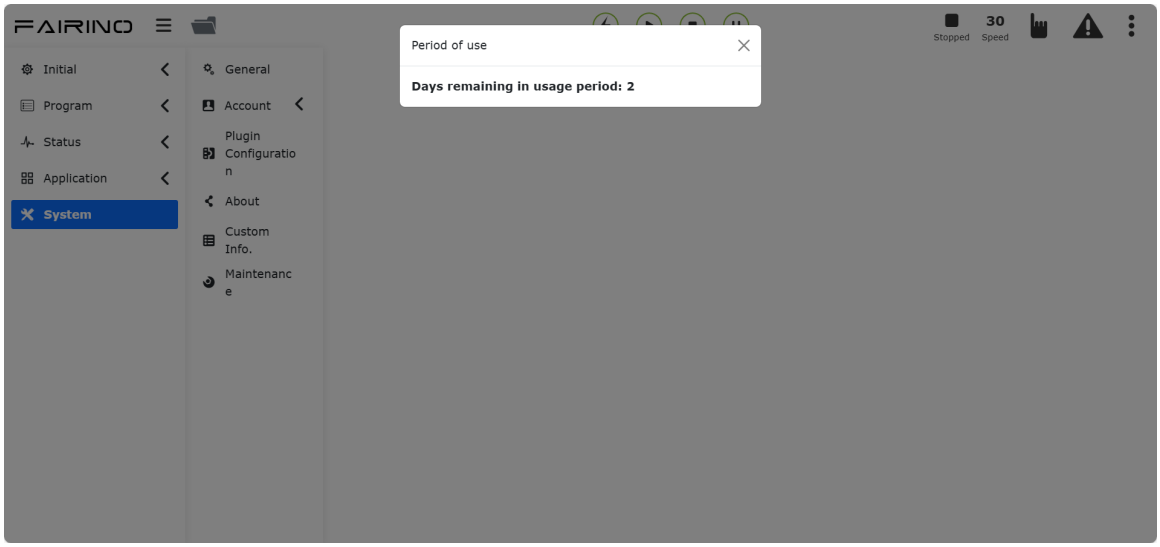


Figure 15.4-5 Boot prompt

2)If the device continues to work, 5 days before the device expires, a pop-up window will automatically pop up at zero o'clock to prompt the remaining days of the service life, which can be eliminated by resetting.



Figure 15.4-6 Continuous work tips

3. Unlock login

When the web interface Robot License Usage Time is turned on, after the device expires, you will directly enter the lock screen interface when you log in to the webApp for the first time. When the device continues to work, it will automatically log out after obtaining the lock screen data at zero point and enter the lock screen interface. At this time, enter the unlock code to unlock and enter the login interface, enter your login information to log in.

Note: The integrator operates to generate an encrypted unlock code.

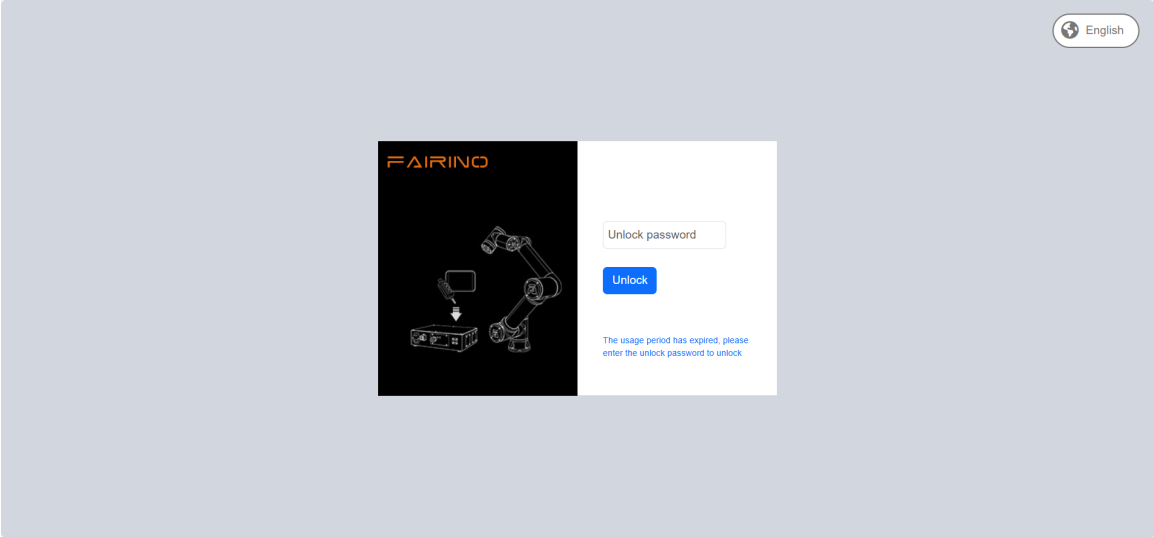


Figure 15.4-7 Lock screen

1.3.16.5 Robot model configuration

Important: If you need to modify the robot model, please contact our technical engineers and proceed under guidance.

After logging into the collaborative robot console Web, select the corresponding model to modify in the "System Settings" -> "Maintenance Mode" -> "Controller Compatibility" configuration item. For the robot model, refer to the table below.

The robot model table is as follows:

Numerical value	Model (Main model - Major - Minor)	Model Abbreviation
0	Not configured	/
1	FR3-V1-000(V5.0)	FR3 V5.0
2	FR3-V1-001(V6.0)	FR3 V6.0
3	FR3-V1-002(V6.0 Mirror)	FR3 V6.0(Mirror)
…	Reserved	/
101	FR5-V1-000	FR5 V4.0
102	FR5-V1-001(V5.0)	FR5 V5.0
103	FR5-V1-002(V6.0)	FR5 V6.0
…	Reserved	/
201	FR10-V1-000(V5.0)	FR10 V5.0
202	FR10-V1-001(V6.0)	FR10 V6.0
…	Reserved	/
301	FR16-V1-000(V5.0)	FR16 V5.0
302	FR16-V1-001(V6.0)	FR16 V6.0
…	Reserved	/
401	FR20-V1-000(V5.0)	FR20 V5.0
402	FR20-V1-001(V6.0)	FR20 V6.0
…	Reserved	/

continues on next page

Table 8 – continued from previous page

501	ART3-V1-000	ART3
…	Reserved	/
601	ART5-V1-000	ART5
…	Reserved	/
702	FRCustom(7)-V1-001(FR3-WML)	FR3-WML
703	FRCustom(7)-V1-001(FR3-WMS)	FR3-WMS
…	Reserved	/
802	FRCustom(8)-V1-001(FR5WM)	FR5WM
803	FRCustom(8)-V1-002(FR5-WML)	FR5-WML
…	Reserved	/
901	FRCustom(9)-V1-001(FR3MT)	FR3MT
902	FRCustom(9)-V1-001(FR10YD)	FR10YD
904	FRCustom(9)-V1-001(FR3-C)	FR3-C
905	FRCustom(9)-V01-001(FR30L)	FR30L
…	Reserved	/
1001	FR30-V1-001(V6.0)	FR30 V6.0
…	Reserved	/

Note: Among them, 10 major version numbers (1-10) are reserved, and 10 minor version numbers (1-10) are reserved.

1.3.17 Teach pendant

1.3.17.1 Teach pendant activation

1. Connect the control box to the teach pendant and start it.
2. Log in with the account admin and password 123. Enter the page, click System Settings-General Settings, and confirm that the teach pendant is enabled.

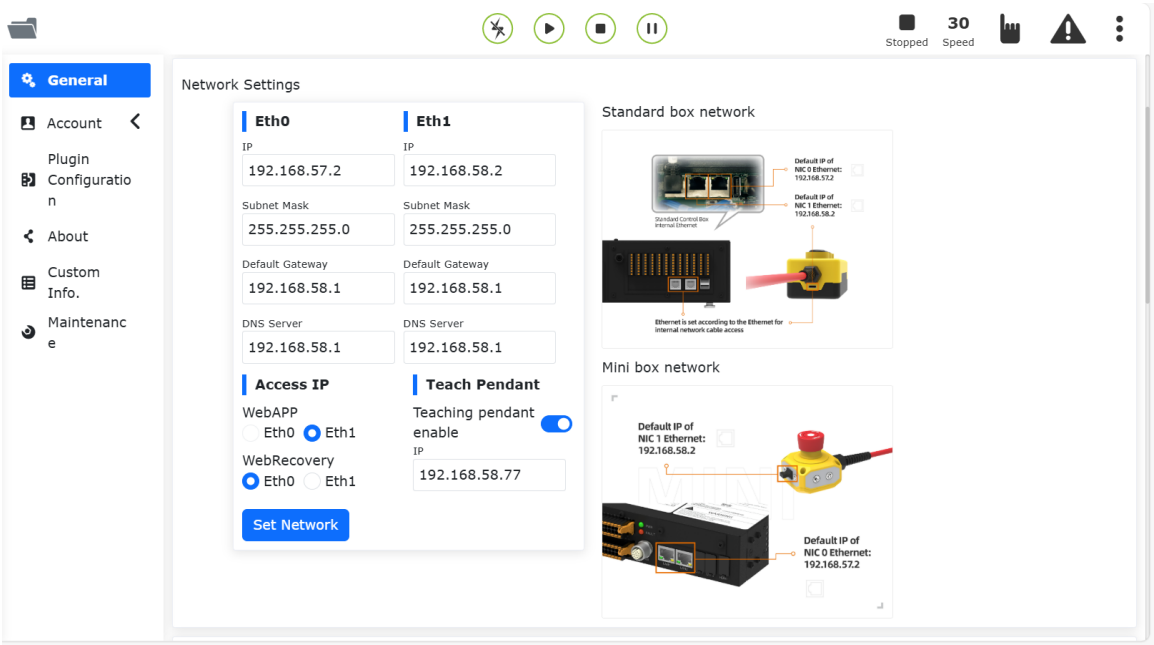


Figure 16.1-1 Teaching pendant activation status

1.3.17.2 Teach pendant multi-language setting

1. On the login interface (or the first activation interface), select the language in the upper right corner.

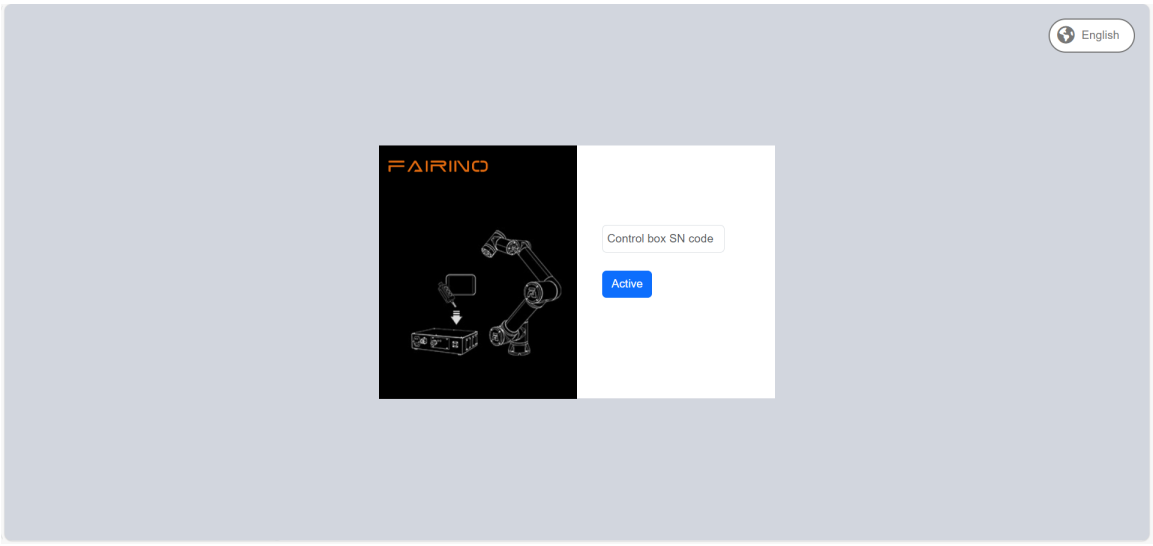


Figure 16.2-1 Activation interface language setting

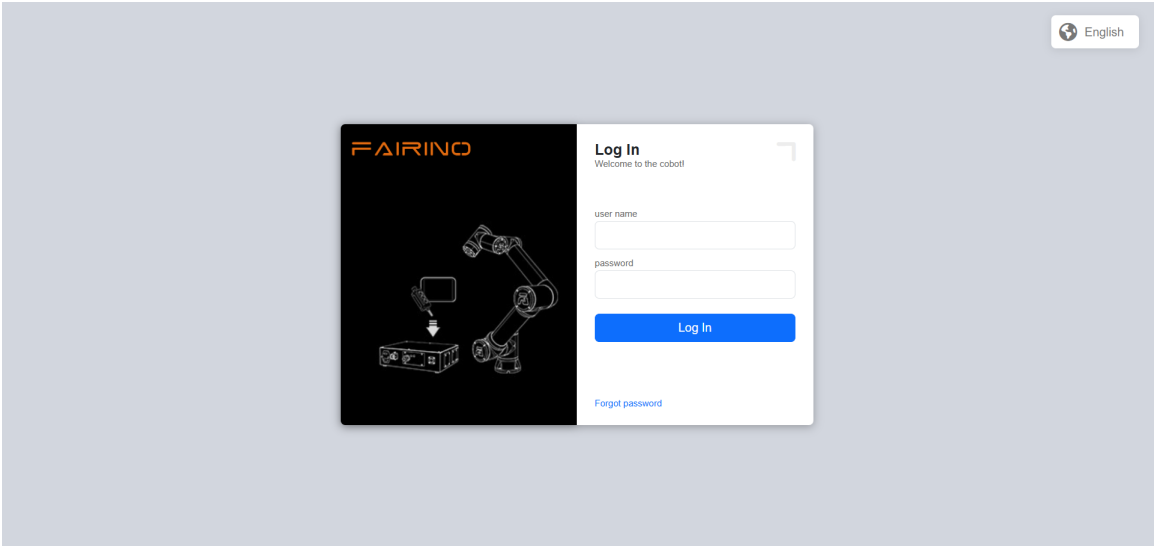


Figure 16.2-2 Login interface language setting

2. Taking the login interface multi-language setting as an example, select the language. If the following prompt (corresponding to different languages) appears, the setting is successful. Restart the control box to complete the language setting.



Figure 16.2-3 Set Chinese

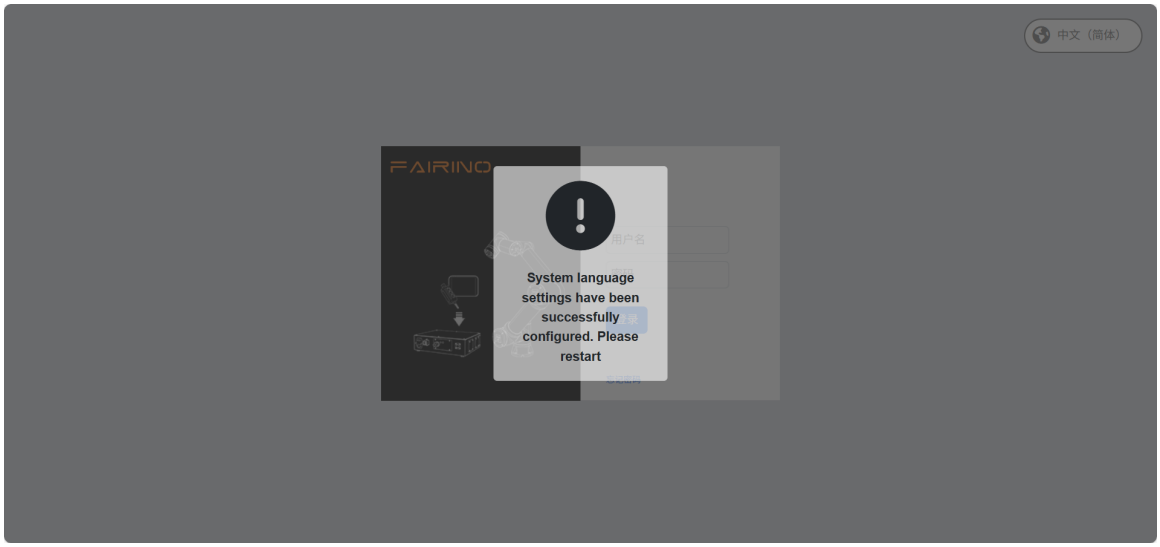


Figure 16.2-4 Set English

1.3.17.2.1 Input method switch

The default input method is English input method.

1. Open the soft keyboard in the lower right corner and click the input box, such as the user name input box.
2. Switch to Chinese Pinyin input method.

Click the CTRL key twice, the key status turns red, and click the spacebar to select the input method. The following is the Chinese input method.



Figure 16.2-5 Chinese Pinyin Input Method

3. Switch to English Input Method

Click the CTRL key twice, the key status turns red, click the spacebar to select the input method, the following is the English input method.



Figure 16.2-6 English Input Method

After successful login, the system will load the model and other data, and enter the initial page after loading.

1.3.17.2.2 The language of the teaching pendant and webApp is inconsistent

After the teach pendant is enabled, the teach pendant and webApp language verification will be triggered on the login interface. When the teach pendant language is inconsistent with the webApp language, the following prompt will appear.



Figure 16.2-7 The language of the teaching pendant and webApp is inconsistent

1.3.17.3 Controller and Teach Pendant IP Reset Function

1.3.17.3.1 Function Overview

This optimization adds IP reset operations for the controller and the physical teach pendant through different methods, mainly enabling the following functions through the following operations:

- 1. Use the webrecovery interface to reset the IP addresses of the controller's Network Card 0 and Network Card 1;
- 2. Use the physical teach pendant's F1 custom key function configured for IP reset (press and hold for 10 seconds) to reset the IP addresses of the controller's Network Card 0, Network Card 1, and the physical teach pendant;
- 3. Use the physical teach pendant's F2 and F4 key combination, press and hold simultaneously for 10 seconds, to reset the IP address of the physical teach pendant device when it is not logged in.

The network port diagrams for the Standard Controller Box and the Mini Controller Box are shown below.

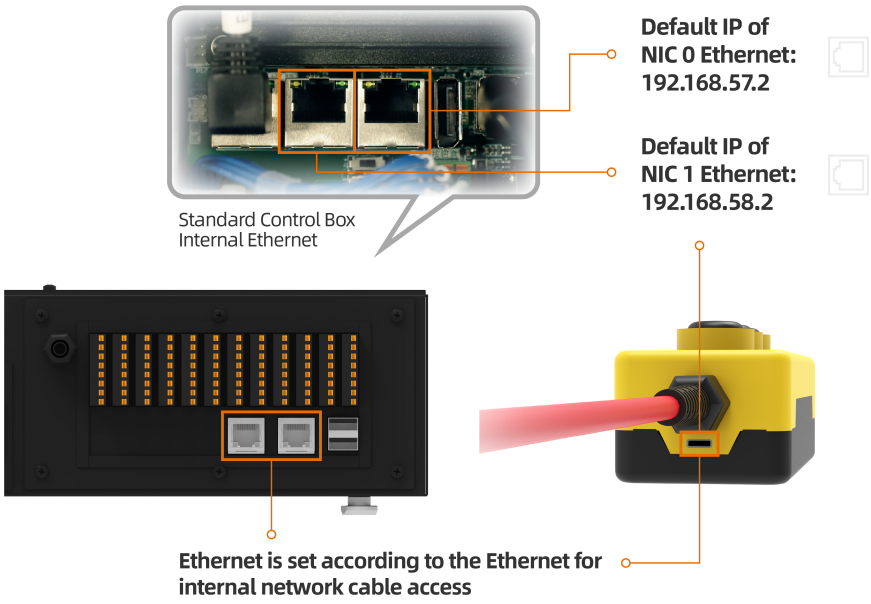


Figure 16.3-1 Standard Controller Box Network Port Diagram

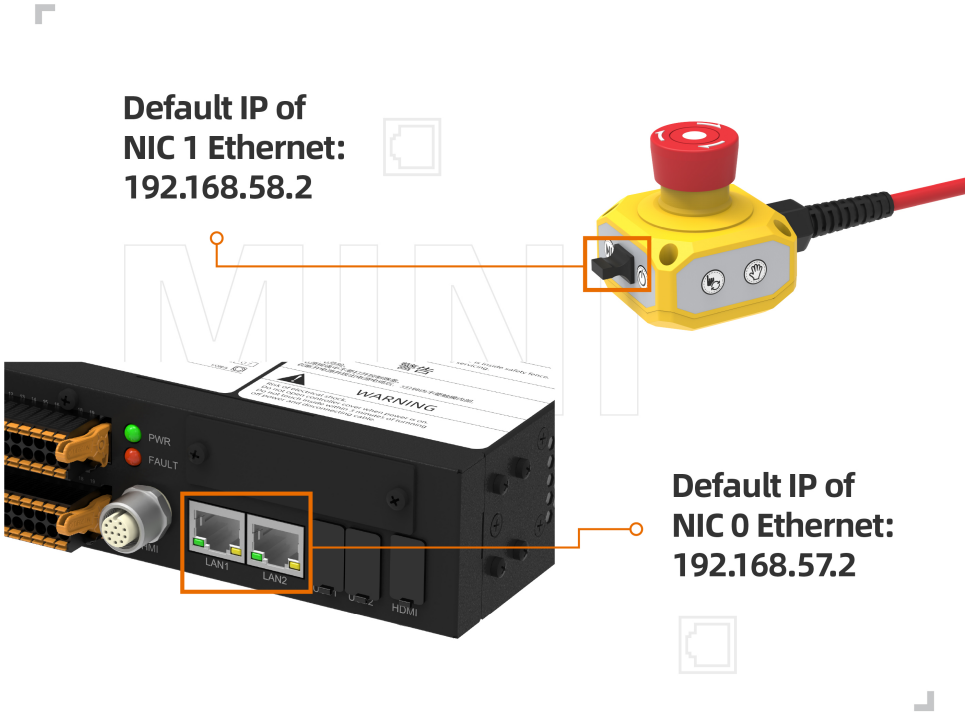


Figure 16.3-2 Mini Controller Box Network Port Diagram

1.3.17.3.2 Webrecovery Interface IP Reset

Access the webrecovery interface using port 8050, for example, log in with the default IP: 192.168.57.2:8050. Click the 'Reset' button for "Reset Controller IP". The page will display a secondary confirmation pop-up. Click 'Confirm' and then click the Reset Controller IP button again to confirm the reset.

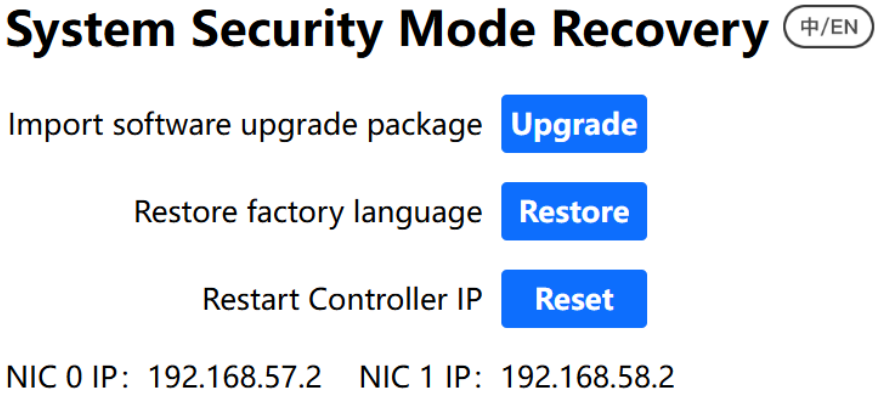


Figure 16.3-3 Webrecovery Interface IP Reset Function

After the secondary confirmation, a prompt will indicate that a restart is required to take effect. After restarting, the controller's Network Card 0 IP will be restored to the default 192.168.57.2, and Network Card 1 IP will be restored to the default 192.168.58.2.

1.3.17.3.3 Physical Teach Pendant F1 Key Custom IP Reset

To use the physical teach pendant's F1 key custom function, you must first log in to the teach pendant interface to configure the F-key custom functions. Click "System Settings", click "General Settings", select the Teach Pendant module, turn on the Enable Teach Pendant switch, configure the F1 key as Reset IP (press and hold for 10 seconds), then click Configure.

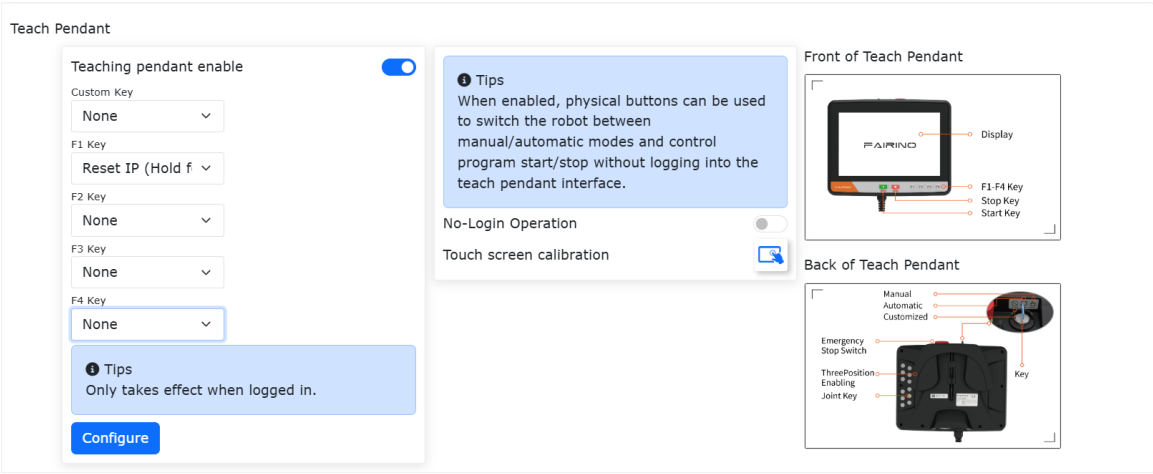


Figure 16.3-4 Physical Teach Pendant F1 Key Custom IP Reset

This function only takes effect when the physical teach pendant is logged into the webapp. After pressing and holding the F1 key for 10 seconds, a prompt will indicate that a restart is required to take effect. After restarting, the controller's Network Card 0 IP will be restored to the default 192.168.57.2, Network Card 1 IP will be restored to the default 192.168.58.2, and the physical teach pendant IP will be restored to the default 192.168.58.77.

1.3.17.3.4 Physical Teach Pendant F2 and F4 Key Combination IP Reset

The physical teach pendant device provides an IP reset function, which can be performed even when not connected to the webapp. Simultaneously press and hold the F2 and F4 key combination for 10 seconds to reset the physical teach pendant's IP address. The IP will be restored to the default 192.168.58.77. After restoration, you need to log back into the webapp. In System Settings - General Settings, set the physical teach pendant IP to 192.168.58.77. Restart to re-establish the teach pendant connection.

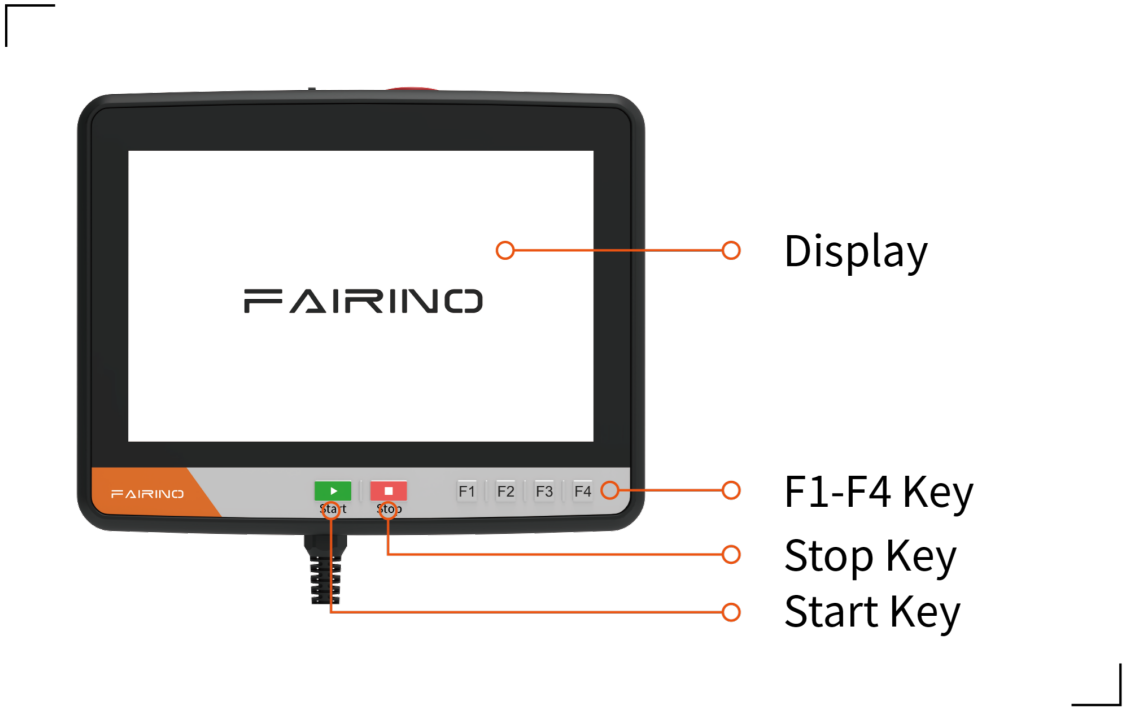


Figure 16.3-5 Physical Teach Pendant F2 and F4 Key Combination IP Reset

1.3.17.4 Teach Pendant Key Customization Function

1.3.17.4.1 Function Overview

This document aims to introduce how to use the teach pendant key customization function.

1.3.17.4.2 Operation Instructions

1.3.17.4.2.1 Function Configuration

1. Open and log in to the webApp;
2. Click the "System Settings" - "General Settings" menu in the left sidebar to enter the teach pendant configuration module interface;

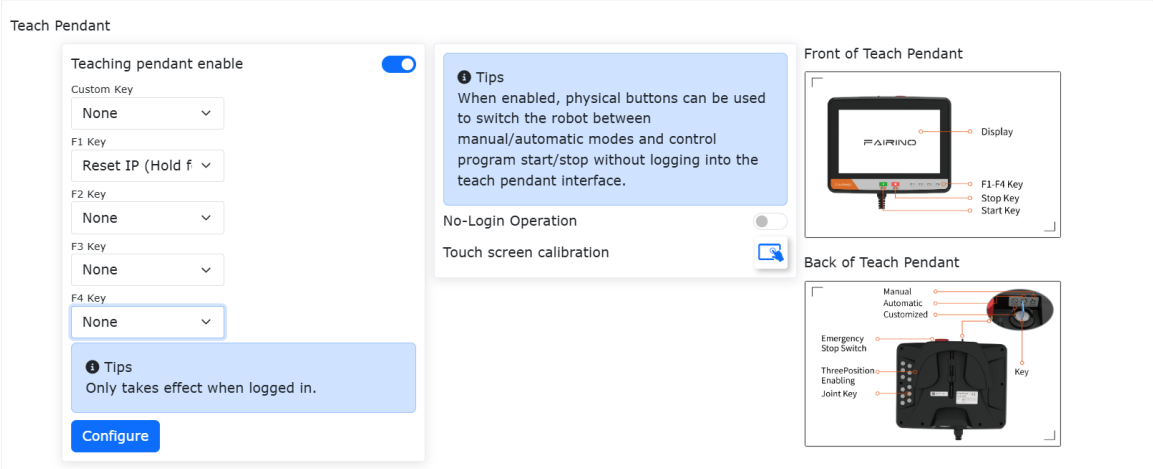


Figure 16.4-1 Teach Pendant Key Function Configuration Interface

3. After the teach pendant is enabled, it includes key custom function and F1-F4 key function configuration. The key custom function can be set to Drag mode. The F1-F4 keys can be configured for Reset IP (press and hold for 10 seconds), One-key Clear Errors, Output DO, Enable Switching, and Start Specified Lua Program.

1.3.17.4.2.2 Key Custom Set to Drag

1. When the key custom is configured to Drag mode, and while logged into the WebApp, if the teach pendant key is rotated to Custom, the interface will pop up a window requiring confirmation of the current load to prevent falling due to incorrect load;



Figure 16.4-2 Teach Pendant Mode Example

2. After confirming the load setting is correct, click Confirm, and the robot will enter Drag mode.

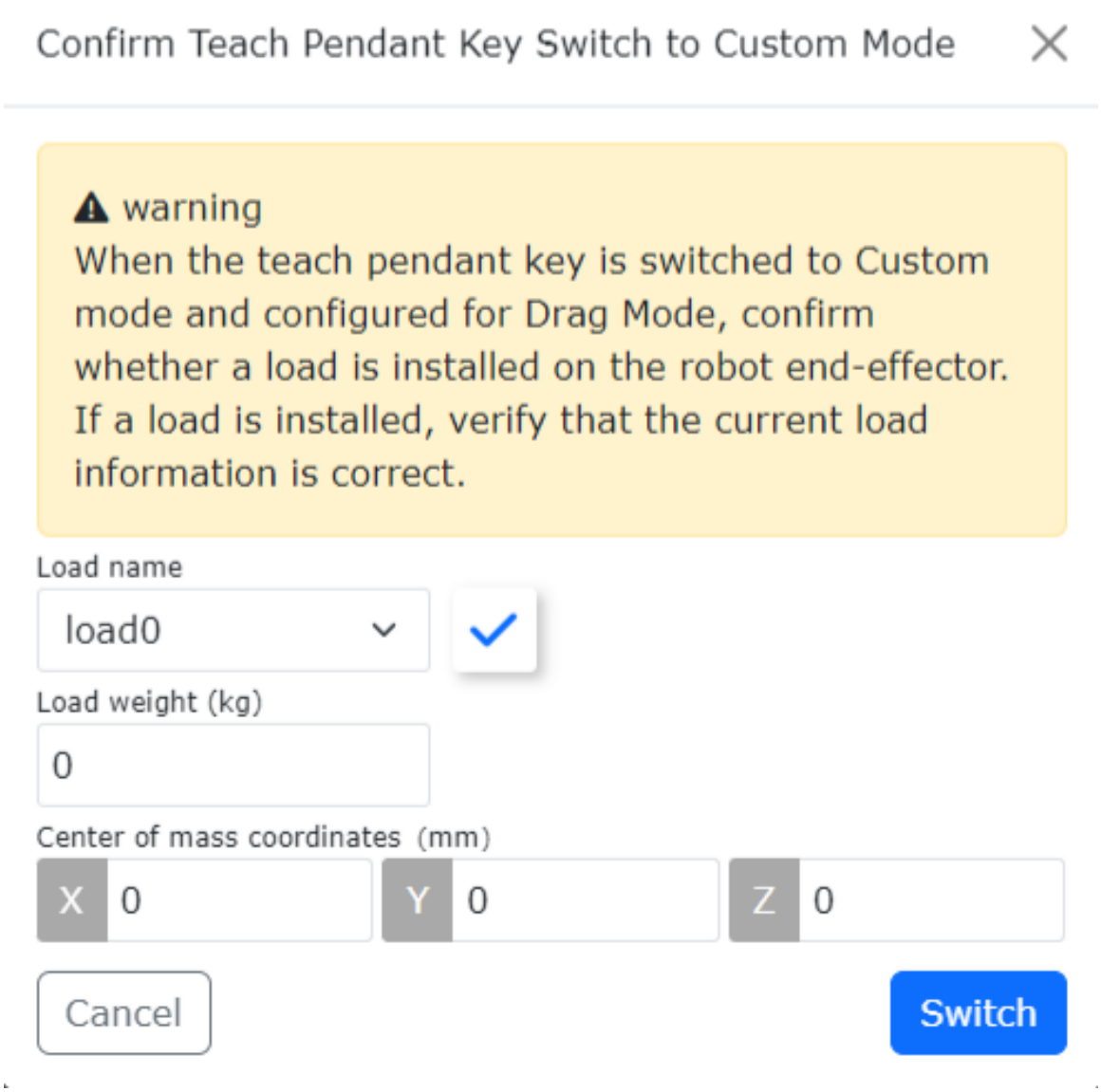


Figure 16.4-3 Confirm Load Before Setting Drag Mode

1.3.17.4.2.3 F1-F4 Key Function Customization

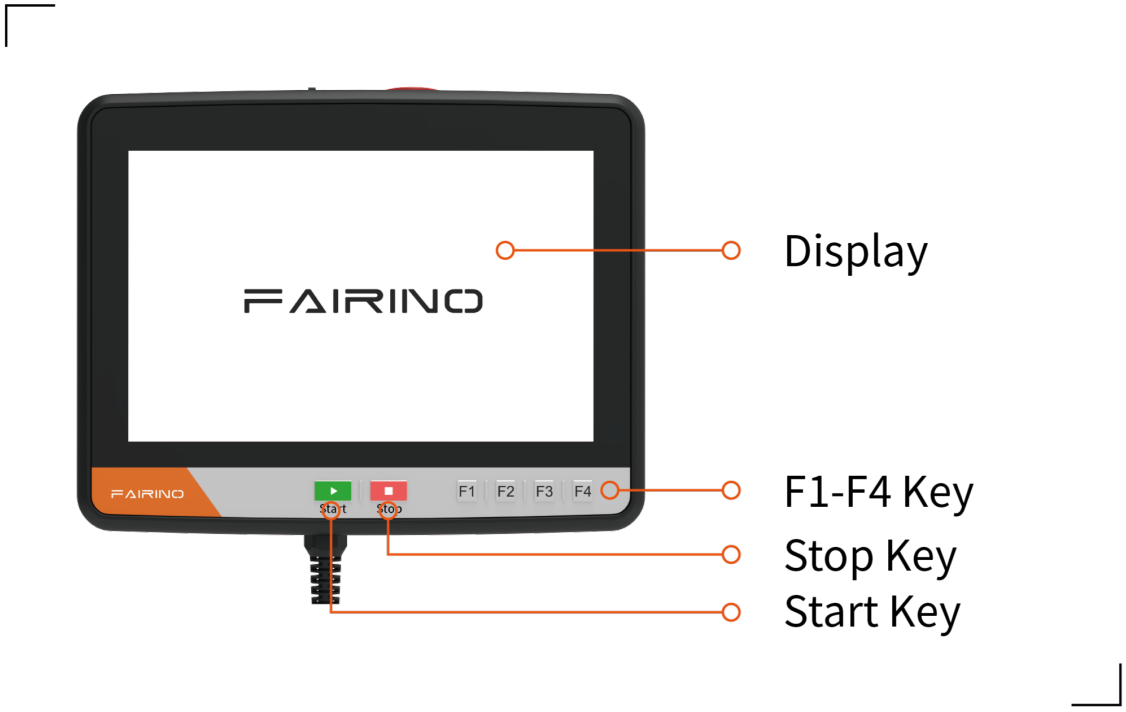


Figure 16.4-4 Teach Pendant Key Example

1. **Reset IP (press and hold for 10 seconds) function**: After configuration, pressing and holding for 10 seconds will prompt that a restart is required to take effect. After restarting, the controller's Network Card 0 IP will be restored to the default 192.168.57.2, Network Card 1 IP to the default 192.168.58.2, and the physical teach pendant IP to the default 192.168.58.77.
2. **One-key Clear Errors function**: When an error message appears on the interface, pressing the corresponding function F key can clear the error.
3. **Output DO function**: After configuring this function and setting the DO number, pressing the corresponding function F key can toggle the state corresponding to the current DO number.
4. **Enable Switching function**: After configuring this function, pressing the corresponding function F key can toggle the current enable state.
5. **Start Lua Program**: After configuring this function and setting the Lua program, pressing the corresponding function F key will cause the robot to automatically run the set Lua program in Automatic mode.

1.3.18 Custom protocol slave commands

1.3.18.1 Overview

1.3.18.2 Industrial Bus Protocol Integration for Robot Motion Control

To facilitate PLC-based robot motion control through various industrial bus protocols (CC-Link IEF Basic, Profinet, Ethernet/IP, and EtherCAT), the integrated mini control cabinet has been equipped with FRH-PCIeN-EC/EIP/CC/PN-RJ-V10 cards, FRJ-PCIeN-EIP/CC/PN-RJ-V10 cards, and FRJ-PCIeN-EC-RJ-V10 cards.

1.3.18.3 Environment Configuration

The required card models and software versions are as follows:

Protocol Type	Card Model	Robot Software Version
CC-Link IEF Basic	FRH-PCIeN-EC/EIP/CC/PN-RJ-V10, FRJ-PCIeN-EIP/CC/PN-RJ-V10	V3.8.0+
Profinet	FRH-PCIeN-EC/EIP/CC/PN-RJ-V10, FRJ-PCIeN-EIP/CC/PN-RJ-V10	V3.8.0+
Ethernet/IP	FRH-PCIeN-EC/EIP/CC/PN-RJ-V10, FRJ-PCIeN-EIP/CC/PN-RJ-V10	V3.8.0+
EtherCAT	FRH-PCIeN-EC/EIP/CC/PN-RJ-V10, FRJ-PCIeN-EC-RJ-V10	V3.8.4.1+

1.3.18.3.1 FRH-PCIeN-EC/EIP/CC/PN-RJ-V10 board hardware environment setup

1. Install the FRH-PCIeN-EC/EIP/CC/PN-RJ-V10 board into the integrated mini control box as shown.

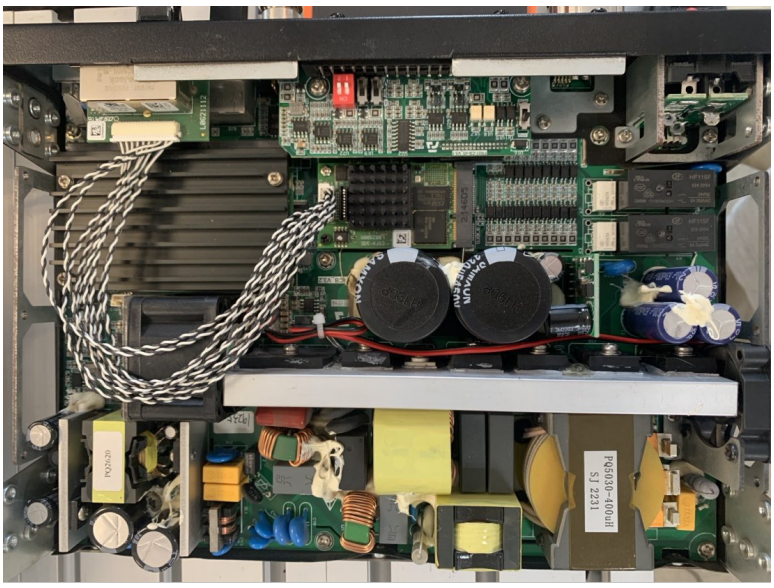


Figure 17.2-1 FRH-PCIeN-EC/EIP/CC/PN-RJ-V10 board installation



Figure 17.2-2 FRH-PCIeN-EC/EIP/CC/PN-RJ-V10 board network port

2. The robot control box and PLC wiring is shown below.



Figure 17.2-3 Control Box & Mitsubishi PLC Wiring Diagrams



Figure 17.2-4 Control Box & Siemens PLC Wiring Diagrams



Figure 17.2-5 Control Box & Omron PLC Wiring Diagrams

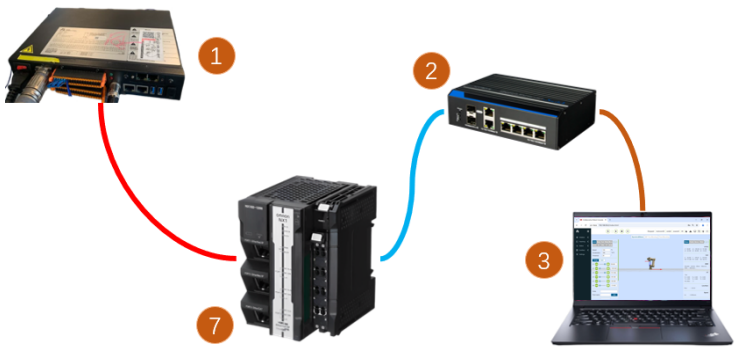


Figure 17.2-6 Control Box & Omron PLC Wiring Diagrams

Note: 1: Robot control box (board input network port); 2: Switch; 3: PC; 4: Mitsubishi PLC (CC-Link IEF Basic network port); 5: Siemens PLC (Profinet network port); 6: Omron PLC (Ethernet/IP network port); 7: Omron PLC (EtherCAT network port);

Important: When the protocol is switched to EtherCAT bus, the board's network port needs to be distinguished as EtherCAT_IN and EtherCAT_OUT. At this time, the PLC's EtherCAT network port is directly connected to the board's EtherCAT_IN through a network cable.

1.3.18.3.2 FRJ-PCIeN Board Hardware Setup

1. Install the board into the integrated mini control box as shown.



Figure 17.2-7 FRJ-PCIeN Board Ethernet Port

2. Wiring between robot control box and PLC is shown below.



Figure 17.2-8 Control Box & Mitsubishi PLC Wiring Diagram



Figure 17.2-9 Control Box & Siemens PLC Wiring Diagram



Figure 17.2-10 Control Box & Inovance PLC Wiring Diagram

Note: 1: Robot control box (board Ethernet port); 2: Switch; 3: Laptop PC; 4: Mitsubishi PLC (CC-Link IEF Basic

port); 5: Siemens PLC (Profinet port); 6: Inovance PLC (Ethernet/IP port);

3. Firmware upgrade is required when switching protocols on FRJ-PCIeN board. For upgrade: - Set PC IP to "192.168.0.xxx" - Open "Gateway Toolset" software - Select PC network adapter - Click "Start" (bottom right) - Click "Search" (top right) to find board devices

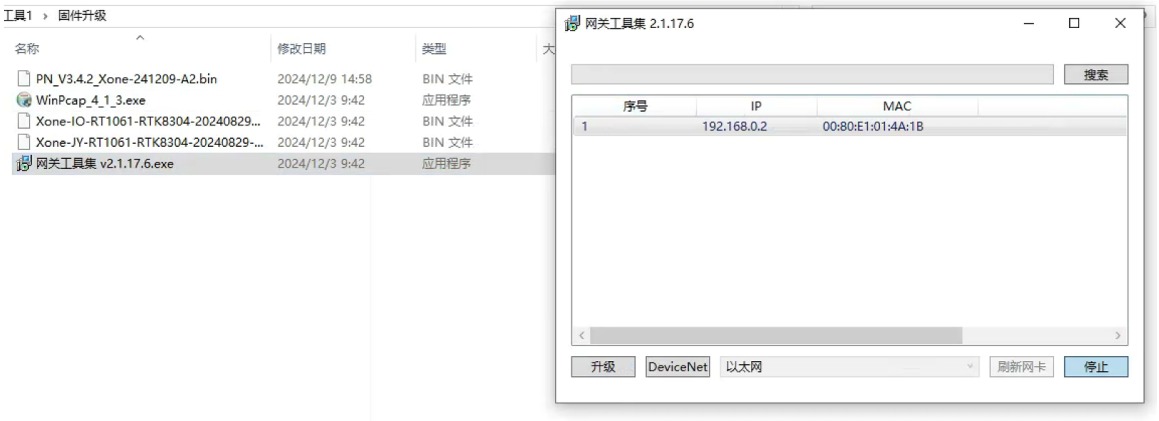


Figure 17.2-11 Connecting Board Device

4. Click "Upgrade" (bottom left) - Select board device - Click "…" (top right) to choose protocol firmware - Click "Upgrade" and wait for completion

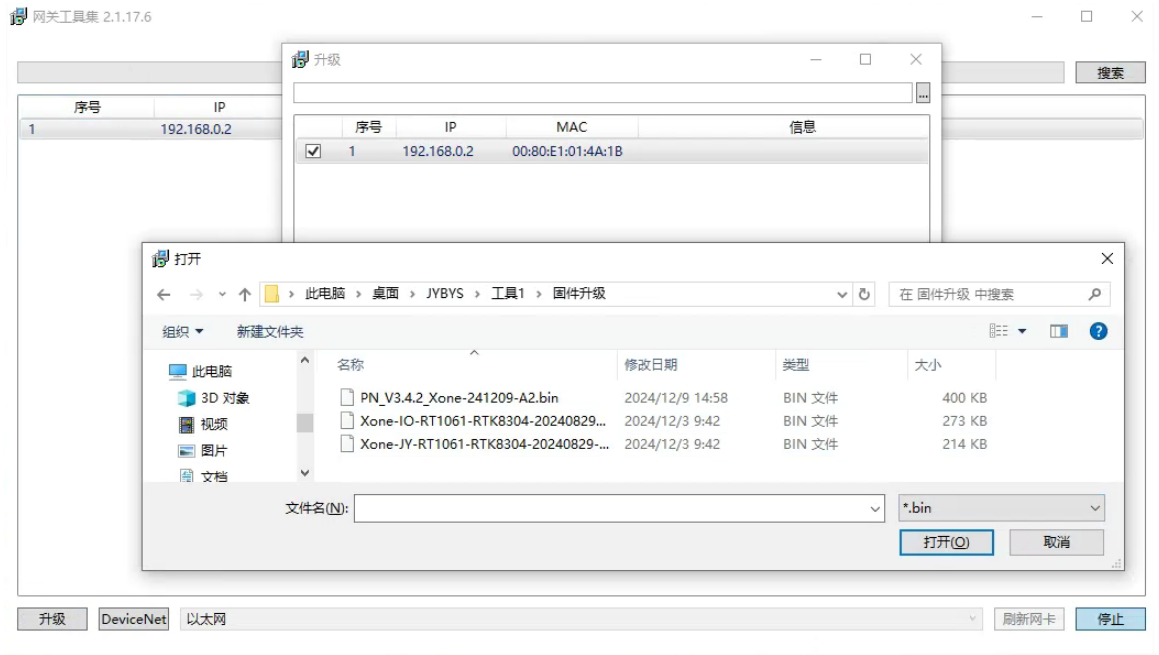


Figure 17.2-12 Board Protocol Switching

Note: IP address changes after protocol switching as shown below.

Table 17.2-1 Board IP Addresses

Protocol	IP Address
CC-Link IEF Basic	192.168.0.113
Ethernet/IP	192.168.0.112
Profinet	192.168.0.2

When configured for CC-Link IEF Basic, controller changes board IP to “192.168.0.113”.

When configured for Ethernet/IP, controller changes board IP to “192.168.0.112”.

When switching to Profinet, if slave device name matches master, master will automatically configure slave IP.

5. FRJ-PCIeN-EC-RJ-V10 Board Firmware Upgrade

Enter the URL 192.169.58.2 to access the robot interface. Click “Initial Settings” -> “Peripherals” -> “Board Communication” interface to obtain the firmware version number of the FRJ-PCIeN-EC-RJ-V10 board. Select the bin file to be upgraded, click Upload, wait for the firmware upgrade to succeed, and then restart the control box.

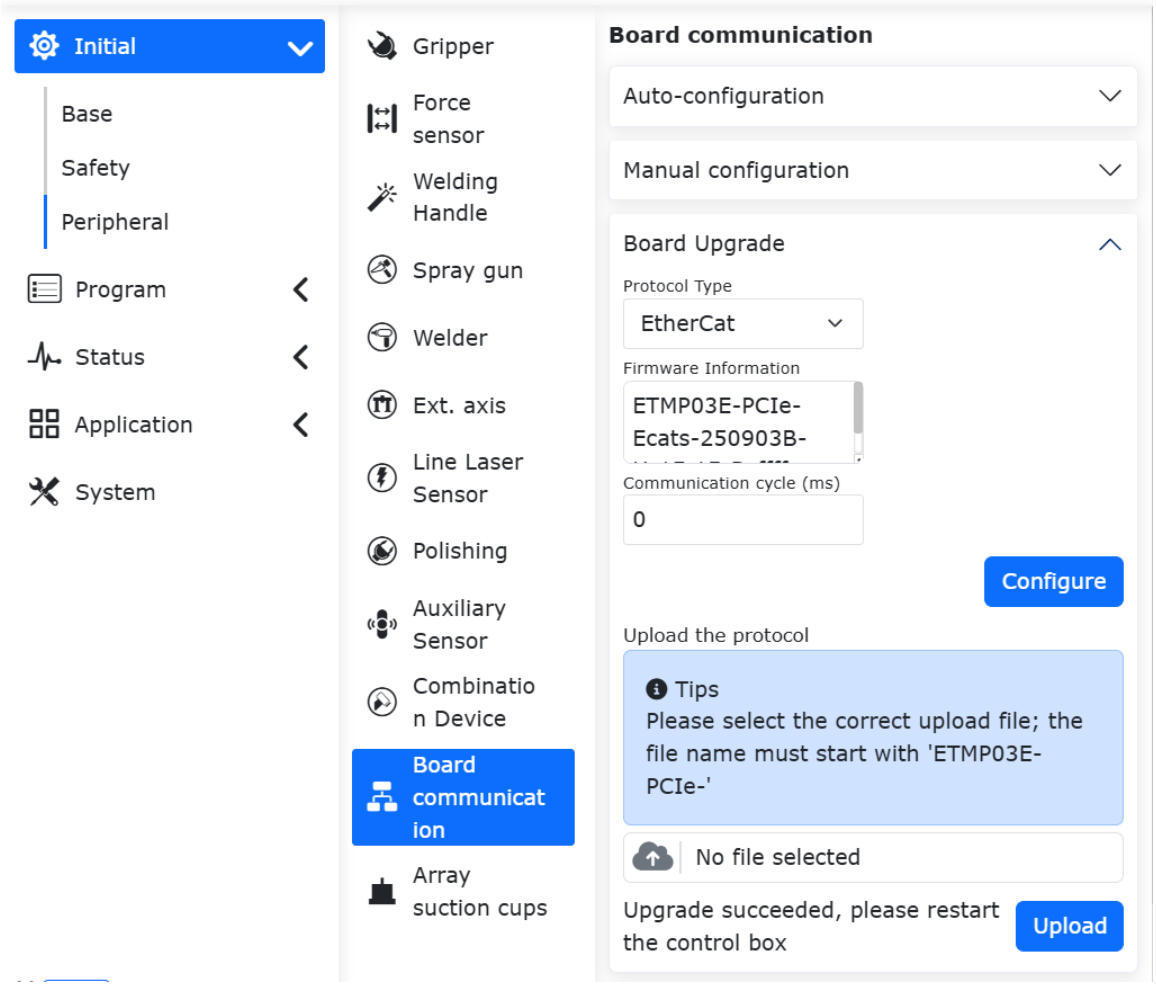


Figure 17.2-13 Board Firmware Upgrade

Note:

1. Only V3.9.2 and later versions support EtherCAT protocol firmware upgrade; 2. Upgrading the EtherCAT protocol firmware requires unloading any currently running open protocols.

1.3.18.3.3 Software environment setup

1. Browser IP input 192.168.58.2, account for admin, password for 123, click 'Login', enter the robot control box Web interface.

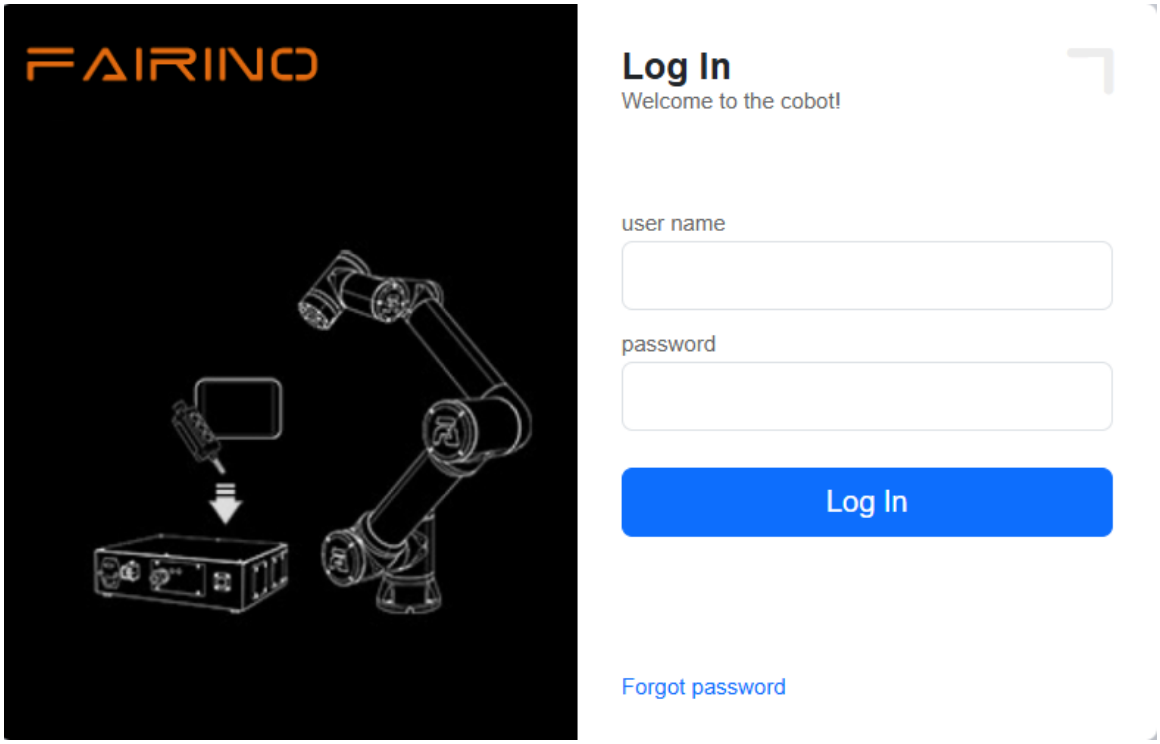


Figure 17.2-14 Web Login Interface

2. Click System Settings -> About Interface, click the Software Upgrade button, select the software.tar.gz file, and upload the upgrade package.

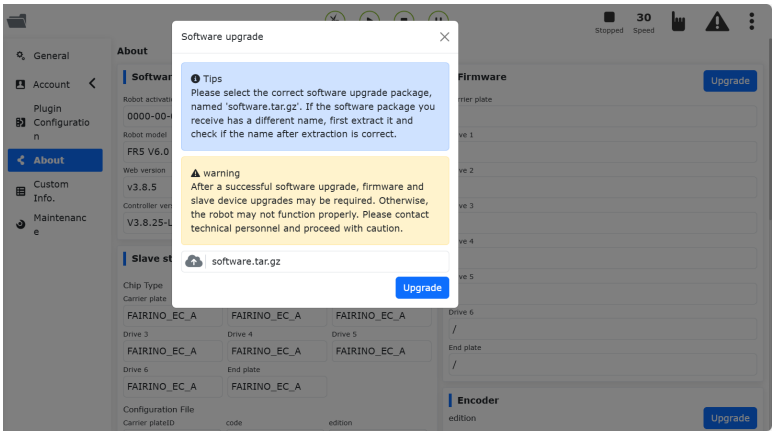


Figure 17.2-15 Upgrade software

Note: QX control box web version needs 3.8.0 and above, LA control box web version needs 3.8.0 and above.

3. Click the extension button in the upper right corner and switch from ‘Local Mode’ to ‘Remote Mode’.

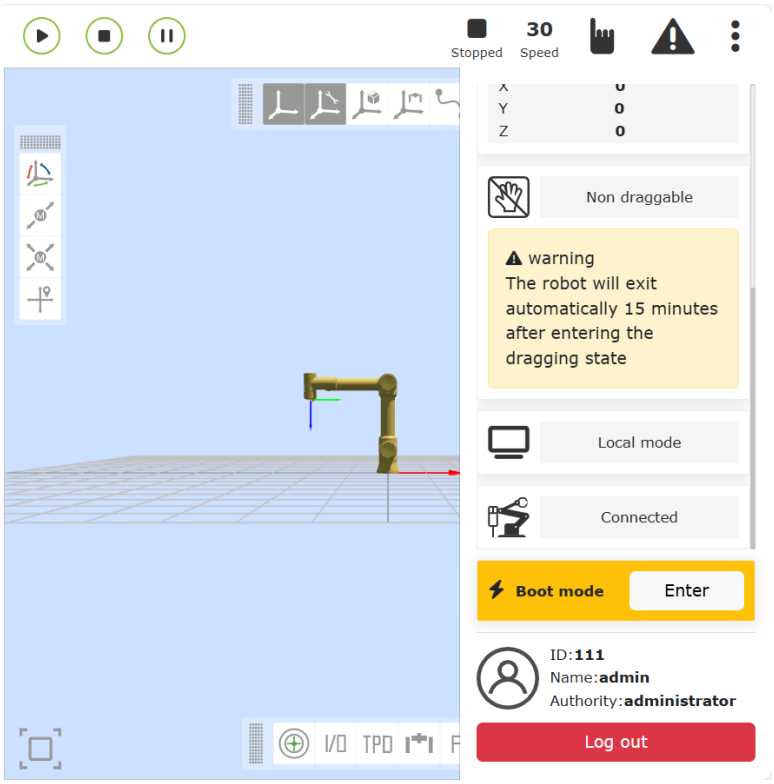


Figure 17.2-16 Switch remote mode

4. Select the controller slave protocol and whether the auto-start function is required, then click the “Set” button. Note: To switch between different protocols, you need to click the “Uninstall” button first before configuring other protocols.

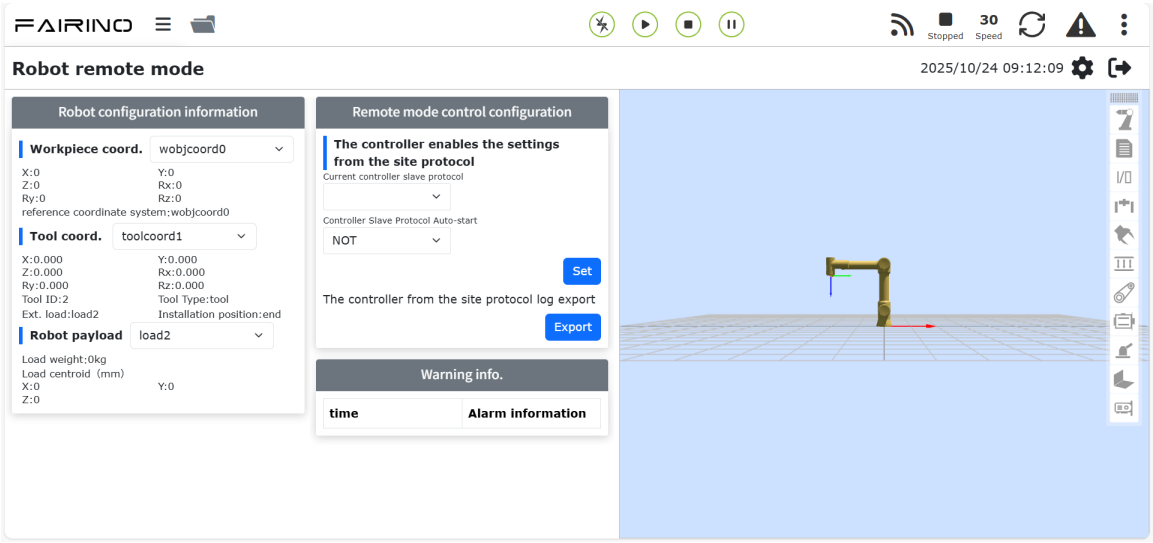


Figure 17.2-17 Configure the communication protocol

Note: Switching different protocols requires restarting the control box before configuring the protocols.

1.3.18.3.4 PLC Environment construction

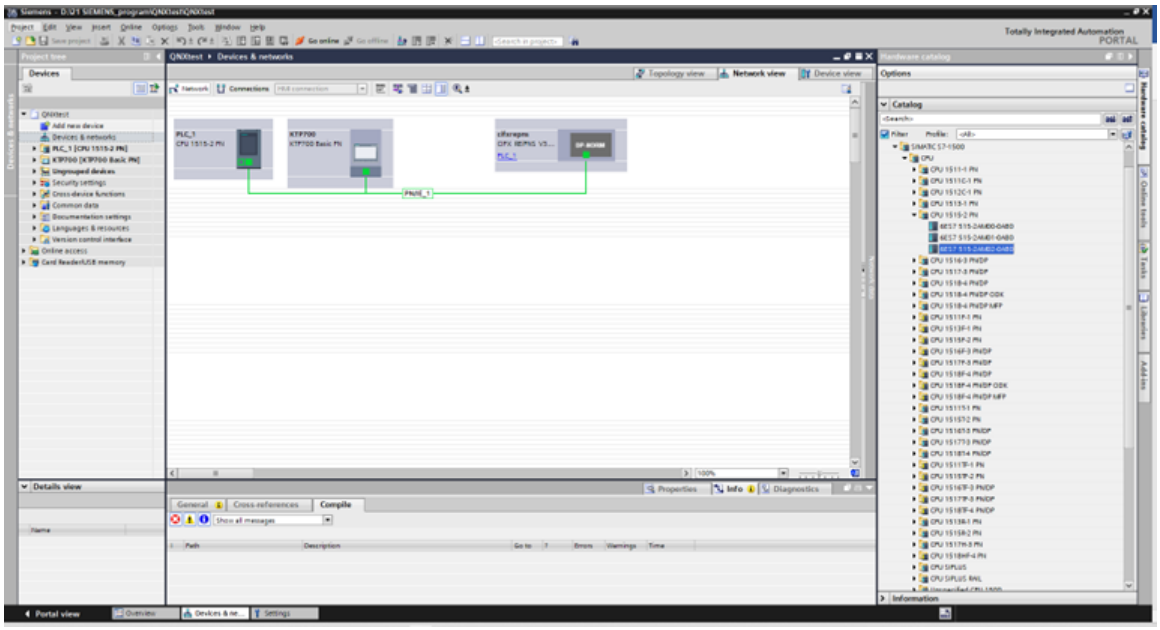
The test environment built to implement the slave commands for each protocol is shown in the table below, which includes the PLC model, firmware version and test software used in each protocol.

Protocol	Brand	Type	Firmware	Software
Profinet	Siemens	CPU 1515-2 PN	6ES75152AM020AB(	TIA Portal V17
CC-Link IEF Basic	Mitsubishi	FX5S-30TR/DS	30MR/ES V1.3	GX Works3 V1.097B
Ethernet/IP	Omron	MX102-1100	V1.3	Sysmac Studio V1.50
EtherCAT	Omron	MX102-1100	V1.3	Sysmac Studio V1.50

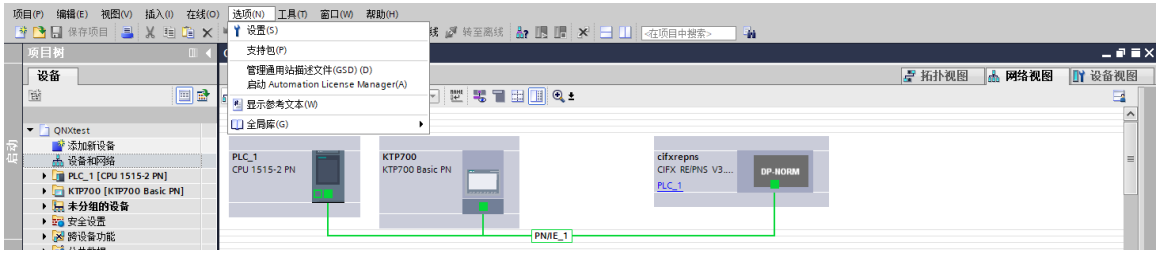
1.3.18.3.4.1 Siemens Profinet

1. GSD file (XML file) importing

Open Siemens programming software TIA Portal V17, create a new PLC project, select 'Devices and Networks', and select 'Hardware Catalogue' on the right side to add PLC module by double clicking 6ES7 515-2AM02-0AB0 to add PLC module.

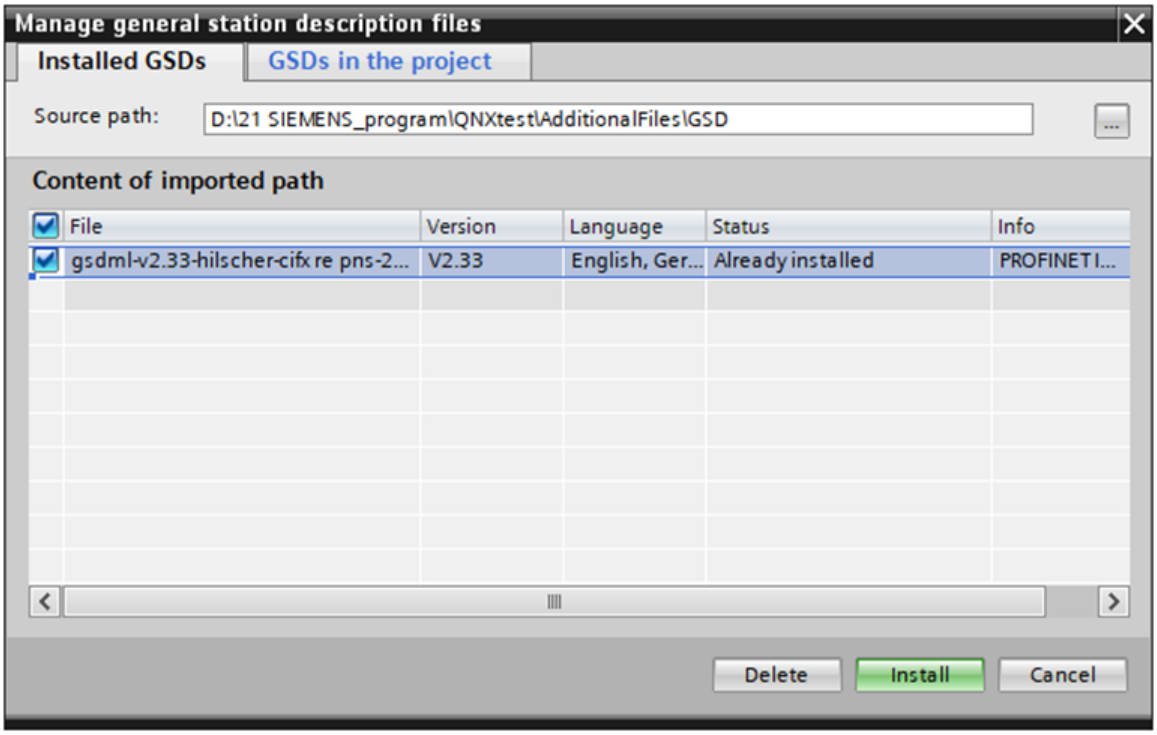


In the TIA PORTAL software, select Options-> Manage Generic Station Description File (GSD) in the menu bar to install or remove a GSD file that has already been installed.

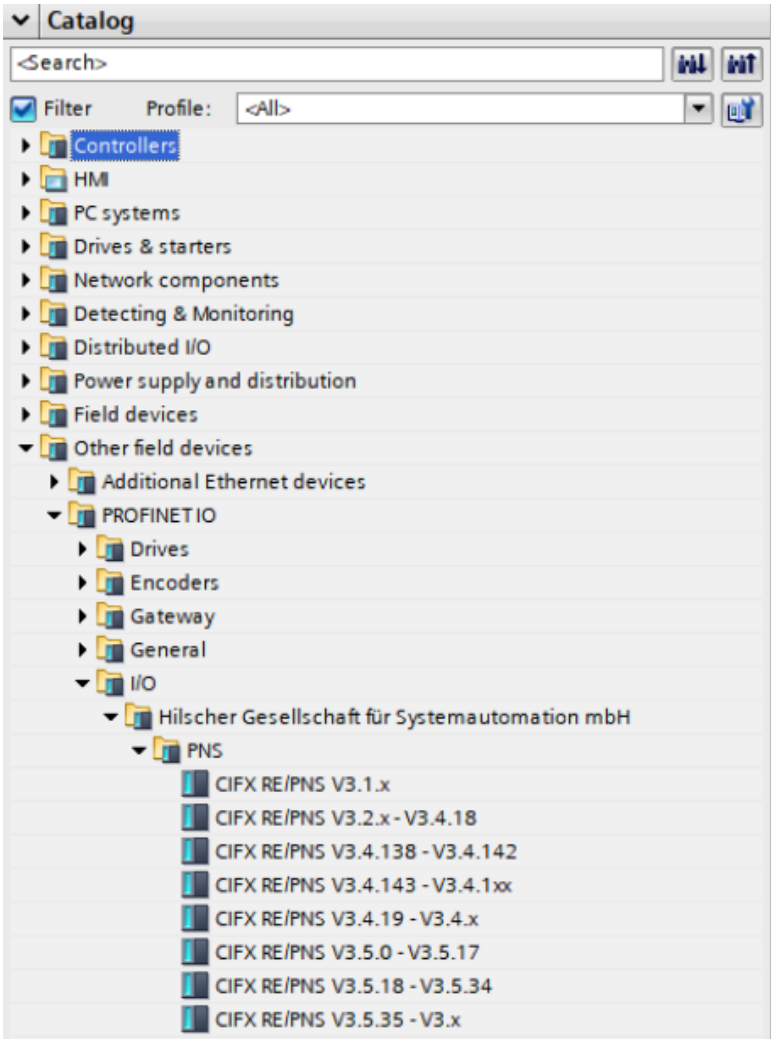


As an example, to install the Herschel GSD file, select 'Manage Generic Station Description File (GSD)' as above, and the 'Manage Generic Station Description File' window will appear.

Select the folder where you want to install the GSD file from the 'Source Path', select one or more files to install from the list of displayed GSD files, and click the 'Install' button. Click the Install button as shown in the following figure.

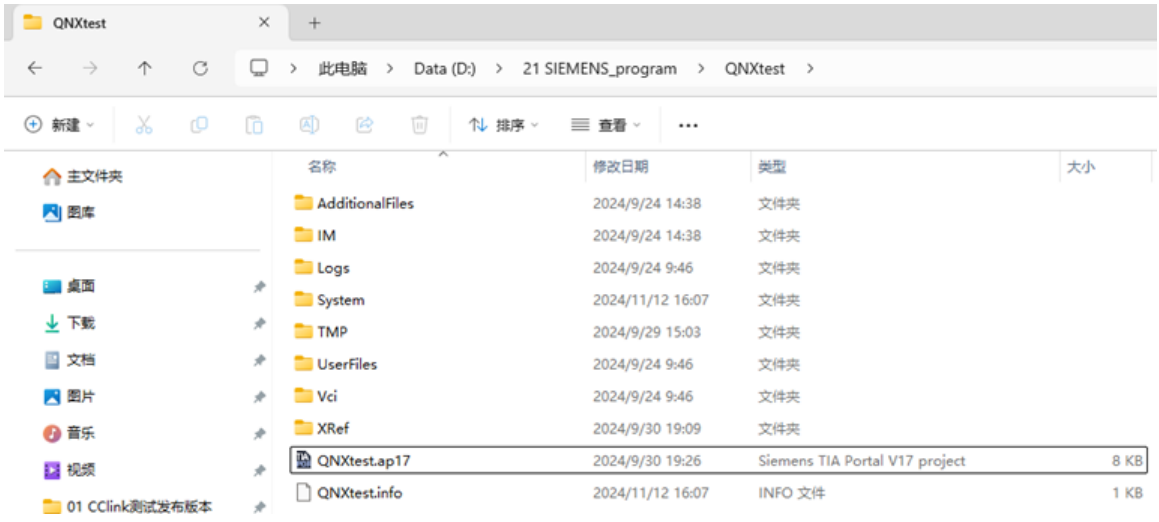


After successful installation, you can find the device with the installed GSD file in the hardware catalogue, other field devices, as shown in the figure below.



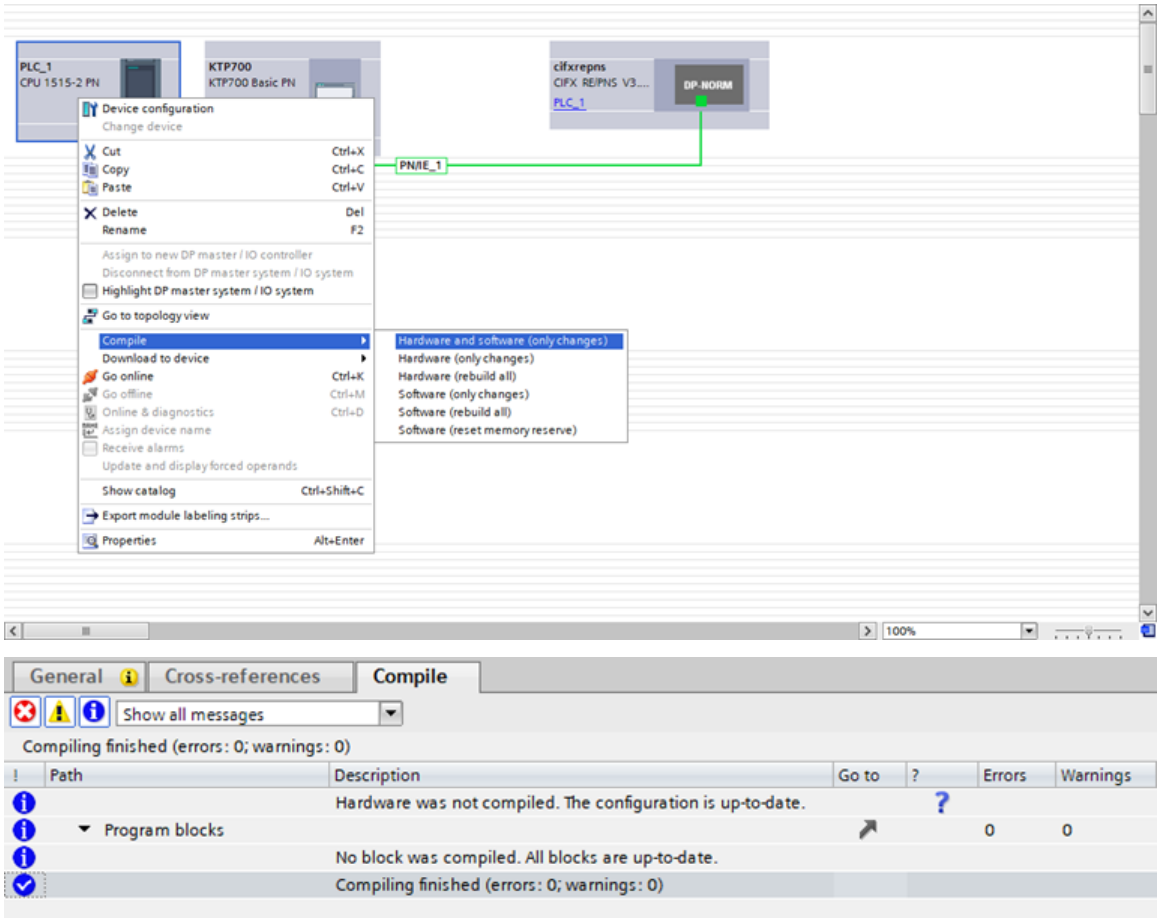
2. Executable programme

Open the project ‘QNXtest’.

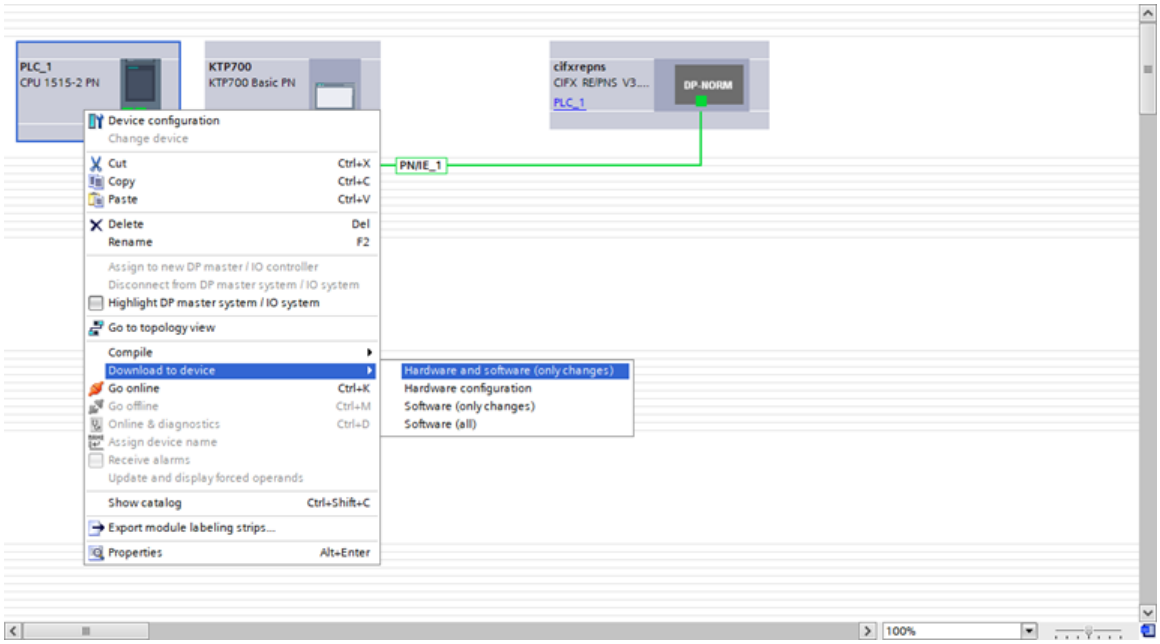


Compiler: Double-click on the left side of the project tree to enter ‘Devices and Networks’, right-click on the ‘PLC_1’

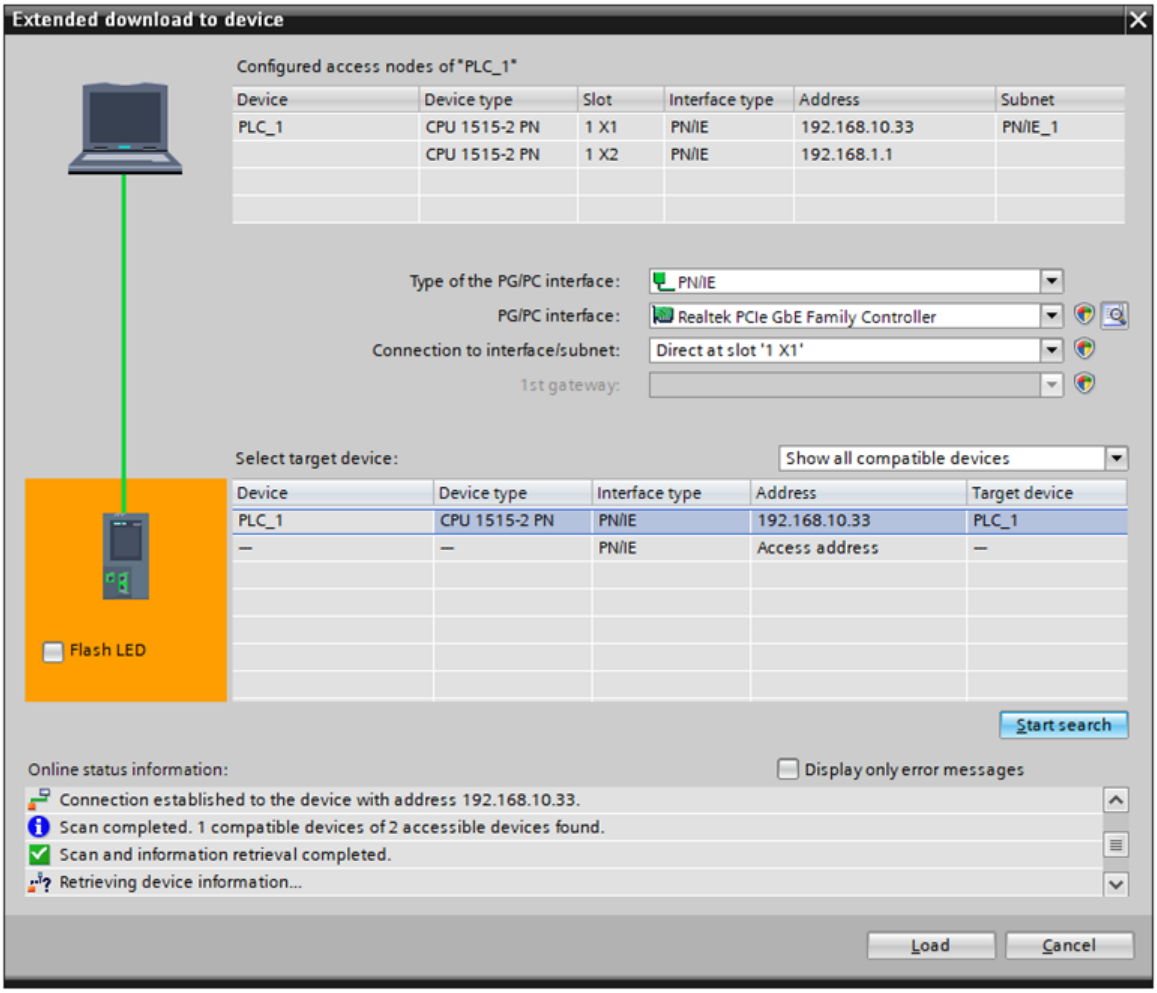
module, select Compile from the drop-down menu, and then select 'Hardware and Software (Changes Only)' from the stand-alone menu. Hardware and Software (Changes Only)'. After the compilation is completed, it will prompt 'Compilation complete' at the bottom of the software view.

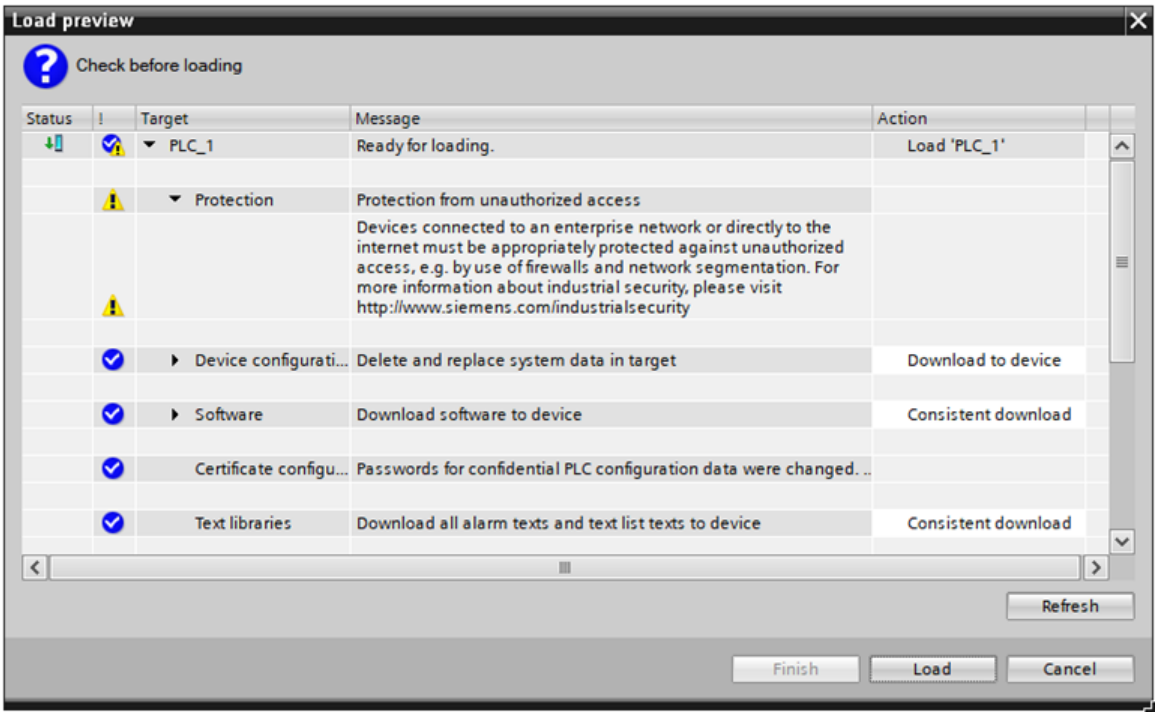


Download the programme to the device: Double click on the left side of the project tree to enter the 'Device and Network', right click on the 'PLC_1' module, and select 'Download to Device' from the drop-down menu. 'Download to Device', "Hardware and Software (change only)".



Search and download devices: After the pop-up window, configure the PG/PC interface type as shown in the following figure, click Start Search, select the device that needs to download the programme, and click Download.

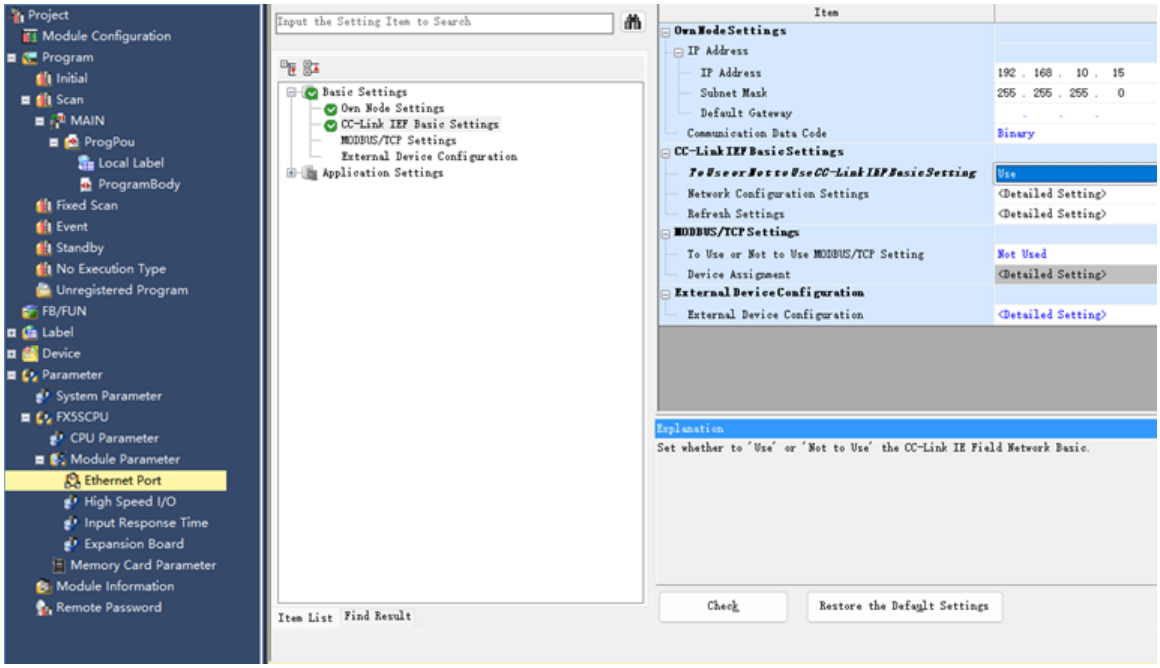




1.3.18.3.4.2 Mitsubishi CC-Link IEF Basic

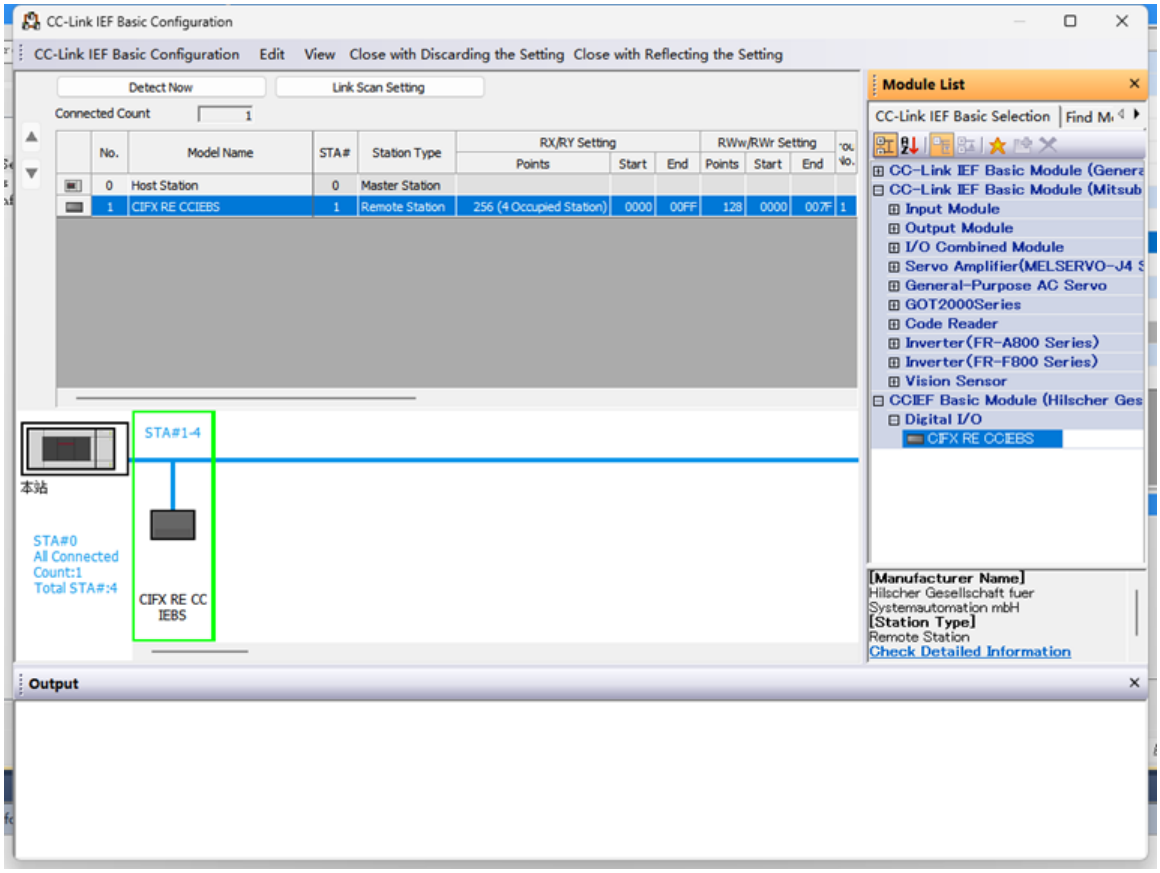
1. CC-Link IEF Basic Setup

Enable CC-Link IEF Basic: Select 'Ethernet Port' in the left menu bar, set the ip address of PLC and make sure it is in the same network segment as the address of Huexun card. Click 'CC-Link IEF Basic' and select 'Use'.

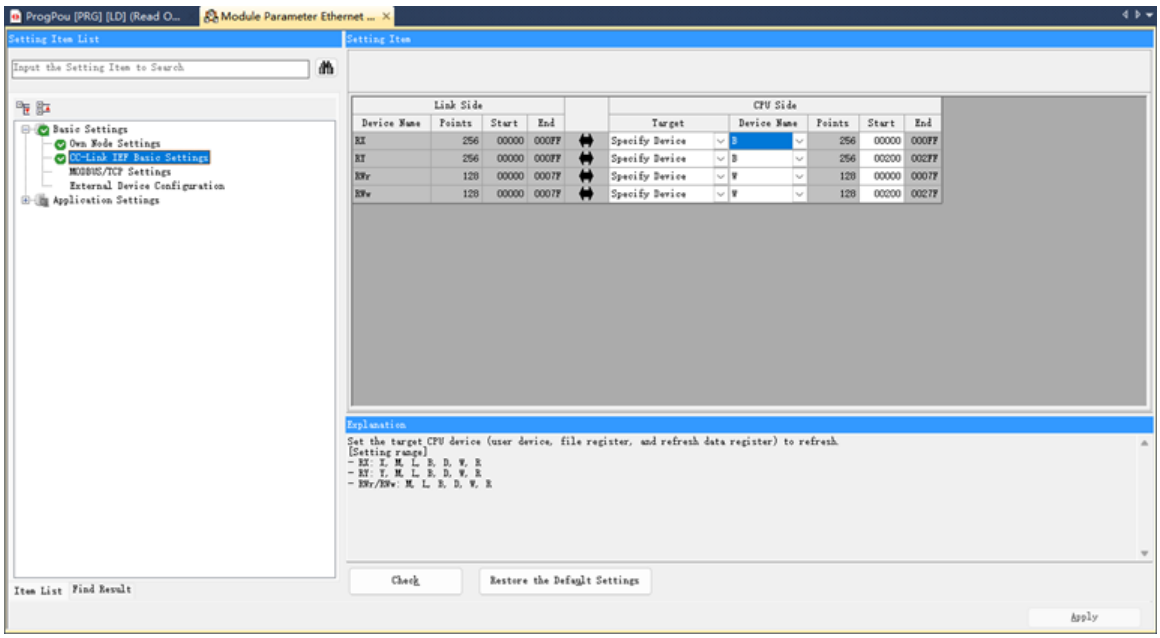


CC-Link IEF Basic Network Configuration Settings: Also in CC-Link IEF Basic Settings, select 'Network Configuration Settings', and choose Hueyoson CIFX Digital I/O module. Drag and drop the module to the bottom left of the

view to complete the hardware configuration.

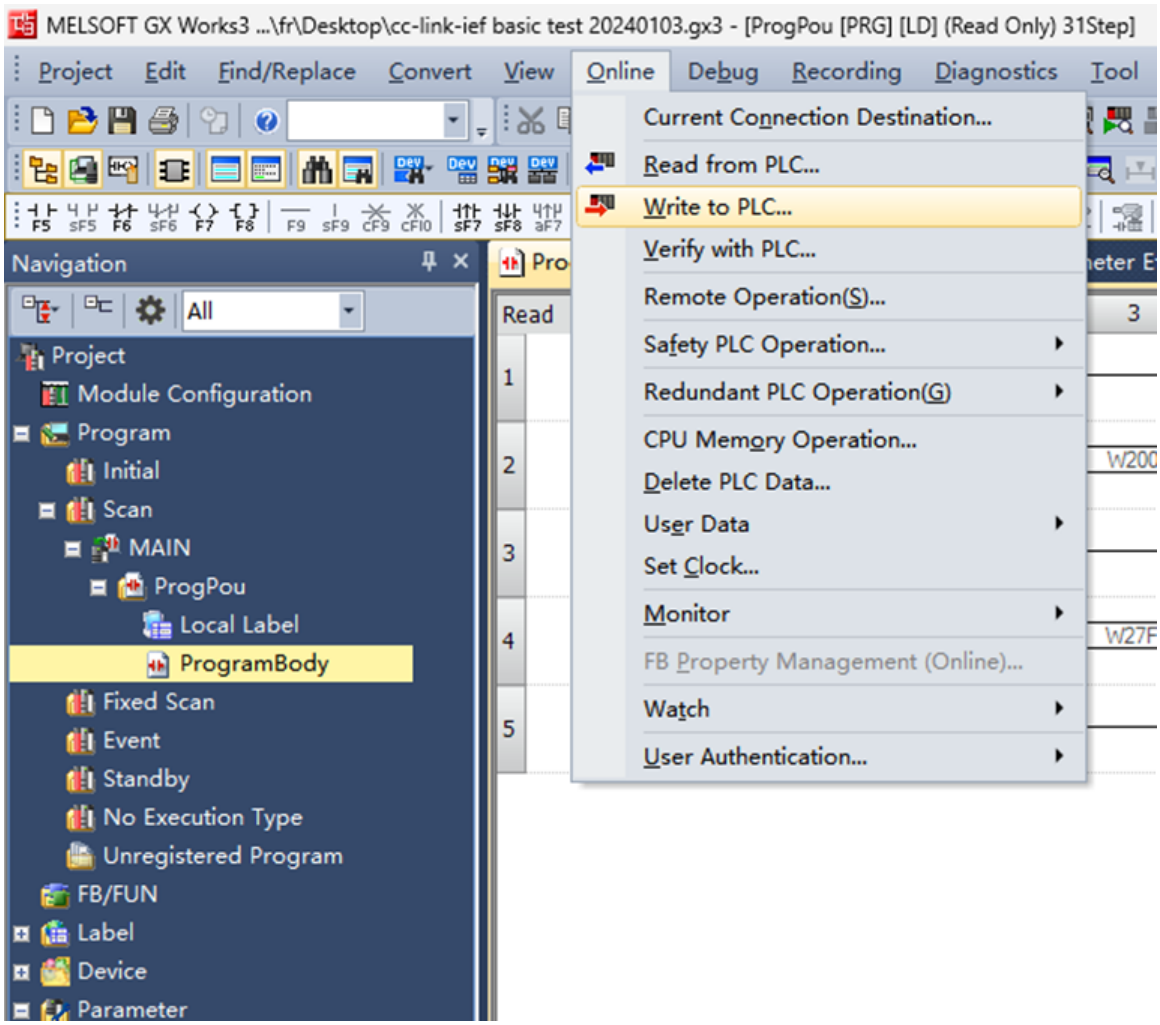


CC-Link IEF Basic Refresh Settings: Also in CC-Link IEF Basic Settings, click Refresh Settings to customise the transmission settings: 256 bytes receive, 256 bytes transmit.

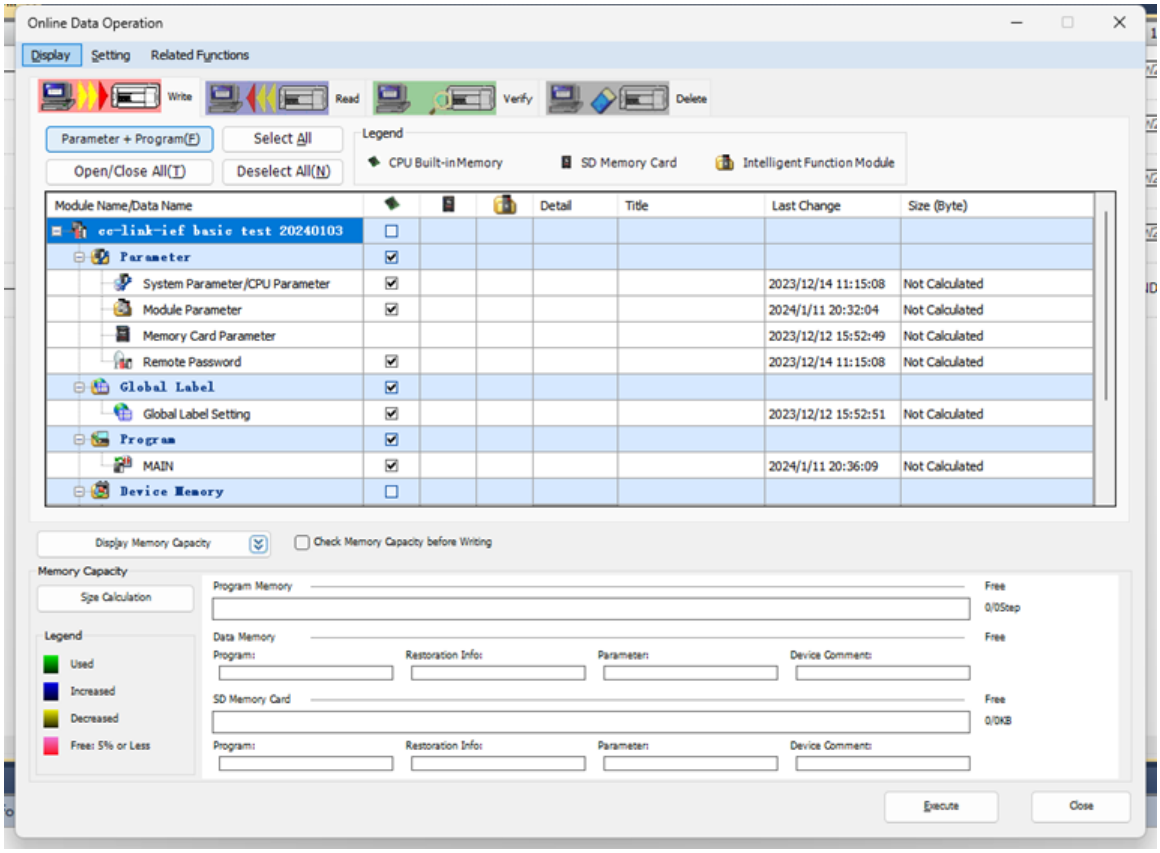


2. Program Download

After opening the test programme, click 'Online'->'Write to Programmable Controller' to enter the download interface.



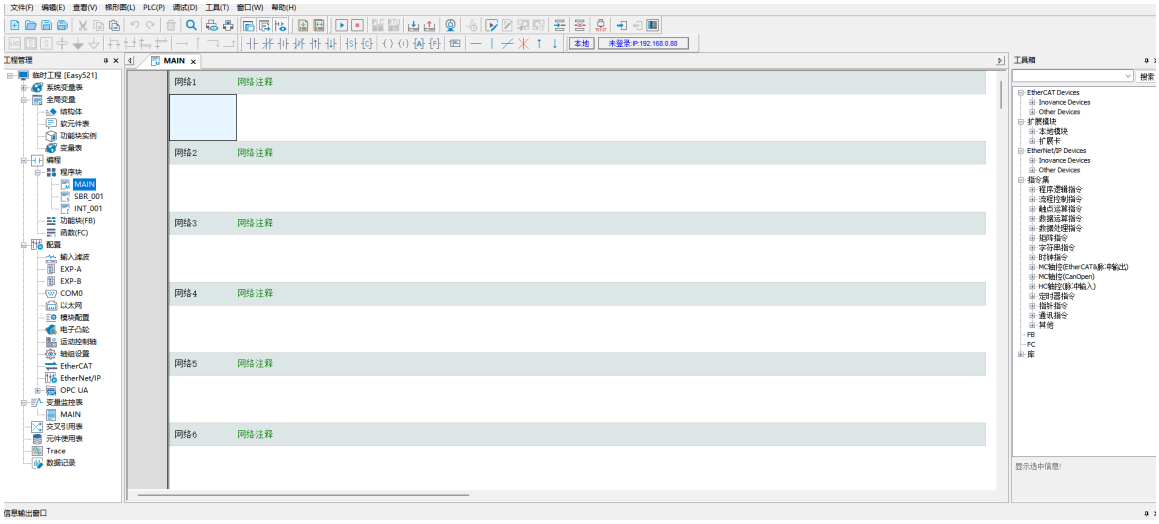
After opening the download interface, click 'Parameter+Programme' on the top left, then click 'Execute' on the bottom right corner to download, wait for the download to complete.



1.3.18.3.4.3 Inovance EtherCAT Configuration

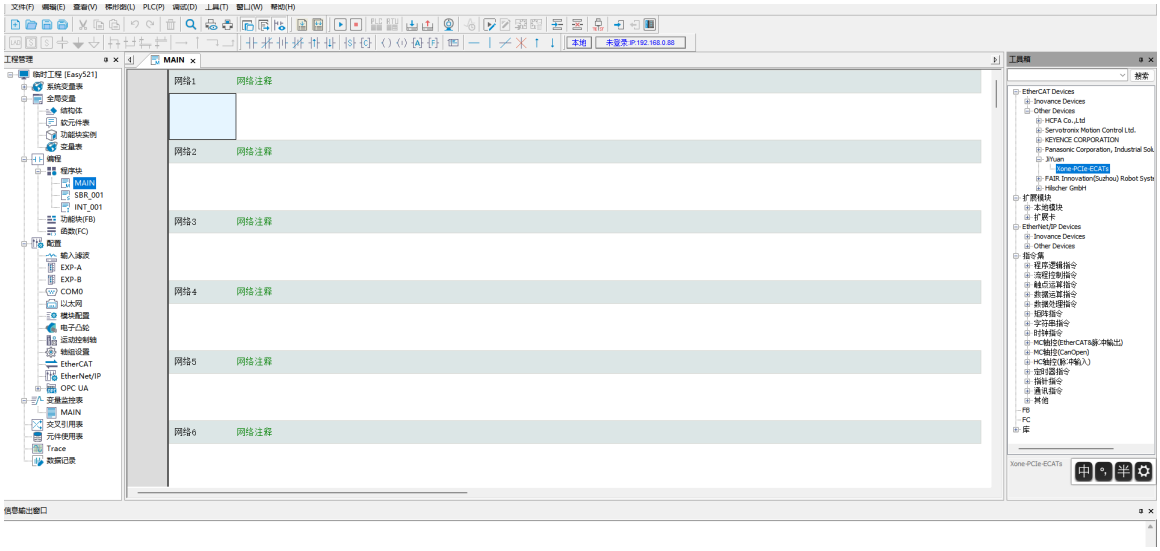
1. XML File Import

Open Inovance AutoShop programming software and create a new PLC project. Select “EtherCAT Devices” from the right toolbox:



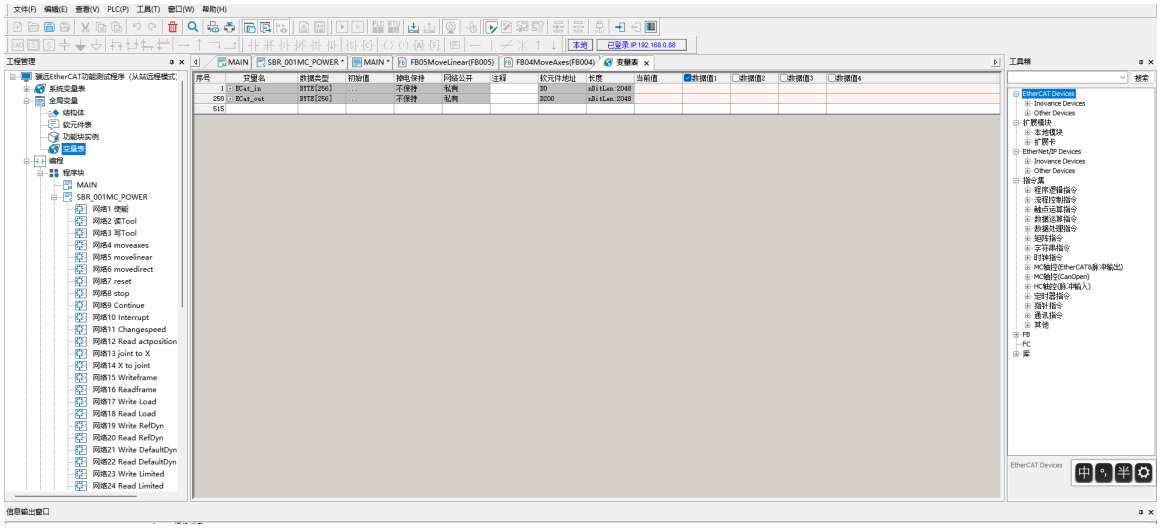
Right-click after selecting “EtherCAT Devices” to open the “Import Device XML” dialog. Locate the folder containing the card’s XML file. After successful import, the card name will appear under “EtherCAT Devices”. Close and reopen

the project to complete the import process.

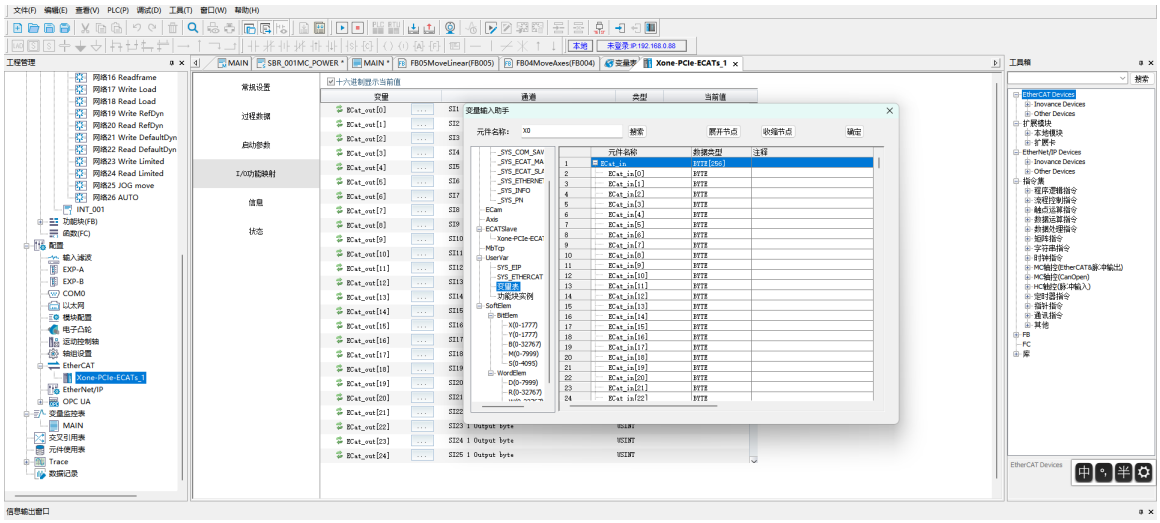


2. Variable Mapping

Double-click the variable table in the left toolbar. Create: - 256-byte input array (Soft element address: D0) - 256-byte output array (Soft element address: D200)

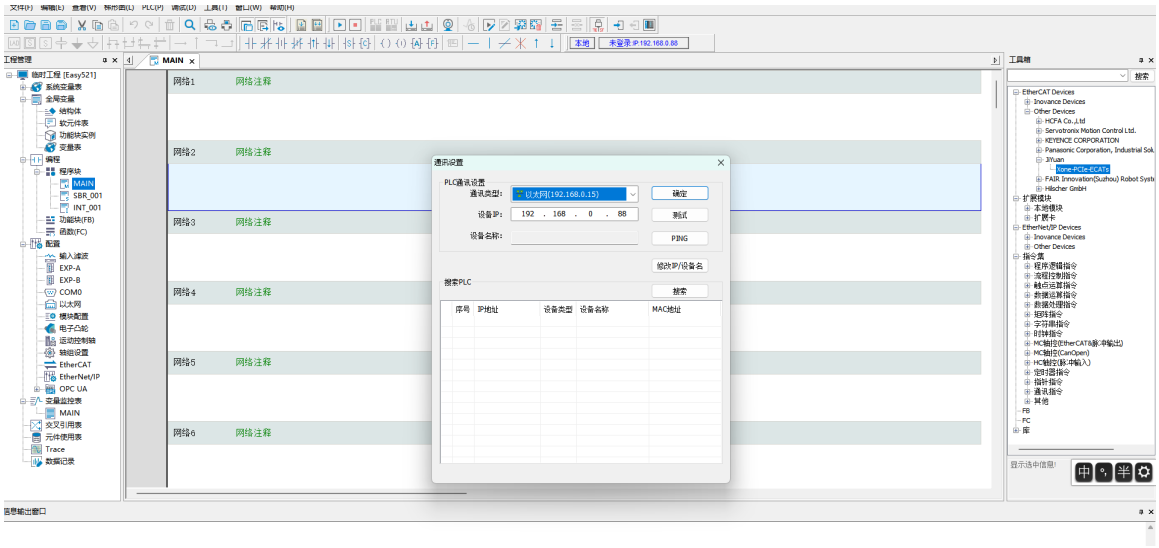


Under “EtherCAT” in the left toolbar, double-click “Xone-PCIe-ECATs”. In the dialog, click “I/O Mapping”, then bind variable addresses by selecting from the variable table. Repeat sequentially for other addresses.

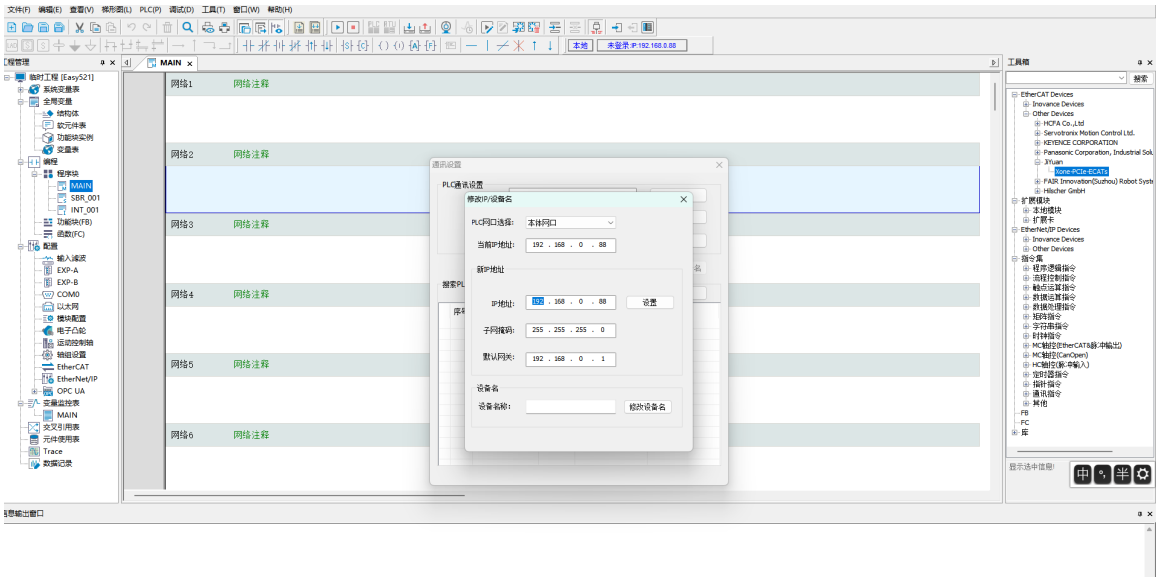


3. Program Download

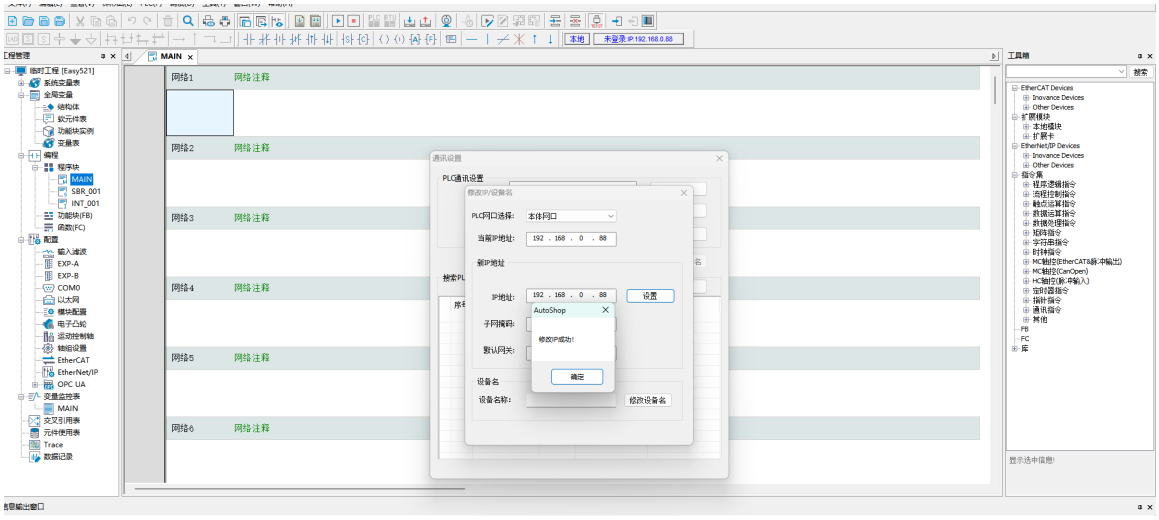
Open the test program and change PLC IP from default “192.168.1.88” to “192.168.0.88”:



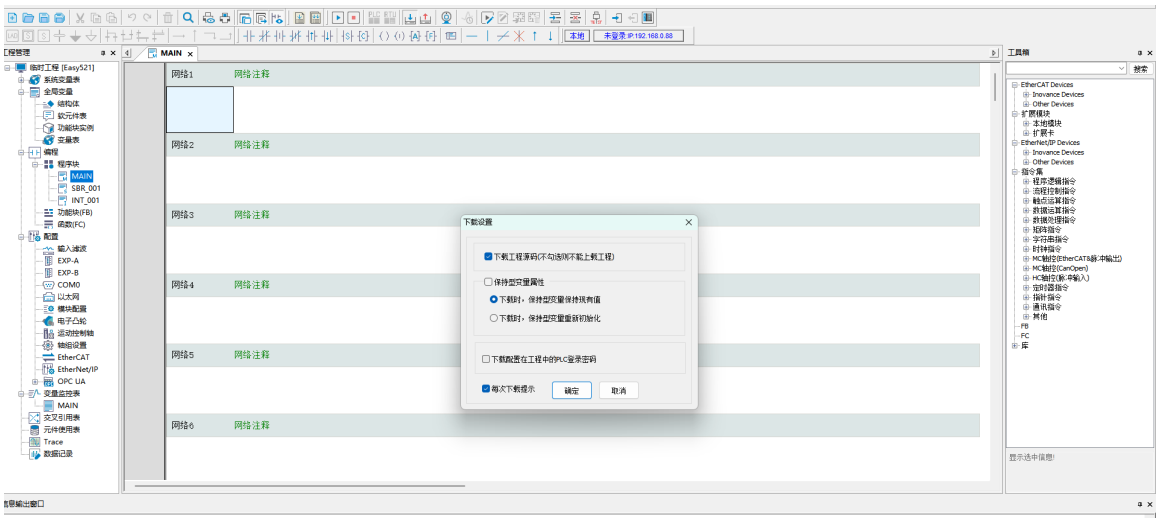
Click “Modify IP/Device Name” and update both IP and gateway to “192.168.0.88”:



Confirm modification by clicking “Yes” in the popup dialog when clicking “Modify IP”.

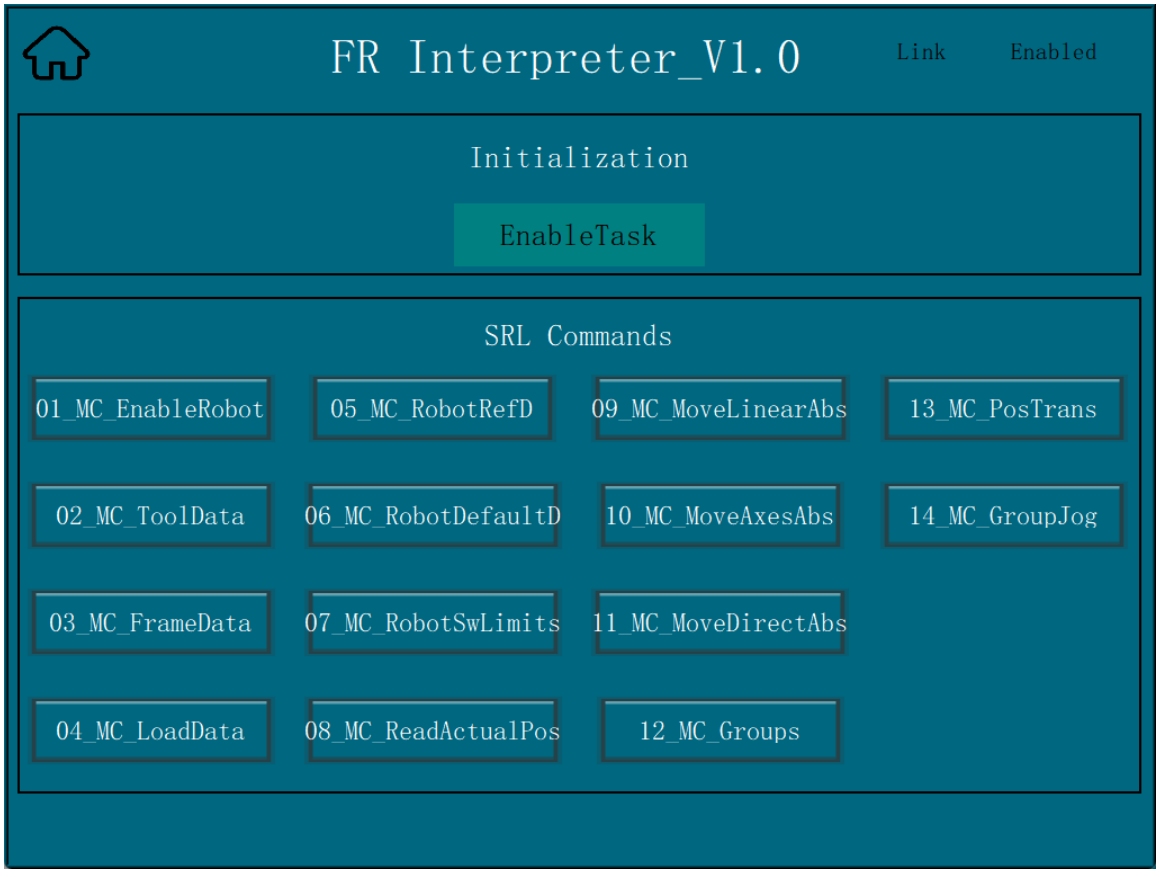


After successful communication, download the PLC program.

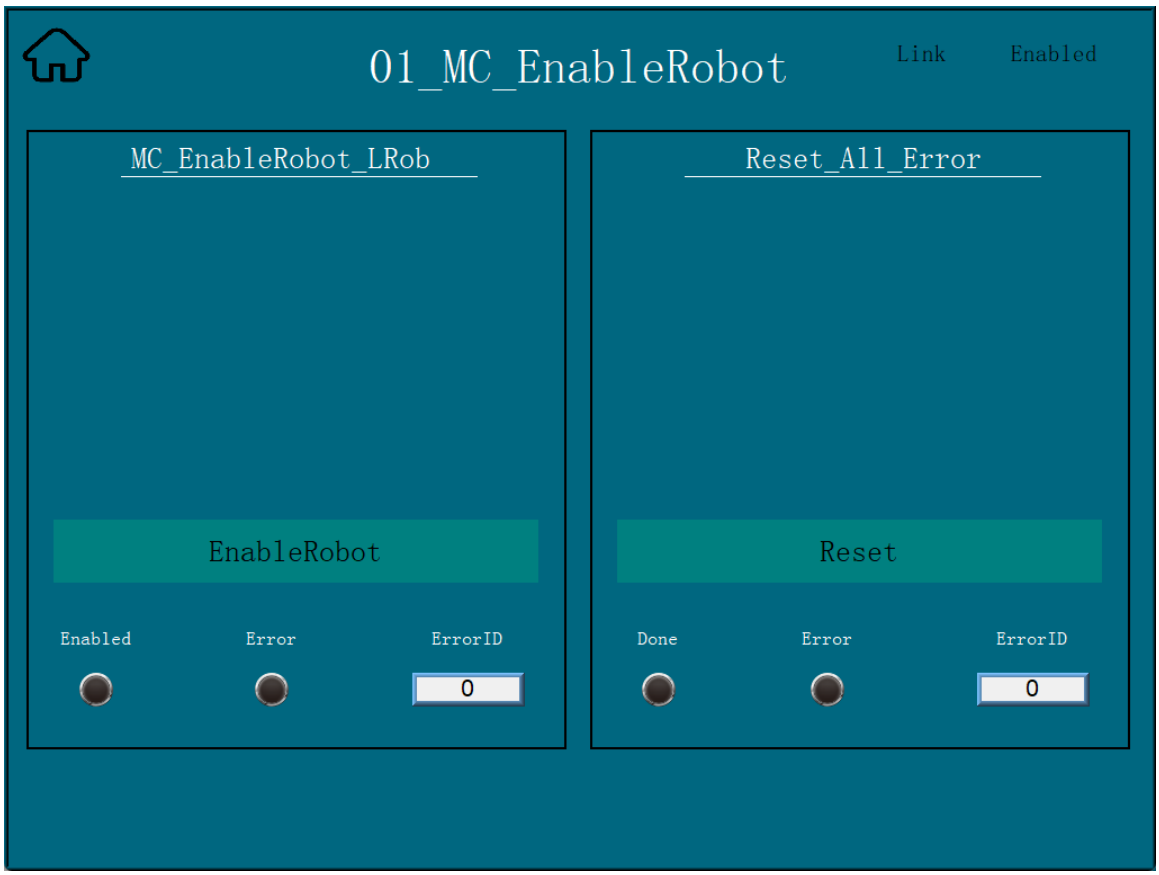


1.3.18.3.5 HMI setting (CC-Link IEF Basic emulation)

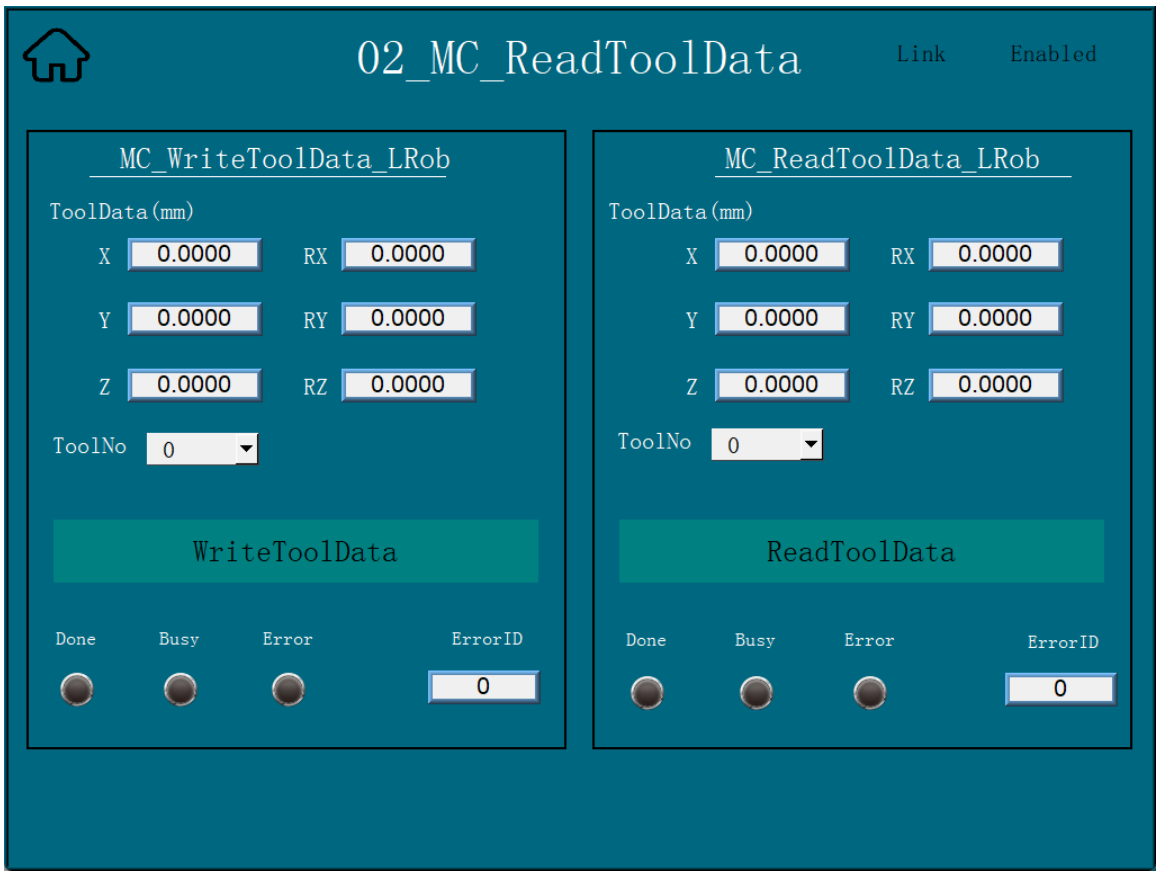
1. After logging into the HMI interface, enable 'Enable Task' to establish the communication connection between PLC and controller.



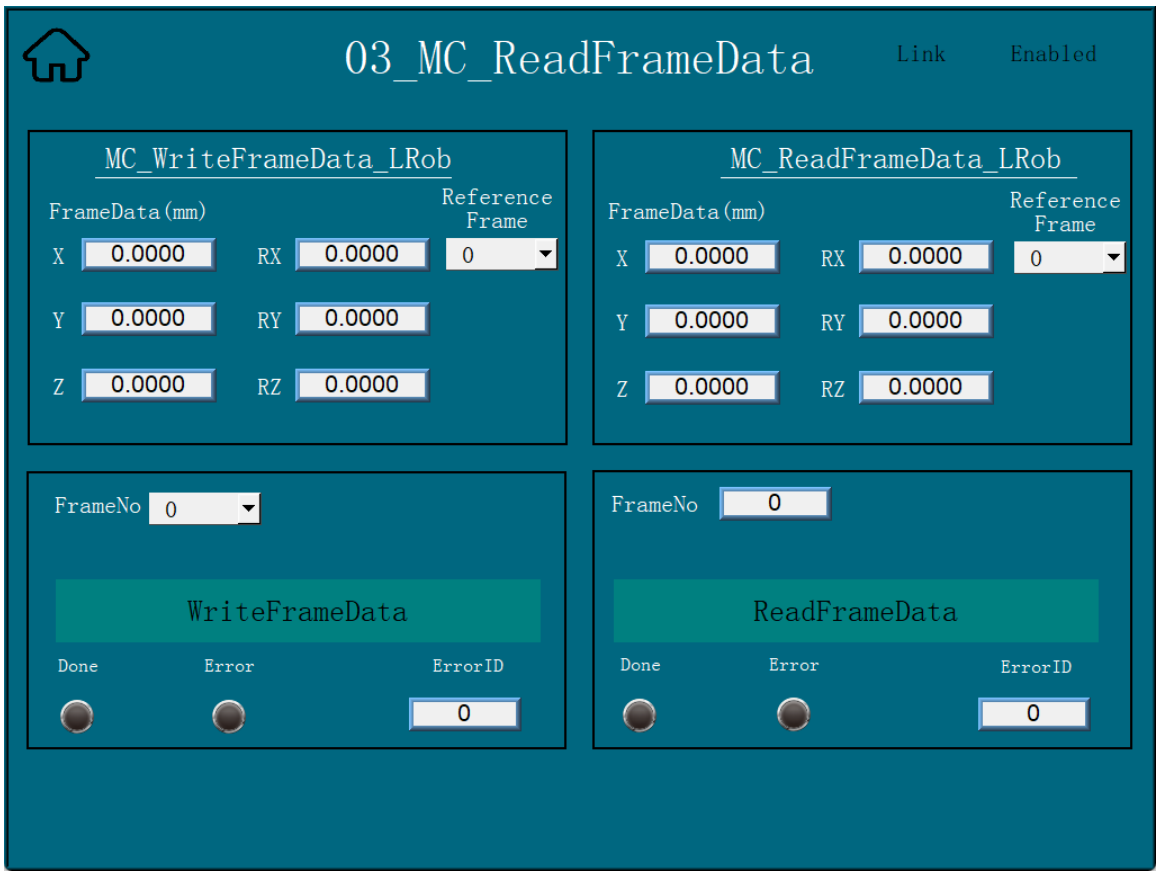
2. Click '01_MC_EnableRobot' interface and then click 'EnableRobot' to enable the robot, and click 'Reset' to reset if there is any error during the process.



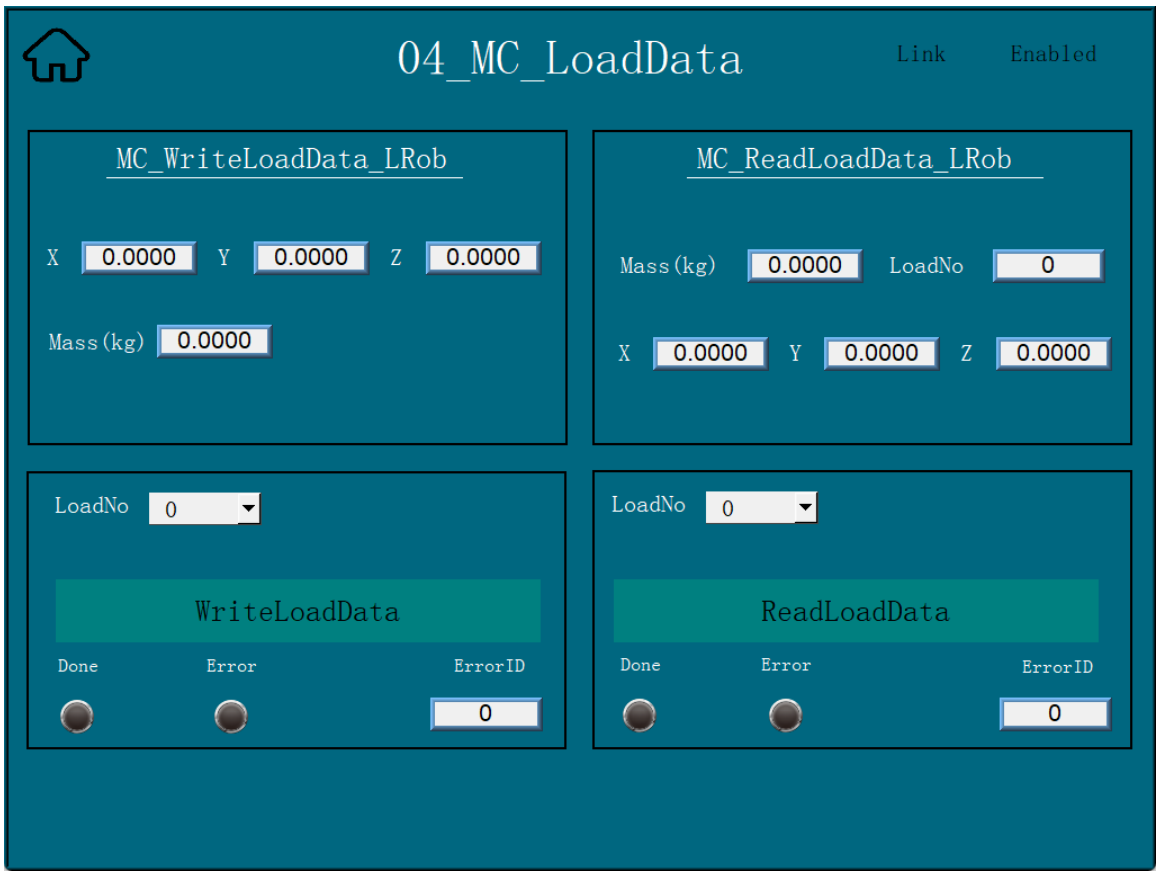
3. Click '02_MC_ToolData' to enter the tool information interface, enter the parameters on the left and click WriteToolData to write the tool information; on the right, click ReadToolData to read the existing tool information.



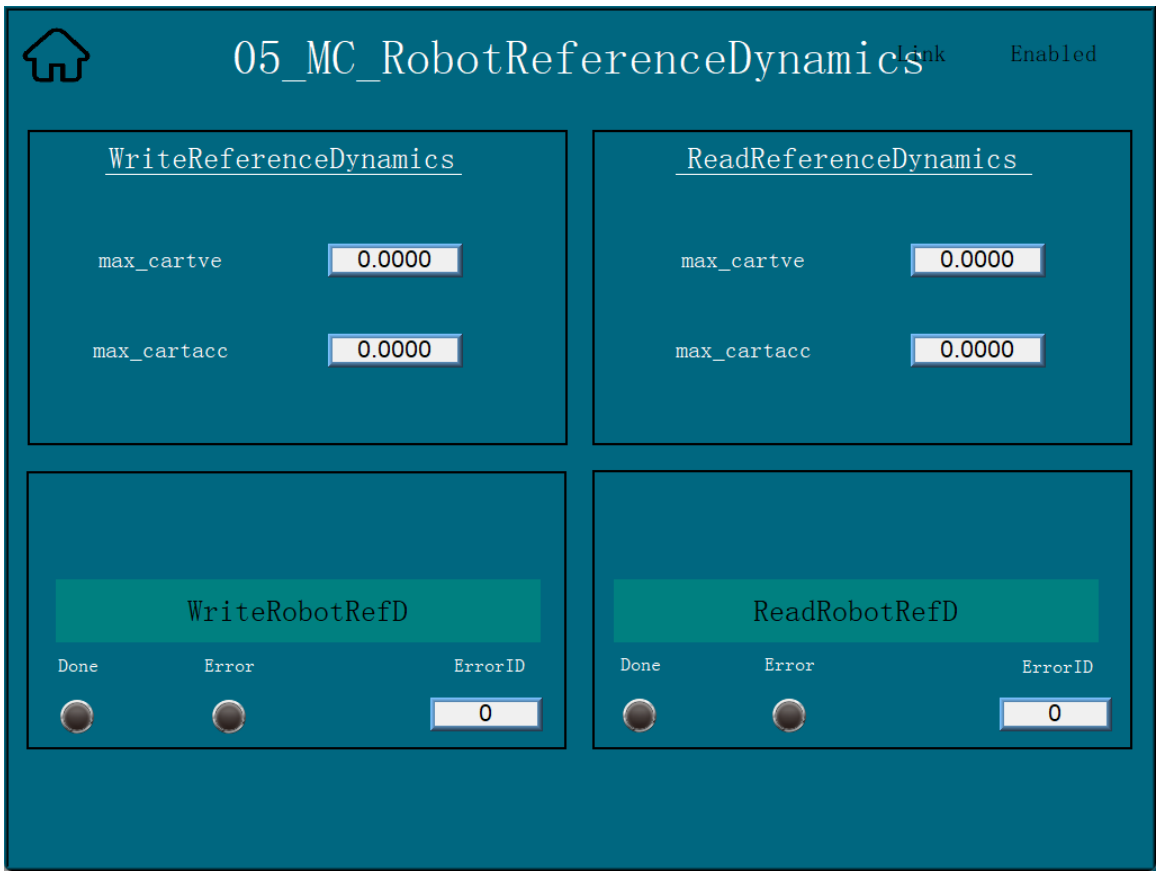
4. Click '03_MC_FrameData' to enter the interface of workpiece information. On the left side, after inputting parameters, click WriteFrameData to write workpiece information; on the right side, click ReadFrameData to read existing workpiece information.



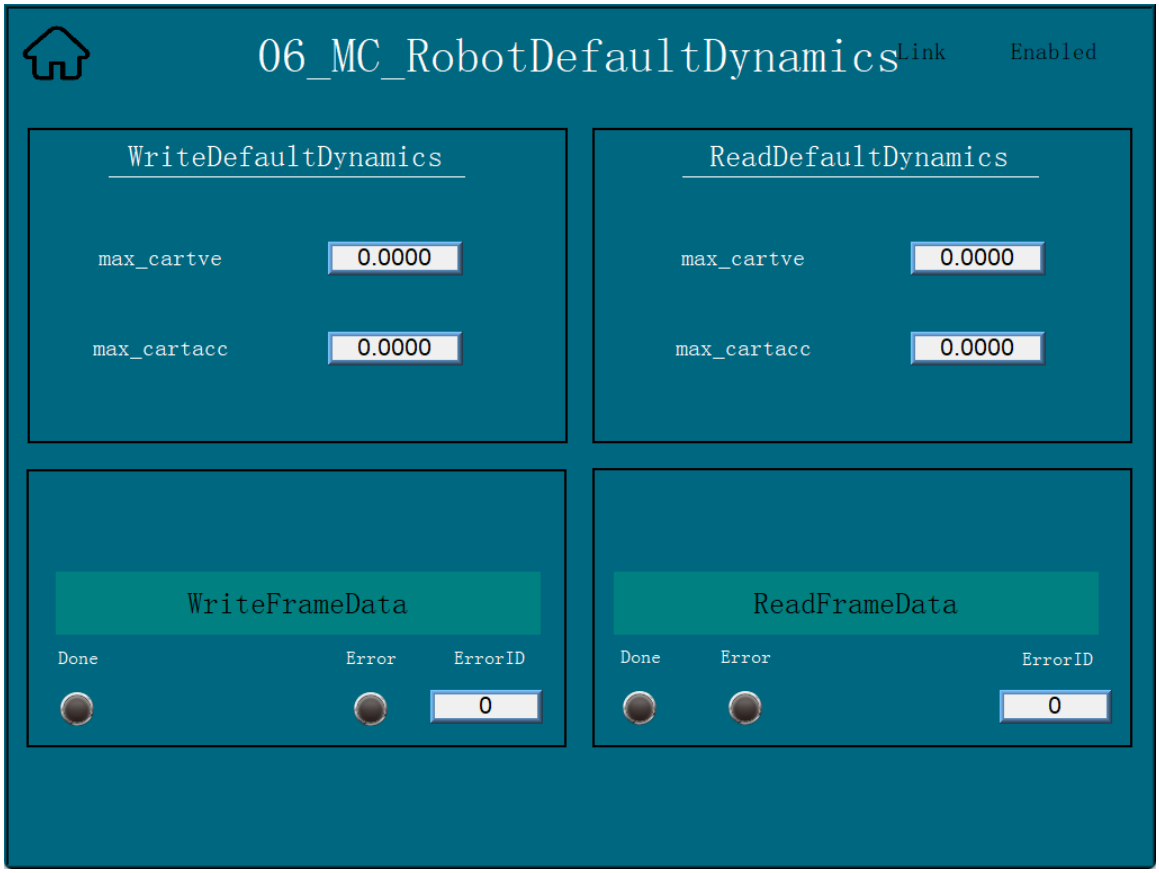
5. Click '04_MC_LoadData' to enter the load information interface, enter the parameters on the left and click WriteLoadData to write load information; on the right, click ReadLoadData to read the existing load information.



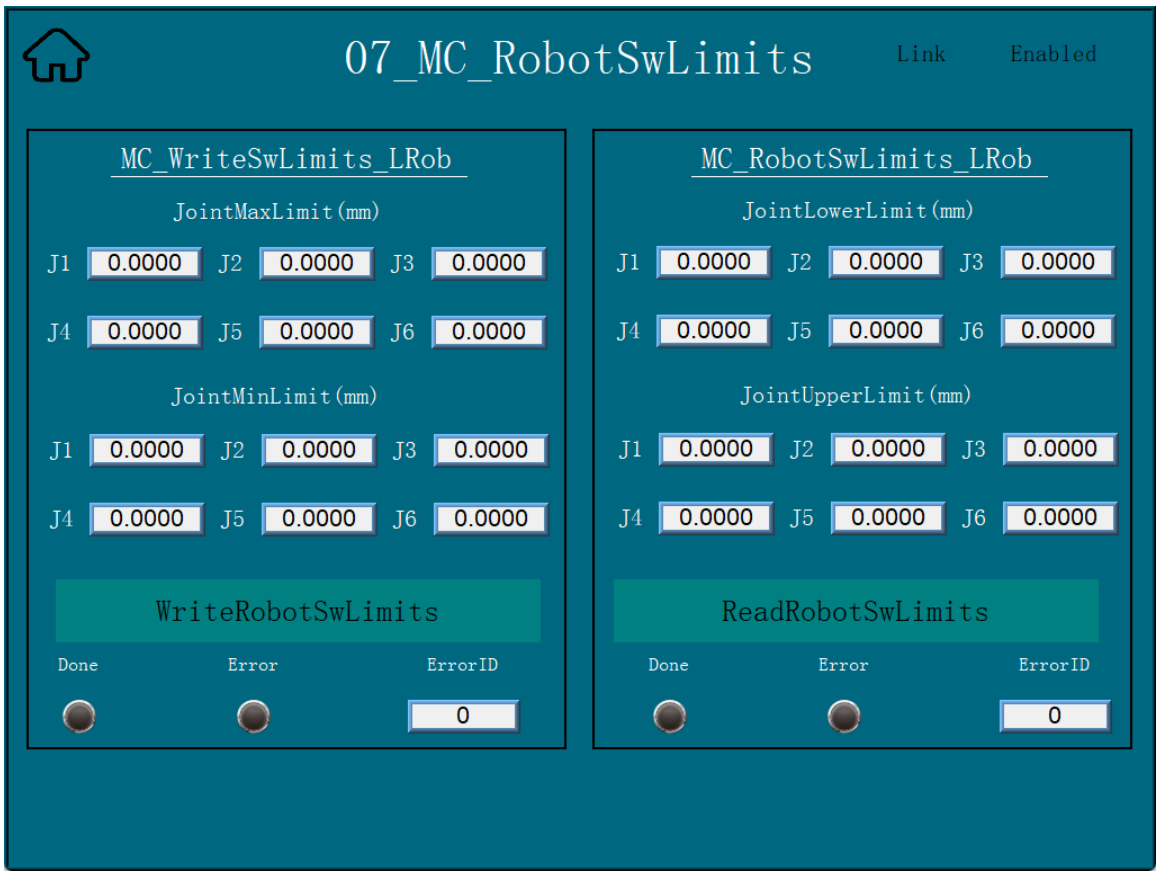
6. Click '05_MC_RobotReferenceDynamics' to enter the interface of Maximum Velocity and Maximum Acceleration of Robot, enter the parameters on the left side and then click WriteRobotRefD to write the information of Maximum Velocity and Maximum Acceleration; click ReadRobotRefD on the right side to read the information of Maximum Velocity and Maximum Acceleration. On the right side, click ReadRobotRefD to read the max speed and max acceleration information.



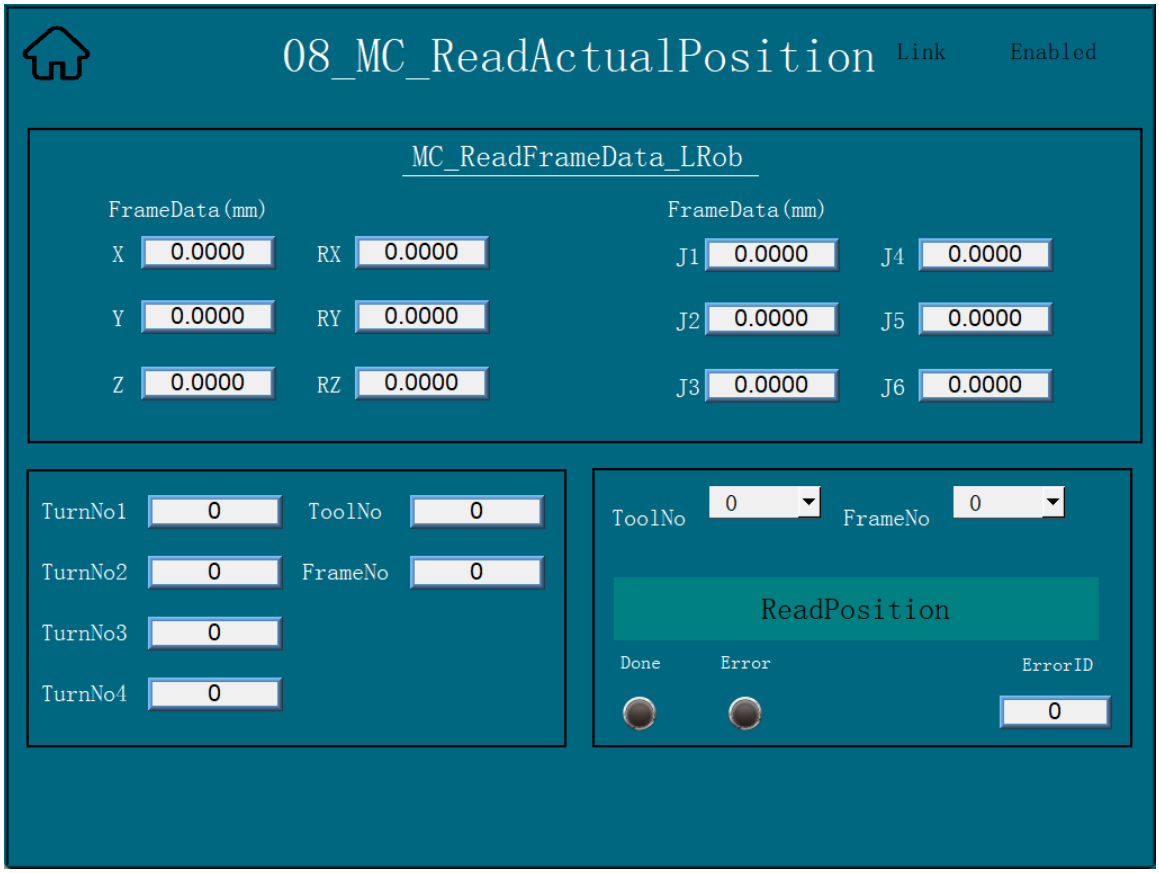
7. Click '06_MC_Robot DefaultDynamics' to enter the interface of robot default speed and default acceleration, enter the parameters on the left side and click WriteRobotDefD to write the default speed and default acceleration information; on the right side, click ReadRobotDefD to read the default speed and default acceleration information. on the left side, and then click WriteRobotDefD to write the default speed and default acceleration information; on the right side, click ReadRobotDefD to read the information.



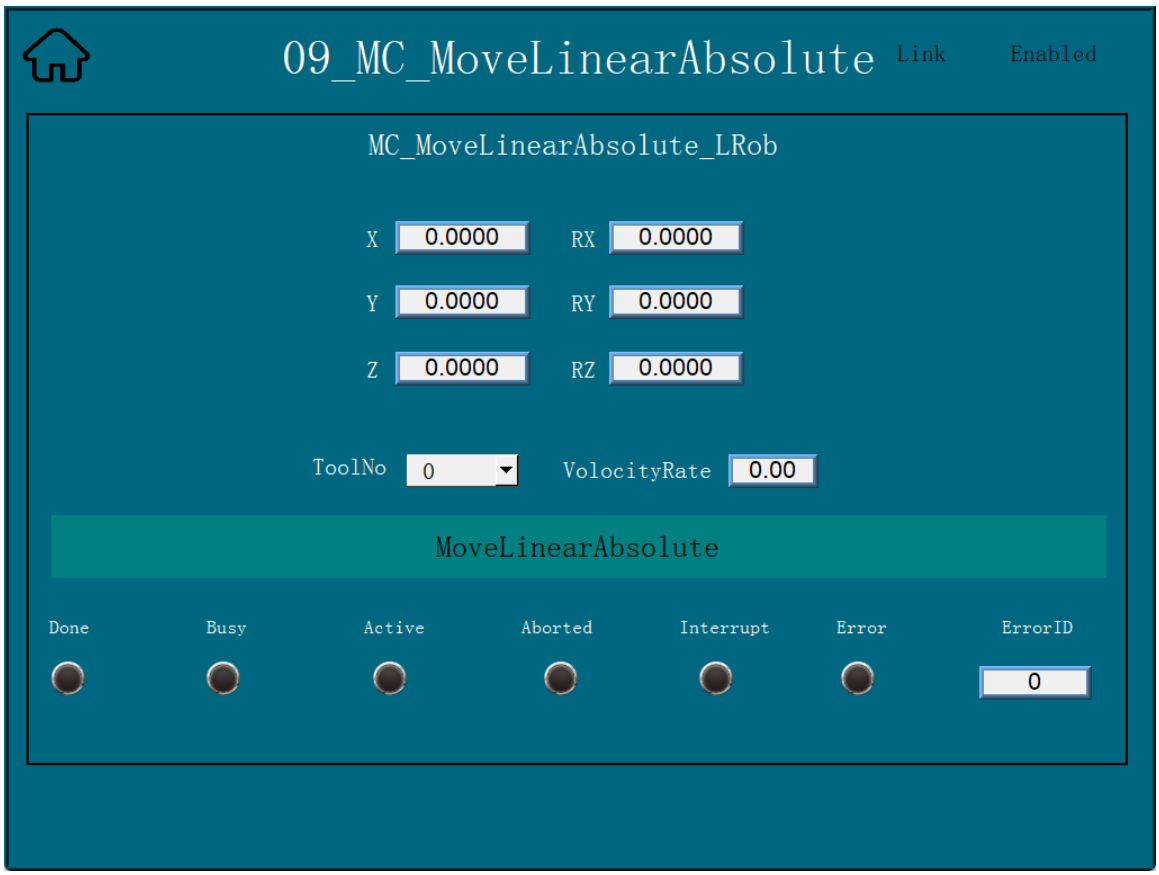
8. Click '07_MC_RobotSwLimits' to enter the coordinate limit interface. On the left side, input the maximum limit and minimum limit parameter values and click WriteRobotSwLimits to write the limit parameter information; on the right side, click ReadRobotSwLimits to read the existing limit parameter information. parameter information.



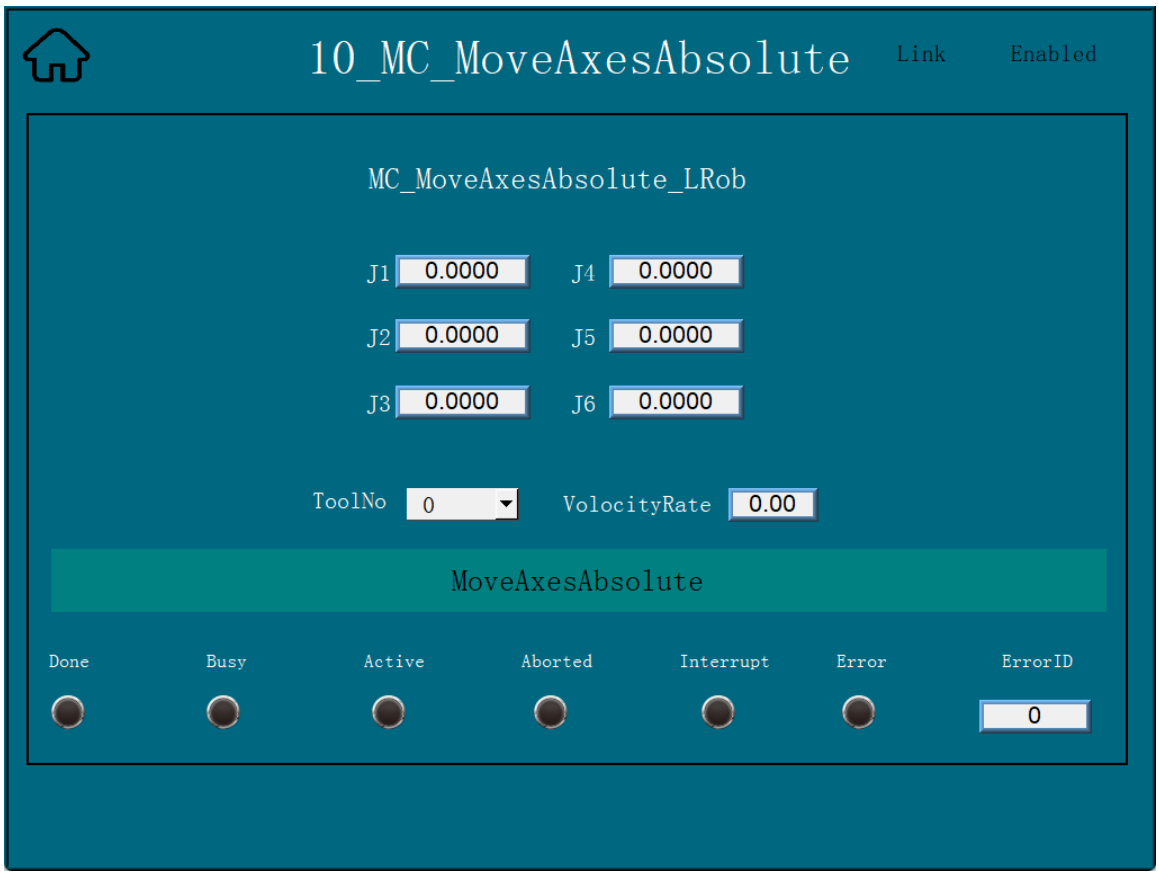
9. Click '08_MC_ReadActualPosition' to enter the read actual position interface, click ReadPosition to read the existing position information.



10. Click '09_MC_MoveLinearAbsolute' to enter the Linear Motion interface, input the coordinate parameter and click MoveLinearAbsolute to make the robot move linearly at the target position.



11. Click '10_MC_MoveAxesAbsolute' to enter the interface of axis coordinate movement, input the coordinate parameter and click MoveAxesAbsolute to make the robot move to the target position with the input axis coordinate as the end point.



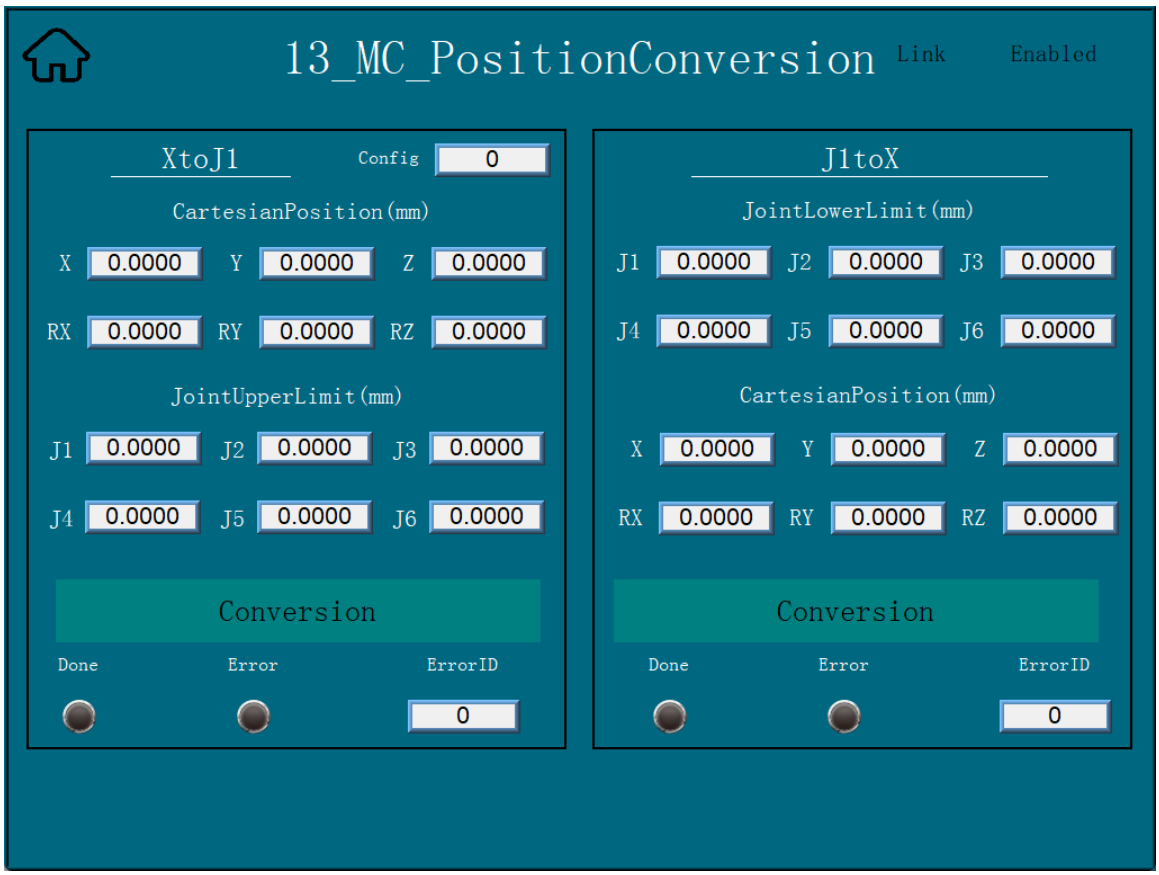
12. Click '11_MC_MoveDirectAbsolute' to enter the direct motion interface, input the coordinate parameter and click MoveDirectAbsolute to make the robot move directly to the target position with the input parameter as the end point.



13. Click '12_MC_Groups' to enter the direct motion interface, in which, clicking GroupInterrupt can interrupt the movement of the robot in the process of movement, and clicking GroupContinue can make the robot continue to move to the target position. Click GroupStop to stop (end) the ongoing position movement. If an alarm or error is triggered during the process, click GroupReset to reset the robot to the error.



14. Click '13_MC_PositionConversion' to enter the position conversion interface, XtoJ1 can be converted from Cartesian position to joint angle, and J1toX can be converted from joint angle to Cartesian position.

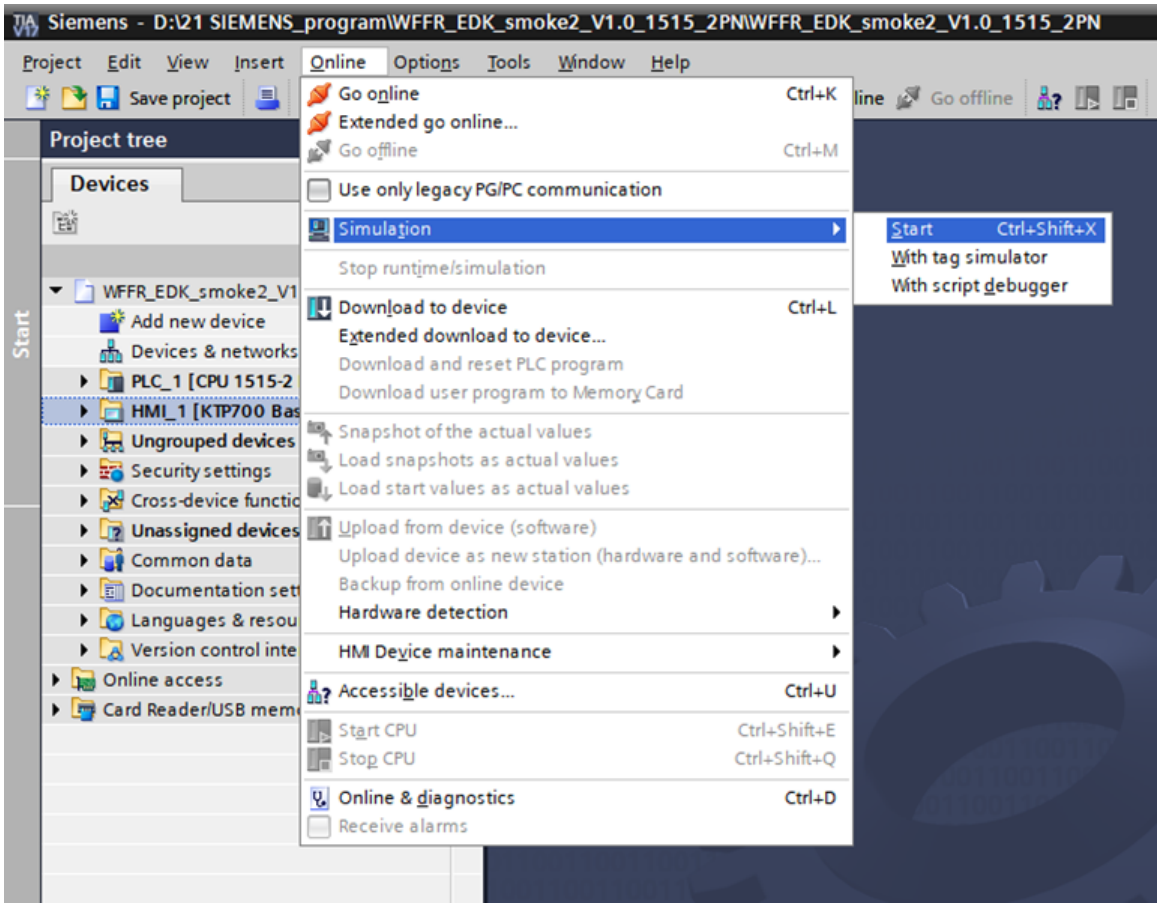


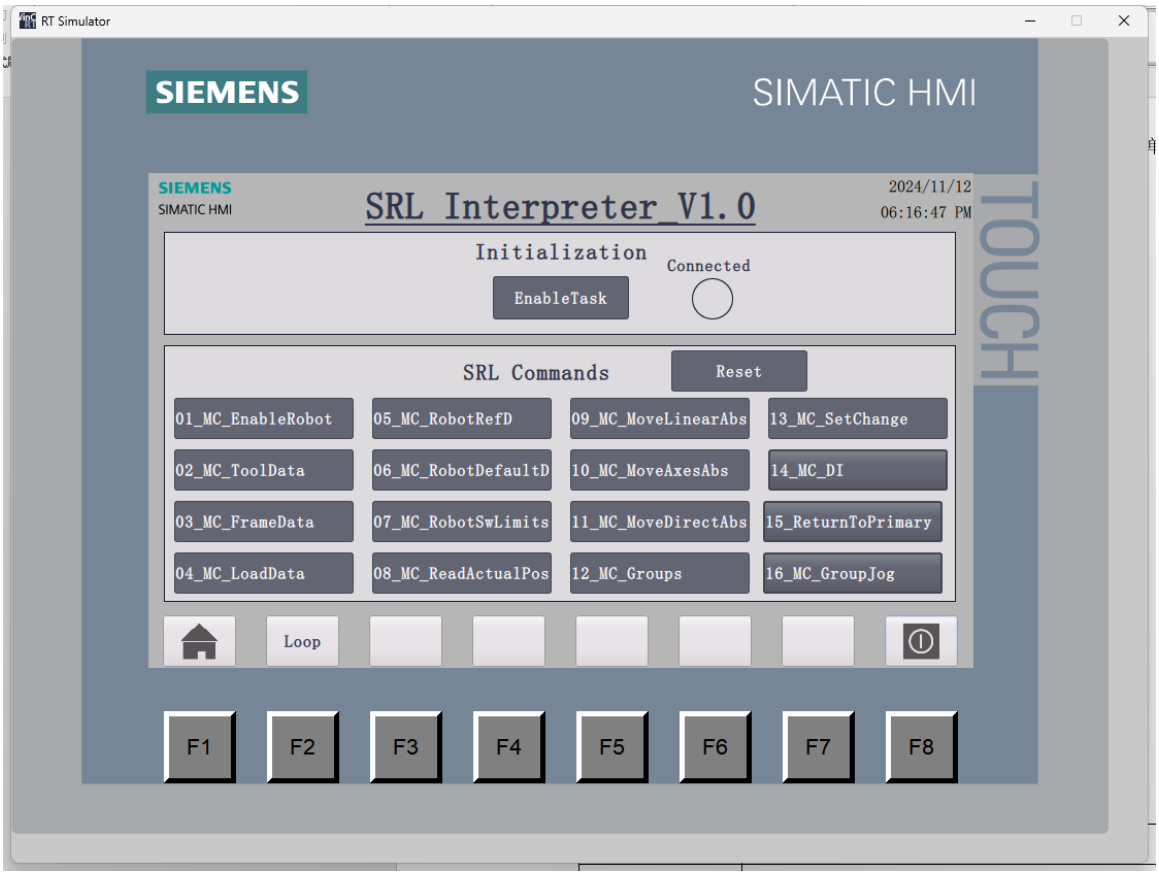
15. Click '14_MC_GroupJog' to enter the interface of robot jogging, after the configuration is finished, drop down the axes to select the axis you need to jog, and then select the rotation direction of the axis. Click JogMove to move. MC_ChangeSpeedOverride on the right side can adjust the moving speed of the robot arm.



1.3.18.3.6 HMI setting (Profinet emulation)

1. After opening the programme, click on 'HMI_1[ktp700 Basic PN]' in the project tree, and then click on 'Online'→'Simulation'→'Start' in the menu bar. Click 'Online'→'Simulation'→'Start' in the menu bar. Wait for the software to compile and simulate.
2. The function after emulation is the same as the content of the Velcro screen (CC-Link IEF Basic). You can refer to the above content to set up.


Siemens - D:\21 SIEMENS_program\WFFR_EDK_smoke2_V1.0_1515_2PN\WFFR_EDK_smoke2_V1.0_1515_2PN
Project
Edit
View
Insert
Online
Options
Tools
Window
Help
Save project
Go offline
Project tree
Devices
Start
WFFR_EDK_smoke2_V1
Add new device
Devices & networks
PLC_1 [CPU 1515-2
HMI_1 [KTP700 Bas
Ungrouped devices
Security settings
Cross-device functio
Unassigned devices
Common data
Documentation set
Languages & resou
Version control inte
Online access
Card Reader/USB mem
Go online Ctrl+K
Extended go online...
Go offline Ctrl+M
Use only legacy PG/PC communication
Simulation
Stop runtime/simulation
Download to device Ctrl+L
Extended download to device...
Download and reset PLC program
Download user program to Memory Card
Snapshot of the actual values
Load snapshots as actual values
Load start values as actual values
Upload from device (software)
Upload device as new station (hardware and software)...
Backup from online device
Hardware detection
HMI Device maintenance
Accessible devices... Ctrl+U
Start CPU Ctrl+Shift+E
Stop CPU Ctrl+Shift+Q
Online & diagnostics Ctrl+D
Receive alarms
Start Ctrl+Shift+X
With tag simulator
With script debugger



1.3.18.4 Robot Slave Mode Operation Manual

1.3.18.4.1 Loading Slave Mode

Step 1: Open the WebApp, navigate to Initial Setup -> Peripherals -> Board Communication -> Manual Configuration.

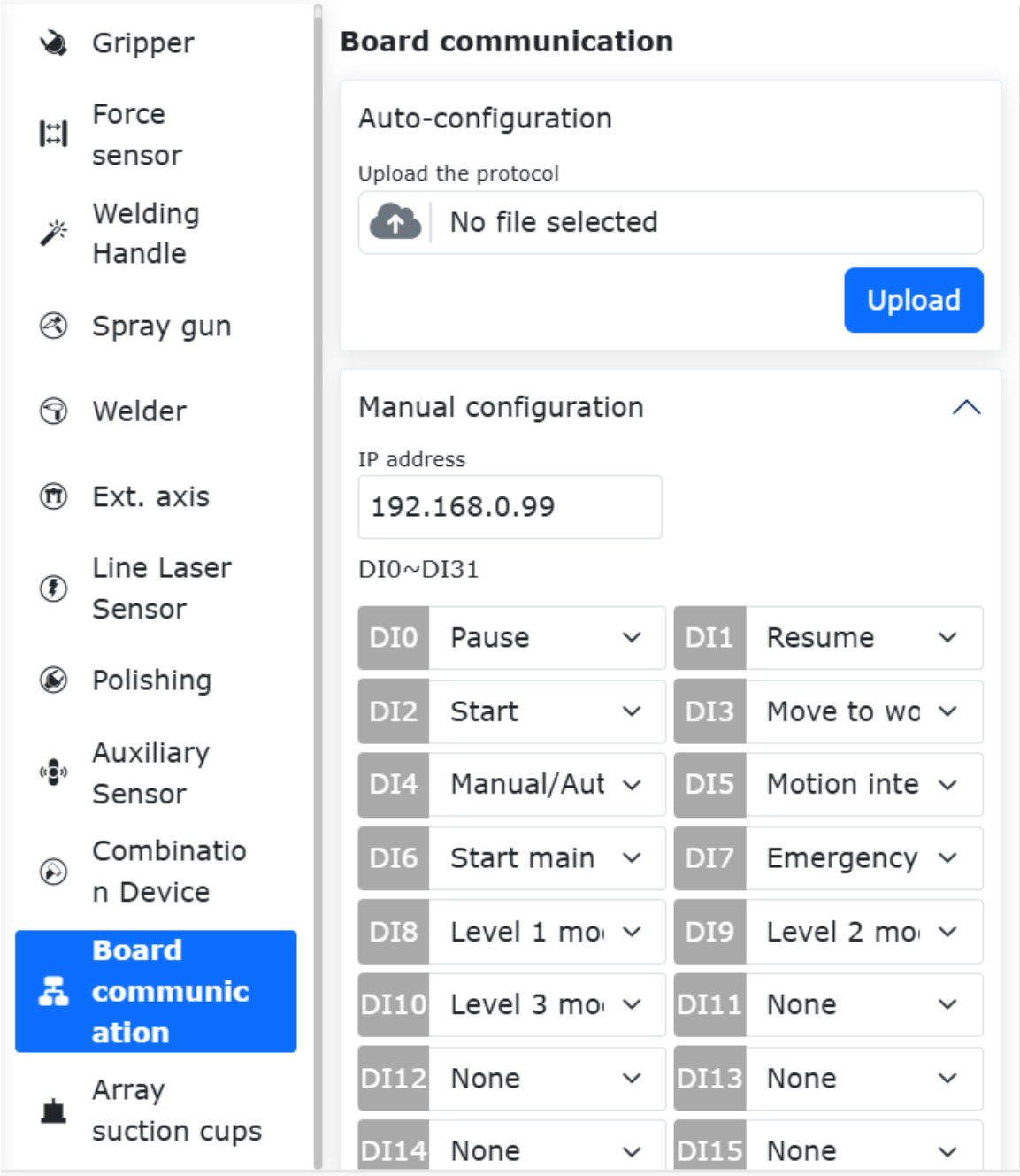


Figure 17.3-1 Board Communication Manual Configuration

First, configure the IP address of the FRJ-PCIeN board. If left blank, the board will use the default IP: 192.168.0.100 for startup configuration. Currently, IP configuration only applies to EIP and CC-Link IEF Basic protocols. For PN protocol, the IP is assigned by the PLC master station scanning slave devices.

Note: After changing the IP address on the page, you need to load the slave mode for the changes to take effect.

Select the required mapping functions for DI, DO, and AO (see Appendix 1). The parameters are defined as follows:

- DI (Robot Control): The robot slave accepts external input signals and executes the mapped functions.

- DO (Robot Status Output): The robot slave feeds back status signals to the master station.
- AO (Robot Status Feedback): The robot slave feeds back status data to the master station. AO0~AO15 are signed integers (int16), and AO16~AO31 are single-precision floating-point numbers (float).

Step 2: Click the "Configure" button to generate the open protocol Lua file.

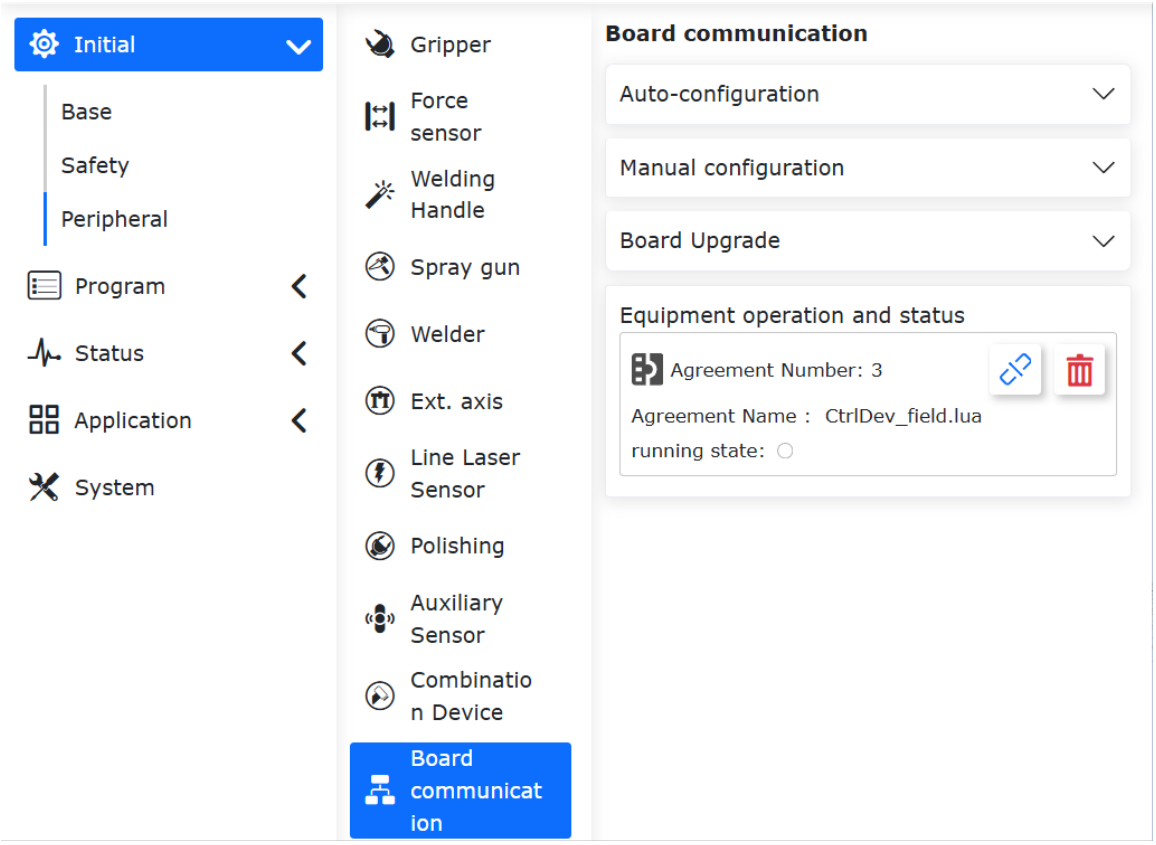


Figure 17.3-2 Device Operation and Status

Note: The open protocol Lua file supports download and can be imported in the auto-configuration interface.

Example generated program:

```
local id = 3
local ctrlDI = {0, 0, 0, 0, 0, 0}
local funcDI = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
local DOState = {0, 0, 0, 0, 0, 0, 0, 0}
local AOState = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
-- Launch the board communication process
LoadFieldBusSlave()
sleep_ms(8000)
while(1) do
   -- Set the DO status
   CtrlBoxDO, CtrlBoxCO, CtrlBoxDI, CtrlBoxCI, errState, motionState, moveToOriginState, robotStartDoneState, modeChangeState, programStartStopState, emergencyState, reduceState, collision, enablestate, safetyStop0, safetyStop1, pauseState,
```

(continues on next page)

(continued from previous page)

```
↪interfereState = GetRobotFuncDOState()
  DOState[1] = CtrlBoxDO
  DOState[2] = CtrlBoxCO
  DOState[3] = CtrlBoxDI
  DOState[4] = CtrlBoxCI
  local ctrlWord0 = 0
  ctrlWord0 = SetBitWithIndex(ctrlWord0, 0, errState)
  ctrlWord0 = SetBitWithIndex(ctrlWord0, 1, motionState)
  ctrlWord0 = SetBitWithIndex(ctrlWord0, 2, moveToOriginState)
  ctrlWord0 = SetBitWithIndex(ctrlWord0, 3, robotStartDoneState)
  ctrlWord0 = SetBitWithIndex(ctrlWord0, 4, modeChangeState)
  ctrlWord0 = SetBitWithIndex(ctrlWord0, 5, programStartStopState)
  ctrlWord0 = SetBitWithIndex(ctrlWord0, 6, emergencyState)
  ctrlWord0 = SetBitWithIndex(ctrlWord0, 7, reduceState)
  DOState[5] = ctrlWord0
  local ctrlWord1 = 0
  ctrlWord1 = SetBitWithIndex(ctrlWord1, 0, collision)
  ctrlWord1 = SetBitWithIndex(ctrlWord1, 1, enablestate)
  ctrlWord1 = SetBitWithIndex(ctrlWord1, 2, safetyStop0)
  ctrlWord1 = SetBitWithIndex(ctrlWord1, 3, safetyStop1)
  ctrlWord1 = SetBitWithIndex(ctrlWord1, 4, pauseState)
  ctrlWord1 = SetBitWithIndex(ctrlWord1, 5, interfereState)
  DOState[6] = ctrlWord1
  SetFieldBusDOState(DOState)

  -- Set the AO status
  mainErrCode, subErrCode, TCPSpeed, axisPos1, axisPos2, axisPos3, axisPos4, axisPos5,␣
↪axisPos6, jointVelFeedback1, jointVelFeedback2, jointVelFeedback3, jointVelFeedback4,␣
↪jointVelFeedback5, jointVelFeedback6, jointCurFeedback1, jointCurFeedback2,␣
↪jointCurFeedback3,jointCurFeedback4,jointCurFeedback5,jointCurFeedback6,␣
↪jointTorqueFeedback1, jointTorqueFeedback2,jointTorqueFeedback3,jointTorqueFeedback4,␣
↪jointTorqueFeedback5, jointTorqueFeedback6, cartPosx, cartPosy, cartPosz, cartPosrx,␣
↪cartPosry, cartPosrz = GetRobotFuncAOState()
  AOState[1] = mainErrCode
  AOState[2] = subErrCode
  AOState[17] = axisPos1
  AOState[18] = axisPos2
  AOState[19] = axisPos3
  AOState[20] = axisPos4
  AOState[21] = axisPos5
  AOState[22] = axisPos6
  AOState[23] = cartPosx
  AOState[24] = cartPosy
  AOState[25] = cartPosz
  AOState[26] = cartPosrx
  AOState[27] = cartPosry
  AOState[28] = cartPosrz
  SetFieldBusAOState(AOState)
  sleep_ms(10)

  -- Set the DI status
  -- Configure the DI function and update it in real-time
```

(continues on next page)

(continued from previous page)

```
   ctrlDI[1],ctrlDI[2],ctrlDI[3],ctrlDI[4],ctrlDI[5],ctrlDI[6] = GetFieldBusDIState()
   funcDI[1] = ctrlDI[1]
   funcDI[2] = ctrlDI[2]
   funcDI[3] = GetBitWithIndex(ctrlDI[3], 0)
   funcDI[4] = GetBitWithIndex(ctrlDI[3], 1)
   funcDI[5] = GetBitWithIndex(ctrlDI[3], 2)
   funcDI[6] = GetBitWithIndex(ctrlDI[3], 3)
   funcDI[7] = GetBitWithIndex(ctrlDI[3], 4)
   funcDI[8] = GetBitWithIndex(ctrlDI[3], 5)
   funcDI[9] = GetBitWithIndex(ctrlDI[3], 6)
   funcDI[10] = GetBitWithIndex(ctrlDI[3], 7)
   funcDI[11] = GetBitWithIndex(ctrlDI[4], 0)
   funcDI[12] = GetBitWithIndex(ctrlDI[4], 1)
   funcDI[13] = GetBitWithIndex(ctrlDI[4], 2)
   funcDI[14] = GetBitWithIndex(ctrlDI[4], 3)
   funcDI[15] = GetBitWithIndex(ctrlDI[4], 4)
   funcDI[16] = GetBitWithIndex(ctrlDI[4], 5)
   SetRobotFuncDIState(funcDI)
   local stopFlag = GetOpenLUAStopFlag(id)
   if(stopFlag ~= 0) then
      UnloadFieldBusSlave()
      break
   end
   sleep_ms(10)
end
```

Step 3: Click the “Load” button to load the robot slave mode.

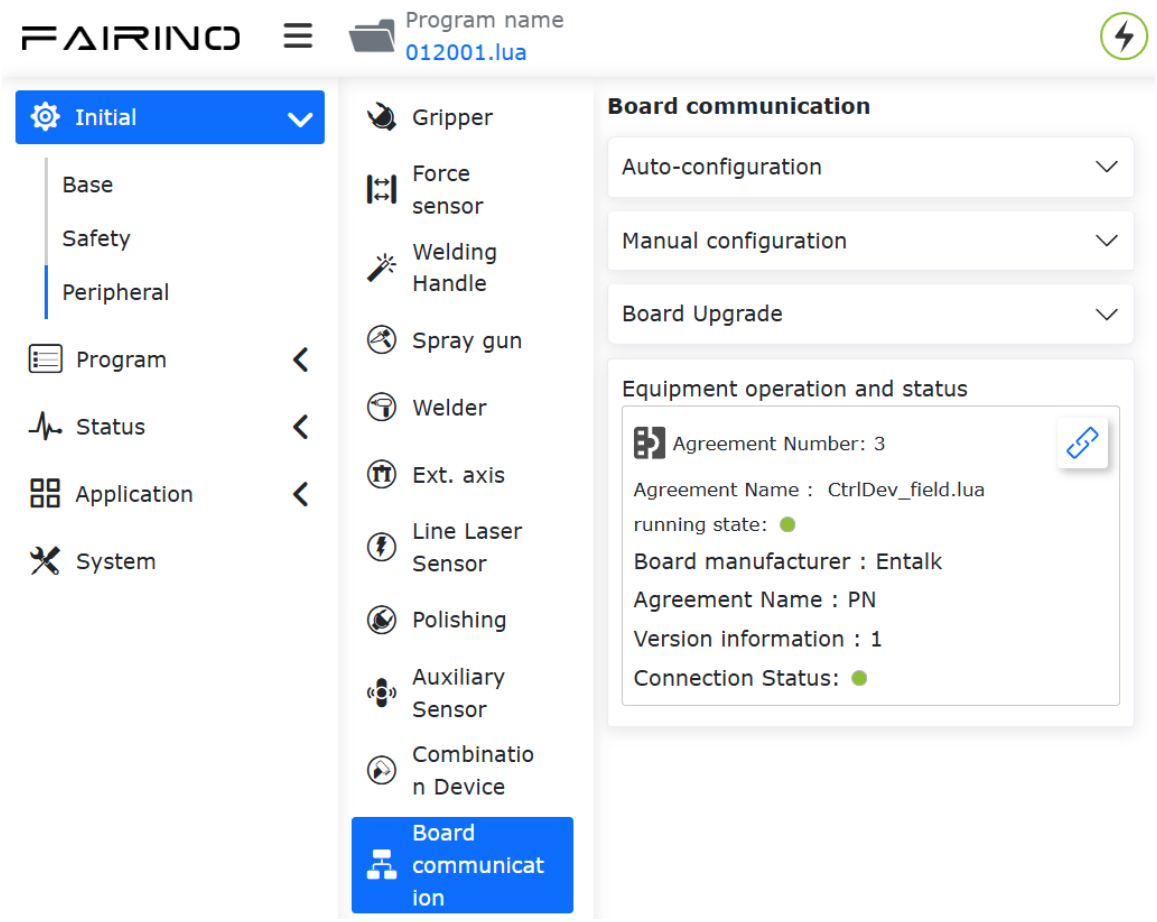


Figure 17.3-3 Loading Slave Mode

Note: After successfully loading the robot slave mode, the auto-start function is supported. To use remote mode, unload the slave mode first.

Step 4: Click the status bar button on the right to monitor DI, DO, AI, and AO interaction information. The parameters are as follows:

- CtrlDO: Input signal value from the master device controlling the robot control box DO.
- DI: Input signal value from the external master control.
- DO: Output signal value fed back by the robot slave.
- AI: Input value from the external master. AI0~AI15 are int16 type, and AI16~AI31 are float type.
- AO: Output value from the robot slave. AO0~AO15 are int16 type, and AO16~AO31 are float type.

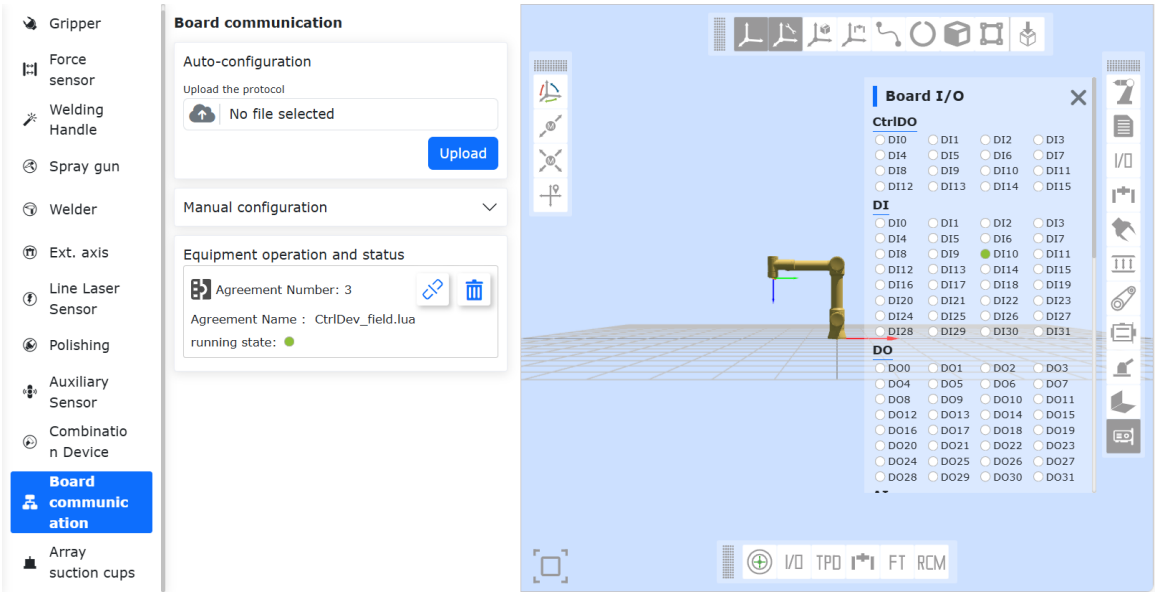


Figure 17.3-4 DI, DO, AI, AO Interaction Information

Step 5:After loading is complete, you can use the Teach Program -> Communication Command -> Board Card to generate Lua commands for the board. This allows you to:

1) Set Slave DO (Digital Output) and AO (Analog Output).
2) Read Slave DI (Digital Input) and AI (Analog Input).
3) Wait for Slave DI and AI signals.

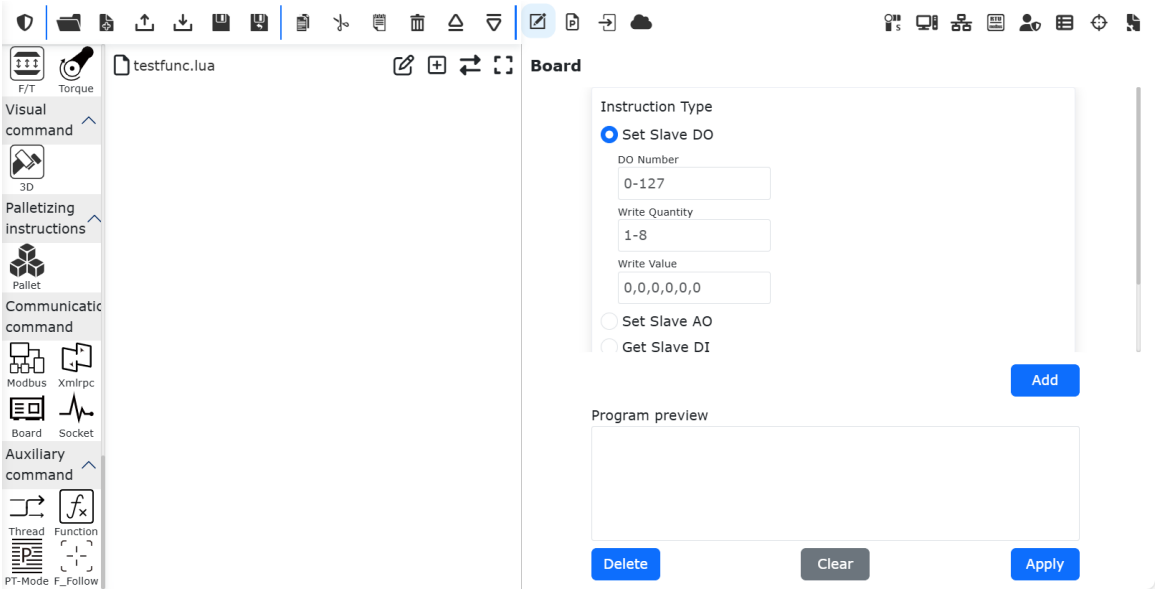


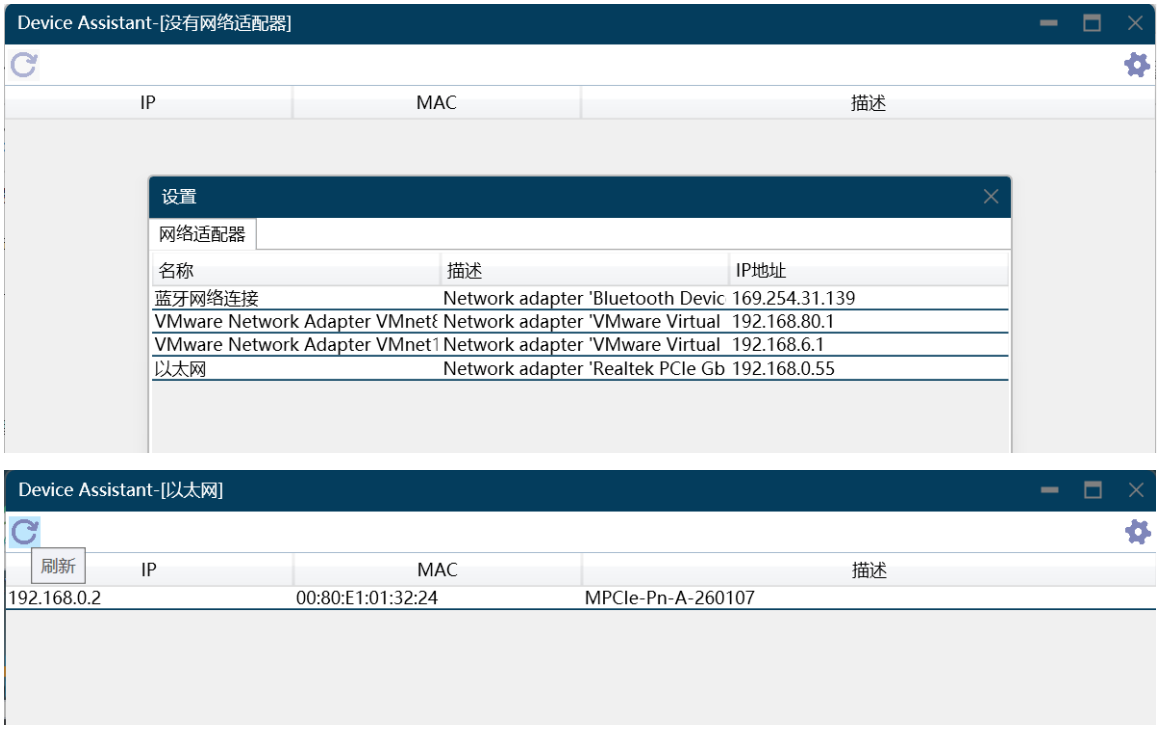
Figure 17.3-5 Board Lua Commands

`Appendix 1: Slave Mode Address Mapping Table`

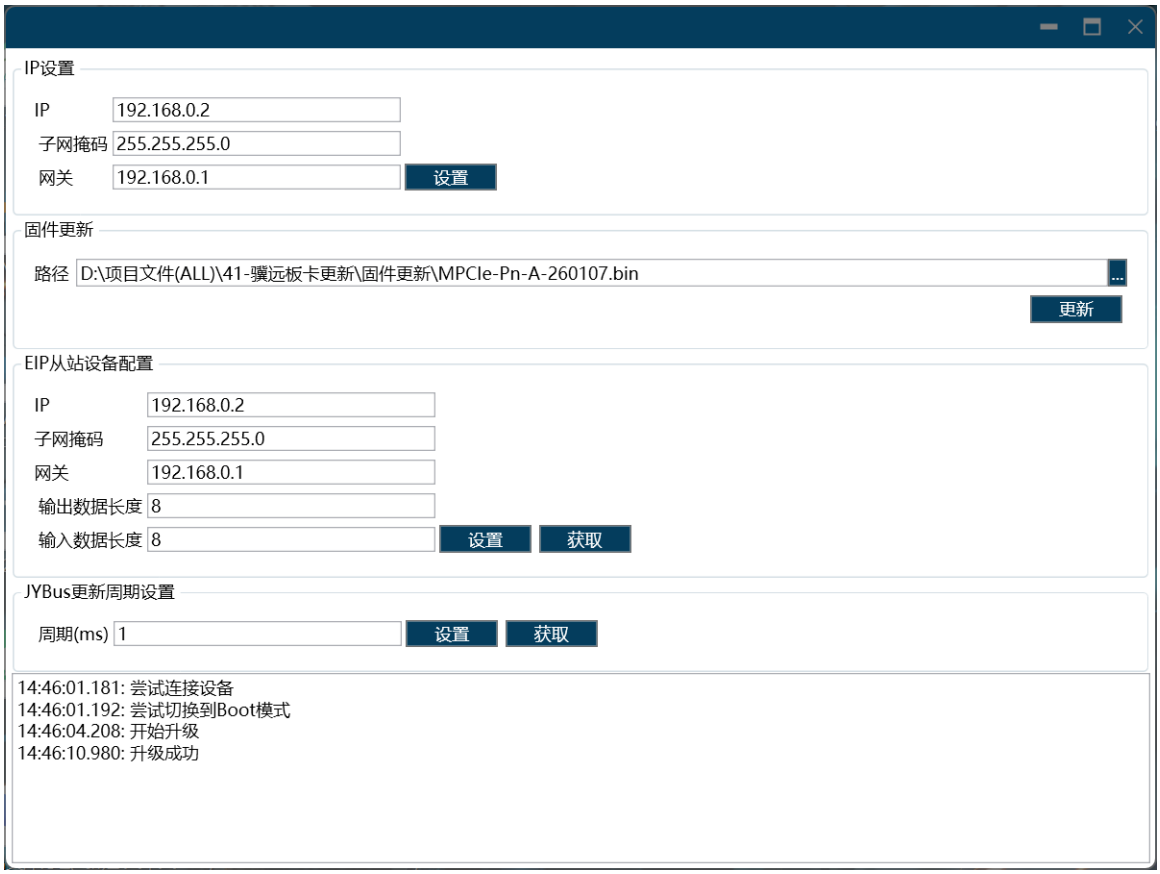
1.3.18.5 Board Communication Cycle Configuration

The communication cycle of the board can be configured via the host computer. Currently, only PN protocol firmware is provided, with future compatibility for EIP, CC-Link IE Basic, and EtherCAT protocols.

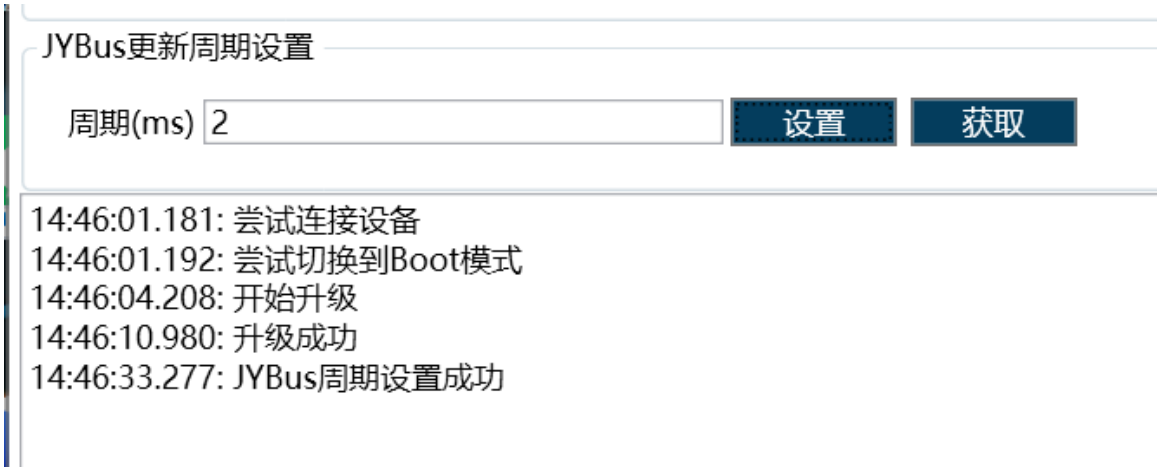
(1) Directly connect the PC (Windows 11 system) network port to the board's network port. Open Device Assistant v1.1.0, double-click "Ethernet," and click the "Refresh" button in the upper left corner to scan for currently connected board devices.



(2) In the firmware update interface, upload the new version of PN firmware and click the "Update" button. A prompt reading "Upgrade Successful" will appear in the lower left corner when printed.



(3) Enter the desired communication cycle (supports 1~100ms) and click the “Set” button. A prompt reading “Cycle Setting Successful” will appear in the lower left corner when printed.



1.3.18.6 Appendice

1.3.18.6.1 Instruction List

Command code	Command description
0x1000	Robot enablement
0x1001	Reset all error
0x1002	Robot stops moving
0x1003	Read actual position
0x1004	Set robot speed
0x1005	Resume robot motion
0x1006	Robot pauses motion
0x1007	Calculate the Cartesian position from the joint position
0x1008	Calculate joint position from Cartesian position
0x2000	Write tool information
0x2001	Read tool information
0x2002	Write workpiece information
0x2003	Read workpiece information
0x2004	Write load information
0x2005	Read load information
0x2006	Write reference dynamic information
0x2007	Read reference dynamic information
0x2008	Write default dynamic information
0x2009	Read default dynamic information
0x2010	Write soft limit information
0x2011	Read soft limit information
0x3000	MoveAxes (based on joint angle)
0x3001	MoveLinear
0x3002	MoveDirect (based on Cartesian coordinate system)
0x3003	jog motion
0x3004	jog stop

1.3.19 Appendix

1.3.19.1 Appendix 1: Motion controller errors and handling methods

Main Fault Code	Sub Fault Code	Description
0-No Fault	0	No fault
1-Command Point Error	1	Joint command point error, resettable
1-Command Point Error	2	Linear target point error (including tool mismatch), resettable
1-Command Point Error	3	Arc intermediate point error (including tool mismatch), resettable
1-Command Point Error	4	Arc target point error (including tool mismatch), resettable

continues on next page

Table 9 – continued from previous page

Main Fault Code	Sub Fault Code	Description
1-Command Point Error	5	Arc command points too close, resettable
1-Command Point Error	6	Full circle/helix intermediate point 1 error (including tool mismatch), resettable
1-Command Point Error	7	Full circle/helix intermediate point 2 error (including tool mismatch), resettable
1-Command Point Error	8	Full circle/helix intermediate point 3 error (including tool mismatch), resettable
1-Command Point Error	9	Full circle/helix command points too close, resettable
1-Command Point Error	10	TPD command point error, resettable
1-Command Point Error	11	TPD command tool does not match current tool, resettable
1-Command Point Error	12	TPD current command and next command start point deviation too large, resettable
1-Command Point Error	13	Internal/external tool switching error, resettable
1-Command Point Error	14	New helix start point error, resettable
1-Command Point Error	15	New spline command point error, resettable
1-Command Point Error	17	PTP joint command limit exceeded, resettable
1-Command Point Error	18	TPD joint command limit exceeded, resettable
1-Command Point Error	19	LIN/ARC joint command limit exceeded, resettable
1-Command Point Error	20	Cartesian space command overspeed, not resettable
1-Command Point Error	21	Joint space torque command limit exceeded, resettable
1-Command Point Error	22	JOG joint command limit exceeded, resettable
1-Command Point Error	23	Axis 1 joint space command speed limit exceeded, resettable
1-Command Point Error	24	Axis 2 joint space command speed limit exceeded, resettable
1-Command Point Error	25	Axis 3 joint space command speed limit exceeded, resettable
1-Command Point Error	26	Axis 4 joint space command speed limit exceeded, resettable
1-Command Point Error	27	Axis 5 joint space command speed limit exceeded, resettable
1-Command Point Error	28	Axis 6 joint space command speed limit exceeded, resettable
1-Command Point Error	29	Joint feedback speed limit exceeded, not resettable

continues on next page

Table 9 – continued from previous page

Main Fault Code	Sub Fault Code	Description
1-Command Error Point	30	Joint command and feedback deviation too large, not resettable, requires restart
1-Command Error Point	31	DMP target point error (including tool mismatch), resettable
1-Command Error Point	33	Next command joint configuration changed (next command has singular pose, use PTP command or change next point), resettable
1-Command Error Point	34	Current command joint configuration changed (next command has singular pose, use PTP command or change next point), resettable
1-Command Error Point	35	LIN command joint speed limit exceeded, resettable
1-Command Error Point	36	LIN command adaptive speed exceeds threshold, resettable
1-Command Error Point	37	Unreachable point in trajectory, resettable
1-Command Error Point	38	Unreachable point in trajectory - singular pose, resettable
1-Command Error Point	49	Command error, only LIN and ARC commands allowed between ARCSTART and ARCEND, resettable
1-Command Error Point	50	Command error, only LIN and ARC commands allowed between WEAVESTART and WEAVEEND, resettable
1-Command Error Point	51	Weaving parameter error, resettable
1-Command Error Point	52	Weaving command points too close, resettable
1-Command Error Point	53	Unreachable point in weaving trajectory - singular pose, resettable
1-Command Error Point	54	Unreachable point in weaving trajectory - joint command limit exceeded, resettable
1-Command Error Point	55	Unreachable point in weaving trajectory - planning anomaly (tool z coincides with forward direction x angle), resettable
1-Command Error Point	56	Unreachable point in weaving trajectory - planning anomaly (arc waypoint error), resettable
1-Command Error Point	65	Laser sensor command deviation too large, resettable
1-Command Error Point	66	Laser sensor command interrupted, seam tracking ended early, resettable
1-Command Error Point	81	External axis command speed limit exceeded, resettable
1-Command Error Point	82	External axis command and feedback deviation too large, not resettable, requires homing or restart
1-Command Error Point	83	Extended peripheral (external axis/IO) communication error, resettable
1-Command Error Point	84	Extended peripheral (external axis/IO) communication packet loss error, resettable
1-Command Error Point	97	Conveyor tracking - start point and reference point pose change too large, resettable
1-Command Error Point	113	Constant force control - X direction exceeds max adjustment distance, resettable

continues on next page

Table 9 – continued from previous page

Main Fault Code	Sub Fault Code	Description
1-Command Point Error	114	Constant force control - Y direction exceeds max adjustment distance, resettable
1-Command Point Error	115	Constant force control - Z direction exceeds max adjustment distance, resettable
1-Command Point Error	116	Constant force control - RX direction exceeds max adjustment angle, resettable
1-Command Point Error	117	Constant force control - RY direction exceeds max adjustment angle, resettable
1-Command Point Error	118	Constant force control - RZ direction exceeds max adjustment angle, resettable
1-Command Point Error	119	External sensor data error, resettable
1-Command Point Error	120	Helix exploration motion failed, resettable
1-Command Point Error	121	Rotational insertion motion failed, resettable
1-Command Point Error	122	Linear insertion motion failed, resettable
1-Command Point Error	123	Surface positioning motion failed, resettable
1-Command Point Error	129	Exceeded max torque record points, resettable
1-Command Point Error	130	Speed switching error, resettable
1-Command Point Error	147	Focus following error, resettable
1-Command Point Error	148	Attitude speed limit exceeded, resettable
1-Command Point Error	149	Joint status word feedback abnormal, resettable
2-Drive Fault	1	Axis 1 drive fault, not resettable
2-Drive Fault	2	Axis 2 drive fault, not resettable
2-Drive Fault	3	Axis 3 drive fault, not resettable
2-Drive Fault	4	Axis 4 drive fault, not resettable
2-Drive Fault	5	Axis 5 drive fault, not resettable
2-Drive Fault	6	Axis 6 drive fault, not resettable
3-Soft Limit Exceeded	1	Axis 1 soft limit exceeded, resettable
3-Soft Limit Exceeded	2	Axis 2 soft limit exceeded, resettable
3-Soft Limit Exceeded	3	Axis 3 soft limit exceeded, resettable
3-Soft Limit Exceeded	4	Axis 4 soft limit exceeded, resettable
3-Soft Limit Exceeded	5	Axis 5 soft limit exceeded, resettable
3-Soft Limit Exceeded	6	Axis 6 soft limit exceeded, resettable

continues on next page

Table 9 – continued from previous page

Main Fault Code	Sub Fault Code	Description
4-Collision Fault	1	Axis 1 collision fault, resettable
4-Collision Fault	2	Axis 2 collision fault, resettable
4-Collision Fault	3	Axis 3 collision fault, resettable
4-Collision Fault	4	Axis 4 collision fault, resettable
4-Collision Fault	5	Axis 5 collision fault, resettable
4-Collision Fault	6	Axis 6 collision fault, resettable
4-Collision Fault	7	End effector collision fault, resettable
5-Active Slave Count Error	1	Active slave count error, not resettable
6-Slave Error	1	Slave offline, not resettable
6-Slave Error	2	Slave status does not match set value, not resettable
6-Slave Error	3	Slave not configured, not resettable
6-Slave Error	4	Slave configuration error, not resettable
6-Slave Error	5	Slave initialization error, not resettable
6-Slave Error	6	Slave mailbox communication initialization error, not resettable
7-IO Error	1	Channel error, resettable
7-IO Error	2	Value error, resettable
7-IO Error	3	WaitDI timeout, resettable
7-IO Error	4	WaitAI timeout, resettable
7-IO Error	5	WaitAxleDI timeout, resettable
7-IO Error	6	WaitAxleAI timeout, resettable
7-IO Error	7	Channel configured function error, resettable
7-IO Error	8	Arc start timeout, resettable
7-IO Error	9	Arc end timeout, resettable
7-IO Error	10	Search positioning timeout, resettable
7-IO Error	11	Conveyor IO detection timeout, resettable
7-IO Error	12	WaitAuxDI timeout, resettable
7-IO Error	13	WaitAuxAI timeout, resettable
7-IO Error	14	Wire search positioning timeout, resettable
8-Gripper Error	1	Gripper motion timeout error, resettable
9-File Error	1	zbt config file version error, initialization error - not resettable
9-File Error	2	zbt config file load failed, initialization error - not resettable
9-File Error	3	user config file version error, initialization error - not resettable
9-File Error	4	user config file load failed, initialization error - not resettable
9-File Error	5	exaxis config file version error, initialization error - not resettable
9-File Error	6	exaxis config file load failed, initialization error - not resettable
9-File Error	7	Robot model inconsistent, requires reconfiguration - not resettable
9-File Error	8	dhpara config file version error, initialization error - not resettable
9-File Error	9	dhpara config file load failed, initialization error - not resettable
9-File Error	10	Robot model not set - not resettable
9-File Error	11	load config file version error, initialization error - not resettable
9-File Error	12	load config file load failed, initialization error - not resettable
9-File Error	13	speed config file version error, initialization error - not resettable
9-File Error	14	speed config file load failed, initialization error - not resettable
10-Singular Pose	1	Singular pose
11-Drive Communication Error	1	Axis 1 drive communication error, not resettable

continues on next page

Table 9 – continued from previous page

Main Fault Code	Sub Fault Code	Description
11-Drive Communication Error	2	Axis 2 drive communication error, not resettable
11-Drive Communication Error	3	Axis 3 drive communication error, not resettable
11-Drive Communication Error	4	Axis 4 drive communication error, not resettable
11-Drive Communication Error	5	Axis 5 drive communication error, not resettable
11-Drive Communication Error	6	Axis 6 drive communication error, not resettable
12-External Axis Soft Limit	1	External axis 1 soft limit exceeded, resettable
12-External Axis Soft Limit	2	External axis 2 soft limit exceeded, resettable
12-External Axis Soft Limit	3	External axis 3 soft limit exceeded, resettable
12-External Axis Soft Limit	4	External axis 4 soft limit exceeded, resettable
13-Parameter Setting Error	1	Tool number out of range, resettable
13-Parameter Setting Error	2	Positioning completion threshold error, resettable
13-Parameter Setting Error	3	Collision level error, resettable
13-Parameter Setting Error	4	Load weight error, resettable
13-Parameter Setting Error	5	Load center of mass X error, resettable
13-Parameter Setting Error	6	Load center of mass Y error, resettable
13-Parameter Setting Error	7	Load center of mass Z error, resettable
13-Parameter Setting Error	8	DI filter time error, resettable
13-Parameter Setting Error	9	AxleDI filter time error, resettable
13-Parameter Setting Error	10	AI filter time error, resettable
13-Parameter Setting Error	11	AxleAI filter time error, resettable
13-Parameter Setting Error	12	DI high/low level range error, resettable
13-Parameter Setting Error	13	DO high/low level range error, resettable
13-Parameter Setting Error	14	Workpiece number out of range, resettable
13-Parameter Setting Error	15	External axis number out of range, resettable

continues on next page

Table 9 – continued from previous page

Main Fault Code	Sub Fault Code	Description
13-Parameter Setting Error	16	Conveyor encoder channel error, resettable
13-Parameter Setting Error	17	Conveyor workpiece axis number error, resettable

1.3.19.2 Appendix 2: Servo driver fault code table

Fault code	Fault name	Processing method
1	Software overcurrent fault	1. Check whether the joint load or resistance becomes larger or abnormal; 2. If the fault is still not eliminated, repair or replace the drive board.
2	Overvoltage fault	Reduce the speed or acceleration of the robot.
3	Under voltage fault	1. Check whether the 48V power supply voltage output of the control box is abnormal; 2. Check the drive plate and joint housing for short circuit; 3. If the fault is still not eliminated, repair or replace the drive board.
4	Overheating fault	Reduce the load or speed of the robot.
5	Overload fault	Reduce the load or speed of the robot.
6	Overspeed fault	1. Check whether the magnetic braiding assembly and the motor shaft fixing jackscrew are loose; 2. Re-perform encoder zero calibration; 3. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
7	Parameter abnormal fault	Repair or replace the drive plate.

continues on next page

Table 10 – continued from previous page

Fault code	Fault name	Processing method
8	Runaway fault	1. Check whether the magnetic braiding assembly and the motor shaft fixing jackscrew are loose; 2. Re-perform encoder zero calibration; 3. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
9	Position error fault	1. Check whether the joint load or resistance becomes larger or abnormal; 2. If the fault is still not eliminated, repair or replace the drive board.
10	Position overflow fault	1. Check whether the hard limit is loose; 2. Re-perform robot zero calibration.
11	Hardware overcurrent fault	Repair or replace the drive plate.
12	Drive inhibit fault	The robot is NOT enabled.
13	Motor locked rotor fault	1. Check whether the brake electromagnet is engaged; 2. Check whether the hard limit is hit; 3. If the fault is still not eliminated, repair or replace the drive board.
14	Power supply failure	The robot is NOT enabled.
15	STO fault	The robot is NOT enabled.
16	Phase current AD zero setting fault	Repair or replace the drive plate.
17	EEPROM fault	Repair or replace the drive plate.
18	Hall fault	1. Check whether the hall harness is inserted firmly and whether there is short circuit or open circuit; 2. If the fault is still not eliminated, repair or replace the joint.
19	Encoder failed	Repair or replace the magnetic braid assembly.

continues on next page

Table 10 – continued from previous page

Fault code	Fault name	Processing method
20	Encoder zero setting fault	1. Re-perform encoder zero calibration; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
21	Encoder Z-phase signal loss fault	The robot is NOT enabled.
22	Encoder count fault	The robot is NOT enabled.
23	Encoder multi turn data overflow fault	The robot is NOT enabled.
24	External clock fault	Repair or replace the drive plate.
25	UVW phase sequence fault	The robot is NOT enabled.
26	FPGA fault	The robot is NOT enabled.
27	Zero return fault	The robot is NOT enabled.
28	Magnetic encoder fault	1. Check whether the magnetic braiding assembly and the motor shaft fixing jackscrew are loose; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
29	Motor power line disconnection fault	1. Check whether the power line of the motor is firmly inserted, and whether there is short circuit or open circuit; 2. If the fault is still not eliminated, repair or replace the drive board.
30	EtherCAT fault	1. Check whether the network cable is firmly plugged, and whether there is short circuit or open circuit; 2. If the fault is still not eliminated, repair or replace the drive board
31	EtherCAT_SM_DOG fault	1. Check whether the network cable is firmly plugged, and whether there is short circuit or open circuit; 2. If the fault is still not eliminated, repair or replace the drive board.

continues on next page

Table 10 – continued from previous page

Fault code	Fault name	Processing method
32	EtherCAT_FATALSYNC failure	1. Check whether the network cable is firmly plugged, and whether there is short circuit or open circuit; 2. If the fault is still not eliminated, repair or replace the drive board.
33	EtherCAT_SYNC fault	1. Check whether the network cable is firmly plugged, and whether there is short circuit or open circuit; 2. If the fault is still not eliminated, repair or replace the drive board.
34	EtherCAT_RFT failure	1. Check whether the network cable is firmly plugged, and whether there is short circuit or open circuit; 2. If the fault is still not eliminated, repair or replace the drive board.
35	Drive shaft address fault	1. Re-configure the drive axis address; 2. If the fault is still not eliminated, repair or replace the drive board.
36	Robot zero calibration fault	1. Re-perform robot zero calibration; 2. First use JLINK to erase FLASH, then download the program again and zero; 3. If the fault is still not eliminated, repair or replace the drive board.
37	Encoder communication failure	1. Check whether the encoder harness is inserted firmly and whether there is short circuit or open circuit; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.

continues on next page

Table 10 – continued from previous page

Fault code	Fault name	Processing method
40	Magnetic encoder module failure - zero calibration failure	1. Re-zero the magnetic braiding assembly; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
41	Magnetic encoder module fault - multi turn fault	1. Check whether the magnetic braiding assembly and the motor shaft fixing jackscrew are loose; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
42	Magnetic encoder module failure - multi turn small magnetic encoder failure	1. Check whether the multi turn small magnetic braiding chip is abnormal; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
43	Magnetic encoder module failure - multi turn large magnetic encoder failure	1. Check whether the multi turn magnetic braiding chip is abnorma; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
44	Magnetic braiding module fault - single turn magnetic braiding fault	1. Check whether the single turn magnetic braiding chip is abnormal; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.
45	Magnetic encoder module failure - optical encoder failure	1. Check whether the optical coding disc is polluted or not stuck; 2. If the fault is still not eliminated, repair or replace the magnetic editor assembly.

1.3.19.3 Appendix 3: End plate 485 upgrade

During field use, it is possible to update the firmware to meet the new requirements. A new upgrade file (XX_XX_MAIN. bin) will be provided to upgrade the terminal board through the 485 interface (USB to 485 module is required). The upgrade steps are as follows:

Step1: 485 wiring. There is a 5Pin communication aviation connector at the end of the robot. The pin pin distribution and pin pin description of the aviation connector are shown in Fig. 1.Connect the 485 - A and 485 - B at the end of the robot with A and B of the USB to 485 tool using twisted pair cables.

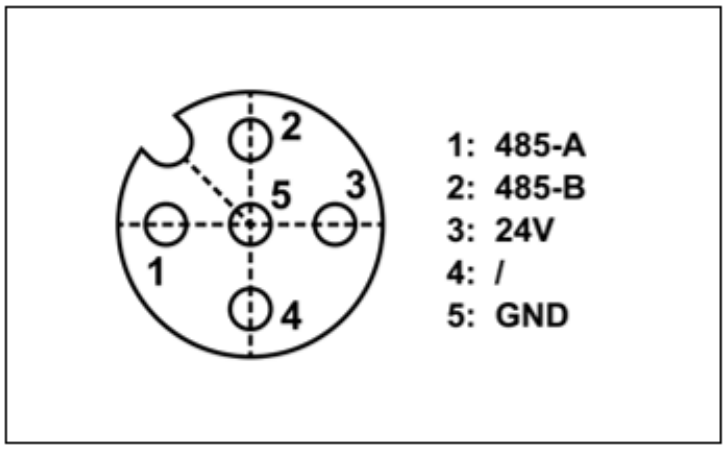


Figure 18.3-1 Pin Pin Distribution of Aviation Connector

Step2: hardware connection. connect the USB end of the USB to 485 tool to the PC, and in the PC device manager, if the USB&485 tool is identified, the following interface will appear.

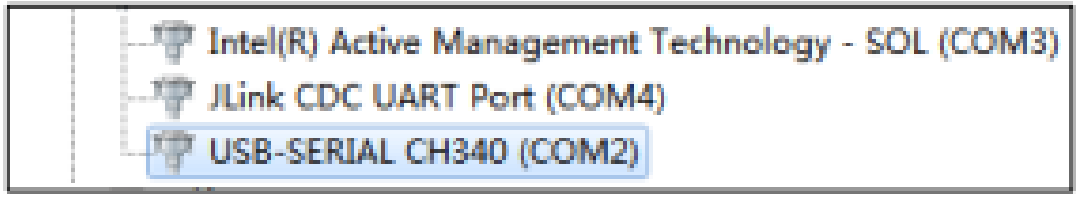


Figure 18.3-2 USB&485 Port Identification Description

Step3:Upgrade the tool. After the wiring is completed, open the "Fair Serial Port Debugging Assistant", click the "Terminal Board" button, and select the above identified serial port in the "Serial Port Parameter Setting" function. The baud rate is 115200, the data bit is 8, the check bit is none, and the stop bit is 1. Then open the serial port. After success, the prompt "Serial port opened successfully" will appear.

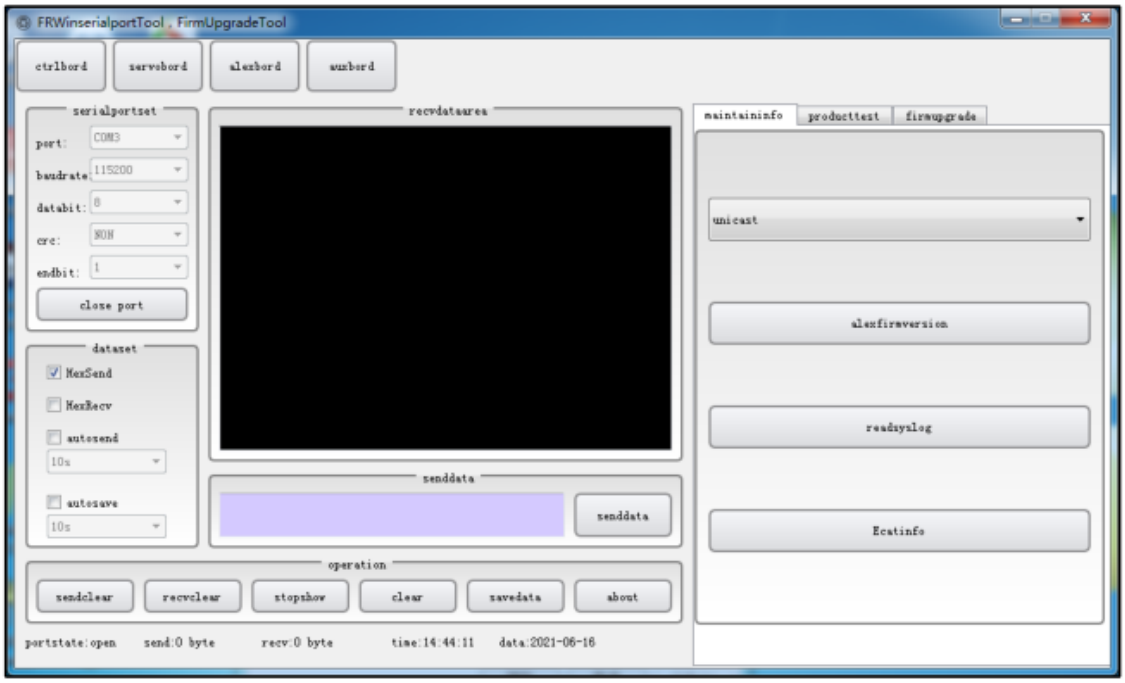
Figure 18.3-3 Serial Port Parameter Settings

Step4:Firmware upgrade. Select "End plate" and click "Firmware upgrade", as shown in the fig.

Figure 18.3-4 End Plate Firmware Upgrade

- First, click "Flash Erase". After the erasure is successful, you will be prompted in the receiving data area that the erasure is successful.
- Open the file (the file to be upgraded), select the path to store it, as shown below. After the selection, the file name to be upgraded will appear in the file name display box.


serialportset
port:
COM3
baudrate:
115200
databit:
8
crc:
NON
endbit:
1
close port


FRWinserialportTool , FirmUpgradeTool
ctrlbord
servobord
alexbord
auxbord
serialportset
port:
COM3
baudrate:
115200
databit:
8
crc:
NON
endbit:
1
close port
recvdataarea
maintaininfo
producttest
firmupgrade
unicast
alexfirmversion
readsyslog
Ecatinfo
dataset
HexSend
HexRecv
autosend
10s
autosave
10s
senddata
senddata
operation
sendclear
recvclear
stopshow
clear
savedata
about
portstate:open
send:0 byte
recv:0 byte
time:14:44:11
data:2021-06-16

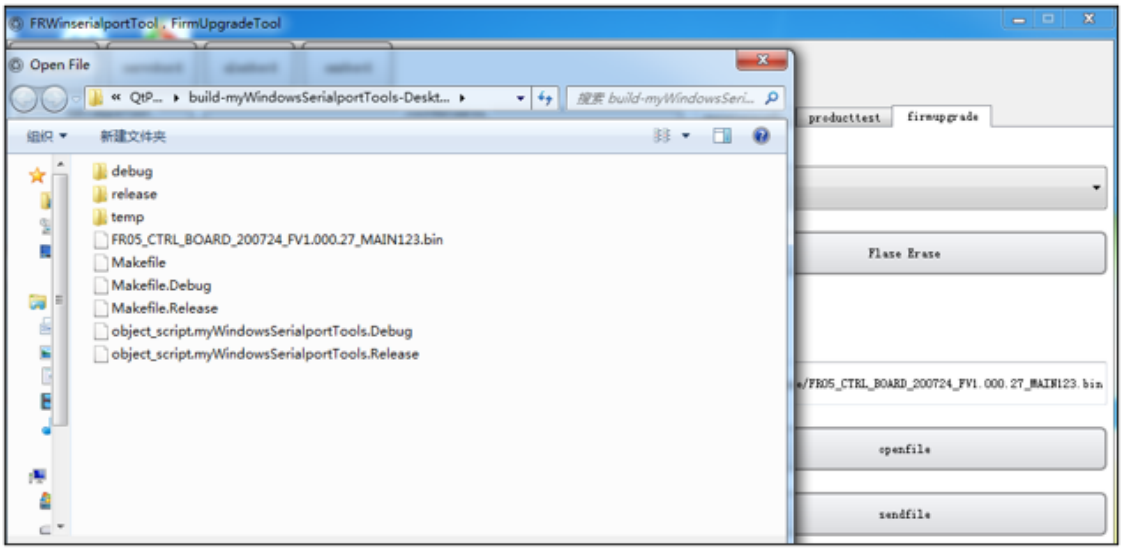


Figure 18.3-5 Select Upgrade File

- Click “Send File”, and when the progress bar displays 100%, it means that the upgrade file has been sent.

Step5:Upgrade verification. The system is restarted and powered on. In the “Maintenance Information” column, select “Query terminal board firmware version information”, and the firmware version information will be displayed in the “Receive Data Area”. If it is consistent with the upgraded file version information, the upgrade is successful, otherwise the upgrade fails.

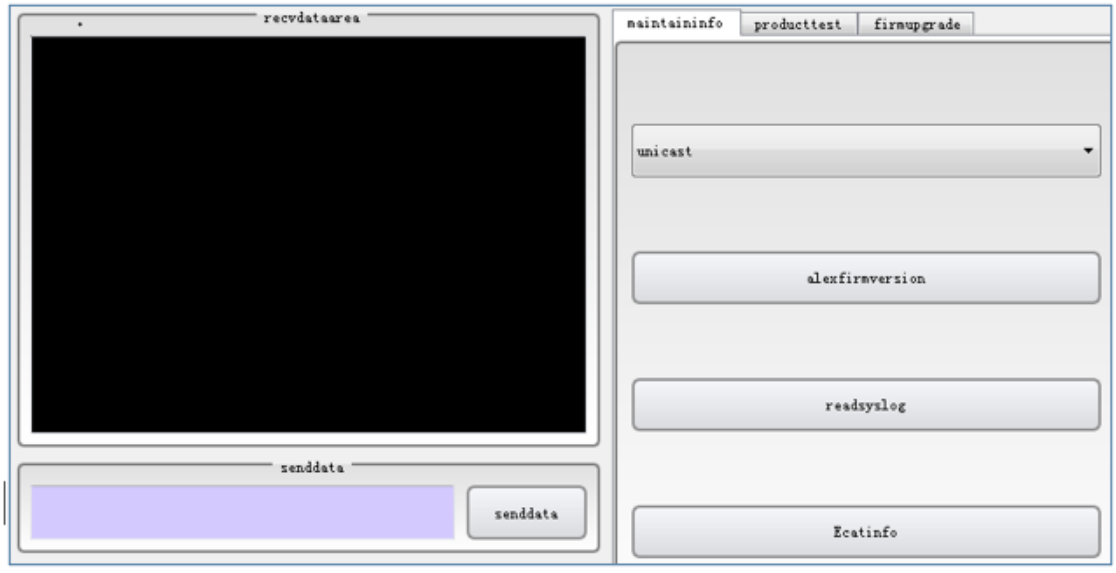


Figure 18.3-6 Querying Firmware Version Information

1.3.19.4 Appendix 4: Control Box 485 Upgrade

There is a "power communication" interface on the robot control box board, and USB&485 tools A and B are respectively connected to the "485-A" and "485-B" of its interface.

The upgrade process is the same as that of the terminal board, and the software can be selected accordingly, which will not be repeated here.

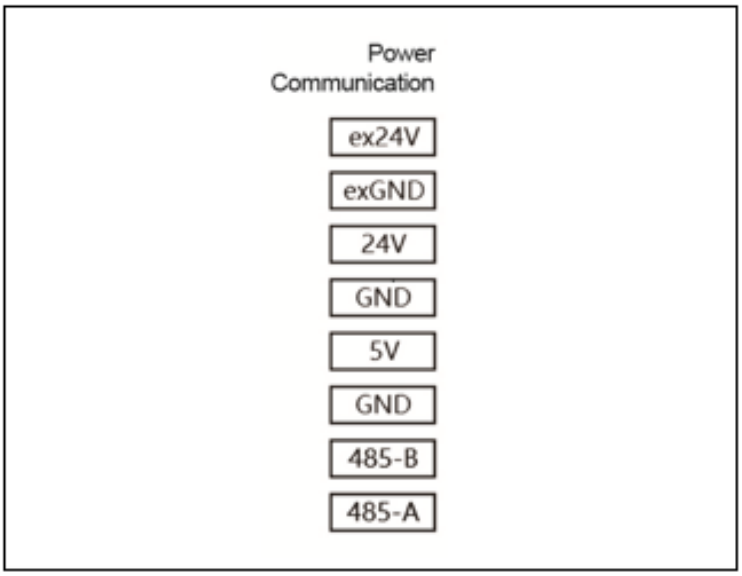


Figure 18.4-1 Power Communication Interface

1.3.19.5 Appendix 5: List of Spare Parts and Vulnerable Parts

Spare Part Name	Part No.	Quantity/piece(s)
M8 * 30 screw	4.0.08.2006185	4
Straight pin A type 8 * 20	4.5.00.2013076	2
Fuse 5x20 6A	/	1

1.3.20 Term

Shutdown Category:

- **Class 0 shutdown**:When the power supply of the robot is cut off, the robot stops working immediately. This is an uncontrollable stop. Since each joint will brake at the fastest speed, the robot may deviate from the path set by the program.This protective stop can be used when the safety assessment limit is exceeded, or when an error occurs in the safety assessment part of the control system. For more information, refer to ENISO13850:2008 or IEC60204-1:2006.
- **Class 1 shutdown**:When the robot is powered to stop, the robot will stop. When the robot stops, the power will be cut off. This is a controllable stop, and the robot will follow the programmed path.Cut off the power supply after one second or once the robot stops stably. For more information, refer to ENISO13850:2008 or IEC60204-1:2006.

- **Class 2 shutdown**:Controllable stop when the robot is powered on. The robot stops all movements within one second.The operation of the safety assessment control system can make the robot stay at the stop position. For more information, refer to IEC60204-1:2006.

Diagnostic coverage rateDC:Used to measure the effectiveness of the diagnosis implemented to achieve the assessed performance level. For more information, see ENISO13849-1:2008..

Integrator:The integrator is the mechanism that designs the final installation of the robot. The integrator is responsible for the final risk assessment and must ensure that the final installation complies with local laws and regulations.

Mean time to dangerous failureMTTFd:The mean time to dangerous failure (MTTFd) refers to the value calculated and detected to reach the assessed performance level. For more information, see ENISO13849-1:2008.

Risk assessment:Risk assessment is the whole process of identifying all risks and reducing them to an appropriate level.The risk assessment shall be recorded and archived.For details, please refer to ISO12100.

Performance Level:Performance Level (PL) is a separate level, which is used to describe the ability of each safety related part of the control system to perform safety functions under predictable conditions. PL is the second highest reliability category, which means that the security function is quite trustworthy.For more information, refer to EN ISO13849-1:2008.

Connecting flange:The structure used to connect external tools, generally called flange.

Robot end: The center point of the last axis or connecting flange of the robot.

Tool center pointTCP:The tool center point is the characteristic point of the robot tool, which is the control point of the robot system. It defaults to the center of the last moving axis or connecting flange when leaving the factory. The tool center point of each tool contains transformations and rotations set relative to the center of the tool output flange. The position coordinates X, Y, Z determine the position of the tool center point, and RX, RY, RZ determine the direction of the tool center.When the values are all zero, the center of the tool coincides with the center of the connecting flange.

Tool position and attitude pointTCF:Reflect the attitude of the tool coordinate system relative to the end link coordinate system on the basis of the tool center point TCP.

Base coordinate system:The origin of the base coordinate system is generally defined at the center point between the first axis of the robot and the mounting surface. The x-axis is forward, and the y-axis is determined according to the right-hand rule in the axial direction.

World coordinate system:A fixed coordinate system established in a work unit or workstation.When there is only one robot, the coordinate system can be considered to coincide with the base coordinate system;When there are multiple robots or external devices, the world coordinate system can provide a unique reference system for these devices. On the premise of satisfying the coordinate system for convenient calibration of other devices, its specific position can be arbitrarily specified.

Joint coordinate system: The joint coordinate system is the coordinate system in the robot joint. Under the joint coordinate system, each axis of the robot can achieve independent forward or reverse movement within the limit range. It is applicable to the robot that needs to move in a large range and does not require the TCP posture of the robot. The single axis inching of the robot in manual mode is carried out in the joint coordinate system.

Tool coordinate system:The coordinate system used to define the position of the tool center point and the tool attitude. If not defined, the tool coordinate system defaults to the center of the connecting flange.After installing the tool, TCP will change and become the center of the end of the tool.

External tool coordinate system: The coordinate system used to define the position and pose of tools fixed outside the robot.

Extension axis:Remove the axis on the robot body, and add additional axis for work. The extension axis mainly includes slide rail, turntable and external servo equipment.

Manual mode: In this mode, all movements of the robot are manually controlled by the user, and external safety facilities such as safety gratings and safety doors do not work, so as to facilitate close debugging.

Automatic mode:This mode is generally used for the robot to run the teaching program. At this time, the external safety facilities are enabled.

Repetitive positioning accuracy:The consistency of the position and attitude measured by the robot in the same condition and the same method for n times.

Teaching pendant:A hand-held unit that programs or moves the robot and is connected to the control system.

1.4 Changelog

1.4.1 Version V3.9.3

Date: 2026-02-11

- **Optimized SmartTool Motion Command Speed Settings:**
  Path: Initial Setup -> Peripherals -> Welding Handle.

  Description: The speed setting range for PTP, LIN, and ARC commands has been adjusted to 0-100% (the original speed setting upper limit was 30%).

- **Optimized Force Sensorless External Force Observation and Improved Drag Experience:**
  Path: Auxiliary Applications -> Tool Applications -> Drag Lock.

  Description: Improved external force observation accuracy to ±0.5N.

- **Optimized Modbus TCP Master Reconnection Mechanism:**
  Path: Teach Program -> Program Programming -> Modbus TCP.

  Description: Added pause logic.

- **Added Extended Axis Debugging Command Function:**
  Path: Initial Setup -> Peripherals -> Extended Axis.

  Description: Implemented robot inverse kinematics solution in the extended axis coordinate system and ServoCart servo motion with extended axes.

- **Added Pause/Resume DO Reset State Configurable Function:**
  Path: Initial Setup -> Peripherals -> Welding Handle.

  Description: Option to choose whether the DO output state after resuming motion is consistent with the state before pause.

1.4.2 Version V3.9.2

Date: 2026-01-26

- **Optimized Joint Torque Sensor Position Loop Drag Teaching:**
  Path: Auxiliary Applications -> Tool Applications -> Drag Lock.

  Description: Resolved insufficient point alignment accuracy within 2 seconds during joint-to-joint dragging.

- **Optimized Extended Board Card**:

  Description: Optimized the extended DO data synchronization mechanism.

- **Optimized Arc Tracking Function:**
  Path: Initial Settings -> Peripherals -> Welding Machine.

  Description: Added real-time welding current acquisition via the peripheral open protocol ModbusTCP.

- **Optimized Controller and Physical Teach Pendant IP Reset Function:**
  Path: System Settings -> Network.

  Description: Added a "Reset Controller IP" function in webrecovery, requiring secondary confirmation; Added an IP reset function to the physical teach pendant; Added the ability for the physical teach pendant to reset its own IP via physical buttons when not connected. The teach pendant IP is 192.168.58.77.

- **Added New FR3C Robot Configuration**:

  Description: Joint operating range is consistent with the FR3 series, upper and lower arms are consistent with WMS, rated load is 3kg.

- **Added Cube Interference Function:**
  Path: Initial Settings -> Safety -> Interference Zone -> Cube Interference.

  Description: Implemented simultaneous interference checking for at least 4 cubes and configurable CO output corresponding to interference signals.

- **Added Slave_interpret Module Function Extension (PROFINET Protocol) and Web Full IO Display Function:**
  Path: Initial Settings -> Peripherals -> Board Communication -> Board Upgrade.

  Description: Added Jiyuan board EtherCAT firmware upgrade, PLC communication detection, configurable cycle, and Web full IO display function (PROFINET protocol).

- **Added Smarttool Open Protocol Functions:**
  Path: Initial Settings -> Peripherals -> Welding Handle.

  Description: Added program creation parsing and detection signal processing based on the open protocol, and added anti-misoperation function to the "Undo, Delete" keys. Added automatic generation function for smarttool open protocol key configuration.

- **Added QX and LA Software Version Backup Package Compatibility:**
  Path: Auxiliary Applications -> Tool Applications -> Data Backup.

  Description: QX supports and can fully import LA backup packages, LA supports and can fully import QX backup packages; Compatible with backup packages from version v3.8.7 onwards.

- **Added Configurable Parameters for End CI and Controller CI:**
  Path: Initial Settings -> Basic -> I/O Settings -> DI.

  Description: Enables configurable motion interruption for End CI and synchronization of some configurable controller CI functions to End CI.

- **Added Robot Laser Data Servo Tracking Function:**
  Path: Initial Settings -> Peripherals -> Line Laser Sensor.

  Description: Enables the robot to directly track using laser sensor data.

- **Added Teach Pendant Configuration Function**:

  Description: The F1-F4 keys on the teach pendant support configuring custom functions via the WEB interface; Key switching to custom mode can be configured as drag mode.

1.4.3 Version V3.9.1

Date: 2025-12-30

- **Robot Background Logging of Fault and Alarm Events Function**:

  Description: Capable of logging fault and alarm information during user interface logouts, viewable after re-login.

- **PTP and LIN Commands Add Current Position Relative Motion Function, WebApp Adds ServoJ Command Addition Function:**
  Path: Teach Program -> Program Programming -> PTP/LIN Commands.

  Description: Solves the issue of complex command operations when the robotic arm lacks taught points but requires offset based on the current position. A single command input achieves the motion.

- **End-effector Lua Open Protocol Debug Switch and Syntax Verification Function:**
  Path: Initial Setup -> Peripherals -> Gripper, Force Sensor, Welding Handle.

  Description: The controller's integrated system has now adapted to end-effector protocols (gripper, force sensor, welding handle). This optimization allows direct selection and application to the end-effector within the webAPP. The webServer now includes Lua syntax verification for protocols, reporting errors directly if verification fails. The webAPP adds an Lua open protocol debug switch, which, when enabled during debugging, runs the end-effector Lua while suppressing sensor communication timeout errors.

- **FRCap Palletizing Scenario Function Optimization:**
  Path: Auxiliary Applications -> Process Package -> Palletizing.

  Description: Palletizing now supports two pick points. After materials arrive on two conveyor lines, the robot sequentially picks materials from the pick points on the two lines and places them onto two corresponding pallets.

- **Physical Teach Pendant Function Optimization:**
  Path: System Settings -> General Settings -> Network

  Description: Previously, users had to switch to the webAPP first before accessing their host computer interface when using the teach pendant, which was cumbersome. After optimization, by rotating the physical key switch without logging into the teach interface, the robot can switch between manual/auto modes.

- **Collision Function Optimization:**
  Path: Initial Setup -> Basic -> Joint -> Collision Level -> False Alarm Detection.

  Description: When the collision level is set to 1, the robot can run at any speed without false alarms due to high speed. Errors and stops occur only upon actual collision or squeezing.

- **Photoelectric Sensor TCP Calibration Function:**
  Path: Initial Setup -> Basic -> Tool Coordinates -> Photoelectric Auto Calibration.

  Description: Supports automatic calibration for straight shank welding torches and bent shank welding torches;

1.4.4 Version V3.9.0

Date: 2025-11-27

- **DARU DFC Force Control Polishing Head Application:**
  Path: Initial Setup -> Peripherals -> Polishing -> DARU DFC Force Control Polishing Head.

  Description: The software component for the DFC intelligent flexible polishing force control peripheral has been adapted and tested, meeting the adaptation requirements of the DFC polishing system.

- **Parameter Calibration of Joint Torque Sensor on the Complete Machine:**
  Path: Auxiliary Applications -> Tool Applications -> Drag Lock -> Joint Torque Sensor Whole Machine Drag.

  Description: Run a predetermined trajectory to complete parameter calibration for the joint torque sensor's sensitivity, linearity, hysteresis error, and repeatability accuracy.

- **New Joint Torque Sensor-Based Guided Drag Teaching Function to Avoid Overshoot:**
  Path: Auxiliary Applications -> Tool Applications -> Drag Lock -> Joint Torque Sensor Whole Machine Drag.

  Description: Mitigates drag overshoot phenomena in the current loop. After enabling this function, drag-to-point operations can be performed conveniently.

- **Intersecting Line Welding Function:**
  Path: Teach Program -> Program Programming -> Intersecting Line Instruction.

  Description: Record 6 taught points on the cross-sections of the main pipe and branch pipe respectively, input parameters such as motion direction, speed, acceleration, and offset values. The robot can then generate the intersecting line trajectory formed by the intersection of the two circular pipes and perform welding.

- **Modbus Wait for Analog Input Setting Adds Equal Judgment Function:**
  Path: Teach Program -> Program Programming -> Modbus Instruction.

  Description: Added equal judgment for the slave station wait for analog input wait status.

- **Teach Point One-Click Update & LIN Single Point Execution Function:**
  Path: Teach Program -> Teach Points -> Teach Management.

  Description: On the teach point interface, update the current robot pose to the corresponding point via the robot body operation bar, with the option to sync the teach program. Added LIN single point execution method.

- **Optimized Point Recording Function in Point Table Mode:**
  Path: Teach Program -> Teach Points -> Teach Management.

  Description: Re-recording points in point table mode can synchronously update the Lua program. Optimized to resolve occasional overspeed in PTP reduction mode and occasional motion stop after arrival in LIN speed adjustment mode.

- **User Configuration File:**
  Description: Optimized handling of robot user configuration file loading failures. Added automatic restart backup configuration file and rollback backup configuration file functions.

- **Web Interface Version:**
  Description: Web interface version updated to 2.0, featuring updated and optimized software operation interface.

- **New V6.5 Robot Models:**
  Description: Through hardware (such as reducers) and algorithm optimizations, robot vibration is reduced and trajectory accuracy is improved. Added FR3, FR5, FR10, FR16, FR20 robot models.

- **New FR5C Robot Model Configuration:**
  Description: The LA version software adds FR5C robot model configuration functionality.

1.4.5 Version V3.8.7

Date: 2025-10-21

- **Linear Rack Guideway Robot Collision Detection Function:**
  Path: Initial Setup -> Basic -> Joints -> Collision Level -> Linear Rack Guideway Robot Collision Detection.

  Description: Enables the guideway to perform an emergency stop upon collision during movement, thereby enhancing operational safety.

- **New Actual Physical Speed Setting Function for New Helix:**
  Path: Teach Program -> Program Programming -> New Helix N-Spiral Command.

  Description: Ensures the constant velocity segment of the robot end-effector's linear speed matches the set value.

- **Safety Stop Recovery Optional Auto Enable Function:**
  Path: Initial Setup -> Safety -> Emergency Stop.

  Description: Added configuration for enable strategy after Category 1b emergency stop reset.

- **Force Sensor Loaded Zeroing and Open Posture Compliance Admittance Parameters:**
  Path: Initial Setup -> Basic -> Load, Teach Program -> Program Programming -> F/T Control Command.

  Description: Added open posture compliance admittance parameters.

- **New Laser Tracking Function for Arcs and Full Circles**:

  Description: Enables real-time laser tracking functionality for arcs and full circles, with posture following changes during motion, allowing laser scanning reproduction for arcs and full circles.

- **Button Box Function**:

  Description: Optimized the Button Box Version 1.0 reset IP function.

1.4.6 Version V3.8.6

Date: 2025-09-19

- **New Robot Impedance Control Function:**
  Path: Teach Program -> Program Programming -> F/T Command.

  Description: Through real-time detection of external forces, the robot actively complies with the external force and deviates from the motion trajectory when the set threshold is reached. It returns to the motion trajectory when the external force drops below the threshold, enabling better human-robot interaction.

- **New Torque Detection Function Before Dragging:**
  Path: Initial Setup -> Basic -> Collision Level.

  Description: Before entering drag mode, the difference between the torque command and feedback for each joint is calculated. If the load or installation method is incorrectly configured, causing the difference to exceed the threshold, entry into drag mode is prevented to avoid robot.

- **New Custom Oscillation Welding Function:**
  Path: Teach Program -> Program Programming -> Weave Command.

  Description: Users can design their own oscillation welding patterns to execute oscillation welding.

- **ModbusTCP:**
  Path: Teach Program -> Program Programming -> ModbusTCP.

  Description: Optimized the ModbusTCP master station timeout detection function.

- **Pneumatic Suction Cup:**
  Path: Initial Setup -> Peripherals -> Array Suction Cup.

  Description: Optimized the web page adaptation function for pneumatic suction cups.

- **Wire Feeding Search Position Fault Error**:

  Description: Optimized WebAPP wire feeding search position fault error reporting, which can be reset.

- **Configurable IO Function**:

  Description: Added configurable IO function for FR3C-FR3MT. Configurable inputs are CI0-CI4, and configurable outputs are CO0-CO4.

1.4.7 Version V3.8.5

Date: 2025-08-19

- **Socket Network Debugging:**
  Path: Teach Program -> Program Programming -> Socket Network Debugging.

  Description: Allows adding, configuring, and deleting socket connection configurations, supporting up to 4 socket connections. Command modules can generate teach programming for opening connections, closing connections, sending data, and receiving data.

- **CAD to G-code to Robot Trajectory Planning Function:**
  Path: Auxiliary Applications -> Tool Applications -> G-code Conversion.

  Description: Generate G-code files for machining trajectories like lines, arcs, circles, and splines in CAM software such as Solidworks and FUSION360, then import the G-code files into the Web.

- **Added Actual Physical Speed Function for Linear, Arc, and Full Circle Motion Commands:**
  Path: Teach Program -> Program Programming, Initial Setup -> Peripherals -> Welding Handle.

  Description: Allows direct definition of the actual running physical speed for the current motion command.

- **Optimized PTP Speed Adjustment Function**:

  Description: Ensures a smooth speed adjustment process and reduces speed fluctuations at the points of speed adjustments.

- **Robot Model Configuration**:

  Description: Added FR30L robot model configuration option.

- **Control Box Adaptation**:

  Description: Added control box adaptation for ETMP03E board (EtherCAT protocol).

1.4.8 Version V3.8.4.1

Date: 2025-07-30

- **Control Box Adaptation for Ethernet Pass-Through Module and Suction Cup/Gripper Control Function:**
  Path: Initial Setup -> Peripherals -> Array Suction Cup.

  Description: Enables control of array suction cups (up to 20) via an Ethernet to 485 pass-through module, based on the page and peripheral open protocol.

- **Robot Look-ahead Trajectory Planning (Constant Velocity Look-ahead):**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: Added a constant velocity switch option. When enabled, the robot performs look-ahead motion at constant velocity.

- **Optimized Control Box Slave Mode Function:**
  Path: Initial Setup -> Peripherals -> Board Communication, Remote Mode.

  Description: Added control box adaptation for Ji Yuan ETMP03E board (EtherCAT protocol).

- **Laser Search Position Point Acquisition Function:**
  Path: Initial Setup -> Peripherals -> Line Laser Sensor.

  Description: After successful laser seam search, the position data stored in "seamPos" can be obtained using SDK or TCP protocol commands.

- **FR5 and FR10 Speed Opening Function**:

  Description: Increased FR5 linear speed from 1.0m/s to 1.7m/s; increased FR10 linear speed from 1.5m/s to 2.0m/s.

- **Robot Model Configuration**:

  Description: Added FR5-WML long-reach robot model configuration option.

- **Optimized Software Upgrade**:

  Description: Shortened upgrade time, compatible with downgrading from high versions to any lower version (3.7.6-3.8.4).

- **Optimized Factory Reset**:

  Description: Factory reset preserves robot model, stiffness, dynamics settings, button box version, etc.

1.4.9 Version V3.8.4

Date: 2025-07-18

- **Implementation of Extension Axis Blending Smoothing Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: Achieves smooth motion between extension axis commands, improving work efficiency.

- **Optimized Control Box Slave Mode Function:**
  Path: Initial Setup -> Peripherals -> Board Communication, Remote Mode.

  Description: Added board IP configuration, interface LUA command issuance, and added self-start function for the controller slave protocol in remote mode.

- **Control Box N2L Board QX Adaptation (EtherCAT):**
  Description: FaRo Industrial real-time bus communication board, miniPCIe board, supporting EtherCAT protocol communication.

- **Robot JOG Motion:**
  Description: Added robot JOG motion CO status output function.

1.4.10 Version V3.8.3

Date: 2025-06-27

- **Control Box Slave Mode Function:**
  Path: Initial Setup -> Peripherals -> Board Communication.

  Description: Added board communication module configuration in the robot peripherals interface, enabling interaction with the robot slave station via EIP, CClink, PN protocols based on domestic expansion boards.

- **Added Collision Detection Function for Force Sensor Assisted Dragging in Manual Mode:**
  Path: Auxiliary Applications -> Tool Applications -> Drag Lock -> Force Sensor Assisted Lock.

  Description: When the force sensor assisted dragging function is enabled and the robot is in manual mode, the collision detection function is also triggered to avoid damage to the end equipment.

- **Added T-shaped Velocity Profile Trajectory Planning + Blending Smoothing Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: Achieves smooth motion between various motion commands, improving work efficiency.

- **WebApp Status Query Interface Function Optimization:**
  Path: Status Information -> Status Query.

  Description: Optimized status query parameters to 6, maximum waveform time to 30s, added data view display with copy capability, and added rename icon function.

- **FR Full Series Robot Drag Force Optimization:**
  Path: Initial Setup -> Basic -> Joint -> Friction Compensation -> Drag Force Compensation.

  Description: Makes dragging more energy-saving by providing compensation torque during the dragging process.

- **Web System Log Storage Function Optimization:**
  Description: Fixed system log file abnormality issues, set default log retention days to 7 days.

1.4.11 Version V3.8.2

Date: 2025-05-29

- **End Lua Open Protocol Adaptation for Welding Handle Function:**
  Path: Initial Setup -> Peripherals -> Welding Handle.

  Description: Enables adaptation of the smarttool welding handle using the open protocol, with support for configuring all parameters.

- **Controller Peripheral Open Protocol Adds Welding Machine Open Protocol:**
  Path: Initial Setup -> Peripherals -> Welding Machine.

  Description: Enables communication with the welding machine for control via the controller peripheral open protocol (ModbusTCP).

- **Added LUA Program Pause Function:**
  Path: Teach Program -> Program Programming.

  Description: During teach program execution, pause can be clicked on any line, including wait-type instructions and communication-type instructions, without occupying timeout time when paused.

- **Added T-shaped Velocity Profiling + Blending Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: After enabling the T-shaped velocity profiling switch, the velocity curve becomes smooth. Currently supports blending types (PTP-PTP, LIN-LIN, ARC-ARC, LIN-ARC, ARC-LIN).

- **FIR Adaptive Parameter Function + FIR Pause/Resume Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming, Initial Setup -> Safety -> Motion Configuration.

  Description: Added global FIR enable and adaptive parameter switch to the FIR function. When enabled, running ordinary PTP/LIN/ARC instructions automatically converts them to FIR planning (cannot use blend radius).

- **Modbus RTU Function Optimization:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming, Teach Program -> Program Programming -> Modbus RTU Settings.

  Description: Allows configuring ModbusRTU master station, baud rate, parity mode, station number, etc., in WebApp. Enables monitoring of real-time ModbusRTU register values.

- **Joint Soft Limit Protection Function:**
  Path: Initial Setup -> Joint -> Soft Limits.

  Description: In drag mode, when a joint approaches the soft limit set in the interface, damping force is applied. After removing the external torque, the joint can return within the soft limit.

- **Robot Oscillation Roll Angle Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: Supports the robot end tool customizing the oscillation angle around the Rx direction of the oscillation coordinate system during oscillation motion.

- **Process Parameter Gradual Change Welding Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: Enables gradual change of travel speed, current, and voltage during the welding process.

- **Analog Arc Tracking Function:**
  Path: Initial Setup -> Basic -> I/O Settings -> AI.

  Description: Directly uses control box DI, DO, AI, AO to connect with the welding machine IO ports, achieving arc tracking based on analog current feedback.

- **FRCap Plugin System + Palletizing Plugin Package Function**:

  Description: Adapted FRCap plugin system and palletizing plugin package for QX x86 version. FRCap can interact with the robot controller through provided official interfaces, or customers can write custom interface commands and processing logic based on actual needs for personalized development.

- **Robot Model and Configuration**:

  Description: Added FR3-C robot model and configuration.

1.4.12 Version V3.8.1

Date: 2025-04-14

- **T-shaped Velocity Function Optimization:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming, Initial Setup -> Basic Module.

  Description: Added trapezoidal velocity optimization mode in motion configuration, optimizing jerk during start/stop phases and increasing acceleration during the acceleration phase. Mainly applied to working conditions prone to collision reports during start/stop, and conditions with significant residual vibration during start/stop.

- **Conveyor Tracking Function Optimization:**
  Path: Auxiliary Applications -> Process Package -> Conveyor Tracking.

  Description: Added tracking inspection motion function in conveyor tracking motion mode, eliminating the need for motion teaching in the workpiece coordinate system, allowing custom trigger delay distance.

- **Added Impedance Callback Function for Axis Interference Zone:**
  Path: Initial Setup -> Safety -> Interference Zone.

  Description: Under force sensor assisted dragging, entering the interference zone can have an impedance callback effect.

- **Welding Handle Adaptation Function:**
  Description: Added welding handle adaptation: A welding handle installed on the robot end can be used for welding program writing and controlling robot start, stop program, manual/auto switching, dragging, etc.

- **WEB Interface Added Joint Limit Ring Display:**
  Description: Enabling the limit ring display function allows real-time display of the joint position.

- **SDK New Functions:**
  Description: SDK added functions for downloading controller logs, all data sources, and data backup packages.

- **Collision Pre/Post Joint Information Recording Function:**
  Description: Records joint position, velocity, acceleration, and torque information before and after a collision, facilitating direct analysis of the cause of collisions at the customer site.

1.4.13 Version V3.8.0

Date: 2025-03-03

- **Offset and Amplitude Monotonic Gradual Change Arc Tracking Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: The oscillation width among oscillation parameters can gradually change during a welding motion, transitioning from the segment start amplitude parameter to the segment end amplitude parameter. The welding center can actively offset from the seam center.

- **Custom Collision Detection Torque Threshold Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: Allows setting the collision detection torque threshold during operation. The joint torque threshold or TCP side torque threshold can be selected as needed. After a collision, joint speed, acceleration, torque, and other information are printed to the log file.

- **Real-time Look-ahead Trajectory Planning Method Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: Achieves smooth connection of multiple trajectory points through fitting, with running speed adaptively adjusted according to path curvature.

- **FR Full Series Model Load Curve Drawing:**
  Description: Updated load curve charts for the FR full series models.

- **Reduced Mode Speed Switching Logic Optimization:**
  Description: Optimized and resolved the issue where occasionally the speed was far lower than the reduced mode speed when the reduced mode was triggered.

- **Teach Point Optimization:**
  Description: Optimized the prompt information added to teach points.

- **CNC Function Package Function Based on FOCAS:**
  Description: Added CNC function based on FOCAS.

- **Slave Command Adaptation Board:**
  Description: Slave command adaptation for EnTalk miniPCIe board (Profinet protocol, Ethernet/IP protocol, CC-Link IEF Basic protocol), adaptation for CIFX 9OE-RE/F/PNS miniPCIe board (Profinet protocol, Ethernet/IP protocol, Ethercat protocol, CC-Link IEF Basic protocol).

- **Checkpoint Timestamp Feedback Function:**
  Description: Servo J motion can receive timestamp results, including command number and issue, enqueue, dequeue, execution timestamps.

- **Laser Sensor Adaptation Controller Peripheral Open Protocol:**
  Description: Added laser peripheral open protocol communication function.

1.4.14 Version V3.7.8

Date: 2025-01-20

- **Extension Axis Plus Laser Data No Transformation Fixed Point Tracking Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming, Initial Setup -> Peripherals -> Line Laser Sensor.

  Description: Mainly for real-time fixed-point tracking of positioners, compensating the workpiece deviation recorded by the laser at the robot tool end when the extension axis is moving.

- **Robot CI Configuration Addition and Optimization Function:**
  Path: Initial Setup -> Basic -> I/O Settings -> DI, Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: 1. Configurable CI input added the function of switching robot manual/auto mode via high/low level. Configure the CI port as "Manual/Auto Switch (High/Low Level)". When the input signal of this port is valid, the robot automatically switches to auto mode; when invalid, it switches to manual mode. 2. After opening a LUA program in WebApp, starting the LUA program via configurable CI automatically runs the currently opened LUA program, not the last saved program.

- **WebApp Status Query Function Optimization:**
  Path: Status Information -> Status Query.

  Description: Changed the front-end and back-end data interaction method from front-end polling request to WebSocket active data sending, reducing CPU usage.

- **Control Box Firmware DO Power-on High/Low Level Configurable Function:**
  Path: Initial Setup -> Basic -> I/O Settings -> DO.

Description: Added "Control Box DO Output During Power-on" configuration. Before the robot is enabled, the control box DO can be configured to the required high/low level state according to specific usage scenarios.

- **Real-time Look-ahead Trajectory Planning Method Function:**
  Path: Teach Program -> Program Programming, Graphical Programming, Node Diagram Programming.

  Description: Added "TrajectoryLA" instruction.

- **Strategy Function After Collision Based on Force Sensor:**
  Path: Initial Setup -> Joint -> Collision Level.

  Description: Added "Safe Speed" parameter to the collision rebound mode.

- **Automatic Lifting Function After Force Sensor Collision:**
  Description: Implemented automatic lifting function of the robot after collision detection based on the force sensor, with the ability to limit the speed during the lifting process.

- **FR Full Series Model Load Curve Drawing:**
  Description: Updated load curve charts for the FR full series models.

1.4.15 Version V3.7.7

Date: 2024-12-30

- **Gripper Status Data Real-time Monitoring Function:**
  Path: Initial Setup -> Peripherals -> Gripper -> Status Monitoring.

  Description: Displays real-time gripper status information such as running speed, torque, and position.

- **Controller Fault Data Collection Function:**
  Path: System Settings -> General Settings -> Fault Data.

  Description: When collisions, command point errors, or other faults occur during robot operation, the controller automatically records the robot's status information (position, speed, etc.) for 15 seconds before and after the fault. WebApp can export fault information in .csv format to help troubleshoot and analyze the cause of the fault.

- **Singularity Avoidance and Traversal Function Based on Force Sensor Assisted Dragging:**
  Path: Auxiliary Applications -> Tool Applications -> Drag Lock -> Force Sensor Assisted Lock.

  Description: 1. In the force sensor assisted dragging interface, when the strategy is set to avoidance, a virtual force is generated when dragging the robot close to a singularity. 2. When the strategy is set to traversal, it switches to drag mode when approaching a singularity, and switches back to force sensor assisted dragging when moving away from the singularity.

- **Old Dynamics Adaptation for 360° Free Installation and Backup Package Import Adaptation Dynamics Function:**
  Path: Initial Setup -> Basic -> Installation, Auxiliary Applications -> Tool Applications -> Data Backup.

  Description: Added verification of robot type, installation method, installation angle, and dynamics configuration type during backup package import. If these parameters are inconsistent, import is prohibited with a prompt.

- **Singularity Traversal Function in Auto Mode:**
  Path: Teach Program -> Peripherals -> LIN/ARC Command.

  Description: Added "Singularity Traversal" motion protection configuration.

- **Function to Automatically Update Corresponding Point LUA Program After Recording Point with End Button:**
  Path: Teach Program -> Program Programming -> Teach Point.

Description: Added function to automatically update the corresponding point LUA program after recording a point with the end button.

- **Status Query Interface Optimization:**
  Path: Status Information -> Status Query.

  Description: Optimized the UI and interaction of the status query interface.

- **WebApp, Teach Pendant Interface Optimization:**
  Description: WebApp and teach pendant added Russian and Traditional Chinese display interfaces.

- **WebApp Display Interface Optimization Function:**
  Description: Added software version and robot model display in the lower left corner of the WebApp interface.

- **Extension Axis Plus Laser Tracking Function:**
  Description: 1. Robot and external axis move synchronously, the laser achieves synchronous tracking in the external axis coordinate system. 2. Robot and external axis move asynchronously, the laser can track in the robot base coordinate system or the external axis coordinate system.

- **Tool Point Dragging Function Based on Force Sensor:**
  Description: In the web interface, set the FT reference coordinate system to a custom coordinate system. After enabling force sensor assisted dragging, the robot moves along the set tool coordinate system.

- **Robot CNDE Function:**
  Description: Clients can use CNDE to obtain robot status feedback and send control data to the robot (UDP communication). The status feedback cycle is configurable (1-200ms). The content of robot status feedback data and client input control data can be freely configured.

1.4.16 Version V3.7.6

Date: 2024-11-18

- **Initial Setup Page Layout Optimization:**
  Path: Initial Setup.

  Description: Optimized the tool setting interface, load configuration interface, and display of some function icons.

- **WebApp Peripheral Extension Axis Configuration Function Interaction Optimization:**
  Path: Initial Setup -> Peripherals -> Extension Axis.

  Description: Optimized the layout and interaction of the extension axis interface.

- **System Log Function Optimization:**
  Path: Status Information -> System Log.

  Description: Added log pagination display and detailed log operation category distinction.

- **XY Direction Horizontal Constant Force Grinding Function:**
  Path: Teach Program -> Program Programming -> Force Control Set -> F/T_Control Command.

  Description: Added “Grinding Disc Radius” parameter, allowing repeated linear/curved motion along the workpiece surface.

- **Laser Point Acquisition Motion Function:**
  Path: Auxiliary Applications -> Tool Applications -> Intersection Point Generation.

  Description: Added the use of LUA script instructions for the three-point and four-point search to find intersection point coordinate function.

- **External Axis Configuration - Two-DOF Cart Function:**
  Path: Initial Setup -> Peripherals -> Extension Axis.

  Description: Added "Two-DOF Cart" extension axis solution. The robot communicates with the PLC via UDP, and then the PLC controls the two-DOF cart via EtherCat.

- **TCP Calibration Method Function Based on Flat Plate Tool:**
  Path: Initial Setup -> Coordinate System -> Tool Coordinate.

  Description: Added "Flat Plate Tool Calibration" coordinate system calibration method.

- **Robot Background Program Function:**
  Path: Teach Program -> Program Programming.

  Description: Background programs can normally obtain I/O interface data, including system I/O, modbus, and extended I/O.

- **Robot Trajectory Automatic Singularity Avoidance Function:**
  Path: Teach Program -> Program Programming -> Linear Lin/Arc Arc Command.

  Description: Added "Singularity Avoidance" motion protection configuration.

- **Rotary Gripper End Adaptation Function:**
  Path: Initial Setup -> Peripherals -> Gripper.

  Description: Added related function code configuration for rotary grippers.

- **Custom Protocol Slave Command:**
  Path: Remote Mode.

  Description: Added "Controller Slave Protocol" configuration.

- **UDP Communication Wire Feeding Search Position Function**:

  Description: The robot can control wire feeding search start/stop and obtain the wire feeding search success signal via UDP extended IO.

- **Backup Package Function Optimization**:

  Description: Supports import and use of old version data packages (QX 3.6.1 and later version data packages).

- **Factory Reset Function Optimization**: Added file verification, increased system factory reset stability.

  Description: Added file verification, increased system factory reset stability.

- **Upgrade Function Optimization**:

  Description: QX3.6.9 and later versions can be directly upgraded to QX3.7.6, retaining current version user data after upgrade.

- **Page Downgrade Function**:

  Description: WebApp page supports downgrade, can be downgraded to any version after QX3.6.9, retaining current version user data after downgrade.

1.4.17 Version V3.7.5

Date: 2024-09-30

- **Wrap Angle Pose Transition Angular Speed Adjustable Function:**
  Path: Teach Program -> Program Programming -> Linear Lin Command.

  Description: Added "Transition Point Angular Speed Adjustable" motion protection configuration.
- **Arc Tracking Multi-layer Multi-pass Welding Function:**
  Path: Auxiliary Applications -> Process Package -> Welding Expert Library -> Multi-layer Multi-pass Welding.

  Description: Added "First Layer Welding Oscillation Function" and "Arc Tracking Function" configurations.
- **485 Extension Axis Configuration Function:**
  Path: Initial Setup -> Peripherals -> Extension Axis.

  Description: Added "Acceleration and Emergency Stop" configuration.
- **Robot Tool TCP Automatic Calibration Function (Self-made Fiber Optic Sensor Fixture):**
  Path: Initial Setup -> Basic -> Tool Coordinate.

  Description: Added "Photoelectric Automatic Calibration" coordinate system calibration.
- **Teach Pendant Multi-language Setting Function:**
  Path: Login Page.

  Description: Added "Language Switch" configuration.

1.4.18 Version V3.7.4

Date: 2024-08-09

- **Software Function Based on Lua End Open Protocol Adaptation (Gripper Part):**
  Path: Initial Setup -> Peripherals -> Gripper.

  Description: Added "End Protocol Enable" configuration.
- **Oblique Zigzag Oscillation Function:**
  Path: Teach Program -> Program Programming -> Oscillation Weave Command.

  Description: Added "Oscillation Direction Azimuth Angle" parameter configuration.
- **Robot Welding Process Package Optimization Function:**
  Path: Initial Setup -> Peripherals -> Welding Machine -> Welding Machine Configuration.

  Description: Added "Welding Process Parameters" configuration.
- **Lin Command Joint Overspeed Handling Function:**
  Path: Teach Program -> Program Programming -> Linear Lin Command.

  Description: Added "Joint Overspeed Protection" motion protection configuration.

1.4.19 Version V3.7.3

Date: 2024-06-28

- **Modbus Slave Control Robot Function:**
  Path: Teach Program -> Program Programming -> ModbusTCP.

  Description: Added "Slave Controller" configuration.
- **Emergency Stop Type Function:**
  Path: Initial Setup -> Safety -> Emergency Stop.

  Description: Added "Category 1a, Category 2a, Category 2" stop types to the stop type.

1.4.20 Version V3.7.2

Date: 2024-06-07

- **Configuration Function Based on Lua End Open Protocol:**
  Path: Initial Setup -> Peripherals -> End Tool -> Open Protocol.

  Description: Added "Open Protocol" configuration.
- **Motion AO Control Command Function:**
  Path: Teach Program -> Program Programming.

  Description: Added "Motion AO Command".
- **Six-Axis Force and Joint Impedance Hybrid Dragging Function:**
  Path: Auxiliary Applications -> Tool Applications -> Drag Lock.

  Description: Added "Six-Axis Force and Joint Impedance Hybrid Dragging".
- **Post-Collision Response Strategy Function:**
  Path: Initial Setup -> Basic -> Joint -> Collision Level.

  Description: Added "Collision Strategy" configuration.
- **Robot First Activation Function:**
  Path: Login Settings.

  Description: Added robot first activation verification function.

1.4.21 Version V3.7.1

Date: 2024-05-10

- **Web Interface Lock Function:**
  Path: System Settings -> Custom Information.

  Description: Added "Web Interface Lock Screen" configuration.
- **Search Position Three-Point and Four-Point Intersection Coordinate Function:**
  Path: Auxiliary Applications -> Tool Applications -> Intersection Point Generation.

  Description: Added "Search Position Three-Point and Four-Point Intersection Coordinate Function".
- **Segment Welding Motion Posture Optimization Function:**
  Path: Teach Program -> Program Programming -> Segment Welding Segment Command.

  Description: Added "Segment Welding Mode" configuration.

- **Virtual Wall Function Based on Force Sensor:**
  Path: Initial Setup -> Peripherals -> Force Sensor, Auxiliary Applications -> Tool Applications -> Drag Lock.

  Description: Added "Force Sensor" related configuration parameters to the end peripheral configuration. Added "Inertia Coefficient" configuration to force sensor assisted lock.

- **SmartTool + Force Sensor Combination New Functions:**
  Path: Initial Setup -> Peripherals -> Welding Handle.

  Description: Added key function configuration.

CHAPTER

TWO

SDK MANUAL

2.1 C++

This manual is the secondary development interface document of C++.

Important: Robot parameter unit description: The robot position unit is millimeter (mm), and the attitude unit is degree (°).

Important:

1) In code examples that are not specifically stated, the robot has been powered on and enabled by default;
2) All code examples in the documentation default to no interference within the robot's workspace;
3) Please use the data of the on-site robot in the actual use test.

Note: The current document applies to SDK-v2.0.0 version and is backward compatible with v1.x version.

2.1.1 Version Update Description

Version	Date	Update Description
V3.9.3	2026-02-11	1.ServoCart() interface added extended axis parameters 2.SetOutputResetCtlBoxDO() interface added parameter for whether to reload DO state before reset after pause resume 3.SetOutputResetCtlBoxAO() interface added parameter for whether to reload DO state before reset after pause resume 4.SetOutputResetAxleDO() interface added parameter for whether to reload DO state before reset after pause resume 5.SetOutputResetAxleAO() interface added parameter for whether to reload DO state before reset after pause resume 6.SetOutputResetExtDO() interface added parameter for whether to reload DO state before reset after pause resume 7.SetOutputResetExtAO() interface added parameter for whether to reload DO state before reset after pause resume 8.SetOutputResetSmartToolDO() interface added parameter for whether to reload DO state before reset after pause resume 9.Added GetInverseKinExaxis() inverse kinematics solution interface including extended axis position
V3.9.2	2026-01-26	1. Added a processing strategy parameter for undetected force/torque to the FT_RotInsertion() interface 2. Added robot fixed-point tracking related parameters to the LaserSensorRecordandReplay() interface 3. Added the MoveStationary() interface 4. Added the TCPComputeRPY() interface 5. Added the TCPComputeXYZ() interface 6. Added the TCPRecordFlangePosStart() interface 7. Added the TCPRecordFlangePosEnd() interface 8. Added the TCPGetRecordFlangePos() interface 9. Added the PhotoelectricSensorTCPCalibration() interface
V3.9.1	2025-12-25	1. Added oacc speed scaling factor parameter / physical acceleration parameter to the MoveL() interface; 2. Added oacc speed scaling factor parameter / physical acceleration parameter to the MoveC() interface; 3. Optimized parameter descriptions for physical speed and physical acceleration in the Circle() interface; 4. Added FT_Control() overloaded function with rx, ry activation thresholds and torque adjustment coefficient parameters; 5. Added SerCoderCompenParams() interface;
V3.9.0	2025-11-26	1. JointSensitivityCalibration() interface adds j1~j6 joint linearity return 2. Added JointHysteresisError() interface 3. Added JointRepeatability() interface 4. Added SetAdmittanceParams() interface 5. Added MoveToIntersectLineStart() interface

2.1.2 Data structure specification

2.1.2.1 Interface call return value type

```
typedef  int errno_t;
```

2.1.2.2 Joint position data type

```
/**
* @brief Joint position data type
*/
typedef  struct
{
    double jPos[6];   /* Six joint positions, unit: deg */
}JointPos;
```

2.1.2.3 Cartesian spatial location data type

```
/**
* @brief Cartesian spatial location data type
*/
typedef struct
{
    double x;    /* X-axis coordinate, unit: mm  */
    double y;    /* Y-axis coordinate, unit: mm  */
    double z;    /* Z-axis coordinate, unit: mm  */
} DescTran;
```

2.1.2.4 Euler Angle attitude data type

```
/**
* @brief Euler Angle attitude data type
*/
typedef struct
{
    double rx;   /* Rotation Angle about fixed axis X, unit: deg  */
    double ry;   /* Rotation Angle about fixed axis y, unit: deg  */
    double rz;   /* Rotation Angle about fixed axis Z, unit: deg  */
} Rpy;
```

2.1.2.5 Cartesian space pose data type

```
/**
*@brief Cartesian space pose type
*/
typedef struct
{
    DescTran tran;      /* Cartesian position  */
    Rpy rpy;            /* Cartesian space attitude  */
} DescPose;
```

2.1.2.6 Extension axis position data type

```
/**
* @brief Extension axis position data type
*/
typedef  struct
{
    double ePos[4];   /* Position of four expansion shafts, unit: mm */
}ExaxisPos;
```

2.1.2.7 Torque sensor data type

```
/**
* @brief The force component and torque component of the force sensor
*/
typedef struct
{
    double fx;  /* Component of force along the x axis, unit: N  */
    double fy;  /* Component of force along the y axis, unit: N  */
    double fz;  /* Component of force along the z axis, unit: N  */
    double tx;  /* Component of torque about the X-axis, unit: Nm */
    double ty;  /* Component of torque about the Y-axis, unit: Nm */
    double tz;  /* Component of torque about the Z-axis, unit: Nm */
} ForceTorque;
```

2.1.2.8 Spiral parameter data type

```
/**
* @brief  Spiral parameter data type
*/
typedef  struct
{
    int    circle_num;          /* Coil number  */
    float  circle_angle;        /* Spiral Angle  */
    float  rad_init;            /* Initial radius of spiral, unit: mm  */
    float  rad_add;             /* Radius increment  */
    float  rotaxis_add;         /* Increment in the direction of the axis of rotation ␣
```

(continues on next page)

(continued from previous page)

```
↪*/
    int rot_direction;  /* Rotation direction, 0- clockwise, 1- counterclockwise  */
    int velAccMode;      /* Velocity acceleration parameter mode: 0-constant angular␣
↪velocity; 1-constant linear velocity */
}SpiralParam;
```

2.1.2.9 feedback packet of robot controller state

Changed in version C++: SDK-v2.1.4.0

```
/**
 * @brief  feedback packet of robot controller state
 */
typedef struct _ROBOT_STATE_PKG
{
    uint16_t frame_head;      /* Frame header, fixed as 0x5A5A */
    uint8_t  frame_cnt;       /* Frame counter, cycles 0-255 */
    uint16_t data_len;        /* Length of data content */
    uint8_t  program_state;   /* Program running state: 1-Stop; 2-Run; 3-Pause; */
    uint8_t  robot_state;     /* Robot motion state: 1-Stop; 2-Run; 3-Pause; 4-Drag */
    int      main_code;       /* Main error code */
    int      sub_code;        /* Sub error code */
    uint8_t  robot_mode;      /* Robot mode: 1-Manual mode; 0-Auto mode; */
    double   jt_cur_pos[6];   /* Current joint positions of 6 axes, unit: deg */
    double   tl_cur_pos[6];   /* Current tool position
                                 tl_cur_pos[0]: Position along x-axis, unit: mm,
                                 tl_cur_pos[1]: Position along y-axis, unit: mm,
                                 tl_cur_pos[2]: Position along z-axis, unit: mm,
                                 tl_cur_pos[3]: Rotation angle around fixed X-axis,␣
↪unit: deg
                                 tl_cur_pos[4]: Rotation angle around fixed y-axis,␣
↪unit: deg
                                 tl_cur_pos[5]: Rotation angle around fixed z-axis,␣
↪unit: deg */
    double   flange_cur_pos[6]; /* Current flange position
                                 flange_cur_pos[0]: Position along x-axis, unit: mm,
                                 flange_cur_pos[1]: Position along y-axis, unit: mm,
                                 flange_cur_pos[2]: Position along z-axis, unit: mm,
                                 flange_cur_pos[3]: Rotation angle around fixed X-axis,␣
↪unit: deg
                                 flange_cur_pos[4]: Rotation angle around fixed y-axis,␣
↪unit: deg
                                 flange_cur_pos[5]: Rotation angle around fixed z-axis,␣
↪unit: deg */
    double   actual_qd[6];      /* Current joint velocities of 6 axes, unit: deg/s */
    double   actual_qdd[6];     /* Current joint accelerations of 6 axes, unit: deg/s^2␣
↪*/
    double   target_TCP_CmpSpeed[2]; /* target_TCP_CmpSpeed[0]: TCP composite command␣
↪speed (position), unit: mm/s
                                 target_TCP_CmpSpeed[1]: TCP composite command speed␣
↪(orientation), unit: deg/s */
```

(continues on next page)

(continued from previous page)

```
    double   target_TCP_Speed[6];   /* TCP command speed
                                    target_TCP_Speed[0]: Velocity along x-axis, unit: mm/s,
                                    target_TCP_Speed[1]: Velocity along y-axis, unit: mm/s,
                                    target_TCP_Speed[2]: Velocity along z-axis, unit: mm/s,
                                    target_TCP_Speed[3]: Angular velocity around fixed X-
↪axis, unit: deg/s
                                    target_TCP_Speed[4]: Angular velocity around fixed y-
↪axis, unit: deg/s
                                    target_TCP_Speed[5]: Angular velocity around fixed z-
↪axis, unit: deg/s */
    double   actual_TCP_CmpSpeed[2];/* actual_TCP_CmpSpeed[0]: TCP composite actual␣
↪speed (position), unit: mm/s
                                    actual_TCP_CmpSpeed[1]: TCP composite actual speed␣
↪(orientation), unit: deg/s */
    double   actual_TCP_Speed[6];   /* TCP actual speed
                                    actual_TCP_Speed[0]: Velocity along x-axis, unit: mm/s,
                                    actual_TCP_Speed[1]: Velocity along y-axis, unit: mm/s,
                                    actual_TCP_Speed[2]: Velocity along z-axis, unit: mm/s,
                                    actual_TCP_Speed[3]: Angular velocity around fixed X-
↪axis, unit: deg/s
                                    actual_TCP_Speed[4]: Angular velocity around fixed y-
↪axis, unit: deg/s
                                    actual_TCP_Speed[5]: Angular velocity around fixed z-
↪axis, unit: deg/s */
    double   jt_cur_tor[6];      /* Current torque of 6 axes, unit: N·m */
    int      tool;               /* Applied tool coordinate system number */
    int      user;               /* Applied workpiece coordinate system number */
    uint8_t  cl_dgt_output_h;    /* Control box digital IO output 15-8 */
    uint8_t  cl_dgt_output_l;    /* Control box digital IO output 7-0 */
    uint8_t  tl_dgt_output_l;    /* Tool digital IO output 7-0, only bit0-bit1 valid */
    uint8_t  cl_dgt_input_h;     /* Control box digital IO input 15-8 */
    uint8_t  cl_dgt_input_l;     /* Control box digital IO input 7-0 */
    uint8_t  tl_dgt_input_l;     /* Tool digital IO input 7-0, only bit0-bit1 valid */
    uint16_t cl_analog_input[2]; /* cl_analog_input[0]: Control box analog input 0
                                    cl_analog_input[1]: Control box analog input 1 */
    uint16_t tl_anglog_input;    /* Tool analog input */
    double   ft_sensor_raw_data[6]; /* Force-torque sensor raw data
                                    ft_sensor_raw_data[0]: Force along x-axis, unit: N
                                    ft_sensor_raw_data[1]: Force along y-axis, unit: N
                                    ft_sensor_raw_data[2]: Force along z-axis, unit: N
                                    ft_sensor_raw_data[3]: Torque around x-axis, unit: Nm
                                    ft_sensor_raw_data[4]: Torque around y-axis, unit: Nm
                                    ft_sensor_raw_data[5]: Torque around z-axis, unit: Nm */
    double   ft_sensor_data[6];     /* Force-torque sensor data
                                    ft_sensor_data[0]: Force along x-axis, unit: N
                                    ft_sensor_data[1]: Force along y-axis, unit: N
                                    ft_sensor_data[2]: Force along z-axis, unit: N
                                    ft_sensor_data[3]: Torque around x-axis, unit: Nm
                                    ft_sensor_data[4]: Torque around y-axis, unit: Nm
                                    ft_sensor_data[5]: Torque around z-axis, unit: Nm */
    uint8_t  ft_sensor_active;   /* Force-torque sensor active state: 0-Reset, 1-Active␣
↪*/
```

(continues on next page)

(continued from previous page)

```
    uint8_t  EmergencyStop;      /* Emergency stop flag: 0-Not pressed, 1-Pressed */
    int      motion_done;        /* Motion completion signal: 1-Completed, 0-Not completed */
    uint8_t  gripper_motiondone; /* Gripper motion completion signal: 1-Completed, 0-Not completed */
    int      mc_queue_len;       /* Motion command queue length */
    uint8_t  collisionState;     /* Collision detection: 1-Collision, 0-No collision */
    int      trajectory_pnum;    /* Trajectory point number */
    uint8_t  safety_stop0_state; /* Safety stop signal SI0 */
    uint8_t  safety_stop1_state; /* Safety stop signal SI1 */
    uint8_t  gripper_fault_id;   /* Faulty gripper number */
    uint16_t gripper_fault;      /* Gripper fault */
    uint16_t gripper_active;     /* Gripper active state */
    uint8_t  gripper_position;   /* Gripper position */
    int8_t   gripper_speed;      /* Gripper speed */
    int8_t   gripper_current;    /* Gripper current */
    int      gripper_temp;       /* Gripper temperature */
    int      gripper_voltage;    /* Gripper voltage */
    robot_aux_state aux_state;
    EXT_AXIS_STATUS extAxisStatus[4];  /* UDP extended axis status */
    uint16_t extDIState[8];        //Extended DI input
    uint16_t extDOState[8];        //Extended DO output
    uint16_t extAIState[4];        //Extended AI input
    uint16_t extAOState[4];        //Extended AO output
    int rbtEnableState;            //Robot enable state
    double   jointDriverTorque[6];        //Robot joint driver torque
    double   jointDriverTemperature[6];   //Robot joint driver temperature
    RobotTime robotTime;           //Robot system time
    int softwareUpgradeState;      //Robot software upgrade state
    uint16_t endLuaErrCode;        //End LUA running state
    uint16_t cl_analog_output[2];  //Control box analog output
    uint16_t tl_analog_output;     //Tool analog output
    float gripperRotNum;           //Rotary gripper current rotation count
    uint8_t gripperRotSpeed;       //Rotary gripper current rotation speed percentage
    uint8_t gripperRotTorque;      //Rotary gripper current rotation torque percentage
    WELDING_BREAKOFF_STATE weldingBreakOffState;  //Welding breakoff state
    double jt_tgt_tor[6];          //Joint command torque
    int smartToolState;            //SmartTool handle button state
    float wideVoltageCtrlBoxTemp;        //Temperature of wide voltage control box
    uint16_t wideVoltageCtrlBoxFanCurrent;   //Wide voltage control box fan current (Ma)
    double toolCoord[6];      //Tool coordinate system
    double wobjCoord[6];      //Workpiece coordinate system
    double extoolCoord[6];     //External tool coordinate system
    double exAxisCoord[6];     //Extended axis coordinate system
    double load;          //Load mass
    double loadCog[3];       //Load center of gravity
    int servoJCmdNum;           //SERVOJ instruction count
    uint16_t check_sum;      /* And verification */
    uint16_t check_sum;      /* Checksum */
}ROBOT_STATE_PKG;
```

2.1.3 Basics

2.1.3.1 Instantiate Robot

```
/**
* @brief  Robot interface class constructor
*/
FRRobot();
```

2.1.3.2 Establish Communication with Controller

```
/**
* @brief  Establish communication with robot controller
* @param  [in] ip  Controller IP address, default is 192.168.58.2
* @return Error code
*/
errno_t  RPC(const char *ip);
```

2.1.3.3 Close Communication with Controller

```
/**
 * @brief  Close communication with robot controller
 * @return Error code
 */
errno_t  CloseRPC();
```

2.1.3.4 Query SDK Version

```
/**
* @brief  Query SDK version
* @param  [out] version   SDK version
* @return  Error code
*/
errno_t  GetSDKVersion(char *version);
```

2.1.3.5 Get Controller IP

```
/**
* @brief  Get controller IP
* @param  [out] ip  Controller IP
* @return  Error code
*/
errno_t  GetControllerIP(char *ip);
```

2.1.3.6 Control Robot to Enter/Exit Drag Teaching Mode

```
/**
* @brief  Control robot to enter/exit drag teaching mode
* @param  [in] state 0-Exit drag teaching mode, 1-Enter drag teaching mode
* @return  Error code
*/
errno_t  DragTeachSwitch(uint8_t state);
```

2.1.3.7 Check if Robot is in Drag Mode

```
/**
* @brief  Check if robot is in drag teaching mode
* @param  [out] state 0-Not in drag teaching mode, 1-In drag teaching mode
* @return  Error code
*/
errno_t  IsInDragTeach(uint8_t *state);
```

2.1.3.8 Enable/Disable Robot

```
/**
* @brief  Enable/disable robot (enabled by default after power on)
* @param  [in] state  0-Disable, 1-Enable
* @return  Error code
*/
errno_t  RobotEnable(uint8_t state);
```

2.1.3.9 Switch Between Manual/Auto Mode

```
/**
* @brief Switch between manual/auto mode
* @param [in] mode 0-Auto mode, 1-Manual mode
* @return Error code
*/
errno_t  Mode(int mode);
```

2.1.3.10 Shut Down Robot OS

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Shut down robot operating system
* @return Error code
*/
errno_t ShutDownRobotOS();
```

2.1.3.11 Set Reconnection Parameters

New in version C++SDK-v2.1.5.0.

```cpp
/**
* @brief Set communication reconnection parameters
* @param [in] enable Enable reconnection when network fails true-enable false-disable
* @param [in] reconnectTime Reconnection time (ms)
* @param [in] period Reconnection interval (ms)
* @return Error code
*/
errno_t SetReConnectParam(bool enable, int reconnectTime = 30000, int period = 50);
```

2.1.3.12 Shut down the robot operating system

```cpp
/**
* @brief Shut down the robot operating system
* @return Error code
*/
int ShutDownRobotOS();
```

2.1.3.13 Initialize Log Parameters

New in version C++SDK-v2.1.2.0.

```cpp
/**
* @brief Initialize log parameters;
* @param output_modelOutput mode, 0-Direct output; 1-Buffered output; 2-Asynchronous output;
* @param file_path: File save path+name (max 256 chars), must be in xxx.log format, e.g. /home/fr/linux/fairino.log;
* @param file_numNumber of rolling files, 1~20. Single file size limit 50MB;
* @return errno_t Error code;
*/
errno_t LoggerInit(int output_model = 0, std::string file_path = "", int file_num = 5);
```

2.1.3.14 Set Log Filter Level

New in version C++SDK-v2.1.2.0.

```cpp
/**
* @brief Set log filter level;
* @param lvl: Filter level, smaller value means less logs, default is 1. 1-error, 2-warning, 3-info, 4-debug;
*/
void SetLoggerLevel(int lvl = 1);
```

2.1.3.15 Basic Robot Control Example

```
int TestRobotCtrl(void)
{
        ROBOT_STATE_PKG pkg = {};
        FRRobot robot;
        robot.LoggerInit();
        robot.SetLoggerLevel(1);
        int rtn = robot.RPC("192.168.58.2");
        robot.SetReConnectParam(true, 30000, 500);
        char ip[64] = "";
        char version[64] = "";
        uint8_t state;
        robot.GetSDKVersion(version);
        printf("SDK version:%s\n", version);
        robot.GetControllerIP(ip);
        printf("controller ip:%s\n", ip);
        robot.Mode(1);
        robot.Sleep(1000);
        robot.DragTeachSwitch(1);
        robot.Sleep(1000);
        robot.IsInDragTeach(&state);
        printf("drag state :%u\n", state);
        robot.Sleep(3000);
        robot.DragTeachSwitch(0);
        robot.Sleep(1000);
        robot.IsInDragTeach(&state);
        printf("drag state :%u\n", state);
        robot.Sleep(3000);
        robot.RobotEnable(0);
        robot.Sleep(3000);
        robot.RobotEnable(1);
        robot.Mode(0);
        robot.Sleep(1000);
        robot.Mode(1);
        robot.Sleep(3000);
        robot.ShutDownRobotOS();
        robot.CloseRPC();
        return 0;
}
```

2.1.3.16 Get Robot Software Version Example

```
/**
* @brief Get robot software version
* @param[out]    robotModel Robot model
* @param[out]    webversion Web version
* @param[out]    controllerVersion Controller version
* @return Error code
*/
errno_t GetSoftwareVersion(char robotModel[64], char webVersion[64], char␣
↪controllerVersion[64]);
```

2.1.3.17 Get Robot Hardware Version

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Get robot hardware version
* @param[out] ctrlBoxBoardversion Control box board hardware version
* @param[out] driver1version Driver 1 hardware version
* @param[out] driver2version Driver 2 hardware version
* @param[out] driver3version Driver 3 hardware version
* @param[out] driver4version Driver 4 hardware version
* @param[out] driver5version Driver 5 hardware version
* @param[out] driver6version Driver 6 hardware version
* @param[out] endBoardversion End board hardware version
*/
errno_t GetHardwareVersion(char ctrlBoxBoardversion[128], char driver1version[128], char␣
↪driver2version[128], char driver3version[128], char driver4version[128], char␣
↪driver5version[128], char driver6version[128], char endBoardversion[128]);
```

2.1.3.18 Get Robot Firmware Version

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Get robot firmware version
* @param[out] ctrlBoxBoardversion Control box board firmware version
* @param[out] driver1version Driver 1 firmware version
* @param[out] driver2version Driver 2 firmware version
* @param[out] driver3version Driver 3 firmware version
* @param[out] driver4version Driver 4 firmware version
* @param[out] driver5version Driver 5 firmware version
* @param[out] driver6version Driver 6 firmware version
* @param[out] endBoardversion End board firmware version
*/
errno_t GetFirmwareVersion(char ctrlBoxBoardversion[128], char driver1version[128], char␣
↪driver2version[128], char driver3version[128], char driver4version[128], char␣
↪driver5version[128], char driver6version[128], char endBoardversion[128]);
```

2.1.3.19 Get Robot Software/Firmware Version Example

```
int TestGetVersions(void)
 {
     ROBOT_STATE_PKG pkg = {};
     FRRobot robot;
     robot.LoggerInit();
     robot.SetLoggerLevel(1);
     int rtn = robot.RPC("192.168.58.2");
     if (rtn != 0)
     {
         return -1;
     }
```

(continues on next page)

(continued from previous page)

```
    robot.SetReConnectParam(true, 30000, 500);
    char robotModel[64] = { 0 };
    char webversion[64] = { 0 };
    char controllerVersion[64] = { 0 };
    char ctrlBoxBoardversion[128] = { 0 };
    char driver1version[128] = { 0 };
    char driver2version[128] = { 0 };
    char driver3version[128] = { 0 };
    char driver4version[128] = { 0 };
    char driver5version[128] = { 0 };
    char driver6version[128] = { 0 };
    char endBoardversion[128] = { 0 };
    rtn = robot.GetSoftwareVersion(robotModel, webversion, controllerVersion);
    printf("Getsoftwareversion rtn is: %d\n", rtn);
    printf("robotmodel is: %s, webversion is: %s, controllerVersion is: %s \n\n",␣
↪robotModel, webversion, controllerVersion);
    rtn = robot.GetHardwareVersion(ctrlBoxBoardversion, driver1version, driver2version,␣
↪driver3version, driver4version, driver5version, driver6version, endBoardversion);
    printf("GetHardwareversion rtn is: %d\n", rtn);
    printf("GetHardwareversion get hardware versoin is: %s, %s, %s, %s, %s, %s, %s, %s\
↪n\n", ctrlBoxBoardversion, driver1version, driver2version, driver3version,␣
↪driver4version, driver5version, driver6version, endBoardversion);
    rtn = robot.GetFirmwareVersion(ctrlBoxBoardversion, driver1version, driver2version,␣
↪driver3version, driver4version, driver5version, driver6version, endBoardversion);
    printf("GetFirmwareversion rtn is: %d\n", rtn);
    printf("GetHardwareversion get hardware versoin is: %s, %s, %s, %s, %s, %s, %s, %s\
↪n\n", ctrlBoxBoardversion, driver1version, driver2version, driver3version,␣
↪driver4version, driver5version, driver6version, endBoardversion);
    robot.CloseRPC();
    return 0;
}
```

2.1.4 Robot Motion

2.1.4.1 JOG Motion

```
/**
* @brief JOG motion
* @param [in] ref 0-Joint JOG, 2-Base coordinate JOG, 4-Tool coordinate JOG, 8-Workpiece␣
↪coordinate JOG
* @param [in] nb 1-Joint1 (or x-axis), 2-Joint2 (or y-axis), 3-Joint3 (or z-axis), 4-
↪Joint4 (or rotation about x-axis), 5-Joint5 (or rotation about y-axis), 6-Joint6 (or␣
↪rotation about z-axis)
* @param [in] dir 0-Negative direction, 1-Positive direction
* @param [in] vel Speed percentage [0~100]
* @param [in] acc Acceleration percentage [0~100]
* @param [in] max_dis Maximum single JOG angle in [°] or distance in [mm]
* @return Error code
*/
errno_t StartJOG(uint8_t ref, uint8_t nb, uint8_t dir, float vel, float acc, float max_
```

(continues on next page)

(continued from previous page)

```
↪dis);
```

2.1.4.2 JOG Deceleration Stop

```
/**
* @brief JOG deceleration stop
* @param [in] ref 1-Joint JOG stop, 3-Base coordinate JOG stop, 5-Tool coordinate JOG␣
↪stop, 9-Workpiece coordinate JOG stop
* @return Error code
*/
errno_t StopJOG(uint8_t ref);
```

2.1.4.3 JOG Immediate Stop

```
/**
* @brief JOG immediate stop
* @return Error code
*/
errno_t ImmStopJOG();
```

2.1.4.4 Robot JOG Control Example

```
int TestJOG(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    for (int i = 0; i < 6; i++)
    {
        robot.StartJOG(0, i + 1, 0, 20.0, 20.0, 30.0);
        robot.Sleep(1000);
        robot.ImmStopJOG();
        robot.Sleep(1000);
    }
    for (int i = 0; i < 6; i++)
    {
        robot.StartJOG(2, i + 1, 0, 20.0, 20.0, 30.0);
        robot.Sleep(1000);
        robot.ImmStopJOG();
        robot.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
    }
    for (int i = 0; i < 6; i++)
    {
        robot.StartJOG(4, i + 1, 0, 20.0, 20.0, 30.0);
        robot.Sleep(1000);
        robot.StopJOG(5);
        robot.Sleep(1000);
    }
    for (int i = 0; i < 6; i++)
    {
        robot.StartJOG(8, i + 1, 0, 20.0, 20.0, 30.0);
        robot.Sleep(1000);
        robot.StopJOG(9);
        robot.Sleep(1000);
    }
    robot.CloseRPC();
    return 0;
}
```

2.1.4.5 Joint Space Motion

```
/**
* @brief Joint space motion
* @param [in] joint_pos Target joint position in deg
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate number [0~14]
* @param [in] user Workpiece coordinate number [0~14]
* @param [in] vel Speed percentage [0~100]
* @param [in] acc Acceleration percentage [0~100] (not yet available)
* @param [in] ovl Speed scaling factor [0~100]
* @param [in] epos Extended axis position in mm
* @param [in] blendT [-1.0]-Move to position (blocking), [0~500.0]-Smoothing time (non-
↪blocking) in ms
* @param [in] offset_flag 0-No offset, 1-Offset in base/workpiece coordinate, 2-Offset␣
↪in tool coordinate
* @param [in] offset_pos Pose offset
* @return Error code
*/
errno_t  MoveJ(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float vel,␣
↪float acc, float ovl, ExaxisPos *epos, float blendT, uint8_t offset_flag, DescPose␣
↪*offset_pos);
```

2.1.4.6 Joint Space Motion (Automatic Forward Kinematics Calculation)

```
/**
* @brief Joint Space Motion (Automatic Forward Kinematics Calculation)
* @param [in] joint_pos Target joint position, unit: deg
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], temporarily not available
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] epos Extended axis position, unit: mm
* @param [in] blendT [-1.0]-Move to position (blocking), [0~500.0]-Smoothing time (non-blocking), unit: ms
* @param [in] offset_flag 0-No offset, 1-Offset in base coordinate system/workpiece coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos Pose offset value
* @return Error code
*/
errno_t MoveJ(JointPos* joint_pos, int tool, int user, float vel, float acc, float ovl, ExaxisPos* epos, float blendT, uint8_t offset_flag, DescPose* offset_pos);
```

2.1.4.7 Cartesian Space Linear Motion

```
/**
* @brief Cartesian Space Linear Motion
* @param [in] joint_pos Target joint positions, unit: deg
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate system index, range [0~14]
* @param [in] user Workpiece/user coordinate system index, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100] (currently unavailable)
* @param [in] ovl Velocity scaling factor [0~100] / physical velocity (mm/s)
* @param [in] blendR [-1.0]-block until motion completes (blocking), [0~1000.0]-blend radius (non-blocking), unit: mm
* @param [in] blendMode Transition method; 0-tangent blend; 1-corner blend
* @param [in] epos Extended axis position, unit: mm
* @param [in] search 0-no wire search, 1-wire search
* @param [in] offset_flag 0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos Pose offset value
* @param [in] oacc Acceleration scaling factor [0-100] / physical acceleration (mm/s^2)
* @param [in] velAccParamMode Velocity/acceleration parameter mode; 0-percentage; 1-physical velocity (mm/s) and acceleration (mm/s^2)
* @param [in] overSpeedStrategy Overspeed handling strategy; 1-standard; 2-stop with error on overspeed; 3-adaptive speed reduction, default 0
* @param [in] speedPercent Allowable speed reduction threshold percentage [0-100], default 10%
* @return Error code
*/
errno_t MoveL(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float vel, float acc, float ovl, float blendR, int blendMode, ExaxisPos *epos, uint8_t search,
```

(continues on next page)

(continued from previous page)

```
↪uint8_t offset_flag, DescPose *offset_pos, float oacc = 100.0, int velAccParamMode = 0,
↪ int overSpeedStrategy = 0, int speedPercent = 10);
```

2.1.4.8 Cartesian Space Linear Motion (Automatic Inverse Kinematics Calculation)

```
/**
* @brief Cartesian Space Linear Motion (Automatic Inverse Kinematics Calculation)
* @param [in] desc_pos  Target Cartesian pose
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], temporarily not available
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Move to position (blocking), [0~1000.0]-Smoothing radius␣
↪(non-blocking), unit: mm
* @param [in] blendMode Transition method; 0-Tangent transition; 1-Corner transition
* @param [in] epos Extended axis position, unit: mm
* @param [in] search 0-No wire search, 1-Wire search
* @param [in] offset_flag 0-No offset, 1-Offset in base coordinate system/workpiece␣
↪coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos Pose offset value
* @param [in] config Inverse kinematics joint space configuration, [-1]-Calculate based␣
↪on current joint position, [0~7]-Solve based on specific joint space configuration
* @param [in] overSpeedStrategy Overspeed handling strategy, 1-Standard; 2-Error stop␣
↪when overspeed; 3-Adaptive speed reduction, default is 0
* @param [in] speedPercent Allowable speed reduction threshold percentage [0-100],␣
↪default 10%
* @return Error code
*/
errno_t MoveL(DescPose* desc_pos, int tool, int user, float vel, float acc, float ovl,␣
↪float blendR, int blendMode, ExaxisPos* epos, uint8_t search, uint8_t offset_flag,␣
↪DescPose* offset_pos, int config = -1, int overSpeedStrategy = 0, int speedPercent =␣
↪10);
```

2.1.4.9 Cartesian Space Circular Motion

```
/**
* @brief  Cartesian Space Circular Motion
* @param  [in] joint_pos_p  Path point joint position, unit: deg
* @param  [in] desc_pos_p   Path point Cartesian pose
* @param  [in] ptool  Tool coordinate number, range [0~14]
* @param  [in] puser  Workpiece coordinate number, range [0~14]
* @param  [in] pvel  Velocity percentage, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], temporarily not available
* @param  [in] epos_p  Extended axis position, unit: mm
* @param  [in] poffset_flag  0-No offset, 1-Offset in base coordinate system/workpiece␣
↪coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos_p  Pose offset value
* @param  [in] joint_pos_t  Target point joint position, unit: deg
```

(continues on next page)

(continued from previous page)

```
* @param  [in] desc_pos_t   Target point Cartesian pose
* @param  [in] ttool  Tool coordinate number, range [0~14]
* @param  [in] tuser  Workpiece coordinate number, range [0~14]
* @param  [in] tvel  Velocity percentage, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], temporarily not available
* @param  [in] epos_t  Extended axis position, unit: mm
* @param  [in] toffset_flag  0-No offset, 1-Offset in base coordinate system/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos_t  Pose offset value
* @param [in] ovl Velocity scaling factor [0~100] / Physical velocity (mm/s)
* @param [in] blendR [-1.0]-Block until motion completes (blocking), [0~1000.0]-Blend radius (non-blocking), unit: mm
* @param [in] oacc Acceleration scaling factor [0-100] / Physical acceleration (mm/s^2)
* @param [in] velAccParamMode Velocity/acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) acceleration (mm/s^2)
* @return Error code
*/
errno_t MoveC(JointPos *joint_pos_p, DescPose *desc_pos_p, int ptool, int puser, float pvel, float pacc, ExaxisPos *epos_p, uint8_t poffset_flag, DescPose *offset_pos_p, JointPos *joint_pos_t, DescPose *desc_pos_t, int ttool, int tuser, float tvel, float tacc, ExaxisPos *epos_t, uint8_t toffset_flag, DescPose *offset_pos_t, float ovl, float blendR, float oacc = 100.0, int velAccParamMode = 0);
```

2.1.4.10 Cartesian Space Circular Motion (Automatic Inverse Kinematics Calculation)

```
/**
* @brief Cartesian Space Circular Motion (Automatic Inverse Kinematics Calculation)
* @param [in] desc_pos_p  Path point Cartesian pose
* @param [in] ptool Tool coordinate number, range [0~14]
* @param [in] puser Workpiece coordinate number, range [0~14]
* @param [in] pvel Velocity percentage, range [0~100]
* @param [in] pacc Acceleration percentage, range [0~100], temporarily not available
* @param [in] epos_p Extended axis position, unit: mm
* @param [in] poffset_flag 0-No offset, 1-Offset in base coordinate system/workpiece coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos_p Pose offset value
* @param [in] desc_pos_t  Target point Cartesian pose
* @param [in] ttool Tool coordinate number, range [0~14]
* @param [in] tuser Workpiece coordinate number, range [0~14]
* @param [in] tvel Velocity percentage, range [0~100]
* @param [in] tacc Acceleration percentage, range [0~100], temporarily not available
* @param [in] epos_t Extended axis position, unit: mm
* @param [in] toffset_flag 0-No offset, 1-Offset in base coordinate system/workpiece coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos_t Pose offset value
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Move to position (blocking), [0~1000.0]-Smoothing radius (non-blocking), unit: mm
* @param [in] config Inverse kinematics joint space configuration, [-1]-Calculate based on current joint position, [0~7]-Solve based on specific joint space configuration
```

(continues on next page)

(continued from previous page)

```
* @return Error code
*/
errno_t MoveC(DescPose* desc_pos_p, int ptool, int puser, float pvel, float pacc,␣
↪ExaxisPos* epos_p, uint8_t poffset_flag, DescPose* offset_pos_p, DescPose* desc_pos_t,␣
↪int ttool, int tuser, float tvel, float tacc, ExaxisPos* epos_t, uint8_t toffset_flag,␣
↪DescPose* offset_pos_t, float ovl, float blendR, int config = -1);
```

2.1.4.11 Cartesian Space Full Circle Motion

```
/**
 *@brief  Cartesian Space Full Circle Motion
 *@param  [in] joint_pos_p  Path point 1 joint position, unit: deg
 *@param  [in] desc_pos_p   Path point 1 Cartesian pose
 *@param  [in] ptool  Tool coordinate number, range [1~15]
 *@param  [in] puser  Workpiece coordinate number, range [1~15]
 *@param  [in] pvel  Velocity percentage, range [0~100]
 *@param  [in] pacc  Acceleration percentage, range [0~100], temporarily not available
 *@param  [in] epos_p  Extended axis position, unit: mm
 *@param  [in] joint_pos_t  Path point 2 joint position, unit: deg
 *@param  [in] desc_pos_t   Path point 2 Cartesian pose
 *@param  [in] ttool  Tool coordinate number, range [1~15]
 *@param  [in] tuser  Workpiece coordinate number, range [1~15]
 *@param  [in] tvel  Velocity percentage, range [0~100]
 *@param  [in] tacc  Acceleration percentage, range [0~100], temporarily not available
 *@param  [in] epos_t  Extended axis position, unit: mm
 *@param  [in] ovl Velocity scaling factor [0~100] / Physical velocity (mm/s)
 *@param  [in] offset_flag 0-No offset, 1-Offset in base/workpiece coordinate system, 2-
↪Offset in tool coordinate system
 *@param  [in] offset_pos Pose offset value
 *@param  [in] oacc Acceleration scaling factor [0-100] / Physical acceleration (mm/s^2)
 *@param  [in] blendR -1: Blocking; 0~1000: Smoothing radius
* @param  [in] velAccParamMode Velocity/acceleration parameter mode; 0-Percentage; 1-
↪Physical velocity (mm/s) acceleration (mm/s^2)
* @return Error code
*/
errno_t Circle(JointPos* joint_pos_p, DescPose* desc_pos_p, int ptool, int puser, float␣
↪pvel, float pacc, ExaxisPos* epos_p, JointPos* joint_pos_t, DescPose* desc_pos_t, int␣
↪ttool, int tuser, float tvel, float tacc, ExaxisPos* epos_t, float ovl, uint8_t offset_
↪flag, DescPose* offset_pos, double oacc = 100.0, double blendR = -1, int␣
↪velAccParamMode = 0);
```

2.1.4.12 Cartesian Space Full Circle Motion (Automatic Inverse Kinematics Calculation)

```
/**
* @brief Cartesian Space Full Circle Motion (Automatic Inverse Kinematics Calculation)
* @param [in] desc_pos_p  Path point 1 Cartesian pose
* @param [in] ptool Tool coordinate number, range [0~14]
* @param [in] puser Workpiece coordinate number, range [0~14]
* @param [in] pvel Velocity percentage, range [0~100]
* @param [in] pacc Acceleration percentage, range [0~100], temporarily not available
* @param [in] epos_p Extended axis position, unit: mm
* @param [in] desc_pos_t  Path point 2 Cartesian pose
* @param [in] ttool Tool coordinate number, range [0~14]
* @param [in] tuser Workpiece coordinate number, range [0~14]
* @param [in] tvel Velocity percentage, range [0~100]
* @param [in] tacc Acceleration percentage, range [0~100], temporarily not available
* @param [in] epos_t Extended axis position, unit: mm
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] offset_flag 0-No offset, 1-Offset in base coordinate system/workpiece␣
↪coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos Pose offset value
* @param [in] oacc Acceleration percentage
* @param [in] blendR -1: Blocking; 0~1000: Smoothing radius
* @param [in] config Inverse kinematics joint space configuration, [-1]-Calculate based␣
↪on current joint position, [0~7]-Solve based on specific joint space configuration
* @return Error code
*/
errno_t Circle(DescPose* desc_pos_p, int ptool, int puser, float pvel, float pacc,␣
↪ExaxisPos* epos_p, DescPose* desc_pos_t, int ttool, int tuser, float tvel, float tacc,␣
↪ExaxisPos* epos_t, float ovl, uint8_t offset_flag, DescPose* offset_pos, double oacc =␣
↪100.0, double blendR = -1, int config = -1);
```

2.1.4.13 Cartesian Space Point-to-Point Motion

```
/**
* @brief  Cartesian Space Point-to-Point Motion
* @param  [in]  desc_pos  Target Cartesian pose or pose increment
* @param  [in] tool  Tool coordinate number, range [0~14]
* @param  [in] user  Workpiece coordinate number, range [0~14]
* @param  [in] vel  Velocity percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], temporarily not available
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendT [-1.0]-Move to position (blocking), [0~500.0]-Smoothing time (non-
↪blocking), unit: ms
* @param  [in] config  Joint space configuration, [-1]-Calculate based on current joint␣
↪position, [0~7]-Calculate based on specific joint space configuration, default is -1
* @return  Error code
*/
errno_t  MoveCart(DescPose *desc_pos, int tool, int user, float vel, float acc, float␣
↪ovl, float blendT, int config);
```

2.1.4.14 Robot Basic Motion Instruction Code Example

```
int TestMove(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;

    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);

    JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3(-29.777, -84.536, 109.275, -114.075, -86.655, 74.257);
    JointPos j4(-31.154, -95.317, 94.276, -88.079, -89.740, 74.256);
    DescPose desc_pos1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose desc_pos3(-487.434, 154.362, 308.576, 176.600, 0.268, -14.061);
    DescPose desc_pos4(-443.165, 147.881, 480.951, 179.511, -0.775, -15.409);
    DescPose offset_pos(0, 0, 0, 0, 0, 0);
    ExaxisPos epos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float oacc = 100.0;
    float blendT = 0.0;
    float blendR = 0.0;
    uint8_t flag = 0;
    uint8_t search = 0;
    int blendMode = 0;
    int velAccMode = 0;
    robot.SetSpeed(20);
    rtn = robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT, flag, &
↪offset_pos);
    printf("movej errcode:%d\n", rtn);
    rtn = robot.MoveL(&j2, &desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, &
↪epos, search, flag, &offset_pos, oacc, velAccMode);
    printf("movel errcode:%d\n", rtn);
    rtn = robot.MoveC(&j3, &desc_pos3, tool, user, vel, acc, &epos, flag, &offset_pos, &
↪j4, &desc_pos4, tool, user, vel, acc, &epos, flag, &offset_pos, ovl, blendR, oacc,␣
↪velAccMode);
    printf("movec errcode:%d\n", rtn);
    rtn = robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT, flag, &
↪offset_pos);
    printf("movej errcode:%d\n", rtn);
    rtn = robot.Circle(&j3, &desc_pos3, tool, user, vel, acc, &epos, &j1, &desc_pos1,␣
```

(continues on next page)

(continued from previous page)

```
↪tool, user, vel, acc, &epos, ovl, flag, &offset_pos, oacc, -1, velAccMode);
    printf("circle errcode:%d\n", rtn);
    rtn = robot.MoveCart(&desc_pos4, tool, user, vel, acc, ovl, blendT, -1);
    printf("MoveCart errcode:%d\n", rtn);
    rtn = robot.MoveJ(&j1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", rtn);
    rtn = robot.MoveL(&desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, &epos,␣
↪search, flag, &offset_pos, -1, velAccMode);
    printf("movel errcode:%d\n", rtn);
    rtn = robot.MoveC(&desc_pos3, tool, user, vel, acc, &epos, flag, &offset_pos, &desc_
↪pos4, tool, user, vel, acc, &epos, flag, &offset_pos, ovl, blendR, -1, velAccMode);
    printf("movec errcode:%d\n", rtn);
    rtn = robot.MoveJ(&j2, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", rtn);
    rtn = robot.Circle(&desc_pos3, tool, user, vel, acc, &epos, &desc_pos1, tool, user,␣
↪vel, acc, &epos, ovl, flag, &offset_pos, oacc, blendR, -1, velAccMode);
    printf("circle errcode:%d\n", rtn);
    robot.CloseRPC();
    return 0;
}
```

2.1.4.15 Cartesian Space Spiral Motion

```
/**
* @brief  Cartesian Space Spiral Motion
* @param  [in] joint_pos  Target joint position, unit: deg
* @param  [in] desc_pos   Target Cartesian pose
* @param  [in] tool  Tool coordinate number, range [0~14]
* @param  [in] user  Workpiece coordinate number, range [0~14]
* @param  [in] vel  Velocity percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], temporarily not available
* @param  [in] epos  Extended axis position, unit: mm
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] offset_flag  0-No offset, 1-Offset in base coordinate system/workpiece␣
↪coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset value
* @param  [in] spiral_param  Spiral parameters
* @return  Error code
*/
errno_t  NewSpiral(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float␣
↪vel, float acc, ExaxisPos *epos, float ovl, uint8_t offset_flag, DescPose *offset_pos,␣
↪SpiralParam spiral_param);
```

2.1.4.16 Cartesian Space Spiral Motion (Automatic Inverse Kinematics Calculation)

```
/**
* @brief Cartesian Space Spiral Motion (Automatic Inverse Kinematics Calculation)
* @param [in] desc_pos  Target Cartesian pose
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], temporarily not available
* @param [in] epos Extended axis position, unit: mm
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] offset_flag 0-No offset, 1-Offset in base coordinate system/workpiece␣
↪coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos Pose offset value
* @param [in] spiral_param Spiral parameters
* @param [in] config Inverse kinematics joint space configuration, [-1]-Calculate based␣
↪on current joint position, [0~7]-Solve based on specific joint space configuration
* @return Error code
*/
errno_t NewSpiral(DescPose* desc_pos, int tool, int user, float vel, float acc,␣
↪ExaxisPos* epos, float ovl, uint8_t offset_flag, DescPose* offset_pos, SpiralParam␣
↪spiral_param, int config = -1);
```

2.1.4.17 Spiral Motion Code Example

```
int TestSpiral(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  JointPos j(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
  DescPose desc_pos(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
  DescPose offset_pos1(50, 0, 0, -30, 0, 0);
  DescPose offset_pos2(50, 0, 0, -5, 0, 0);
  ExaxisPos epos(0, 0, 0, 0);
  SpiralParam sp;
  sp.circle_num = 5;
  sp.circle_angle = 5.0;
  sp.rad_init = 50.0;
  sp.rad_add = 10.0;
  sp.rotaxis_add = 10.0;
  sp.rot_direction = 0;
  int tool = 0;
  int user = 0;
```

(continues on next page)

(continued from previous page)

```
  float vel = 100.0;
  float acc = 100.0;
  float ovl = 100.0;
  float blendT = 0.0;
  uint8_t flag = 2;
  robot.SetSpeed(20);
  rtn = robot.MoveJ(&j, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos1);
  printf("movej errcode:%d\n", rtn);
  rtn = robot.NewSpiral(&desc_pos, tool, user, vel, acc, &epos, ovl, flag, &offset_pos2,␣
↪sp);
  printf("newspiral errcode:%d\n", rtn);
  robot.CloseRPC();
  return 0;
}
```

2.1.4.18 Servo Motion Start

```
/**
* @brief Servo motion start, used with ServoJ and ServoCart commands
* @return Error code
*/
errno_t ServoMoveStart();
```

2.1.4.19 Servo Motion End

```
/**
* @brief Servo motion end, used with ServoJ and ServoCart commands
* @return Error code
*/
errno_t ServoMoveEnd();
```

2.1.4.20 Joint Space Servo Mode Motion

```
/**
* @brief Joint space servo mode motion
* @param [in] joint_pos Target joint position in deg
* @param [in] acc Acceleration percentage [0~100] (not yet available, default 0)
* @param [in] vel Speed percentage [0~100] (not yet available, default 0)
* @param [in] cmdT Command cycle in seconds, recommended range [0.001~0.0016]
* @param [in] filterT Filter time in seconds (not yet available, default 0)
* @param [in] gain Target position proportional amplifier (not yet available, default 0)
* @param  [in] gain  The target position of the proportional amplifier, temporarily␣
↪closed, the default is 0
* @param  [in] id servoJInstruction ID, which defaults to 0
* @return  error code
*/
errno_t ServoJ(JointPos *joint_pos, ExaxisPos* axisPos, float acc, float vel, float cmdT,
↪ float filterT, float gain, int id = 0);
```

2.1.4.21 Joint Space Servo Mode Motion Example

```
int TestServoJ(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos j(0, 0, 0, 0, 0, 0);
    ExaxisPos epos(0, 0, 0, 0);
    float vel = 0.0;
    float acc = 0.0;
    float cmdT = 0.008;
    float filterT = 0.0;
    float gain = 0.0;
    uint8_t flag = 0;
    int count = 500;
    float dt = 0.1;
    int ret = robot.GetActualJointPosDegree(flag, &j);
    if (ret == 0)
    {
        robot.ServoMoveStart();
        while (count)
        {
            robot.ServoJ(&j, &epos, acc, vel, cmdT, filterT, gain);
            j.jPos[0] += dt;
            count -= 1;
            robot.WaitMs(cmdT * 1000);
        }
        robot.ServoMoveEnd();
    }
    else
    {
        printf("GetActualJointPosDegree errcode:%d\n", ret);
    }
    robot.CloseRPC();
    return 0;
}
```

2.1.4.22 Joint Torque Control Start

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Joint torque control start
* @return Error code
*/
errno_t ServoJTStart();
```

2.1.4.23 Joint Torque Control

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Joint Torque Control 1
* @param  [in] torque Torque for joints j1~j6, unit: Nm
* @param  [in] interval Command cycle, unit: s, range: [0.001~0.008]
* @param  [in] checkFlag Detection strategy 0-No restrictions; 1-Power limit; 2-Velocity␣
↪limit; 3-Both power and velocity limits
* @param  [in] jPowerLimit Maximum joint power limit (W)
* @param  [in] jVelLimit Maximum joint velocity (°/s)
* @return  Error code
*/
errno_t ServoJT(float torque[], double interval);
```

2.1.4.24 Joint Torque Control End

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Joint torque control end
* @return Error code
*/
errno_t ServoJTEnd();
```

2.1.4.25 Joint Torque Control Example

New in version C++SDK-v2.1.5.0.

```
int TestServoJT(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
```

(continues on next page)

(continued from previous page)

```
    }
    robot.SetReConnectParam(true, 30000, 500);
    robot.DragTeachSwitch(1);
    float torques[] = { 0, 0, 0, 0, 0, 0 };
    robot.GetJointTorques(1, torques);
    int count = 100;
    robot.ServoJTStart();
    int error = 0;
    while (count > 0)
    {
        error = robot.ServoJT(torques, 0.001);
        count = count - 1;
        robot.Sleep(1);
    }
    error = robot.ServoJTEnd();
    robot.DragTeachSwitch(0);
    robot.CloseRPC();
    return 0;
}
```

2.1.4.26 Joint Torque Control Code Example with Overspeed Protection

```
int ServoJTWithSafety(FRRobot* robot)
{
    robot->ResetAllError();
    robot->Sleep(500);
    float torques[] = { 0, 0, 0, 0, 0, 0 };
    robot->GetJointTorques(1, torques);
    robot->ServoJTStart();
    ROBOT_STATE_PKG pkg = {};
    robot->DragTeachSwitch(1);
    int checkFlag = 3;
    //double jPowerLimit[6] = {1, 1, 1, 1, 1, 1};
    double jPowerLimit[6] = { 10.0, 10.0, 10.0, 10.0, 10.0, 10.0 };
    double jVelLimit[6] = { 181, 80, 80, 80, 80, 80 };
    int count = 800000;
    int error = 0;
    while (count > 0)
    {
        torques[2] = torques[2] + 0.01;
        error = robot->ServoJT(torques, 0.008, checkFlag, jPowerLimit, jVelLimit);
        if (error != 0)
        {
            robot->ServoJTEnd();
        }
        printf("ServoJT rtn is %d\n", error);
        count = count - 1;
        robot->Sleep(1);
        robot->GetRobotRealTimeState(&pkg);
        printf("maincode %d, subcode %d\n", pkg.main_code, pkg.sub_code);
```

(continues on next page)

(continued from previous page)

```
    }
    robot->DragTeachSwitch(0);
    error = robot->ServoJTEnd();
    return 0;
}
```

2.1.4.27 Cartesian Space Servo Mode Motion

```
/**
* @brief Cartesian space servo mode motion
* @param [in] mode 0-Absolute motion (base coordinate system), 1-Incremental motion (base coordinate system), 2-Incremental motion (tool coordinate system)
* @param [in] desc_pos Target Cartesian pose or pose increment
* @param [in] exaxis Extended axis position
* @param [in] pos_gain Pose increment proportionality coefficient, only effective in incremental motion, range [0~1]
* @param [in] acc Acceleration percentage, range [0~100], temporarily not available, default 0
* @param [in] vel Velocity percentage, range [0~100], temporarily not available, default 0
* @param [in] cmdT Command transmission period, unit s, recommended range [0.001~0.016]
* @param [in] filterT Filter time, unit s, temporarily not available, default 0
* @param [in] gain Proportional amplifier for target position, temporarily not available, default 0
* @return Error code
*/
errno_t ServoCart(int mode, DescPose *desc_pose, ExaxisPos exaxis, float pos_gain[6], float acc, float vel, float cmdT, float filterT, float gain);
```

2.1.4.28 Cartesian Space Servo Mode Motion Code Example

```
int TestServoCart(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    DescPose desc_pos_dt = { 83.00800, 50.525000 , 29.246 , 179.629 , -7.138 , -166.975 };
    ExaxisPos exaxis = { 100.0, 0.0, 0.0, 0.0 };
    float pos_gain[6] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
    int mode = 0;
    float vel = 0.0;
```

(continues on next page)

(continued from previous page)

```
    float acc = 0.0;
    float cmdT = 0.001;
    float filterT = 0.0;
    float gain = 0.0;
    uint8_t flag = 0;
    int count = 5000;
    robot.SetSpeed(20);
    while (count)
    {
        rtn = robot.ServoCart(mode, &desc_pos_dt, exaxis, pos_gain, acc, vel, cmdT,␣
↪filterT, gain);
        printf("ServoCart rtn is %d\n", rtn);
        count -= 1;
        desc_pos_dt.tran.x += 0.01;
        exaxis.ePos[0] += 0.01;
    }
    robot.CloseRPC();
    return 0;
}
```

2.1.4.29 Spline Motion Start

```
/**
* @brief Spline motion start
* @return Error code
*/
errno_t SplineStart();
```

2.1.4.30 Joint Space Spline Motion (Auto Forward Kinematics Calculation)

```
/**
* @brief Joint space spline motion (auto forward kinematics calculation)
* @param [in] joint_pos Target joint position, unit: deg
* @param [in] tool Tool coordinate system number, range [0~14]
* @param [in] user User coordinate system number, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], currently not open
* @param [in] ovl Velocity scaling factor, range [0~100]
* @return Error code
*/
errno_t SplinePTP(JointPos* joint_pos, int tool, int user, float vel, float acc, float␣
↪ovl);
```

2.1.4.31 Spline PTP Motion

```
/**
* @brief Joint space spline motion
* @param [in] joint_pos Target joint position in deg
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate number [0~14]
* @param [in] user Workpiece coordinate number [0~14]
* @param [in] vel Speed percentage [0~100]
* @param [in] acc Acceleration percentage [0~100] (not yet available)
* @param [in] ovl Speed scaling factor [0~100]
* @return Error code
*/
errno_t SplinePTP(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float vel,
↪ float acc, float ovl);
```

2.1.4.32 Spline Motion End

```
/**
* @brief Spline motion end
* @return Error code
*/
errno_t SplineEnd();
```

2.1.4.33 Spline Motion Example

```
int TestSpline(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3(-61.954, -84.409, 108.153, -116.316, -91.283, 74.260);
    JointPos j4(-89.575, -80.276, 102.713, -116.302, -91.284, 74.267);
    DescPose desc_pos1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose desc_pos3(-327.622, 402.230, 320.402, -178.067, 2.127, -46.207);
    DescPose desc_pos4(-104.066, 544.321, 327.023, -177.715, 3.371, -73.818);
    DescPose offset_pos(0, 0, 0, 0, 0, 0);
    ExaxisPos epos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
```

(continues on next page)

(continued from previous page)

```
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = -1.0;
    uint8_t flag = 0;
    robot.SetSpeed(20);
    int err1 = robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT,␣
↪flag, &offset_pos);
    printf("movej errcode:%d\n", err1);
    robot.SplineStart();
    robot.SplinePTP(&j1, &desc_pos1, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j2, &desc_pos2, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j3, &desc_pos3, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j4, &desc_pos4, tool, user, vel, acc, ovl);
    robot.SplineEnd();
    err1 = robot.MoveJ(&j1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", err1);
    robot.SplineStart();
    robot.SplinePTP(&j1, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j2, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j3, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j4, tool, user, vel, acc, ovl);
    robot.SplineEnd();
    robot.CloseRPC();
    return 0;
}
```

2.1.4.34 New Spline Motion Start

Changed in version C++SDK-v2.1.3.0.

```
/**
* @brief New spline motion start
* @param [in] type 0-Circular transition, 1-Given points as path points
* @param [in] averageTime Global average transition time in ms (10 ~ ), default 2000
* @return Error code
*/
errno_t NewSplineStart(int type, int averageTime=2000);
```

2.1.4.35 New Spline Command Point

```
/**
* @brief New spline command point
* @param [in] joint_pos Target joint position in deg
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate number [0~14]
* @param [in] user Workpiece coordinate number [0~14]
* @param [in] vel Speed percentage [0~100]
* @param [in] acc Acceleration percentage [0~100] (not yet available)
```

(continues on next page)

(continued from previous page)

```
* @param [in] ovl Speed scaling factor [0~100]
* @param [in] blendR [-1.0]-Move to position (blocking), [0~1000.0]-Smoothing radius (non-blocking) in mm
* @param [in] lastFlag Whether it's the last point, 0-No, 1-Yes
* @return Error code
*/
errno_t NewSplinePoint(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float vel, float acc, float ovl, float blendR, int lastFlag);
```

2.1.4.36 New Spline Command Point (Auto Inverse Kinematics Calculation)

```
/**
* @brief New spline command point (auto inverse kinematics calculation)
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate system number, range [0~14]
* @param [in] user User coordinate system number, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], currently not open
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Move to position (blocking), [0~1000.0]-Blending radius (non-blocking), unit: mm
* @param [in] lastFlag Whether it is the last point, 0-No, 1-Yes
* @param [in] config Inverse kinematics joint space configuration, [-1]-Calculate based on current joint position, [0~7]-Solve according to specific joint space configuration
* @return Error code
*/
errno_t NewSplinePoint(DescPose* desc_pos, int tool, int user, float vel, float acc, float ovl, float blendR, int lastFlag, int config = -1);
```

2.1.4.37 New Spline Motion End

```
/**
* @brief New spline motion end
* @return Error code
*/
errno_t NewSplineEnd();
```

2.1.4.38 New Spline Motion Example

```
int TestNewSpline(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
```

(continues on next page)

(continued from previous page)

```
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3(-61.954, -84.409, 108.153, -116.316, -91.283, 74.260);
    JointPos j4(-89.575, -80.276, 102.713, -116.302, -91.284, 74.267);
    JointPos j5(-95.228, -54.621, 73.691, -112.245, -91.280, 74.268);
    DescPose desc_pos1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose desc_pos3(-327.622, 402.230, 320.402, -178.067, 2.127, -46.207);
    DescPose desc_pos4(-104.066, 544.321, 327.023, -177.715, 3.371, -73.818);
    DescPose desc_pos5(-33.421, 732.572, 275.103, -177.907, 2.709, -79.482);
    DescPose offset_pos(0, 0, 0, 0, 0, 0);
    ExaxisPos epos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = -1.0;
    uint8_t flag = 0;
    robot.SetSpeed(20);
    int err1 = robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT,␣
↪flag, &offset_pos);
    printf("movej errcode:%d\n", err1);
    robot.NewSplineStart(1, 2000);
    robot.NewSplinePoint(&j1, &desc_pos1, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(&j2, &desc_pos2, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(&j3, &desc_pos3, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(&j4, &desc_pos4, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(&j5, &desc_pos5, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplineEnd();
    err1 = robot.MoveJ(&j1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", err1);
    robot.NewSplineStart(1, 2000);
    robot.NewSplinePoint(&desc_pos1, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(&desc_pos2, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(&desc_pos3, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(&desc_pos4, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(&desc_pos5, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplineEnd();
    robot.CloseRPC();
    return 0;
}
```

2.1.4.39 Stop Motion

```
/**
* @brief Stop motion
* @return Error code
*/
errno_t StopMotion();
```

2.1.4.40 Pause Motion

```
/**
* @brief Pause motion
* @return Error code
*/
errno_t PauseMotion();
```

2.1.4.41 Resume Motion

```
/**
* @brief Resume motion
* @return Error code
*/
errno_t ResumeMotion();
```

2.1.4.42 Motion Pause, Resume, Stop Example

```
int TestPause(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j5(-95.228, -54.621, 73.691, -112.245, -91.280, 74.268);
    DescPose desc_pos1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos5(-33.421, 732.572, 275.103, -177.907, 2.709, -79.482);
    DescPose offset_pos(0, 0, 0, 0, 0, 0);
    ExaxisPos epos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
```

(continues on next page)

(continued from previous page)

```
    float ovl = 100.0;
    float blendT = -1.0;
    uint8_t flag = 0;
    robot.SetSpeed(20);
    rtn = robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT, flag, &
↪offset_pos);
    rtn = robot.MoveJ(&j5, &desc_pos5, tool, user, vel, acc, ovl, &epos, 1, flag, &
↪offset_pos);
    robot.Sleep(1000);
    robot.PauseMotion();
    robot.Sleep(1000);
    robot.ResumeMotion();
    robot.Sleep(1000);
    robot.StopMotion();
    robot.Sleep(1000);
    robot.CloseRPC();
    return 0;
}
```

2.1.4.43 Point Global Offset Start

```
/**
* @brief Point global offset start
* @param [in] flag 0-Offset in base/workpiece coordinate, 2-Offset in tool coordinate
* @param [in] offset_pos Pose offset
* @return Error code
*/
errno_t PointsOffsetEnable(int flag, DescPose *offset_pos);
```

2.1.4.44 Point Global Offset End

```
/**
* @brief Point global offset end
* @return Error code
*/
errno_t PointsOffsetDisable();
```

2.1.4.45 Point Offset Example

```
int TestOffset(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
```

(continues on next page)

(continued from previous page)

```
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose desc_pos1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose offset_pos(0, 0, 0, 0, 0, 0);
    DescPose offset_pos1(0, 0, 50, 0, 0, 0);
    ExaxisPos epos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = -1.0;
    uint8_t flag = 0;
    robot.SetSpeed(20);
    robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_
↪pos);
    robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_
↪pos);
    robot.Sleep(1000);
    robot.PointsOffsetEnable(0, &offset_pos1);
    robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_
↪pos);
    robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_
↪pos);
    robot.PointsOffsetDisable();
    robot.CloseRPC();
    return 0;
}
```

2.1.4.46 Control Box AO Flying Start

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Control box AO flying start
* @param [in] AONum Control box AO number
* @param [in] maxTCPSpeed Maximum TCP speed value [1-5000mm/s], default 1000
* @param [in] maxAOPercent AO percentage corresponding to maximum TCP speed, default 100%
* @param [in] zeroZoneCmp Dead zone compensation value AO percentage, integer, default 20
↪%, range [0-100]
* @return Error code
*/
errno_t MoveAOStart(int AONum, int maxTCPSpeed, int maxAOPercent, int zeroZoneCmp);
```

2.1.4.47 Control Box AO Flying Stop

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Control box AO flying stop
* @return Error code
*/
errno_t MoveAOStop();
```

2.1.4.48 End AO Flying Start

New in version C++SDK-v2.1.4.0.

```
/**
* @brief End AO flying start
* @param [in] AONum End AO number
* @param [in] maxTCPSpeed Maximum TCP speed value [1-5000mm/s], default 1000
* @param [in] maxAOPercent AO percentage corresponding to maximum TCP speed, default 100%
* @param [in] zeroZoneCmp Dead zone compensation value AO percentage, integer, default 20%, range [0-100]
* @return Error code
*/
errno_t MoveToolAOStart(int AONum, int maxTCPSpeed, int maxAOPercent, int zeroZoneCmp);
```

2.1.4.49 End AO Flying Stop

New in version C++SDK-v2.1.4.0.

```
/**
* @brief End AO flying stop
* @return Error code
*/
errno_t MoveToolAOStop();
```

2.1.4.50 AO Flying Example

```
int TestMoveAO(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
```

(continues on next page)

(continued from previous page)

```
    JointPos j2(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose desc_pos1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose offset_pos(0, 0, 0, 0, 0, 0);
    DescPose offset_pos1(0, 0, 50, 0, 0, 0);
    ExaxisPos epos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    float vel = 20.0;
    float acc = 20.0;
    float ovl = 100.0;
    float blendT = -1.0;
    uint8_t flag = 0;
    robot.SetSpeed(20);
    robot.MoveAOStart(0, 100, 100, 20);
    robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_
↪pos);
    robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_
↪pos);
    robot.MoveAOStop();
    robot.Sleep(1000);
    robot.MoveToolAOStart(0, 100, 100, 20);
    robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_
↪pos);
    robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_
↪pos);
    robot.MoveToolAOStop();
    robot.CloseRPC();
    return 0;
}
```

2.1.4.51 Start Ptp Motion FIR Filter

New in version V3.7.7.

```
/**
* @brief Start Ptp motion FIR filter
* @param [in] maxAcc Maximum acceleration limit (deg/s2)
* @param [in] maxJek Unified joint jerk limit (deg/s3), default 1000
* @return Error code
*/
errno_t PtpFIRPlanningStart(double maxAcc, double maxJek = 1000);
```

2.1.4.52 Close Ptp Motion FIR Filter

New in version V3.7.7.

```
/**
* @brief Close Ptp motion FIR filter
* @return Error code
*/
errno_t PtpFIRPlanningEnd();
```

2.1.4.53 Start LIN, ARC Motion FIR Filter

New in version V3.7.7.

```
/**
* @brief Start LIN, ARC motion FIR filter
* @param [in] maxAccLin Linear acceleration limit (mm/s2)
* @param [in] maxAccDeg Angular acceleration limit (deg/s2)
* @param [in] maxJerkLin Linear jerk limit (mm/s3)
* @param [in] maxJerkDeg Angular jerk limit (deg/s3)
* @return Error code
*/
errno_t LinArcFIRPlanningStart(double maxAccLin, double maxAccDeg, double maxJerkLin,␣
↪double maxJerkDeg);
```

2.1.4.54 Close LIN, ARC Motion FIR Filter

New in version V3.7.7.

```
/**
* @brief Close LIN, ARC motion FIR filter
* @return Error code
*/
errno_t LinArcFIRPlanningEnd();
```

2.1.4.55 FIR Filter Example

```
int TestFIR(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos startjointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
```

(continues on next page)

(continued from previous page)

```
    JointPos midjointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos endjointPos(-29.777, -84.536, 109.275, -114.075, -86.655, 74.257);
    DescPose startdescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose middescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose enddescPose(-487.434, 154.362, 308.576, 176.600, 0.268, -14.061);
    ExaxisPos exaxisPos(0, 0, 0, 0);
    DescPose offdese(0, 0, 0, 0, 0, 0);
    rtn = robot.PtpFIRPlanningStart(1000, 1000);
    cout << "PtpFIRPlanningStart rtn is " << rtn << endl;
    robot.MoveJ(&startjointPos, &startdescPose, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
    robot.MoveJ(&endjointPos, &enddescPose, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
    robot.PtpFIRPlanningEnd();
    cout << "PtpFIRPlanningEnd rtn is " << rtn << endl;
    robot.LinArcFIRPlanningStart(1000, 1000, 1000, 1000);
    cout << "LinArcFIRPlanningStart rtn is " << rtn << endl;
    robot.MoveL(&startjointPos, &startdescPose, 0, 0, 100, 100, 100, -1, &exaxisPos, 0,␣
↪0, &offdese, 1, 1);
    robot.MoveC(&midjointPos, &middescPose, 0, 0, 100, 100, &exaxisPos, 0, &offdese, &
↪endjointPos, &enddescPose, 0, 0, 100, 100, &exaxisPos, 0, &offdese, 100, -1);
    robot.LinArcFIRPlanningEnd();
    cout << "LinArcFIRPlanningEnd rtn is " << rtn << endl;
    robot.CloseRPC();
    return 0;
}
```

2.1.4.56 Acceleration Smoothing Start

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Acceleration smoothing start
* @param [in] saveFlag Whether to save after power off
* @return Error code
*/
errno_t AccSmoothStart(bool saveFlag);
```

2.1.4.57 Acceleration Smoothing End

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Acceleration smoothing end
* @param [in] saveFlag Whether to save after power off
* @return Error code
*/
errno_t AccSmoothEnd(bool saveFlag);
```

2.1.4.58 Acceleration Smoothing Example

```
int TestAccSmooth(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos startjointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos endjointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose startdescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose enddescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    ExaxisPos exaxisPos(0, 0, 0, 0);
    DescPose offdese(0, 0, 0, 0, 0, 0);
    rtn = robot.AccSmoothStart(0);
    cout << "AccSmoothStart rtn is " << rtn << endl;
    robot.MoveJ(&startjointPos, &startdescPose, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
    robot.MoveJ(&endjointPos, &enddescPose, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
    rtn = robot.AccSmoothEnd(0);
    cout << "AccSmoothEnd rtn is " << rtn << endl;
    robot.CloseRPC();
    return 0;
}
```

2.1.4.59 Specified Pose Speed Start

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Specified pose speed start
* @param [in] ratio Pose speed percentage [0-300]
* @return Error code
*/
errno_t AngularSpeedStart(int ratio);
```

2.1.4.60 Specified Pose Speed End

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Specified pose speed end
* @return Error code
*/
errno_t AngularSpeedEnd();
```

2.1.4.61 Robot Specified Pose Speed Example

```
int TestAngularSpeed(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos startjointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos endjointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose startdescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose enddescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    ExaxisPos exaxisPos(0, 0, 0, 0);
    DescPose offdese(0, 0, 0, 0, 0, 0);
    rtn = robot.AngularSpeedStart(50);
    cout << "AngularSpeedStart rtn is " << rtn << endl;
    robot.MoveJ(&startjointPos, &startdescPose, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
    robot.MoveJ(&endjointPos, &enddescPose, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
    rtn = robot.AngularSpeedEnd();
    cout << "AngularSpeedEnd rtn is " << rtn << endl;
    robot.CloseRPC();
    return 0;
}
```

2.1.4.62 Start Singular Pose Protection

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Start singular pose protection
* @param [in] protectMode Singular protection mode, 0: Joint mode; 1-Cartesian mode
* @param [in] minShoulderPos Shoulder singular adjustment range (mm), default 100
* @param [in] minElbowPos Elbow singular adjustment range (mm), default 50
* @param [in] minWristPos Wrist singular adjustment range (°), default 10
* @return Error code
*/
errno_t SingularAvoidStart(int protectMode, double minShoulderPos, double minElbowPos,␣
↪double minWristPos);
```

2.1.4.63 Stop Singular Pose Protection

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Stop singular pose protection
* @return Error code
*/
errno_t SingularAvoidEnd();
```

2.1.4.64 Robot Singular Pose Protection Example

```
int TestAngularSpeed(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos startjointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos endjointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose startdescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose enddescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    ExaxisPos exaxisPos(0, 0, 0, 0);
    DescPose offdese(0, 0, 0, 0, 0, 0);
    rtn = robot.SingularAvoidStart(2, 10, 5, 5);
    cout << "SingularAvoidStart rtn is " << rtn << endl;
    robot.MoveJ(&startjointPos, &startdescPose, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
    robot.MoveJ(&endjointPos, &enddescPose, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
```

(continues on next page)

(continued from previous page)

```
    rtn = robot.SingularAvoidEnd();
    cout << "SingularAvoidEnd rtn is " << rtn << endl;
    robot.CloseRPC();
    return 0;
}
```

2.1.4.65 Clear the motion command queue

```
/**
* @brief Clear the motion command queue
* @return error code
*/
errno_t MotionQueueClear();
```

2.1.4.66 Move to Intersecting Line Start Point

```
/**
* @brief Move to intersecting line start point
* @param [in] mainPoint Cartesian poses of 6 taught points on the main pipe
* @param [in] mainExaxisPos Extended axis positions for 6 taught points on the main pipe
* @param [in] piecePoint Cartesian poses of 6 taught points on the branch pipe
* @param [in] pieceExaxisPos Extended axis positions for 6 taught points on the branch␣
↪pipe
* @param [in] extAxisFlag Whether to enable extended axis; 0-Disable; 1-Enable
* @param [in] exaxisPos Start point extended axis position
* @param [in] tool Tool coordinate system number
* @param [in] wobj Workpiece coordinate system number
* @param [in] vel Velocity percentage
* @param [in] acc Acceleration percentage
* @param [in] ovl Velocity scaling factor
* @param [in] oacc Acceleration scaling factor
* @param [in] moveType Motion type; 0-PTP; 1-LIN
* @param [in] moveDirection Motion direction; 0-Clockwise; 1-Counterclockwise
* @param [in] offset Offset value
* @return Error code
*/
errno_t MoveToIntersectLineStart(DescPose mainPoint[6], ExaxisPos mainExaxisPos[6],␣
↪DescPose piecePoint[6], ExaxisPos pieceExaxisPos[6], int extAxisFlag, ExaxisPos␣
↪exaxisPos, int tool, int wobj, double vel, double acc, double ovl, double oacc, int␣
↪moveType, int moveDirection, DescPose offset);
```

2.1.4.67 Intersecting Line Motion

```
/**
* @brief Intersecting line motion
* @param [in] mainPoint Cartesian poses of 6 taught points on the main pipe
* @param [in] mainExaxisPos Extended axis positions for 6 taught points on the main pipe
* @param [in] piecePoint Cartesian poses of 6 taught points on the branch pipe
* @param [in] pieceExaxisPos Extended axis positions for 6 taught points on the branch␣
↪pipe
* @param [in] extAxisFlag Whether to enable extended axis; 0-Disable; 1-Enable
* @param [in] exaxisPos Start point extended axis positions
* @param [in] tool Tool coordinate system number
* @param [in] wobj Workpiece coordinate system number
* @param [in] vel Velocity percentage
* @param [in] acc Acceleration percentage
* @param [in] ovl Velocity scaling factor
* @param [in] oacc Acceleration scaling factor
* @param [in] moveDirection Motion direction; 0-Clockwise; 1-Counterclockwise
* @param [in] offset Offset value
* @return Error code
*/
errno_t MoveIntersectLine(DescPose mainPoint[6], ExaxisPos mainExaxisPos[6], DescPose␣
↪piecePoint[6], ExaxisPos pieceExaxisPos[6], int extAxisFlag, ExaxisPos exaxisPos[4],␣
↪int tool, int wobj, double vel, double acc, double ovl, double oacc, int moveDirection,
↪ DescPose offset);
```

2.1.4.68 Robot Intersecting Line Motion Code Example

```
void TestIntersectLineMove()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(3);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return ;
    }
    robot.SetReConnectParam(true, 30000, 500);
    DescPose mainPoint[6] = {};
    DescPose piecePoint[6] = {};
    ExaxisPos mainExaxisPos[6] = {};
    ExaxisPos pieceExaxisPos[6] = {};
    int extAxisFlag = 1;
    ExaxisPos exaxisPos[4] = {};
    DescPose offset = { 0.0, 2.0 ,30.0, -2.0, 0.0, 0.0 };
    mainPoint[0] = {490.004, -383.194, 402.735, -9.332, -1.528, 69.594};
    mainPoint[1] = {444.950, -407.117, 389.011, -5.546, -2.196, 65.279};
    mainPoint[2] = {445.168, -463.605, 355.759, -1.544, -10.886, 57.104};
    mainPoint[3] = {507.529, -485.385, 343.013, -0.786, -4.834, 61.799};
```

(continues on next page)

(continued from previous page)

```
    mainPoint[4] = {554.390, -442.647, 367.701, -4.761, -10.181, 64.925};
    mainPoint[5] = {532.552, -394.003, 396.467, -13.732, -13.592, 67.411};
    mainExaxisPos[0] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[1] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[2] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[3] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[4] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[5] = { -29.996, 0.000, 0.000, 0.000 };
    piecePoint[0] = { 505.571, -192.408, 316.759, 38.098, 37.051, 139.447 };
    piecePoint[1] = {533.837, -201.558, 332.340, 34.644, 42.339, 137.748};
    piecePoint[2] = {530.386, -225.085, 373.808, 35.431, 45.111, 137.560};
    piecePoint[3] = {485.646, -229.195, 383.778, 33.870, 45.173, 137.064};
    piecePoint[4] = {460.551, -212.161, 354.256, 28.856, 45.602, 135.930};
    piecePoint[5] = {474.217, -197.124, 324.611, 42.469, 41.133, 148.167};
    pieceExaxisPos[0] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[1] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[2] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[3] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[4] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[5] = { -29.996, -0.000, 0.000, 0.000 };
    exaxisPos[0] = {-29.996, -0.000, 0.000, 0.000};
    exaxisPos[1] = {-44.994, 90.000, 0.000, 0.000};
    exaxisPos[2] = {-59.992, 0.002, 0.000, 0.000};
    exaxisPos[3] = {-44.994, -89.997, 0.000, 0.000};
    int tool = 2;
    int wobj = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 12.0;
    double oacc = 12.0;
    int moveType = 1;
    int moveDirection = 1;
    rtn = robot.MoveToIntersectLineStart(mainPoint, mainExaxisPos, piecePoint,␣
↪pieceExaxisPos, extAxisFlag, exaxisPos[0], tool, wobj, vel, acc, ovl, oacc, moveType,␣
↪moveDirection, offset);
    printf("MoveToIntersectLineStart rtn is %d\n", rtn);
    rtn = robot.MoveIntersectLine(mainPoint, mainExaxisPos, piecePoint, pieceExaxisPos,␣
↪extAxisFlag, exaxisPos, tool, wobj, vel, acc, 5.0, 5.0, moveDirection, offset);
    printf("MoveIntersectLine rtn is %d\n", rtn);
    robot.CloseRPC();
    return ;
}
```

2.1.4.69 Stationary Air Motion

```
/**
* @brief Stationary Air Motion
* @return Error code
*/
errno_t MoveStationary();
```

2.1.4.70 Stationary Air Motion Code Example

```
int TestLaserStationary(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return 0;
    }
    robot.SetReConnectParam(true, 30000, 500);
    rtn = robot.LaserSensorRecordandReplay(0, 10, 1, 0, 0.1, 1, 0, 10, 100);
    printf("LaserSensorRecordandReplay rtn is %d\n", rtn);
    rtn = robot.MoveStationary();
    printf("MoveStationary rtn is %d\n", rtn);
    rtn = robot.LaserSensorRecord1(0, 10);
    printf("LaserSensorRecordandReplay rtn is %d\n", rtn);
    robot.CloseRPC();
    robot.Sleep(9999999);
    return 0;
}
```

2.1.5 IO

2.1.5.1 Set Control Box Digital Output

```
/**
@brief Set control box digital output
@param [in] id IO number, range [0~15]
@param [in] status 0-off, 1-on
@param [in] smooth 0-non-smooth, 1-smooth
@param [in] block 0-blocking, 1-non-blocking
@return Error code
*/
errno_t SetDO(int id, uint8_t status, uint8_t smooth, uint8_t block);
```

2.1.5.2 Set Tool Digital Output

```
/**
@brief Set tool digital output
@param [in] id IO number, range [0~1]
@param [in] status 0-off, 1-on
@param [in] smooth 0-non-smooth, 1-smooth
@param [in] block 0-blocking, 1-non-blocking
@return Error code
*/
errno_t SetToolDO(int id, uint8_t status, uint8_t smooth, uint8_t block);
```

2.1.5.3 Set Control Box Analog Output

```
/**
@brief Set control box analog output
@param [in] id IO number, range [0~1]
@param [in] value Current or voltage percentage, range [0~100] corresponding to current␣
↪[0~20mA] or voltage [0~10V]
@param [in] block 0-blocking, 1-non-blocking
@return Error code
*/
errno_t SetAO(int id, float value, uint8_t block);
```

2.1.5.4 Set Tool Analog Output

```
/**
@brief Set tool analog output
@param [in] id IO number, range [0]
@param [in] value Current or voltage percentage, range [0~100] corresponding to current␣
↪[0~20mA] or voltage [0~10V]
@param [in] block 0-blocking, 1-non-blocking
@return Error code
*/
errno_t SetToolAO(int id, float value, uint8_t block);
```

2.1.5.5 Code Example for Setting Digital and Analog Outputs

```
int TestAODO(void)
{
ROBOT_STATE_PKG pkg = {};
FRRobot robot;
robot.LoggerInit();
robot.SetLoggerLevel(1);
int rtn = robot.RPC("192.168.58.2");
if (rtn != 0)
{
return -1;
```

(continues on next page)

(continued from previous page)

```
}
robot.SetReConnectParam(true, 30000, 500);
uint8_t status = 1;
uint8_t smooth = 0;
uint8_t block = 0;
for (int i = 0; i < 16; i++)
{
robot.SetDO(i, status, smooth, block);
robot.Sleep(300);
}
status = 0;
for (int i = 0; i < 16; i++)
{
robot.SetDO(i, status, smooth, block);
robot.Sleep(300);
}
status = 1;
for (int i = 0; i < 2; i++)
{
robot.SetToolDO(i, status, smooth, block);
robot.Sleep(1000);
}
status = 0;
for (int i = 0; i < 2; i++)
{
robot.SetToolDO(i, status, smooth, block);
robot.Sleep(1000);
}
for (int i = 0; i < 100; i++)
{
robot.SetAO(0, i * 40.96, block);
robot.Sleep(30);
}
for (int i = 0; i < 100; i++)
{
robot.SetToolAO(0, i * 40.96, block);
robot.Sleep(30);
}
robot.CloseRPC();
return 0;
}
```

2.1.5.6 Get Control Box Digital Input

```
/**
@brief Get control box digital input
@param [in] id IO number, range [0~15]
@param [in] block 0-blocking, 1-non-blocking
@param [out] result 0-low level, 1-high level
@return Error code
*/
errno_t GetDI(int id, uint8_t block, uint8_t *result);
```

2.1.5.7 Get Tool Digital Input

```
/**
@brief Get tool digital input
@param [in] id IO number, range [0~1]
@param [in] block 0-blocking, 1-non-blocking
@param [out] result 0-low level, 1-high level
@return Error code
*/
errno_t GetToolDI(int id, uint8_t block, uint8_t *result);
```

2.1.5.8 Get Control Box Analog Input

```
/**
@brief Get control box analog input
@param [in] id IO number, range [0~1]
@param [in] block 0-blocking, 1-non-blocking
@param [out] result Input current or voltage percentage, range [0~100] corresponding to␣
↪current [0~20mS] or voltage [0~10V]
@return Error code
*/
errno_t GetAI(int id, uint8_t block, float *result);
```

2.1.5.9 Get Tool Analog Input

```
/**
@brief Get tool analog input
@param [in] id IO number, range [0]
@param [in] block 0-blocking, 1-non-blocking
@param [out] result Input current or voltage percentage, range [0~100] corresponding to␣
↪current [0~20mS] or voltage [0~10V]
@return Error code
*/
errno_t GetToolAI(int id, uint8_t block, float *result);
```

2.1.5.10 Get Robot End Point Record Button Status

```
/**
@brief Get robot end point record button status
@param [out] state Button status, 0-pressed, 1-released
@return Error code
*/
errno_t GetAxlePointRecordBtnState(uint8_t *state);
```

2.1.5.11 Get Robot End DO Output Status

```
/**
@brief Get robot end DO output status
@param [out] do_state DO output status, do0~do1 correspond to bit1~bit2, starting from bit0
@return Error code
*/
errno_t GetToolDO(uint8_t *do_state);
```

2.1.5.12 Get Robot Controller DO Output Status

```
/**
@brief Get robot controller DO output status
@param [out] do_state_h DO output status, co0~co7 correspond to bit0~bit7
@param [out] do_state_l DO output status, do0~do7 correspond to bit0~bit7
@return Error code
*/
errno_t GetDO(uint8_t *do_state_h, uint8_t *do_state_l);
```

2.1.5.13 Code Example for Getting Robot DI and DO Status

```
int TestGetDIAI(void)
{
ROBOT_STATE_PKG pkg = {};
FRRobot robot;
robot.LoggerInit();
robot.SetLoggerLevel(1);
int rtn = robot.RPC("192.168.58.2");
if (rtn != 0)
{
return -1;
}
robot.SetReConnectParam(true, 30000, 500);
uint8_t status = 1;
uint8_t smooth = 0;
uint8_t block = 0;
uint8_t di = 0, tool_di = 0;
float ai = 0.0, tool_ai = 0.0;
```

(continues on next page)

(continued from previous page)

```
float value = 0.0;
robot.GetDI(0, block, &di);
printf("di0:%u\n", di);
tool_di = robot.GetToolDI(1, block, &tool_di);
printf("tool_di1:%u\n", tool_di);
robot.GetAI(0, block, &ai);
printf("ai0:%f\n", ai);
tool_ai = robot.GetToolAI(0, block, &tool_ai);
printf("tool_ai0:%f\n", tool_ai);
uint8_t _button_state = 0;
robot.GetAxlePointRecordBtnState(&_button_state);
printf("_button_state is: %u\n", _button_state);
uint8_t tool_do_state = 0;
robot.GetToolDO(&tool_do_state);
printf("tool DO state is: %u\n", tool_do_state);
uint8_t do_state_h = 0;
uint8_t do_state_l = 0;
robot.GetDO(&do_state_h, &do_state_l);
printf("DO state high is: %u \n DO state low is: %u\n", do_state_h, do_state_l);
robot.CloseRPC();
return 0;
}
```

2.1.5.14 Wait for Control Box Digital Input

```
/**
@brief Wait for control box digital input
@param [in] id IO number, range [0~15]
@param [in] status 0-off, 1-on
@param [in] max_time Maximum wait time, unit ms
@param [in] opt Strategy after timeout, 0-stop program and prompt timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
@return Error code
*/
errno_t WaitDI(int id, uint8_t status, int max_time, int opt);
```

2.1.5.15 Wait for Control Box Multi-Channel Digital Input

```
/**
@brief Wait for control box multi-channel digital input
@param [in] mode 0-multi AND, 1-multi OR
@param [in] id IO number, bit0~bit7 correspond to DI0~DI7, bit8~bit15 correspond to CI0~
↪CI7
@param [in] status 0-off, 1-on
@param [in] max_time Maximum wait time, unit ms
@param [in] opt Strategy after timeout, 0-stop program and prompt timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
@return Error code
```

(continues on next page)

(continued from previous page)

```
*/
errno_t WaitMultiDI(int mode, int id, uint8_t status, int max_time, int opt);
```

2.1.5.16 Wait for Tool Digital Input

```
/**
@brief Wait for tool digital input
@param [in] id IO number, range [0~1]
@param [in] status 0-off, 1-on
@param [in] max_time Maximum wait time, unit ms
@param [in] opt Strategy after timeout, 0-stop program and prompt timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
@return Error code
*/
errno_t WaitToolDI(int id, uint8_t status, int max_time, int opt);
```

2.1.5.17 Wait for Control Box Analog Input

```
/**
@brief Wait for control box analog input
@param [in] id IO number, range [0~1]
@param [in] sign 0-greater than, 1-less than
@param [in] value Input current or voltage percentage, range [0~100] corresponding to␣
↪current [0~20mS] or voltage [0~10V]
@param [in] max_time Maximum wait time, unit ms
@param [in] opt Strategy after timeout, 0-stop program and prompt timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
@return Error code
*/
errno_t WaitAI(int id, int sign, float value, int max_time, int opt);
```

2.1.5.18 Wait for Tool Analog Input

```
/**
@brief Wait for tool analog input
@param [in] id IO number, range [0]
@param [in] sign 0-greater than, 1-less than
@param [in] value Input current or voltage percentage, range [0~100] corresponding to␣
↪current [0~20mS] or voltage [0~10V]
@param [in] max_time Maximum wait time, unit ms
@param [in] opt Strategy after timeout, 0-stop program and prompt timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
@return Error code
*/
errno_t WaitToolAI(int id, int sign, float value, int max_time, int opt);
```

2.1.5.19 Code Example for Waiting for Control Box Digital and Analog Input Signals

Changed in version C++SDK-v2.1.2.0.

```
int TestWaitDIAI(void)
{
ROBOT_STATE_PKG pkg = {};
FRRobot robot;
robot.LoggerInit();
robot.SetLoggerLevel(1);
int rtn = robot.RPC("192.168.58.2");
if (rtn != 0)
{
return -1;
}
robot.SetReConnectParam(true, 30000, 500);
uint8_t status = 1;
uint8_t smooth = 0;
uint8_t block = 0;
uint8_t di = 0, tool_di = 0;
float ai = 0.0, tool_ai = 0.0;
float value = 0.0;
rtn = robot.WaitDI(0, 1, 1000, 1);
cout << "WaitDI over; rtn is: " << rtn << endl;
robot.WaitMultiDI(1, 3, 3, 1000, 1);
cout << "WaitDI over; rtn is: " << rtn << endl;
robot.WaitToolDI(1, 1, 1000, 1);
cout << "WaitDI over; rtn is: " << rtn << endl;
robot.WaitAI(0, 0, 50, 1000, 1);
cout << "WaitDI over; rtn is: " << rtn << endl;
robot.WaitToolAI(0, 0, 50, 1000, 1);
cout << "WaitDI over; rtn is: " << rtn << endl;
robot.CloseRPC();
return 0;
}
```

2.1.5.20 Set Whether to Reset Control Box DO Output After Stop/Pause

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set whether to reset control box DO output after stop/pause
* @param [in] resetFlag 0-do not reset; 1-reset
* @param [in] reloadFlag Whether to reload after pause recovery, 0-do not load; 1-load
* @return Error code
*/
errno_t SetOutputResetCtlBoxDO(int resetFlag, int reloadFlag = 0);
```

2.1.5.21 Set Whether to Reset Control Box AO Output After Stop/Pause

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set whether to reset control box AO output after stop/pause
* @param [in] resetFlag  0-do not reset; 1-reset
* @param [in] reloadFlag Whether to reload after pause recovery, 0-do not load; 1-load
* @return Error code
*/
errno_t SetOutputResetCtlBoxAO(int resetFlag, int reloadFlag = 0);
```

2.1.5.22 Set Whether to Reset End Tool DO Output After Stop/Pause

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set whether to reset end tool DO output after stop/pause
* @param [in] resetFlag  0-do not reset; 1-reset
* @param [in] reloadFlag Whether to reload after pause recovery, 0-do not load; 1-load
* @return Error code
*/
errno_t SetOutputResetAxleDO(int resetFlag, int reloadFlag = 0);
```

2.1.5.23 Set Whether to Reset End Tool AO Output After Stop/Pause

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set whether to reset end tool AO output after stop/pause
* @param [in] resetFlag  0-do not reset; 1-reset
* @param [in] reloadFlag Whether to reload after pause recovery, 0-do not load; 1-load
* @return  Error code
*/
errno_t SetOutputResetAxleAO(int resetFlag, int reloadFlag = 0);
```

2.1.5.24 Set Whether to Reset Extension DO Output After Stop/Pause

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set whether to reset extension DO output after stop/pause
* @param [in] resetFlag  0-do not reset; 1-reset
* @param [in] reloadFlag Whether to reload after pause recovery, 0-do not load; 1-load
* @return  Error code
*/
errno_t SetOutputResetExtDO(int resetFlag, int reloadFlag = 0);
```

2.1.5.25 Set Whether to Reset Extension AO Output After Stop/Pause

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set whether to reset extension AO output after stop/pause
* @param [in] resetFlag  0-do not reset; 1-reset
* @param [in] reloadFlag Whether to reload after pause recovery, 0-do not load; 1-load
* @return  Error code
*/
errno_t SetOutputResetExtAO(int resetFlag, int reloadFlag = 0);
```

2.1.5.26 Set Whether to Reset SmartTool Output After Stop/Pause

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set whether to reset SmartTool output after stop/pause
* @param [in] resetFlag  0-do not reset; 1-reset
* @param [in] reloadFlag Whether to reload after pause recovery, 0-do not load; 1-load
* @return  Error code
*/
errno_t SetOutputResetSmartToolDO(int resetFlag, int reloadFlag = 0);
```

2.1.5.27 Example Code for Setting LUA Program Output Reset After Stop/Pause

New in version C++SDK-v2.1.5.0.

```
int TestDOReset(void)
{
ROBOT_STATE_PKG pkg = {};
FRRobot robot;
robot.LoggerInit();
robot.SetLoggerLevel(3);
int rtn = robot.RPC("192.168.58.2");
if (rtn != 0)
{
    return -1;
}
robot.SetReConnectParam(true, 30000, 500);
for (int i = 0; i < 16; i++)
{
    robot.SetDO(i, 1, 0, 0);
    robot.Sleep(200);
}
int resetFlag = 0;
int resumeReloadFlag = 0;
rtn = robot.SetOutputResetCtlBoxDO(resetFlag, resumeReloadFlag);
robot.SetOutputResetCtlBoxAO(resetFlag, resumeReloadFlag);
robot.SetOutputResetAxleDO(resetFlag, resumeReloadFlag);
robot.SetOutputResetAxleAO(resetFlag, resumeReloadFlag);
```

(continues on next page)

(continued from previous page)

```
robot.SetOutputResetExtDO(resetFlag, resumeReloadFlag);
robot.SetOutputResetExtAO(resetFlag, resumeReloadFlag);
robot.SetOutputResetSmartToolDO(resetFlag, resumeReloadFlag);
robot.ProgramLoad("/fruser/test.lua");
robot.ProgramRun();
robot.Sleep(2000);
robot.PauseMotion();
robot.Sleep(2000);
robot.ResumeMotion();
robot.Sleep(2000);
robot.CloseRPC();
return 0;
}
```

2.1.6 Common Settings

2.1.6.1 Set Tool Reference Point - Six-Point Method

```
/**
 * @brief Set tool reference point - six-point method
 * @param [in] point_num Point number, range [1~6]
 * @return Error code
 */
errno_t SetToolPoint(int point_num);
```

2.1.6.2 Calculate Tool Coordinate System

```
/**
 * @brief Calculate tool coordinate system
 * @param [out] tcp_pose Tool coordinate system
 * @return Error code
 */
errno_t ComputeTool(DescPose *tcp_pose);
```

2.1.6.3 Set Tool Reference Point - Four-Point Method

```
/**
 * @brief Set tool reference point - four-point method
 * @param [in] point_num Point number, range [1~4]
 * @return Error code
 */
errno_t SetTcp4RefPoint(int point_num);
```

2.1.6.4 Calculate Tool Coordinate System

```
/**
 * @brief Calculate tool coordinate system
 * @param [out] tcp_pose Tool coordinate system
 * @return Error code
 */
errno_t ComputeTcp4(DescPose *tcp_pose);
```

2.1.6.5 Calculate Tool Coordinate System from Points

New in version C++SDK-v2.1.8-3.7.8.

```
/**
 * @brief Calculate tool coordinate system from point data
 * @param [in] method Calculation method: 0-four-point method; 1-six-point method
 * @param [in] pos Joint position array (4 points for four-point method, 6 points for six-point method)
 * @param [out] coord Resulting tool coordinate system
 * @return Error code
 */
errno_t ComputeToolCoordWithPoints(int method, JointPos pos[], DescPose& coord);
```

2.1.6.6 Set Tool Coordinate System

Changed in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set tool coordinate system
 * @param [in] id Coordinate system ID, range [0~14]
 * @param [in] coord Tool center point pose relative to end flange center
 * @param [in] type 0-tool coordinate system, 1-sensor coordinate system
 * @param [in] install Installation position: 0-robot end, 1-external to robot
 * @param [in] toolID Tool ID
 * @param [in] loadNum Load number
 * @return Error code
 */
errno_t SetToolCoord(int id, DescPose *coord, int type, int install, int toolID, int loadNum);
```

2.1.6.7 Set Tool Coordinate System List

Changed in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set tool coordinate system list
 * @param [in] id Coordinate system ID, range [0~14]
 * @param [in] coord Tool center point pose relative to end flange center
 * @param [in] type 0-tool coordinate system, 1-sensor coordinate system
 * @param [in] install Installation position: 0-robot end, 1-external to robot
```

(continues on next page)

(continued from previous page)

```
 * @param [in] loadNum Load number
 * @return Error code
 */
errno_t SetToolList(int id, DescPose *coord, int type, int install, int loadNum);
```

2.1.6.8 Get Current Tool Coordinate System

```
/**
 * @brief Get current tool coordinate system
 * @param [in] flag 0-blocking, 1-non-blocking
 * @param [out] desc_pos Tool coordinate system pose
 * @return Error code
 */
errno_t GetTCPOffset(uint8_t flag, DescPose *desc_pos);
```

2.1.6.9 Robot Tool Coordinate System Operation Example

```
int TestTCPCompute(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    DescPose p1Desc(186.331, 487.913, 209.850, 149.030, 0.688, -114.347);
    JointPos p1Joint(-127.876, -75.341, 115.417, -122.741, -59.820, 74.300);
    DescPose p2Desc(69.721, 535.073, 202.882, -144.406, -14.775, -89.012);
    JointPos p2Joint(-101.780, -69.828, 110.917, -125.740, -127.841, 74.300);
    DescPose p3Desc(146.861, 578.426, 205.598, 175.997, -36.178, -93.437);
    JointPos p3Joint(-112.851, -60.191, 86.566, -80.676, -97.463, 74.300);
    DescPose p4Desc(136.284, 509.876, 225.613, 178.987, 1.372, -100.696);
    JointPos p4Joint(-116.397, -76.281, 113.845, -128.611, -88.654, 74.299);
    DescPose p5Desc(138.395, 505.972, 298.016, 179.134, 2.147, -101.110);
    JointPos p5Joint(-116.814, -82.333, 109.162, -118.662, -88.585, 74.302);
    DescPose p6Desc(105.553, 454.325, 232.017, -179.426, 0.444, -99.952);
    JointPos p6Joint(-115.649, -84.367, 122.447, -128.663, -90.432, 74.303);
    ExaxisPos exaxisPos(0, 0, 0, 0);
    DescPose offdese(0, 0, 0, 0, 0, 0);
    JointPos posJ[6] = { p1Joint , p2Joint , p3Joint , p4Joint , p5Joint , p6Joint };
    DescPose coordRtn = {};
    rtn = robot.ComputeToolCoordWithPoints(1, posJ, coordRtn);
    printf("ComputeToolCoordWithPoints    %d  coord is %f %f %f %f %f %f \n", rtn,␣
↪coordRtn.tran.x, coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry,␣
↪coordRtn.rpy.rz);
```

(continues on next page)

(continued from previous page)

```
    robot.MoveJ(&p1Joint, &p1Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetToolPoint(1);
    robot.MoveJ(&p2Joint, &p2Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetToolPoint(2);
    robot.MoveJ(&p3Joint, &p3Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetToolPoint(3);
    robot.MoveJ(&p4Joint, &p4Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetToolPoint(4);
    robot.MoveJ(&p5Joint, &p5Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetToolPoint(5);
    robot.MoveJ(&p6Joint, &p6Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetToolPoint(6);
    rtn = robot.ComputeTool(&coordRtn);
    printf("6 Point ComputeTool        %d  coord is %f %f %f %f %f %f \n", rtn, coordRtn.
↪tran.x, coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry, coordRtn.
↪rpy.rz);
    robot.SetToolList(1, &coordRtn, 0, 0, 0);
    robot.MoveJ(&p1Joint, &p1Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetTcp4RefPoint(1);
    robot.MoveJ(&p2Joint, &p2Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetTcp4RefPoint(2);
    robot.MoveJ(&p3Joint, &p3Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetTcp4RefPoint(3);
    robot.MoveJ(&p4Joint, &p4Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetTcp4RefPoint(4);
    rtn = robot.ComputeTcp4(&coordRtn);
    printf("4 Point ComputeTool        %d  coord is %f %f %f %f %f %f \n", rtn, coordRtn.
↪tran.x, coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry, coordRtn.
↪rpy.rz);
    robot.SetToolCoord(2, &coordRtn, 0, 0, 1, 0);
    DescPose getCoord = {};
    rtn = robot.GetTCPOffset(0, &getCoord);
    printf("GetTCPOffset    %d  coord is %f %f %f %f %f %f \n", rtn, coordRtn.tran.x,␣
↪coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry, coordRtn.rpy.rz);
    robot.CloseRPC();
    return 0;
}
```

2.1.6.10 Set External Tool Reference Point - Six-Point Method

```
/**
 * @brief Set external tool reference point - six-point method
 * @param [in] point_num Point number, range [1~4]
 * @return Error code
 */
errno_t SetExTCPPoint(int point_num);
```

2.1.6.11 Calculate External Tool Coordinate System

```
/**
 * @brief Calculate external tool coordinate system
 * @param [out] tcp_pose External tool coordinate system
 * @return Error code
 */
errno_t ComputeExTCF(DescPose *tcp_pose);
```

2.1.6.12 Set External Tool Coordinate System

Changed in version C++SDK-v2.1.2.0.

```
/**
 * @brief Set external tool coordinate system
 * @param [in] id Coordinate system ID, range [0~14]
 * @param [in] etcp Tool center point pose relative to end flange center
 * @param [in] etool To be determined
 * @return Error code
 */
errno_t SetExToolCoord(int id, DescPose *etcp, DescPose *etool);
```

2.1.6.13 Set External Tool Coordinate System List

Changed in version C++SDK-v2.1.2.0.

```
/**
 * @brief Set external tool coordinate system list
 * @param [in] id Coordinate system ID, range [0~14]
 * @param [in] etcp Tool center point pose relative to end flange center
 * @param [in] etool To be determined
 * @return Error code
 */
errno_t SetExToolList(int id, DescPose *etcp, DescPose *etool);
```

2.1.6.14 Robot External Tool Coordinate System Operation Example

```
int TestExtCoord(void)
{
   ROBOT_STATE_PKG pkg = {};
   FRRobot robot;
   robot.LoggerInit();
   robot.SetLoggerLevel(1);
   int rtn = robot.RPC("192.168.58.2");
   if (rtn != 0)
   {
      return -1;
   }
   robot.SetReConnectParam(true, 30000, 500);
```

(continues on next page)

(continued from previous page)

```
    DescPose p1Desc(-89.606, 779.517, 193.516, 178.000, 0.476, -92.484);
    JointPos p1Joint(-108.145, -50.137, 85.818, -125.599, -87.946, 74.329);
    DescPose p2Desc(-24.656, 850.384, 191.361, 177.079, -2.058, -95.355);
    JointPos p2Joint(-111.024, -41.538, 69.222, -114.913, -87.743, 74.329);
    DescPose p3Desc(-99.813, 766.661, 241.878, -176.817, 1.917, -91.604);
    JointPos p3Joint(-107.266, -56.116, 85.971, -122.560, -92.548, 74.331);
    ExaxisPos exaxisPos(0, 0, 0, 0);
    DescPose offdese(0, 0, 0, 0, 0, 0);
    DescPose posTCP[3] = { p1Desc , p2Desc , p3Desc };
    DescPose coordRtn = {};
    robot.MoveJ(&p1Joint, &p1Desc, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetExTCPPoint(1);
    robot.MoveJ(&p2Joint, &p2Desc, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetExTCPPoint(2);
    robot.MoveJ(&p3Joint, &p3Desc, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetExTCPPoint(3);
    rtn = robot.ComputeExTCF(&coordRtn);
    printf("ComputeExTCF          %d coord is %f %f %f %f %f %f \n", rtn, coordRtn.tran.x,
↪ coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry, coordRtn.rpy.rz);
    robot.SetExToolCoord(1, &coordRtn, &offdese);
    robot.SetExToolList(1, &coordRtn, &offdese);
    robot.CloseRPC();
    return 0;
}
```

2.1.6.15 Set Workpiece Reference Point - Three-Point Method

```
/**
 * @brief Set workpiece reference point - three-point method
 * @param [in] point_num Point number, range [1~3]
 * @return Error code
 */
errno_t SetWObjCoordPoint(int point_num);
```

2.1.6.16 Calculate Workpiece Coordinate System

Changed in version C++SDK-v2.1.5.0.

```
/**
 * @brief Calculate workpiece coordinate system
 * @param [in] method Calculation method: 0-origin-x-axis-z-axis, 1-origin-x-axis-xy-
↪plane
 * @param [in] refFrame Reference coordinate system
 * @param [out] wobj_pose Workpiece coordinate system
 * @return Error code
 */
errno_t ComputeWObjCoord(int method, int refFrame, DescPose *wobj_pose);
```

2.1.6.17 Set Workpiece Coordinate System

Changed in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set workpiece coordinate system
 * @param [in] id Coordinate system ID, range [0~14]
 * @param [in] coord Workpiece coordinate system pose relative to end flange center
 * @param [in] refFrame Reference coordinate system
 * @return Error code
 */
errno_t SetWObjCoord(int id, DescPose *coord, int refFrame);
```

2.1.6.18 Set Workpiece Coordinate System List

Changed in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set workpiece coordinate system list
 * @param [in] id Coordinate system ID, range [0~14]
 * @param [in] coord Workpiece coordinate system pose relative to end flange center
 * @param [in] refFrame Reference coordinate system
 * @return Error code
 */
errno_t SetWObjList(int id, DescPose *coord, int refFrame);
```

2.1.6.19 Calculate Workpiece Coordinate System from Points

New in version C++SDK-v2.1.8-3.7.8.

```
/**
 * @brief Calculate workpiece coordinate system from point data
 * @param [in] method Calculation method: 0-origin-x-axis-z-axis, 1-origin-x-axis-xy-
↪plane
 * @param [in] pos Three TCP position array
 * @param [in] refFrame Reference coordinate system
 * @param [out] coord Resulting workpiece coordinate system
 * @return Error code
 */
errno_t ComputeWObjCoordWithPoints(int method, DescPose pos[], int refFrame, DescPose&␣
↪coord);
```

2.1.6.20 Get Current Workpiece Coordinate System

```
/**
 * @brief Get current workpiece coordinate system
 * @param [in] flag 0-blocking, 1-non-blocking
 * @param [out] desc_pos Workpiece coordinate system pose
 * @return Error code
 */
errno_t GetWObjOffset(uint8_t flag, DescPose *desc_pos);
```

2.1.6.21 Robot Workpiece Coordinate System Operation Example

```
int TestWobjCoord(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    DescPose p1Desc(-89.606, 779.517, 193.516, 178.000, 0.476, -92.484);
    JointPos p1Joint(-108.145, -50.137, 85.818, -125.599, -87.946, 74.329);
    DescPose p2Desc(-24.656, 850.384, 191.361, 177.079, -2.058, -95.355);
    JointPos p2Joint(-111.024, -41.538, 69.222, -114.913, -87.743, 74.329);
    DescPose p3Desc(-99.813, 766.661, 241.878, -176.817, 1.917, -91.604);
    JointPos p3Joint(-107.266, -56.116, 85.971, -122.560, -92.548, 74.331);
    ExaxisPos exaxisPos(0, 0, 0, 0);
    DescPose offdese(0, 0, 0, 0, 0, 0);
    DescPose posTCP[3] = { p1Desc , p2Desc , p3Desc };
    DescPose coordRtn = {};
    rtn = robot.ComputeWObjCoordWithPoints(1, posTCP, 0, coordRtn);
    printf("ComputeWObjCoordWithPoints    %d  coord is %f %f %f %f %f %f \n", rtn,␣
↪coordRtn.tran.x, coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry,␣
↪coordRtn.rpy.rz);
    robot.MoveJ(&p1Joint, &p1Desc, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetWObjCoordPoint(1);
    robot.MoveJ(&p2Joint, &p2Desc, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetWObjCoordPoint(2);
    robot.MoveJ(&p3Joint, &p3Desc, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.SetWObjCoordPoint(3);
    rtn = robot.ComputeWObjCoord(1, 0, &coordRtn);
    printf("ComputeWObjCoord                  %d  coord is %f %f %f %f %f %f \n", rtn,␣
↪coordRtn.tran.x, coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry,␣
↪coordRtn.rpy.rz);
    robot.SetWObjCoord(1, &coordRtn, 0);
    robot.SetWObjList(1, &coordRtn, 0);
    DescPose getWobjDesc = {};
```

(continues on next page)

(continued from previous page)

```
    rtn = robot.GetWObjOffset(0, &getWobjDesc);
    printf("GetWObjOffset                    %d  coord is %f %f %f %f %f %f \n", rtn,␣
↪coordRtn.tran.x, coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry,␣
↪coordRtn.rpy.rz);
    robot.CloseRPC();
    return 0;
}
```

2.1.6.22 Set Global Speed

```
/**
 * @brief Set global speed
 * @param [in] vel Speed percentage, range [0~100]
 * @return Error code
 */
errno_t SetSpeed(int vel);
```

2.1.6.23 Set Robot Acceleration

```
/**
 * @brief Set robot acceleration
 * @param [in] acc Robot acceleration percentage
 * @return Error code
 */
errno_t SetOaccScale(double acc);
```

2.1.6.24 Get Robot Default Speed

```
/**
 * @brief Get robot default speed
 * @param [out] vel Speed in mm/s
 * @return Error code
 */
errno_t GetDefaultTransVel(float *vel);
```

2.1.6.25 Set End Load Weight

Changed in version C++SDK-v2.1.8-3.7.8.

```
/**
 * @brief Set end load weight
 * @param [in] loadNum Load number
 * @param [in] weight Load weight in kg
 * @return Error code
 */
errno_t SetLoadWeight(int loadNum = 0, float weight);
```

2.1.6.26 Set End Effector Payload Center of Gravity Coordinates

New in version C++SDK-v3.8.6.

```
/**
* @brief Set End Effector Payload Center of Gravity Coordinates
* @param [in] loadNum Payload number
* @param [in] coord Center of gravity coordinates, unit mm
* @return Error code
*/
errno_t SetLoadCoord(int loadNum, DescTran* coord);
```

2.1.6.27 Get Current Load Weight

```
/**
 * @brief Get current load weight
 * @param [in] flag 0-blocking, 1-non-blocking
 * @param [out] weight Load weight in kg
 * @return Error code
 */
errno_t GetTargetPayload(uint8_t flag, float *weight);
```

2.1.6.28 Get Current Load Center of Gravity

```
/**
 * @brief Get current load center of gravity
 * @param [in] flag 0-blocking, 1-non-blocking
 * @param [out] cog Load center of gravity in mm
 * @return Error code
 */
errno_t GetTargetPayloadCog(uint8_t flag, DescTran *cog);
```

2.1.6.29 Set Robot Installation Method

```
/**
 * @brief Set robot installation method
 * @param [in] install Installation method: 0-standard, 1-wall-mounted, 2-ceiling-mounted
 * @return Error code
 */
errno_t SetRobotInstallPos(uint8_t install);
```

2.1.6.30 Set Robot Installation Angle

```
/**
 * @brief Set robot installation angle (free installation)
 * @param [in] yangle Tilt angle
 * @param [in] zangle Rotation angle
 * @return Error code
 */
errno_t SetRobotInstallAngle(double yangle, double zangle);
```

2.1.6.31 Get Robot Installation Angle

```
/**
 * @brief Get robot installation angle
 * @param [out] yangle Tilt angle
 * @param [out] zangle Rotation angle
 * @return Error code
 */
errno_t GetRobotInstallAngle(float *yangle, float *zangle);
```

2.1.6.32 Set System Variable Value

```
/**
 * @brief Set system variable value
 * @param [in] id Variable number, range [1~20]
 * @param [in] value Variable value
 * @return Error code
 */
errno_t SetSysVarValue(int id, float value);
```

2.1.6.33 Get System Variable Value

```
/**
 * @brief Get system variable value
 * @param [in] id System variable number, range [1~20]
 * @param [out] value System variable value
 * @return Error code
 */
errno_t GetSysVarValue(int id, float *value);
```

2.1.6.34 Robot Common Settings Example

```
int TestLoadInstall(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    for (int i = 1; i < 100; i++)
    {
        robot.SetSpeed(i);
        robot.SetOaccScale(i);
        robot.Sleep(30);
    }
    float defaultVel = 0.0;
    robot.GetDefaultTransVel(&defaultVel);
    printf("GetDefaultTransVel is %f\n", defaultVel);
    for (int i = 1; i < 21; i++)
    {
        robot.SetSysVarValue(i, i + 0.5);
        robot.Sleep(100);
    }
    for (int i = 1; i < 21; i++)
    {
        float value = 0;
        robot.GetSysVarValue(i, &value);
        printf("sys value  %d is :%f\n", i, value);
        robot.Sleep(100);
    }
    robot.SetLoadWeight(0, 2.5);
    DescTran loadCoord = {};
    loadCoord.x = 3.0;
    loadCoord.y = 4.0;
    loadCoord.z = 5.0;
    robot.SetLoadCoord(&loadCoord);
    robot.Sleep(1000);
    float getLoad = 0.0;
    robot.GetTargetPayload(0, &getLoad);
    DescTran getLoadTran = {};
    robot.GetTargetPayloadCog(0, &getLoadTran);
    printf("get load is %f; get load cog is %f %f %f\n", getLoad, getLoadTran.x,␣
↪getLoadTran.y, getLoadTran.z);
    robot.SetRobotInstallPos(0);
    robot.SetRobotInstallAngle(15.0, 25.0);
    float anglex = 0.0;
    float angley = 0.0;
    robot.GetRobotInstallAngle(&anglex, &angley);
```

(continues on next page)

(continued from previous page)

```
    printf("GetRobotInstallAngle x:  %f;  y:  %f\n", anglex, angley);
    robot.CloseRPC();
    return 0;
}
```

2.1.6.35 Joint Friction Compensation Switch

```
/**
 * @brief Joint friction compensation switch
 * @param [in] state 0-off, 1-on
 * @return Error code
 */
errno_t FrictionCompensationOnOff(uint8_t state);
```

2.1.6.36 Set Joint Friction Compensation Coefficient - Standard Installation

```
/**
 * @brief Set joint friction compensation coefficient - standard installation
 * @param [in] coeff Six joint compensation coefficients, range [0~1]
 * @return Error code
 */
errno_t SetFrictionValue_level(float coeff[6]);
```

2.1.6.37 Set Joint Friction Compensation Coefficient - Wall Installation

```
/**
 * @brief Set joint friction compensation coefficient - wall installation
 * @param [in] coeff Six joint compensation coefficients, range [0~1]
 * @return Error code
 */
errno_t SetFrictionValue_wall(float coeff[6]);
```

2.1.6.38 Set Joint Friction Compensation Coefficient - Ceiling Installation

```
/**
 * @brief Set joint friction compensation coefficient - ceiling installation
 * @param [in] coeff Six joint compensation coefficients, range [0~1]
 * @return Error code
 */
errno_t SetFrictionValue_ceiling(float coeff[6]);
```

2.1.6.39 Set Joint Friction Compensation Coefficient - Free Installation

```
/**
 * @brief Set joint friction compensation coefficient - free installation
 * @param [in] coeff Six joint compensation coefficients, range [0~1]
 * @return Error code
 */
errno_t SetFrictionValue_freedom(float coeff[6]);
```

2.1.6.40 Robot Joint Friction Compensation Example

```
int TestFriction(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;

    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    float lcoeff[6] = { 0.9,0.9,0.9,0.9,0.9,0.9 };
    float wcoeff[6] = { 0.4,0.4,0.4,0.4,0.4,0.4 };
    float ccoeff[6] = { 0.6,0.6,0.6,0.6,0.6,0.6 };
    float fcoeff[6] = { 0.5,0.5,0.5,0.5,0.5,0.5 };
    rtn = robot.FrictionCompensationOnOff(1);
    printf("FrictionCompensationOnOff rtn is %d\n", rtn);
    rtn = robot.SetFrictionValue_level(lcoeff);
    printf("SetFrictionValue_level rtn is %d\n", rtn);
    rtn = robot.SetFrictionValue_wall(wcoeff);
    printf("SetFrictionValue_wall rtn is %d\n", rtn);
    rtn = robot.SetFrictionValue_ceiling(ccoeff);
    printf("SetFrictionValue_ceiling rtn is %d\n", rtn);
    rtn = robot.SetFrictionValue_freedom(fcoeff);
    printf("SetFrictionValue_freedom rtn is %d\n", rtn);
    robot.CloseRPC();
    return 0;
}
```

2.1.6.41 Query Robot Error Code

```
/**
 * @brief Query robot error code
 * @param [out] maincode Main error code
 * @param [out] subcode Sub error code
 * @return Error code
 */
errno_t GetRobotErrorCode(int *maincode, int *subcode);
```

2.1.6.42 Error State Clear

```
/**
 * @brief Error state clear
 * @return Error code
 */
errno_t ResetAllError();
```

2.1.6.43 Robot Fault State Acquisition and Error Clear Example

```
int TestGetError(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    int maincode, subcode;
    robot.GetRobotErrorCode(&maincode, &subcode);
    printf("robot maincode is %d; subcode is %d\n", maincode, subcode);
    robot.ResetAllError();
    robot.Sleep(1000);
    robot.GetRobotErrorCode(&maincode, &subcode);
    printf("robot maincode is %d; subcode is %d\n", maincode, subcode);
    robot.CloseRPC();
    return 0;
}
```

2.1.6.44 Set wide voltage control box temperature and fan current monitoring parameters

```
/**
*@brief setting wide voltage control box temperature and fan current monitoring␣
↪parameters
*@param [ in ] enable 0-do not enable monitoring; 1-enable monitoring
*@param [ in ] period (s) , range 1-100
*@return error code
*/
errno_t SetWideBoxTempFanMonitorParam(int enable, int period);
```

2.1.6.45 Obtain wide voltage control box temperature and fan current monitoring parameters

```
/**
* @brief Obtain wide voltage control box temperature and fan current monitoring␣
↪parameters
*@param [ in ] enable 0-do not enable monitoring; 1-enable monitoring
*@param [ in ] period (s) , range 1-100
*@return error code
*/
errno_t GetWideBoxTempFanMonitorParam(int &enable, int &period);
```

2.1.6.46 Wide voltage control box temperature and fan current state acquisition code example

```
int TestWideVoltageCtrlBoxtemp(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    printf("robot rpc rtn is %d\n", rtn);
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    robot.SetWideBoxTempFanMonitorParam(1, 2);
    int enable = 0;
    int period = 0;
    robot.GetWideBoxTempFanMonitorParam(enable, period);
    printf("GetWideBoxTempFanMonitorParam enable is %d   period is %d\n", enable,␣
↪period);
    for (int i = 0; i < 100; i++)
    {
        robot.GetRobotRealTimeState(&pkg);
        printf("robot ctrl box temp is %f,  fan current is %d\n", pkg.
↪wideVoltageCtrlBoxTemp, pkg.wideVoltageCtrlBoxFanCurrent);
        robot.Sleep(100);
    }
```

(continues on next page)

(continued from previous page)

```
    rtn = robot.SetWideBoxTempFanMonitorParam(0, 2);
    printf("SetWideBoxTempFanMonitorParam rtn is %d\n", rtn);
    enable = 0;
    period = 0;
    robot.GetWideBoxTempFanMonitorParam(enable, period);
    printf("GetWideBoxTempFanMonitorParam enable is %d   period is %d\n", enable, period);
    for (int i = 0; i < 100; i++)
    {
        robot.GetRobotRealTimeState(&pkg);
        printf("robot ctrl box temp is %f,  fan current is %d\n", pkg.wideVoltageCtrlBoxTemp, pkg.wideVoltageCtrlBoxFanCurrent);
        robot.Sleep(100);
    }
    robot.CloseRPC();
    robot.Sleep(2000);
    return 0;
}
```

2.1.6.47 Calculate Focus Calibration Results

```
/**
* @brief Calculate focus calibration results
* @param [in] pointNum Number of calibration points
* @param [out] resultPos Calibration result XYZ
* @param [out] accuracy Calibration accuracy error
* @return Error code
*/
errno_t ComputeFocusCalib(int pointNum, DescTran& resultPos, float& accuracy);
```

2.1.6.48 Set Focus Position

```
/**
* @brief Set focus position
* @param [in] pos Focus position XYZ
* @return Error code
*/
errno_t SetFocusPosition(DescTran pos);
```

2.1.6.49 Enable Focus Following

```
/**
* @brief Enable focus following
* @param [in] kp Proportional parameter (default: 50.0)
* @param [in] kpredict Feedforward parameter (default: 19.0)
* @param [in] aMax Maximum angular acceleration limit (default: 1440°/s²)
* @param [in] vMax Maximum angular velocity limit (default: 180°/s)
* @param [in] type X-axis locking mode (0-reference input vector; 1-horizontal; 2-vertical)
* @return Error code
*/
errno_t FocusStart(double kp, double kpredict, double aMax, double vMax, int type);
```

2.1.6.50 Disable Focus Following

```
/**
* @brief Disable focus following
* @return Error code
*/
errno_t FocusEnd();
```

2.1.6.51 Robot Focus Following Code Example

```
int TestFocus()
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  DescPose p1Desc(186.331, 487.913, 209.850, 149.030, 0.688, -114.347);
  JointPos p1Joint(-127.876, -75.341, 115.417, -122.741, -59.820, 74.300);
  DescPose p2Desc(69.721, 535.073, 202.882, -144.406, -14.775, -89.012);
  JointPos p2Joint(-101.780, -69.828, 110.917, -125.740, -127.841, 74.300);
  DescPose p3Desc(146.861, 578.426, 205.598, 175.997, -36.178, -93.437);
  JointPos p3Joint(-112.851, -60.191, 86.566, -80.676, -97.463, 74.300);
  DescPose p4Desc(136.284, 509.876, 225.613, 178.987, 1.372, -100.696);
  JointPos p4Joint(-116.397, -76.281, 113.845, -128.611, -88.654, 74.299);
  DescPose p5Desc(138.395, 505.972, 298.016, 179.134, 2.147, -101.110);
  JointPos p5Joint(-116.814, -82.333, 109.162, -118.662, -88.585, 74.302);
  DescPose p6Desc(105.553, 454.325, 232.017, -179.426, 0.444, -99.952);
  JointPos p6Joint(-115.649, -84.367, 122.447, -128.663, -90.432, 74.303);
  ExaxisPos exaxisPos(0, 0, 0, 0);
  DescPose offdese(0, 0, 100, 0, 0, 0);
```

(continues on next page)

(continued from previous page)

```
  robot.MoveJ(&p1Joint, &p1Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
  robot.SetTcp4RefPoint(1);
  robot.MoveJ(&p2Joint, &p2Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
  robot.SetTcp4RefPoint(2);
  robot.MoveJ(&p3Joint, &p3Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
  robot.SetTcp4RefPoint(3);
  robot.MoveJ(&p4Joint, &p4Desc, 0, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
  robot.SetTcp4RefPoint(4);
  DescPose coordRtn = {};
  rtn = robot.ComputeTcp4(&coordRtn);
  printf("4 Point ComputeTool    %d coord is %f %f %f %f %f %f \n", rtn, coordRtn.tran.x,
↪ coordRtn.tran.y, coordRtn.tran.z, coordRtn.rpy.rx, coordRtn.rpy.ry, coordRtn.rpy.rz);
  robot.SetToolCoord(1, &coordRtn, 0, 0, 1, 0);
  robot.GetForwardKin(&p1Joint, &p1Desc);
  robot.GetForwardKin(&p2Joint, &p2Desc);
  robot.GetForwardKin(&p3Joint, &p3Desc);
  robot.SetFocusCalibPoint(1, p1Desc);
  robot.SetFocusCalibPoint(2, p2Desc);
  robot.SetFocusCalibPoint(3, p3Desc);
  DescTran resultPos = {};
  float accuracy = 0.0;
  rtn = robot.ComputeFocusCalib(3, resultPos, accuracy);
  printf("ComputeFocusCalib coord is %d %f %f %f accuracy is %f\n", rtn, resultPos.x,␣
↪resultPos.y, resultPos.z, accuracy);
  rtn = robot.SetFocusPosition(resultPos);
  robot.GetForwardKin(&p5Joint, &p5Desc);
  robot.GetForwardKin(&p6Joint, &p6Desc);
  robot.MoveL(&p5Joint, &p5Desc, 1, 0, 10, 100, 100, -1, 0, &exaxisPos, 0, 1, &offdese);
  robot.MoveL(&p6Joint, &p6Desc, 1, 0, 10, 100, 100, -1, 0, &exaxisPos, 0, 1, &offdese);
  robot.FocusStart(50, 19, 710, 90, 0);
  robot.MoveL(&p5Joint, &p5Desc, 1, 0, 10, 100, 100, -1, 0, &exaxisPos, 0, 1, &offdese);
  robot.MoveL(&p6Joint, &p6Desc, 1, 0, 10, 100, 100, -1, 0, &exaxisPos, 0, 1, &offdese);
  robot.FocusEnd();
  robot.CloseRPC();
  return 0;
}
```

2.1.6.52 Open the joint torque sensor sensitivity calibration function

```
/**
* @brief Open the joint torque sensor sensitivity calibration function
* @param [ in ] status 0-off; 1-on
* @return  error code
*/
errno_t JointSensitivityEnable(int status);
```

2.1.6.53 Data acquisition of joint torque sensor sensitivity

```
/**
* @brief Data acquisition of joint torque sensor sensitivity
* @return error code
*/
errno_t JointSensitivityCollect();
```

2.1.6.54 Obtain joint torque sensor sensitivity calibration results

```
/**
* @brief Get joint torque sensor sensitivity calibration results
* @param [out] calibResult j1~j6 joint sensitivity [0-1]
* @param [out] linearity j1~j6 joint linearity [0-1]
* @return Error code
*/
errno_t JointSensitivityCalibration(double calibResult[6], double linearity[6]);
```

2.1.6.55 Get Joint Torque Sensor Hysteresis Error

```
/**
* @brief Get joint torque sensor hysteresis error
* @param [out] hysteresisError j1~j6 joint hysteresis error
* @return Error code
*/
errno_t JointHysteresisError(double hysteresisError[6]);
```

2.1.6.56 Get Joint Torque Sensor Repeatability

```
/**
* @brief Get joint torque sensor repeatability
* @param [out] repeatability j1~j6 joint torque sensor repeatability
* @return Error code
*/
errno_t JointRepeatability(double repeatability[6]);
```

2.1.6.57 Set Joint Force Sensor Parameters

```
/**
* @brief Set joint force sensor parameters
* @param [in] M J1-J6 mass coefficient [0.001 ~ 10]
* @param [in] B J1-J6 damping coefficient [0.001 ~ 10]
* @param [in] K J1-J6 stiffness coefficient [0.001 ~ 10]
* @param [in] threshold Force control threshold, Nm
* @param [in] sensitivity Sensitivity, Nm/V, [0 ~ 10]
* @param [in] setZeroFlag Function enable flag; 0-Off; 1-On; 2-Record zero point at␣
↪position 1; 3-Record zero point at position 2
```

(continues on next page)

(continued from previous page)

```
* @return Error code
*/
errno_t SetAdmittanceParams(double M[6], double B[6], double K[6], double threshold[6],␣
↪double sensitivity[6], int setZeroFlag);
```

2.1.6.58 An example of joint torque sensor sensitivity auto-calibration code

```
int TestSensitivityCalib()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    robot.SetReConnectParam(true, 30000, 500);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return 0;
    }
    rtn = robot.JointSensitivityEnable(0);
    rtn = robot.JointSensitivityEnable(1);
    printf("JointSensitivityEnable rtn is %d\n", rtn);
    JointPos curJPos = {};
    robot.GetActualJointPosDegree(0, &curJPos);
    ExaxisPos epos = { 0,0,0,0 };
    DescPose offset_pos = { 0,0,0,0,0,0 };
    JointPos jointPos1 = { curJPos.jPos[0], 0, 0, -90, 0.02, curJPos.jPos[5] };
    DescPose descPos1 = {};
    robot.GetForwardKin(&jointPos1, &descPos1);
    robot.MoveJ(&jointPos1, &descPos1, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(200);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 1 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos2 = { curJPos.jPos[0], -30, 0, -90, 0.02, curJPos.jPos[5] };
    DescPose descPos2 = {};
    robot.GetForwardKin(&jointPos2, &descPos2);
    robot.MoveJ(&jointPos2, &descPos2, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 2 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos3 = { curJPos.jPos[0], -60, 0, -90, 0.02, curJPos.jPos[5] };
    DescPose descPos3 = {};
    robot.GetForwardKin(&jointPos3, &descPos3);
    robot.MoveJ(&jointPos3, &descPos3, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 3 rtn is %d\n", rtn);
    robot.Sleep(100);
```

(continues on next page)

(continued from previous page)

```
    JointPos jointPos4 = { curJPos.jPos[0], -90, 0, -90, 0.02, curJPos.jPos[5] };
    DescPose descPos4 = {};
    robot.GetForwardKin(&jointPos4, &descPos4);
    robot.MoveJ(&jointPos4, &descPos4, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 4 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos5 = { curJPos.jPos[0], -120, 0, -90, 0.02, curJPos.jPos[5] };
    DescPose descPos5 = {};
    robot.GetForwardKin(&jointPos5, &descPos5);
    robot.MoveJ(&jointPos5, &descPos5, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 5 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos6 = { curJPos.jPos[0], -150, 0, -90, 0.02, curJPos.jPos[5] };
    DescPose descPos6 = {};
    robot.GetForwardKin(&jointPos6, &descPos6);
    robot.MoveJ(&jointPos6, &descPos6, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 6 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos7 = { curJPos.jPos[0], -180, 0, -90, 0.02, curJPos.jPos[5] };
    DescPose descPos7 = {};
    robot.GetForwardKin(&jointPos7, &descPos7);
    robot.MoveJ(&jointPos7, &descPos7, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 7 rtn is %d\n", rtn);
    robot.Sleep(100);
    //-------------------
    robot.MoveJ(&jointPos6, &descPos6, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 8 rtn is %d\n", rtn);
    robot.Sleep(100);
    robot.MoveJ(&jointPos5, &descPos5, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 9 rtn is %d\n", rtn);
    robot.Sleep(100);
    robot.MoveJ(&jointPos4, &descPos4, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 10 rtn is %d\n", rtn);
    robot.Sleep(100);
    robot.MoveJ(&jointPos3, &descPos3, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 11 rtn is %d\n", rtn);
```

(continues on next page)

(continued from previous page)

```
    robot.Sleep(100);
    robot.MoveJ(&jointPos2, &descPos2, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 12 rtn is %d\n", rtn);
    robot.Sleep(100);
    robot.MoveJ(&jointPos1, &descPos1, 0, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
    robot.Sleep(200);
    rtn = robot.JointSensitivityCollect();
    printf("JointSensitivityCollect 13 rtn is %d\n", rtn);
    robot.Sleep(100);
    double calibResult[6] = { 0.0 };
    double linearity[6] = { 0.0 };
    rtn = robot.JointSensitivityCalibration(calibResult, linearity);
    printf("JointSensitivityCalibration rtn is %d\n", rtn);
    rtn = robot.JointSensitivityEnable(0);
    printf("JointSensitivityEnable rtn is %d\n", rtn);
    printf("jointSensor Calib result is %f %f %f %f %f %f\njointSensor linearity is %f
↪%f %f %f %f %f\n",
        calibResult[0], calibResult[1], calibResult[2],
        calibResult[3], calibResult[4], calibResult[5],
        linearity[0], linearity[1], linearity[2],
        linearity[3], linearity[4], linearity[5]);
    double hysteresisError[6] = { 0.0 };
    rtn = robot.JointHysteresisError(hysteresisError);
    printf("JointHysteresisError result is %f %f %f %f %f %f\n",
        hysteresisError[0], hysteresisError[1], hysteresisError[2],
        hysteresisError[3], hysteresisError[4], hysteresisError[5]);
    double repeatability[6] = { 0.0 };
    rtn = robot.JointRepeatability(repeatability);
    printf("JointRepeatability result is %f %f %f %f %f %f\n",
        repeatability[0], repeatability[1], repeatability[2],
        repeatability[3], repeatability[4], repeatability[5]);
    double M[6] = { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    double B[6] = { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    double K[6] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
    double threshold[6] = { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    int setZeroFlag = 1;
    rtn = robot.SetAdmittanceParams(M, B, K, threshold, calibResult, setZeroFlag);
    printf("SetAdmittanceParams rtn is %d\n", rtn);
    robot.CloseRPC();
}
```

2.1.6.59 Gets 8 slave error frames

```
/**
* @brief  gets 8 slave error frames
* @param [ out ] in recverr
* @param [ out -RSB- inCRCErr enCRC CRC error frames
* @param [ out ] in forwarding error frames
* @param [ out ] inlinker R enter link error frames
* @param [ out ] out recverr output received error frames
* @param [ out ] outCRCErr output CRC error frames
* @param [ out ] outtransmitterr output forwarding error frames
* @param [ out ] outLinkErr output link error frames
* @return  error code
*/
errno_t GetSlavePortErrCounter(int inRecvErr[8], int inCRCErr[8], int inTransmitErr[8],␣
↪int inLinkErr[8], int outRecvErr[8], int outCRCErr[8], int outTransmitErr[8], int␣
↪outLinkErr[8]);
```

2.1.6.60 Error frame zeroing at slave port

```
/**
* @brief Error frame zeroing at slave port
* @param [in] slaveID Slave station number 0 ~ 7
* @return error code
*/
errno_t SlavePortErrCounterClear(int slaveID);
```

2.1.6.61 Gets an example of a slave port error frame code

```
int TestSlavePortErr()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return 0;
    }
    robot.SetReConnectParam(true, 30000, 500);
    int inRecvErr[8] = {0.0};
    int inCRCErr[8] = { 0.0 };
    int inTransmitErr[8] = { 0.0 };
    int inLinkErr[8] = { 0.0 };
    int outRecvErr[8] = { 0.0 };
    int outCRCErr[8] = { 0.0 };
    int outTransmitErr[8] = { 0.0 };
    int outLinkErr[8] = { 0.0 };
    robot.GetSlavePortErrCounter(inRecvErr,  inCRCErr, inTransmitErr, inLinkErr,
```

(continues on next page)

(continued from previous page)

```
        outRecvErr, outCRCErr, outTransmitErr, outLinkErr);
    for (int i = 0; i < 8; i++)
    {
        if (inRecvErr[i] != 0)
        {
            printf("inRecvErr %d is %d\n", i, inRecvErr[i]);
        }
        if (inCRCErr[i] != 0)
        {
            printf("inRecvErr %d is %d\n", i, inCRCErr[i]);
        }
        if (inTransmitErr[i] != 0)
        {
            printf("inRecvErr %d is %d\n", i, inTransmitErr[i]);
        }
        if (inLinkErr[i] != 0)
        {
            printf("inRecvErr %d is %d\n", i, inLinkErr[i]);
        }
        if (outRecvErr[i] != 0)
        {
            printf("outRecvErr %d is %d\n", i, outRecvErr[i]);
        }
        if (outCRCErr[i] != 0)
        {
            printf("outCRCErr %d is %d\n", i, outCRCErr[i]);
        }
        if (outTransmitErr[i] != 0)
        {
            printf("outTransmitErr %d is %d\n", i, outTransmitErr[i]);
        }
        if (outLinkErr[i] != 0)
        {
            printf("outLinkErr %d is %d\n", i, outLinkErr[i]);
        }
    }
    printf("others has no err!\n");
    for (int i = 0; i < 8; i++)
    {
        robot.SlavePortErrCounterClear(i);
    }
    robot.CloseRPC();
    return 0;
}
```

2.1.6.62 Set feed-forward coefficients for each axis

```
/**
* @brief Set feed-forward coefficients for each axis
* @param [in] radio Feed-forward coefficient of velocity of each axis
* @return error code
*/
errno_t SetVelFeedForwardRatio(double radio[6]);
```

2.1.6.63 Obtain the feed-forward coefficients of each axis

```
/**
* @brief Obtain the feed-forward coefficients of each axis
* @param [out] radio Feed-forward coefficient of velocity of each axis
* @return error code
*/
errno_t GetVelFeedForwardRatio(double radio[6]);
```

2.1.6.64 Example of robot velocity feed-forward coefficient code

```
int TestVelFeedForwardRatio()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return 0;
    }
    robot.SetReConnectParam(true, 30000, 500);
    double setRadio[6] = { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    robot.SetVelFeedForwardRatio(setRadio);
    double getRadio[6] = { 0.0 };
    robot.GetVelFeedForwardRatio(getRadio);
    printf(" %f %f %f %f %f %f\n", getRadio[0], getRadio[1], getRadio[2], getRadio[3],␣
↪getRadio[4], getRadio[5]);
    robot.CloseRPC();
    return 0;
}
```

2.1.6.65 Photoelectric Sensor TCP Calibration - Compute Tool RPY

```
/**
* @brief Photoelectric Sensor TCP Calibration - Compute Tool RPY
* @param [in] Btool Robot Cartesian position
* @param [in] Etool Current tool coordinate values
* @param [in] sensor Current sensor coordinate values (not yet available)
* @param [in] radius Circular motion radius in mm (not yet available)
* @param [in] dz Movement distance along the negative Z-axis of the base coordinate system; when dz = 10000, the function directly returns tool RPY
* @param [out] TCPRPY Tool RPY values
* @return Error code
*/
errno_t TCPComputeRPY(DescPose Btool, DescPose Etool, DescPose sensor, double radius, double dz, Rpy& TCPRPY);
```

2.1.6.66 Photoelectric Sensor TCP Calibration - Compute Tool XYZ

```
/**
* @brief Photoelectric Sensor TCP Calibration - Compute Tool XYZ
* @param [in] select 0-Compute tool TCP; 1-Compute sensor origin; 2-Compute sensor orientation; 3-Directly return tool TCP; 4-Record current workpiece coordinate system and tool coordinate system
* @param [in] originDirection 0-X direction; 1-Y direction; 2-Z direction
* @param [in] pos1 Robot Cartesian position 1
* @param [in] pos2 Robot Cartesian position 2
* @param [in] pos3 Robot Cartesian position 3
* @param [in] pos4 Robot Cartesian position 4
* @param [out] TCP Tool XYZ values
* @return Error code
*/
errno_t TCPComputeXYZ(int select, double originDirection, DescTran pos1, DescTran pos2, DescTran pos3, DescTran pos4, DescTran& TCP);
```

2.1.6.67 Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position

```
/**
* @brief Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position
* @return Error code
*/
errno_t TCPRecordFlangePosStart();
```

2.1.6.68 Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position

```
/**
* @brief Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position
* @return Error code
*/
errno_t TCPRecordFlangePosEnd();
```

2.1.6.69 Photoelectric Sensor TCP Calibration - Get Tool Center Point Position

```
/**
* @brief Photoelectric Sensor TCP Calibration - Get Tool Center Point Position
* @param [out] TCP Tool center point position (x, y, z)
* @return Error code
*/
errno_t TCPGetRecordFlangePos(DescTran& TCP);
```

2.1.6.70 Photoelectric Sensor TCP Calibration

```
/**
* @brief Photoelectric Sensor TCP Calibration
* @param [in] luaPath Automatic calibration Lua program path: For QX version robots - "/
↪fruser/FR_CalibrateTheToolTcp.lua"; For LA version robots - "/usr/local/etc/controller/
↪lua/FR_CalibrateTheToolTcp.lua"
* @param [in] offsetX Teaching point offset (x, y, z) in mm
* @param [out] TCP Calibrated tool coordinate system (x, y, z, rx, ry, rz)
* @return Error code
*/
errno_t PhotoelectricSensorTCPCalibration(std::string luaPath, DescTran offset, DescPose&
↪ TCP);
```

2.1.6.71 Photoelectric Sensor TCP Calibration Code Example

```
int TestPhotoelectricSensorTCPCalib(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return 0;
    }
    robot.SetReConnectParam(true, 30000, 500);
    DescTran offset = { 10.0, 10.0, 3.0 };
    DescPose TCP = {};
    rtn = robot.PhotoelectricSensorTCPCalibration("/fruser/FR_CalibrateTheToolTcp.lua",␣
```

(continues on next page)

(continued from previous page)

```
↪offset, TCP);
    printf("PhotoelectricSensorTCPCalibration rtn is  %d %f %f %f %f %f %f \n", rtn, TCP.
↪tran.x, TCP.tran.y, TCP.tran.z, TCP.rpy.rx, TCP.rpy.ry, TCP.rpy.rz);
    robot.CloseRPC();
    robot.Sleep(9999999);
    return 0;
}
```

2.1.7 Security settings

2.1.7.1 Set Collision Level

```
/**
* @brief Set collision level
* @param  [in]  mode  0-Level, 1-Percentage
* @param  [in]  level Collision threshold, level range [], percentage range [0~1]
* @param  [in]  config 0-Don't update config file, 1-Update config file
* @return  Error code
*/
errno_t  SetAnticollision(int mode, float level[6], int config);
```

2.1.7.2 Set Post-Collision Strategy

Changed in version C++SDK-v2.1.5.0.

```
/**
     * @brief  Set post-collision strategy
     * @param  [in] strategy  0-Error pause; 1-Continue running; 2-Error stop; 3-Gravity␣
↪torque mode; 4-Oscillation response mode; 5-Collision rebound mode
     * @param  [in] safeTime  Safe stop time [1000 - 2000]ms
     * @param  [in] safeDistance  Safe stop distance [1-150]mm
     * @param  [in] safetyMargin  J1-J6 safety factor [1-10]
     * @return  Error code
     */
    errno_t SetCollisionStrategy(int strategy, int safeTime, int safeDistance, int␣
↪safetyMargin[]);
```

2.1.7.3 Custom Collision Detection Threshold Start

New in version C++SDK-v2.2.0-3.8.0.

```
    /**
     * @brief  Start custom collision detection threshold, set joint and TCP collision␣
↪detection thresholds
     * @param  [in] flag 1-Joint detection only; 2-TCP detection only; 3-Both joint and␣
↪TCP detection
     * @param  [in] jointDetectionThreshould Joint collision detection threshold j1-j6
     * @param  [in] tcpDetectionThreshould TCP collision detection threshold, xyzabc
```

(continues on next page)

(continued from previous page)

```
     * @param  [in] block 0-Non-blocking; 1-Blocking
     * @return  Error code
     */
    errno_t CustomCollisionDetectionStart(int flag, double jointDetectionThreshould[6],␣
↪double tcpDetectionThreshould[6], int block);
```

2.1.7.4 Custom Collision Detection Threshold End

New in version C++SDK-v2.2.0-3.8.0.

```
    /**
     * @brief  Close custom collision detection threshold function
     * @return  Error code
     */
    errno_t CustomCollisionDetectionEnd();
```

2.1.7.5 Robot Collision Level Setting Code Example

```
int TestCollision(void)
 {
     ROBOT_STATE_PKG pkg = {};
     FRRobot robot;
     robot.LoggerInit();
     robot.SetLoggerLevel(1);
     int rtn = robot.RPC("192.168.58.2");
     if (rtn != 0)
     {
         return -1;
     }
     robot.SetReConnectParam(true, 30000, 500);
     int mode = 0;
     int config = 1;
     float level1[6] = { 1.0,2.0,3.0,4.0,5.0,6.0 };
     float level2[6] = { 50.0,20.0,30.0,40.0,50.0,60.0 };
     rtn = robot.SetAnticollision(mode, level1, config);
     printf("SetAnticollision mode 0 rtn is %d\n", rtn);
     mode = 1;
     rtn = robot.SetAnticollision(mode, level2, config);
     printf("SetAnticollision mode 1 rtn is %d\n", rtn);
     JointPos p1Joint(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
     JointPos p2Joint(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
     DescPose p1Desc(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
     DescPose p2Desc(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
     ExaxisPos exaxisPos(0.0, 0.0, 0.0, 0.0);
     DescPose offdese(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
     robot.MoveL(&p2Joint, &p2Desc, 0, 0, 100, 100, 100, 2, &exaxisPos, 0, 0, &offdese);
     robot.ResetAllError();
     int safety[6] = { 5,5,5,5,5,5 };
     rtn = robot.SetCollisionStrategy(3, 1000, 150, 250, safety);
```

(continues on next page)

(continued from previous page)

```
    printf("SetCollisionStrategy rtn is %d\n", rtn);
    double jointDetectionThreshould[6] = { 0.1, 0.1, 0.1, 0.1, 0.1, 0.1 };
    double tcpDetectionThreshould[6] = { 60,60,60,60,60,60 };
    rtn = robot.CustomCollisionDetectionStart(3, jointDetectionThreshould,␣
↪tcpDetectionThreshould, 0);
    cout << "CustomCollisionDetectionStart rtn is " << rtn << endl;
    robot.MoveL(&p1Joint, &p1Desc, 0, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &offdese);
    robot.MoveL(&p2Joint, &p2Desc, 0, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &offdese);
    rtn = robot.CustomCollisionDetectionEnd();
    cout << "CustomCollisionDetectionEnd rtn is " << rtn << endl;
    robot.CloseRPC();
    return 0;
}
```

2.1.7.6 Set Positive Limit

```
/**
* @brief  Set positive limit
* @param  [in] limit Six joint positions, unit deg
* @return  Error code
*/
errno_t  SetLimitPositive(float limit[6]);
```

2.1.7.7 Set Negative Limit

```
/**
* @brief  Set negative limit
* @param  [in] limit Six joint positions, unit deg
* @return  Error code
*/
errno_t  SetLimitNegative(float limit[6]);
```

2.1.7.8 Get Joint Soft Limit Angles

```
/**
* @brief  Get joint soft limit angles
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] negative  Negative limit angle, unit deg
* @param  [out] positive  Positive limit angle, unit deg
* @return  Error code
*/
errno_t  GetJointSoftLimitDeg(uint8_t flag, float negative[6], float positive[6]);
```

2.1.7.9 Robot Limit Setting Code Example

```
int TestLimit(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  float plimit[6] = { 170.0,80.0,150.0,80.0,170.0,160.0 };
  robot.SetLimitPositive(plimit);
  float nlimit[6] = { -170.0,-260.0,-150.0,-260.0,-170.0,-160.0 };
  robot.SetLimitNegative(nlimit);
  float neg_deg[6] = { 0.0 }, pos_deg[6] = { 0.0 };
  robot.GetJointSoftLimitDeg(0, neg_deg, pos_deg);
  printf("neg limit deg:%f,%f,%f,%f,%f,%f\n", neg_deg[0], neg_deg[1], neg_deg[2], neg_
↪deg[3], neg_deg[4], neg_deg[5]);
  printf("pos limit deg:%f,%f,%f,%f,%f,%f\n", pos_deg[0], pos_deg[1], pos_deg[2], pos_
↪deg[3], pos_deg[4], pos_deg[5]);
  robot.CloseRPC();
  return 0;
}
```

2.1.7.10 Set Robot Collision Detection Method

```
/**
* @brief Set robot collision detection method
* @param [in] method Collision detection method: 0-Current mode; 1-Dual encoder; 2-Both␣
↪current and dual encoder
* @param [in] thresholdMode Collision level threshold method: 0-Fixed collision level␣
↪threshold; 1-Custom collision detection threshold
* @return  Error code
*/
errno_t SetCollisionDetectionMethod(int method, int thresholdMode = 0);
```

2.1.7.11 Set Static Collision Detection On/Off

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set static collision detection on/off
 * @param [in] status 0-Off; 1-On
 * @return Error code
 */
errno_t SetStaticCollisionOnOff(int status);
```

2.1.7.12 Robot Collision Detection Method Code Example

New in version C++SDK-v2.1.5.0.

```
int TestCollisionMethod(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  rtn = robot.SetCollisionDetectionMethod(0, 0);
  printf("SetCollisionDetectionMethod rtn is %d\n", rtn);
  rtn = robot.SetStaticCollisionOnOff(1);
  printf("SetStaticCollisionOnOff On rtn is %d\n", rtn);
  rtn = robot.Sleep(5000);
  rtn = robot.SetStaticCollisionOnOff(0);
  printf("SetStaticCollisionOnOff Off rtn is %d\n", rtn);
  robot.CloseRPC();
  return 0;
}
```

2.1.7.13 Joint Torque Power Detection

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Joint torque power detection
 * @param [in] status 0-Off; 1-On
 * @param [in] power Set maximum power (W);
 * @return Error code
 */
errno_t SetPowerLimit(int status, double power);
```

2.1.7.14 Joint Torque Power Detection Code Example

```
int TestPowerLimit(void)
{
   ROBOT_STATE_PKG pkg = {};
   FRRobot robot;
   robot.LoggerInit();
   robot.SetLoggerLevel(1);
   int rtn = robot.RPC("192.168.58.2");
   if (rtn != 0)
   {
      return -1;
```

(continues on next page)

(continued from previous page)

```
    }
    robot.SetReConnectParam(true, 30000, 500);
    robot.DragTeachSwitch(1);
    robot.SetPowerLimit(1, 200);
    float torques[] = { 0, 0, 0, 0, 0, 0 };
    robot.GetJointTorques(1, torques);
    int count = 100;
    robot.ServoJTStart();
    int error = 0;
    while (count > 0)
    {
        error = robot.ServoJT(torques, 0.001);
        count = count - 1;
        robot.Sleep(1);
    }
    error = robot.ServoJTEnd();
    robot.DragTeachSwitch(0);
    robot.CloseRPC();
    return 0;
}
```

2.1.8 Status query

2.1.8.1 Get Current Joint Positions (Degrees)

```
/**
* @brief  Get current joint positions (degrees)
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] jPos Six joint positions, unit deg
* @return  Error code
*/
errno_t  GetActualJointPosDegree(uint8_t flag, JointPos *jPos);
```

2.1.8.2 Get Joint Feedback Speed

```
/**
 * @brief  Get joint feedback speed - deg/s
 * @param  [in] flag 0-Blocking, 1-Non-blocking
 * @param  [out] speed Six joint speeds
 * @return  Error code
 */
errno_t  GetActualJointSpeedsDegree(uint8_t flag, float speed[6]);
```

2.1.8.3 Get Joint Feedback Acceleration

```
/**
 * @brief  Get joint feedback acceleration - deg/s^2
 * @param  [in] flag 0-Blocking, 1-Non-blocking
 * @param  [out] acc Six joint accelerations
 * @return  Error code
 */
errno_t  GetActualJointAccDegree(uint8_t flag, float acc[6]);
```

2.1.8.4 Get TCP Command Composite Speed

```
/**
 * @brief  Get TCP command composite speed
 * @param  [in] flag 0-Blocking, 1-Non-blocking
 * @param  [out] tcp_speed Linear speed
 * @param  [out] ori_speed Orientation speed
 * @return  Error code
 */
errno_t  GetTargetTCPCompositeSpeed(uint8_t flag, float *tcp_speed, float *ori_speed);
```

2.1.8.5 Get TCP Feedback Composite Speed

```
/**
 * @brief  Get TCP feedback composite speed
 * @param  [in] flag 0-Blocking, 1-Non-blocking
 * @param  [out] tcp_speed Linear speed
 * @param  [out] ori_speed Orientation speed
 * @return  Error code
 */
errno_t  GetActualTCPCompositeSpeed(uint8_t flag, float *tcp_speed, float *ori_speed);
```

2.1.8.6 Get TCP Command Speed

```
/**
 * @brief  Get TCP command speed
 * @param  [in] flag 0-Blocking, 1-Non-blocking
 * @param  [out] speed [x,y,z,rx,ry,rz] speed
 * @return  Error code
 */
errno_t  GetTargetTCPSpeed(uint8_t flag, float speed[6]);
```

2.1.8.7 Get TCP Feedback Speed

```
/**
 * @brief  Get TCP feedback speed
 * @param  [in] flag 0-Blocking, 1-Non-blocking
 * @param  [out] speed [x,y,z,rx,ry,rz] speed
 * @return  Error code
 */
errno_t  GetActualTCPSpeed(uint8_t flag, float speed[6]);
```

2.1.8.8 Get Current Tool Pose

```
/**
* @brief  Get current tool pose
* @param  [in] flag  0-Blocking, 1-Non-blocking
* @param  [out] desc_pos  Tool pose
* @return  Error code
*/
errno_t  GetActualTCPPose(uint8_t flag, DescPose *desc_pos);
```

2.1.8.9 Get Current Tool Frame Number

```
/**
* @brief  Get current tool frame number
* @param  [in] flag  0-Blocking, 1-Non-blocking
* @param  [out] id  Tool frame number
* @return  Error code
*/
errno_t  GetActualTCPNum(uint8_t flag, int *id);
```

2.1.8.10 Get Current Work Object Frame Number

```
/**
* @brief  Get current work object frame number
* @param  [in] flag  0-Blocking, 1-Non-blocking
* @param  [out] id  Work object frame number
* @return  Error code
*/
errno_t  GetActualWObjNum(uint8_t flag, int *id);
```

2.1.8.11 Get Current End Flange Pose

```
/**
* @brief  Get current end flange pose
* @param  [in] flag  0-Blocking, 1-Non-blocking
* @param  [out] desc_pos  Flange pose
* @return  Error code
*/
errno_t  GetActualToolFlangePose(uint8_t flag, DescPose *desc_pos);
```

2.1.8.12 Get Current Joint Torque

```
/**
* @brief Get current joint torque
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] torques Joint torque
* @return  Error code
*/
errno_t  GetJointTorques(uint8_t flag, float torques[6]);
```

2.1.8.13 Get System Time

```
/**
* @brief  Get system time
* @param  [out] t_ms Unit ms
* @return  Error code
*/
errno_t  GetSystemClock(float *t_ms);
```

2.1.8.14 Check If Robot Motion Is Complete

```
/**
* @brief  Check if robot motion is complete
* @param  [out]  state  0-Not complete, 1-Complete
* @return  Error code
*/
errno_t  GetRobotMotionDone(uint8_t *state);
```

2.1.8.15 Query Robot Motion Queue Length

```
/**
 * @brief  Query robot motion queue length
 * @param  [out]  len  Queue length
 * @return  Error code
 */
errno_t  GetMotionQueueLength(int *len);
```

2.1.8.16 Get Robot Emergency Stop Status

```
/**
* @brief Get robot emergency stop status
* @param [out] state Emergency stop status, 0-Normal, 1-Emergency stop
* @return Error code
*/
errno_t GetRobotEmergencyStopState(uint8_t *state);
```

2.1.8.17 Get SDK-Robot Communication Status

```
/**
* @brief Get SDK-robot communication status
* @param [out] state Communication status, 0-Normal, 1-Abnormal
*/
errno_t GetSDKComState(int *state);
```

2.1.8.18 Get Safety Stop Signal

```
/**
* @brief Get safety stop signal
* @param [out] si0_state Safety stop signal SI0, 0-Invalid, 1-Valid
* @param [out] si1_state Safety stop signal SI1, 0-Invalid, 1-Valid
*/
errno_t GetSafetyStopState(uint8_t *si0_state, uint8_t *si1_state);
```

2.1.8.19 Get Robot Joint Driver Temperature(°C)

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get robot joint driver temperature(°C)
* @return Error code
*/
errno_t GetJointDriverTemperature(double temperature[]);
```

2.1.8.20 Get Robot Joint Driver Torque(Nm)

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get robot joint driver torque(Nm)
* @return Error code
*/
errno_t GetJointDriverTorque(double torque[]);
```

2.1.8.21 Get Robot Real-time Status Structure

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Get robot real-time status structure
* @param [out] pkg Robot real-time status structure
* @return Error code
*/
errno_t GetRobotRealTimeState(ROBOT_STATE_PKG *pkg);
```

2.1.8.22 Robot Status Query Code Example

```
int TestGetStatus(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  float yangle, zangle;
  robot.GetRobotInstallAngle(&yangle, &zangle);
  printf("yangle:%f,zangle:%f\n", yangle, zangle);
  JointPos j_deg = {};
  robot.GetActualJointPosDegree(0, &j_deg);
  printf("joint pos deg:%f,%f,%f,%f,%f,%f\n", j_deg.jPos[0], j_deg.jPos[1], j_deg.
↪jPos[2], j_deg.jPos[3], j_deg.jPos[4], j_deg.jPos[5]);
  float jointSpeed[6] = { 0.0 };
  robot.GetActualJointSpeedsDegree(0, jointSpeed);
  printf("joint speeds deg:%f,%f,%f,%f,%f,%f\n", jointSpeed[0], jointSpeed[1],␣
↪jointSpeed[2], jointSpeed[3], jointSpeed[4], jointSpeed[5]);
  float jointAcc[6] = { 0.0 };
  robot.GetActualJointAccDegree(0, jointAcc);
  printf("joint acc deg:%f,%f,%f,%f,%f,%f\n", jointAcc[0], jointAcc[1], jointAcc[2],␣
↪jointAcc[3], jointAcc[4], jointAcc[5]);
  float tcp_speed = 0.0;
  float ori_speed = 0.0;
  robot.GetTargetTCPCompositeSpeed(0, &tcp_speed, &ori_speed);
  printf("GetTargetTCPCompositeSpeed tcp %f; ori %f\n", tcp_speed, ori_speed);
  robot.GetActualTCPCompositeSpeed(0, &tcp_speed, &ori_speed);
  printf("GetActualTCPCompositeSpeed tcp %f; ori %f\n", tcp_speed, ori_speed);
  float targetSpeed[6] = { 0.0 };
  robot.GetTargetTCPSpeed(0, targetSpeed);
  printf("GetTargetTCPSpeed %f,%f,%f,%f,%f,%f\n", targetSpeed[0], targetSpeed[1],␣
↪targetSpeed[2], targetSpeed[3], targetSpeed[4], targetSpeed[5]);
  float actualSpeed[6] = { 0.0 };
  robot.GetActualTCPSpeed(0, actualSpeed);
```

(continues on next page)

(continued from previous page)

```
  printf("GetTargetTCPSpeed %f,%f,%f,%f,%f,%f\n", actualSpeed[0], actualSpeed[1],␣
↪actualSpeed[2], actualSpeed[3], actualSpeed[4], actualSpeed[5]);
  DescPose tcp = {};
  robot.GetActualTCPPose(0, &tcp);
  printf("tcp pose:%f,%f,%f,%f,%f,%f\n", tcp.tran.x, tcp.tran.y, tcp.tran.z, tcp.rpy.rx,␣
↪tcp.rpy.ry, tcp.rpy.rz);
  DescPose flange = {};
  robot.GetActualToolFlangePose(0, &flange);
  printf("flange pose:%f,%f,%f,%f,%f,%f\n", flange.tran.x, flange.tran.y, flange.tran.z,␣
↪flange.rpy.rx, flange.rpy.ry, flange.rpy.rz);
  int id = 0;
  robot.GetActualTCPNum(0, &id);
  printf("tcp num:%d\n", id);
  robot.GetActualWObjNum(0, &id);
  printf("wobj num:%d\n", id);
  float jtorque[6] = { 0.0 };
  robot.GetJointTorques(0, jtorque);
  printf("torques:%f,%f,%f,%f,%f,%f\n", jtorque[0], jtorque[1], jtorque[2], jtorque[3],␣
↪jtorque[4], jtorque[5]);
  float t_ms = 0.0;
  robot.GetSystemClock(&t_ms);
  printf("system clock:%f\n", t_ms);
  int config = 0;
  robot.GetRobotCurJointsConfig(&config);
  printf("joint config:%d\n", config);
  uint8_t motionDone = 0;
  robot.GetRobotMotionDone(&motionDone);
  printf("GetRobotMotionDone :%d\n", motionDone);
  int len = 0;
  robot.GetMotionQueueLength(&len);
  printf("GetMotionQueueLength :%d\n", len);
  uint8_t emergState = 0;
  robot.GetRobotEmergencyStopState(&emergState);
  printf("GetRobotEmergencyStopState :%d\n", emergState);
  int comstate = 0;
  robot.GetSDKComState(&comstate);
  printf("GetSDKComState :%d\n", comstate);
  uint8_t si0_state, si1_state;
  robot.GetSafetyStopState(&si0_state, &si1_state);
  printf("GetSafetyStopState :%d %d\n", si0_state, si1_state);
  double temp[6] = { 0.0 };
  robot.GetJointDriverTemperature(temp);
  printf("Temperature:%f,%f,%f,%f,%f,%f\n", temp[0], temp[1], temp[2], temp[3], temp[4],␣
↪temp[5]);
  double torque[6] = { 0.0 };
  robot.GetJointDriverTorque(torque);
  printf("torque:%f,%f,%f,%f,%f,%f\n", torque[0], torque[1], torque[2], torque[3],␣
↪torque[4], torque[5]);
  robot.GetRobotRealTimeState(&pkg);
  robot.CloseRPC();
  return 0;
}
```

2.1.8.23 Inverse Kinematics Calculation

```
/**
* @brief  Inverse kinematics calculation
* @param  [in] type 0-Absolute pose (base frame), 1-Incremental pose (base frame), 2-Incremental pose (tool frame)
* @param  [in] desc_pos Cartesian pose
* @param  [in] config Joint space configuration, [-1]-Calculate based on current joint position, [0~7]-Solve according to specific joint space configuration
* @param  [out] joint_pos Joint position
* @return  Error code
*/
errno_t  GetInverseKin(int type, DescPose *desc_pos, int config, JointPos *joint_pos);
```

2.1.8.24 Inverse Kinematics Calculation (Reference Position)

```
/**
* @brief  Inverse kinematics calculation, solve based on specified joint position
* @param  [in] type 0-Absolute pose (base frame), 1-Incremental pose (base frame), 2-Incremental pose (tool frame)
* @param  [in] desc_pos Cartesian pose
* @param  [in] joint_pos_ref Reference joint position
* @param  [out] joint_pos Joint position
* @return  Error code
*/
errno_t  GetInverseKinRef(int type, DescPose *desc_pos, JointPos *joint_pos_ref, JointPos *joint_pos);
```

2.1.8.25 Inverse Kinematics Solution, Cartesian Space Including Extended Axis Position

```
/**
* @brief Inverse kinematics solution, Cartesian space including extended axis position
* @param [in] type 0-absolute pose (base coordinate system), 1-incremental pose (base coordinate system), 2-incremental pose (tool coordinate system)
* @param [in] desc_pos Cartesian pose
* @param [in] exaxis Extended axis position
* @param [in] tool Tool number
* @param [in] workPiece Workpiece number
* @param [out] joint_pos Joint position
* @return Error code
*/
errno_t GetInverseKinExaxis(int type, DescPose desc_pos, ExaxisPos exaxis, int tool, int workPiece, JointPos& joint_pos);
```

2.1.8.26 Example Code for Inverse Kinematics Solution Including Extended Axis Position

```
int TestInverseKinExaxis()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return 0;
    }
    robot.SetReConnectParam(true, 30000, 500);
    DescPose desc(99.957, -0.002, 29.994, -176.569, -6.757, -167.462);
    ExaxisPos exaxis(100.0, 0.0, 0.0, 0.0);
    JointPos jointPos = {};
    DescPose offsetPos = {};
    robot.GetRobotRealTimeState(&pkg);
    int toolnum = pkg.tool;
    int workPcsNum = pkg.user;
    robot.GetInverseKinExaxis(0, desc, exaxis, toolnum, workPcsNum, jointPos);
    printf("GetInverseKinExaxis joint is %f, %f, %f, %f, %f, %f\n", jointPos.jPos[0],␣
↪jointPos.jPos[1], jointPos.jPos[2], jointPos.jPos[3], jointPos.jPos[4], jointPos.
↪jPos[5]);
    robot.ExtAxisMove(exaxis, 100, -1);
    robot.MoveJ(&jointPos, &desc, toolnum, workPcsNum, 100.0, 100.0, 100.0, &exaxis, -1,␣
↪0, &offsetPos);
    robot.CloseRPC();
    robot.Sleep(9999999);
    return 0;
}
```

2.1.8.27 Check If Inverse Kinematics Has Solution

```
/**
* @brief  Inverse kinematics calculation, check if solution exists based on specified␣
↪joint position
* @param  [in] type 0-Absolute pose (base frame), 1-Incremental pose (base frame), 2-
↪Incremental pose (tool frame)
* @param  [in] desc_pos Cartesian pose
* @param  [in] joint_pos_ref Reference joint position
* @param  [out] result 0-No solution, 1-Solution exists
* @return  Error code
*/
errno_t  GetInverseKinHasSolution(int type, DescPose *desc_pos, JointPos *joint_pos_ref,␣
↪uint8_t *result);
```

2.1.8.28 Forward Kinematics Calculation

```
/**
* @brief  Forward kinematics calculation
* @param  [in] joint_pos Joint position
* @param  [out] desc_pos Cartesian pose
* @return  Error code
*/
errno_t  GetForwardKin(JointPos *joint_pos, DescPose *desc_pos);
```

2.1.8.29 Robot Kinematics Calculation Code Example

```
int TestInverseKin(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
  DescPose desc_pos1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
  JointPos inverseRtn = {};
  robot.GetInverseKin(0, &desc_pos1, -1, &inverseRtn);
  printf("dcs1 GetInverseKin rtn is %f %f %f %f %f %f \n", inverseRtn.jPos[0],␣
↪inverseRtn.jPos[1], inverseRtn.jPos[2], inverseRtn.jPos[3], inverseRtn.jPos[4],␣
↪inverseRtn.jPos[5]);
  robot.GetInverseKinRef(0, &desc_pos1, &j1, &inverseRtn);
  printf("dcs1 GetInverseKinRef rtn is %f %f %f %f %f %f \n", inverseRtn.jPos[0],␣
↪inverseRtn.jPos[1], inverseRtn.jPos[2], inverseRtn.jPos[3], inverseRtn.jPos[4],␣
↪inverseRtn.jPos[5]);
  uint8_t hasResut = 0;
  robot.GetInverseKinHasSolution(0, &desc_pos1, &j1, &hasResut);
  printf("dcs1 GetInverseKinRef result %d\n", hasResut);
  DescPose forwordResult = {};
  robot.GetForwardKin(&j1, &forwordResult);
  printf("jpos1 forwordResult rtn is %f %f %f %f %f %f \n", forwordResult.tran.x,␣
↪forwordResult.tran.y, forwordResult.tran.z, forwordResult.rpy.rx, forwordResult.rpy.ry,
↪ forwordResult.rpy.rz);
  robot.CloseRPC();
  return 0;
}
```

2.1.8.30 Query Robot Teaching Point Data

```
/**
 * @brief  Query robot teaching point data
 * @param  [in]  name  Point name
 * @param  [out]  data   Point data
 * @return  Error code
 */
errno_t  GetRobotTeachingPoint(char name[64], float data[20]);
```

2.1.8.31 Get Robot DH Parameter Compensation Values

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Get robot DH parameter compensation values
* @param [out] dhCompensation Robot DH parameter compensation values(mm) [cmpstD1,
↪cmpstA2,cmpstA3,cmpstD4,cmpstD5,cmpstD6]
* @return Error code
*/
errno_t GetDHCompensation(double dhCompensation[6]);
```

2.1.8.32 Get Controller SN Code

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Get controller SN code
* @param [out] SNCode Controller SN code
* @return Error code
*/
errno_t GetRobotSN(std::string& SNCode);
```

2.1.8.33 Query Robot Teaching Point Data Code Example

```
int TestGetTeachPoint(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  char name[64] = "P1";
  float data[20] = { 0 };
```

(continues on next page)

(continued from previous page)

```
  rtn = robot.GetRobotTeachingPoint(name, data);
  printf(" %d name is: %s \n", rtn, name);
  for (int i = 0; i < 20; i++)
  {
    printf("data is: %f \n", data[i]);
  }
  int que_len = 0;
  rtn = robot.GetMotionQueueLength(&que_len);
  printf("GetMotionQueueLength rtn is: %d, queue length is: %d \n", rtn, que_len);
  double dh[6] = { 0 };
  int retval = 0;
  retval = robot.GetDHCompensation(dh);
  cout << "retval is: " << retval << endl;
  cout << "dh is: " << dh[0] << " " << dh[1] << " " << dh[2] << " " << dh[3] << " " << dh[4] << " " << dh[5] << endl;
  string SN = "";
  robot.GetRobotSN(SN);
  cout << "robot SN is " << SN << endl;
  robot.CloseRPC();
  return 0;
}
```

2.1.8.34 Get Tool Coordinate System by ID

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Tool Coordinate System by ID
* @param [in] id Tool coordinate system ID
* @param [out] coord Coordinate system values
* @return Error code
*/
errno_t GetToolCoordWithID(int id, DescPose& coord);
```

2.1.8.35 Get Work Object Coordinate System by ID

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Work Object Coordinate System by ID
* @param [in] id Work object coordinate system ID
* @param [out] coord Coordinate system values
* @return Error code
*/
errno_t GetWObjCoordWithID(int id, DescPose& coord);
```

2.1.8.36 Get External Tool Coordinate System by ID

New in version C++SDK-v3.8.6.

```
/**
* @brief Get External Tool Coordinate System by ID
* @param [in] id External tool coordinate system ID
* @param [out] coord Coordinate system values
* @return Error code
*/
errno_t GetExToolCoordWithID(int id, DescPose& coord);
```

2.1.8.37 Get Extended Axis Coordinate System by ID

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Extended Axis Coordinate System by ID
* @param [in] id External tool coordinate system ID
* @param [out] coord Coordinate system values
* @return Error code
*/
errno_t GetExAxisCoordWithID(int id, DescPose& coord);
```

2.1.8.38 Get Payload Mass and Center of Gravity by ID

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Payload Mass and Center of Gravity by ID
* @param [in] id Payload ID
* @param [out] weight Payload mass
* @param [out] cog Payload center of gravity
* @return Error code
*/
errno_t GetTargetPayloadWithID(int id, double& weight, DescTran& cog);
```

2.1.8.39 Get Current Tool Coordinate System

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Current Tool Coordinate System
* @param [out] coord Coordinate system values
* @return Error code
*/
errno_t GetCurToolCoord(DescPose& coord);
```

2.1.8.40 Get Current Work Object Coordinate System

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Current Work Object Coordinate System
* @param [out] coord Coordinate system values
* @return Error code
*/
errno_t GetCurWObjCoord(DescPose& coord);
```

2.1.8.41 Get Current External Tool Coordinate System

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Current External Tool Coordinate System
* @param [out] coord Coordinate system values
* @return Error code
*/
errno_t GetCurExToolCoord(DescPose& coord);
```

2.1.8.42 Get Current Extended Axis Coordinate System

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Current Extended Axis Coordinate System
* @param [out] coord Coordinate system values
* @return Error code
*/
errno_t GetCurExAxisCoord(DescPose& coord);
```

2.1.8.43 Get Robot Coordinate Systems and Payload Code Example

New in version C++SDK-v3.8.6.

```
int TestCoord()
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return 0;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int id = 1;
```

(continues on next page)

(continued from previous page)

```
  DescPose toolCoord = {};
  DescPose extoolCoord = {};
  DescPose wobjCoord = {};
  DescPose exAxisCoord = {};
  robot.GetToolCoordWithID(id, toolCoord);
  printf("GetToolCoordWithID %d, %f %f %f %f %f %f\n", id,
    toolCoord.tran.x, toolCoord.tran.y, toolCoord.tran.z,
    toolCoord.rpy.rx, toolCoord.rpy.ry, toolCoord.rpy.rz);
  robot.GetWObjCoordWithID(id, wobjCoord);
  printf("GetWObjCoordWithID %d, %f %f %f %f %f %f\n", id,
    wobjCoord.tran.x, wobjCoord.tran.y, wobjCoord.tran.z,
    wobjCoord.rpy.rx, wobjCoord.rpy.ry, wobjCoord.rpy.rz);

  robot.GetExToolCoordWithID(id, extoolCoord);
  printf("GetExToolCoordWithID %d, %f %f %f %f %f %f\n", id,
    extoolCoord.tran.x, extoolCoord.tran.y, extoolCoord.tran.z,
    extoolCoord.rpy.rx, extoolCoord.rpy.ry, extoolCoord.rpy.rz);

  robot.GetExAxisCoordWithID(id, exAxisCoord);
  printf("GetExAxisCoordWithID %d, %f %f %f %f %f %f\n", id,
    exAxisCoord.tran.x, exAxisCoord.tran.y, exAxisCoord.tran.z,
    exAxisCoord.rpy.rx, exAxisCoord.rpy.ry, exAxisCoord.rpy.rz);
  double weight = 0.0;
  DescTran cog = {};
  robot.GetTargetPayloadWithID(id, weight, cog);
  printf("GetTargetPayloadWithID %d, %f %f %f %f\n", id, weight,
    cog.x, cog.y, cog.z);
  robot.GetCurToolCoord(toolCoord);
  printf("GetCurToolCoord %f %f %f %f %f %f\n",
    toolCoord.tran.x, toolCoord.tran.y, toolCoord.tran.z,
    toolCoord.rpy.rx, toolCoord.rpy.ry, toolCoord.rpy.rz);
  robot.GetCurWObjCoord(wobjCoord);
  printf("GetCurWObjCoord %f %f %f %f %f %f\n",
    wobjCoord.tran.x, wobjCoord.tran.y, wobjCoord.tran.z,
    wobjCoord.rpy.rx, wobjCoord.rpy.ry, wobjCoord.rpy.rz);
  robot.GetCurExToolCoord(extoolCoord);
  printf("GetExToolCoordWithID %f %f %f %f %f %f\n",
    extoolCoord.tran.x, extoolCoord.tran.y, extoolCoord.tran.z,
    extoolCoord.rpy.rx, extoolCoord.rpy.ry, extoolCoord.rpy.rz);
  robot.GetCurExAxisCoord(exAxisCoord);
  printf("GetCurExAxisCoord %f %f %f %f %f %f\n",
    exAxisCoord.tran.x, exAxisCoord.tran.y, exAxisCoord.tran.z,
    exAxisCoord.rpy.rx, exAxisCoord.rpy.ry, exAxisCoord.rpy.rz);
  float weightT = 0.0;
  DescTran cogT = {};
  robot.GetTargetPayload(0, &weightT);
  robot.GetTargetPayloadCog(0, &cogT);
  printf("GetTargetPayload %f %f %f %f\n", weightT,
    cogT.x, cogT.y, cogT.z);
  DescPose coordSet(0,1,2,3,4,5);
  robot.SetToolCoord(1, &coordSet, 0, 0, 1, 0);
  robot.SetWObjCoord(1, &coordSet, 0);
```

(continues on next page)

(continued from previous page)

```
  robot.SetLoadWeight(1, 1.3);
  DescTran cog = {};
  cog.x = 10;
  cog.y = 20;
  cog.z = 30;
  robot.SetLoadCoord(1, &cog);
  DescPose etcp(0, 0, 100, 0, 0, 0);
  DescPose etool(0, 0, 50, 0, 0, 0);
  rtn = robot.SetExToolCoord(1, &etcp, &etool);
  printf("SetExToolCoord rtn is %d\n", rtn);
  robot.ExtAxisActiveECoordSys(1, 1, coordSet, 1);
  robot.CloseRPC();
  return 0;
}
```

2.1.9 Trajectory recurrence

2.1.9.1 Set TPD Trajectory Recording Parameters

```
/**
* @brief  Set TPD trajectory recording parameters
* @param  [in] type  Recording data type, 1-Joint position
* @param  [in] name  Trajectory file name
* @param  [in] period_ms  Data sampling period, fixed value 2ms or 4ms or 8ms
* @param  [in] di_choose  DI selection, bit0~bit7 correspond to controller DI0~DI7, bit8~bit9 correspond to end DI0~DI1, 0-Not selected, 1-Selected
* @param  [in] do_choose  DO selection, bit0~bit7 correspond to controller DO0~DO7, bit8~bit9 correspond to end DO0~DO1, 0-Not selected, 1-Selected
* @return  Error code
*/
errno_t  SetTPDParam(int type, char name[30], int period_ms, uint16_t di_choose, uint16_t do_choose);
```

2.1.9.2 Start TPD Trajectory Recording

```
/**
* @brief  Start TPD trajectory recording
* @param  [in] type  Recording data type, 1-Joint position
* @param  [in] name  Trajectory file name
* @param  [in] period_ms  Data sampling period, fixed value 2ms or 4ms or 8ms
* @param  [in] di_choose  DI selection, bit0~bit7 correspond to controller DI0~DI7, bit8~bit9 correspond to end DI0~DI1, 0-Not selected, 1-Selected
* @param  [in] do_choose  DO selection, bit0~bit7 correspond to controller DO0~DO7, bit8~bit9 correspond to end DO0~DO1, 0-Not selected, 1-Selected
* @return  Error code
*/
errno_t  SetTPDStart(int type, char name[30], int period_ms, uint16_t di_choose, uint16_t do_choose);
```

2.1.9.3 Stop TPD Trajectory Recording

```
/**
* @brief  Stop TPD trajectory recording
* @return  Error code
*/
errno_t  SetWebTPDStop();
```

2.1.9.4 Delete TPD Trajectory Recording

```
/**
* @brief  Delete TPD trajectory recording
* @param  [in] name  Trajectory file name
* @return  Error code
*/
errno_t  SetTPDDelete(char name[30]);
```

2.1.9.5 TPD Trajectory Preload

```
/**
* @brief  TPD trajectory preload
* @param  [in] name  Trajectory file name
* @return  Error code
*/
errno_t  LoadTPD(char name[30]);
```

2.1.9.6 TPD Trajectory Playback

```
/**
* @brief  TPD trajectory playback
* @param  [in] name  Trajectory file name
* @param  [in] blend 0-No smoothing, 1-Smoothing
* @param  [in] ovl  Speed scaling percentage, range [0~100]
* @return  Error code
*/
errno_t  MoveTPD(char name[30], uint8_t blend, float ovl);
```

2.1.9.7 Get TPD Start Pose

```
/**
 * @brief  Get TPD start pose
 * @param  [in] name TPD file name, no file extension needed
 * @return  Error code
 */
errno_t  GetTPDStartPose(char name[30], DescPose *desc_pose);
```

2.1.9.8 Robot TPD Trajectory Recording Code Example

```
int TestTPD(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int type = 1;
  char name[30] = "tpd2025";
  int period_ms = 4;
  uint16_t di_choose = 0;
  uint16_t do_choose = 0;
  robot.SetTPDParam(type, name, period_ms, di_choose, do_choose);
  robot.Mode(1);
  robot.Sleep(1000);
  robot.DragTeachSwitch(1);
  robot.SetTPDStart(type, name, period_ms, di_choose, do_choose);
  robot.Sleep(10000);
  robot.SetWebTPDStop();
  robot.DragTeachSwitch(0);
  float ovl = 100.0;
  uint8_t blend = 0;
  DescPose start_pose = {};
  rtn = robot.LoadTPD(name);
  printf("LoadTPD rtn is: %d\n", rtn);
  robot.GetTPDStartPose(name, &start_pose);
  printf("start pose, xyz is: %f %f %f. rpy is: %f %f %f \n", start_pose.tran.x, start_
↪pose.tran.y, start_pose.tran.z, start_pose.rpy.rx, start_pose.rpy.ry, start_pose.rpy.
↪rz);
  robot.MoveCart(&start_pose, 0, 0, 100, 100, ovl, -1, -1);
  robot.Sleep(1000);
  rtn = robot.MoveTPD(name, blend, ovl);
  printf("MoveTPD rtn is: %d\n", rtn);
  std::this_thread::sleep_for(std::chrono::milliseconds(5000));
  robot.SetTPDDelete(name);
  robot.CloseRPC();
  return 0;
}
```

2.1.9.9 Trajectory Preprocessing

```
/**
 * @brief  Trajectory preprocessing
 * @param  [in] name  Trajectory file name
 * @param  [in] ovl Speed scaling percentage, range [0~100]
 * @param  [in] opt 1-Control point, default is 1
 * @return  Error code
 */
errno_t  LoadTrajectoryJ(char name[30], float ovl, int opt);
```

2.1.9.10 Trajectory Playback

```
/**
 * @brief  Trajectory playback
 * @return  Error code
 */
errno_t  MoveTrajectoryJ();
```

2.1.9.11 Get Trajectory Start Pose

```
/**
 * @brief  Get trajectory start pose
 * @param  [in] name Trajectory file name
 * @return  Error code
 */
errno_t  GetTrajectoryStartPose(char name[30], DescPose *desc_pose);
```

2.1.9.12 Get Trajectory Point Number

```
/**
 * @brief  Get trajectory point number
 * @return  Error code
 */
errno_t  GetTrajectoryPointNum(int *pnum);
```

2.1.9.13 Set Trajectory Running Speed

```
/**
 * @brief  Set trajectory running speed
 * @param  [in] ovl Speed percentage
 * @return  Error code
 */
errno_t  SetTrajectoryJSpeed(float ovl);
```

2.1.9.14 Set Trajectory Running Force and Torque

```
/**
 * @brief  Set trajectory running force and torque
 * @param  [in] ft Force and torque in three directions, unit N and Nm
 * @return  Error code
 */
errno_t  SetTrajectoryJForceTorque(ForceTorque *ft);
```

2.1.9.15 Set Trajectory Running Force in X Direction

```
/**
 * @brief  Set trajectory running force in X direction
 * @param  [in] fx Force in X direction, unit N
 * @return  Error code
 */
errno_t  SetTrajectoryJForceFx(double fx);
```

2.1.9.16 Set Trajectory Running Force in Y Direction

```
/**
 * @brief  Set trajectory running force in Y direction
 * @param  [in] fy Force in Y direction, unit N
 * @return  Error code
 */
errno_t  SetTrajectoryJForceFy(double fy);
```

2.1.9.17 Set Trajectory Running Force in Z Direction

```
/**
 * @brief  Set trajectory running force in Z direction
 * @param  [in] fz Force in X direction, unit N
 * @return  Error code
 */
errno_t  SetTrajectoryJForceFz(double fz);
```

2.1.9.18 Set Trajectory Running Torque Around X Axis

```
/**
 * @brief  Set trajectory running torque around X axis
 * @param  [in] tx Torque around X axis, unit Nm
 * @return  Error code
 */
errno_t  SetTrajectoryJTorqueTx(double tx);
```

2.1.9.19 Set Trajectory Running Torque Around Y Axis

```
/**
 * @brief  Set trajectory running torque around Y axis
 * @param  [in] ty Torque around Y axis, unit Nm
 * @return  Error code
 */
errno_t  SetTrajectoryJTorqueTy(double ty);
```

2.1.9.20 Set Trajectory Running Torque Around Z Axis

```
/**
 * @brief  Set trajectory running torque around Z axis
 * @param  [in] tz Torque around Z axis, unit Nm
 * @return  Error code
 */
errno_t  SetTrajectoryJTorqueTz(double tz);
```

2.1.9.21 Upload Trajectory J File

New in version V3.7.7.

```
/**
     * @brief Upload trajectory J file
     * @param [in] filePath Full path name of trajectory file to upload   C://test/testJ.
↪txt
     * @return Error code
     */
    errno_t TrajectoryJUpLoad(const std::string& filePath);
```

2.1.9.22 Delete Trajectory J File

New in version V3.7.7.

```
/**
     * @brief Delete trajectory J file
     * @param [in] fileName File name testJ.txt
     * @return Error code
     */
    errno_t TrajectoryJDelete(const std::string& fileName);
```

2.1.9.23 Robot Trajectory J File Playback Code Example

```
int TestTraj(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  rtn = robot.TrajectoryJUpLoad("D://zUP/traj1.txt");
  printf("Upload TrajectoryJ A %d\n", rtn);
  char traj_file_name[30] = "/fruser/traj/traj1.txt";
  rtn = robot.LoadTrajectoryJ(traj_file_name, 100, 1);
  printf("LoadTrajectoryJ %s, rtn is: %d\n", traj_file_name, rtn);
  DescPose traj_start_pose;
  memset(&traj_start_pose, 0, sizeof(DescPose));
  rtn = robot.GetTrajectoryStartPose(traj_file_name, &traj_start_pose);
  printf("GetTrajectoryStartPose is: %d\n", rtn);
  printf("desc_pos:%f,%f,%f,%f,%f,%f\n", traj_start_pose.tran.x, traj_start_pose.tran.y,␣
↪traj_start_pose.tran.z, traj_start_pose.rpy.rx, traj_start_pose.rpy.ry, traj_start_
↪pose.rpy.rz);
  std::this_thread::sleep_for(std::chrono::seconds(1));
  robot.SetSpeed(50);
  robot.MoveCart(&traj_start_pose, 0, 0, 100, 100, 100, -1, -1);
  int traj_num = 0;
  rtn = robot.GetTrajectoryPointNum(&traj_num);
  printf("GetTrajectoryStartPose rtn is: %d, traj num is: %d\n", rtn, traj_num);
  rtn = robot.SetTrajectoryJSpeed(50.0);
  printf("SetTrajectoryJSpeed is: %d\n", rtn);
  ForceTorque traj_force;
  memset(&traj_force, 0, sizeof(ForceTorque));
  traj_force.fx = 10;
  rtn = robot.SetTrajectoryJForceTorque(&traj_force);
  printf("SetTrajectoryJForceTorque rtn is: %d\n", rtn);
  rtn = robot.SetTrajectoryJForceFx(10.0);
  printf("SetTrajectoryJForceFx rtn is: %d\n", rtn);
  rtn = robot.SetTrajectoryJForceFy(0.0);
  printf("SetTrajectoryJForceFy rtn is: %d\n", rtn);
  rtn = robot.SetTrajectoryJForceFz(0.0);
  printf("SetTrajectoryJForceFz rtn is: %d\n", rtn);
  rtn = robot.SetTrajectoryJTorqueTx(10.0);
  printf("SetTrajectoryJTorqueTx rtn is: %d\n", rtn);
  rtn = robot.SetTrajectoryJTorqueTy(10.0);
  printf("SetTrajectoryJTorqueTy rtn is: %d\n", rtn);
  rtn = robot.SetTrajectoryJTorqueTz(10.0);
  printf("SetTrajectoryJTorqueTz rtn is: %d\n", rtn);
  rtn = robot.MoveTrajectoryJ();
  printf("MoveTrajectoryJ rtn is: %d\n", rtn);
```

(continues on next page)

(continued from previous page)

```
  robot.CloseRPC();
  return 0;
}
```

2.1.9.24 Trajectory Preprocessing (Trajectory Lookahead)

```
/**
     * @brief  Trajectory preprocessing (Trajectory Lookahead)
     * @param  [in] name  Trajectory file name
     * @param  [in] mode Sampling mode, 0-No sampling; 1-Equal data interval sampling; 2-Equal error limit sampling
     * @param  [in] errorLim Error limit, takes effect when using linear fitting
     * @param  [in] type Smoothing method, 0-Bezier smoothing
     * @param  [in] precision Smoothing precision, takes effect when using Bezier smoothing
     * @param  [in] vamx Set maximum speed, mm/s
     * @param  [in] amax Set maximum acceleration, mm/s2
     * @param  [in] jmax Set maximum jerk, mm/s3
     * @param [in] flag Constant velocity lookahead switch; 0-Disable; 1-Enable
     * @return Error code
     */
     errno_t LoadTrajectoryLA(char name[30], int mode, double errorLim, int type, double precision, double vamx, double amax, double jmax, int flag = 0);
```

2.1.9.25 Trajectory Playback (Trajectory Lookahead)

```
/**
* @brief  Trajectory playback (Trajectory Lookahead)
* @return  Error code
*/
errno_t MoveTrajectoryLA();
```

2.1.9.26 Trajectory Playback (Trajectory Lookahead) Code Example

```
int TestLoadTrajLA(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  rtn = robot.TrajectoryJUpLoad("D://zUP/traj.txt");
```

(continues on next page)

(continued from previous page)

```
  printf("Upload TrajectoryJ A %d\n", rtn);
  char traj_file_name[30] = "/fruser/traj/traj.txt";
  rtn = robot.LoadTrajectoryLA(traj_file_name, 1, 2, 0, 2, 100, 200, 1000);
  printf("LoadTrajectoryLA %s, rtn is: %d\n", traj_file_name, rtn);
  DescPose traj_start_pose;
  memset(&traj_start_pose, 0, sizeof(DescPose));
  rtn = robot.GetTrajectoryStartPose(traj_file_name, &traj_start_pose);
  printf("GetTrajectoryStartPose is: %d\n", rtn);
  printf("desc_pos:%f,%f,%f,%f,%f,%f\n", traj_start_pose.tran.x, traj_start_pose.tran.y,␣
↪traj_start_pose.tran.z, traj_start_pose.rpy.rx, traj_start_pose.rpy.ry, traj_start_
↪pose.rpy.rz);
  std::this_thread::sleep_for(std::chrono::seconds(1));
  robot.SetSpeed(50);
  robot.MoveCart(&traj_start_pose, 0, 0, 100, 100, 100, -1, -1);
  rtn = robot.MoveTrajectoryLA();
  printf("MoveTrajectoryLA rtn is: %d\n", rtn);
  robot.CloseRPC();
  return 0;
}
```

2.1.10 WebAPP program use

2.1.10.1 Set Default Program to Load Automatically on Startup

```
/**
* @brief  Set default program to load automatically on startup
* @param  [in] flag  0-Do not load default program on startup, 1-Load default program on␣
↪startup
* @param  [in] program_name Program name and path, e.g. "/fruser/movej.lua", where "/
↪fruser/" is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the fixed␣
↪path for LA.
* @return  Error code
*/
errno_t  LoadDefaultProgConfig(uint8_t flag, char program_name[64]);
```

2.1.10.2 Load Specified Program

```
/**
* @brief  Load specified program
* @param  [in] program_name Program name and path, e.g. "/fruser/movej.lua", where "/
↪fruser/" is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the fixed␣
↪path for LA.
* @return  Error code
*/
errno_t  ProgramLoad(char program_name[64]);
```

2.1.10.3 Get Loaded Program Name

```
/**
* @brief  Get loaded program name
* @param  [out] program_name Program name and path, e.g. "/fruser/movej.lua", where "/fruser/" is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the fixed path for LA.
* @return  Error code
*/
errno_t  GetLoadedProgram(char program_name[64]);
```

2.1.10.4 Get Current Program Execution Line Number

```
/**
* @brief  Get current program execution line number
* @param  [out] line  Line number
* @return  Error code
*/
errno_t  GetCurrentLine(int *line);
```

2.1.10.5 Run Currently Loaded Program

```
/**
* @brief  Run currently loaded program
* @return  Error code
*/
errno_t  ProgramRun();
```

2.1.10.6 Pause Currently Running Program

```
/**
* @brief  Pause currently running program
* @return  Error code
*/
errno_t  ProgramPause();
```

2.1.10.7 Resume Currently Paused Program

```
/**
* @brief  Resume currently paused program
* @return  Error code
*/
errno_t  ProgramResume();
```

2.1.10.8 Stop Currently Running Program

```
/**
* @brief  Stop currently running program
* @return  Error code
*/
errno_t  ProgramStop();
```

2.1.10.9 Get Program Execution State

```
/**
* @brief  Get program execution state
* @param  [out]  state 1-Program stopped or no program running, 2-Program running, 3-Program paused
* @return  Error code
*/
errno_t  GetProgramState(uint8_t *state);
```

2.1.10.10 Robot LUA Program Operation Code Example

```
int TestLuaOp(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  char program_name[64] = "/fruser/test.lua";
  char loaded_name[64] = "";
  uint8_t state;
  int line;
  robot.Mode(0);
  robot.LoadDefaultProgConfig(0, program_name);
  robot.ProgramLoad(program_name);
  robot.ProgramRun();
  robot.Sleep(1000);
  robot.ProgramPause();
  robot.GetProgramState(&state);
  printf("program state:%u\n", state);
  robot.GetCurrentLine(&line);
  printf("current line:%d\n", line);
  robot.GetLoadedProgram(loaded_name);
  printf("program name:%s\n", loaded_name);
  robot.Sleep(1000);
  robot.ProgramResume();
```

(continues on next page)

(continued from previous page)

```
  robot.Sleep(1000);
  robot.ProgramStop();
  robot.Sleep(1000);
  robot.CloseRPC();
  return 0;
}
```

2.1.10.11 Download Lua File

New in version C++SDK-v2.1.2.0.

```
/**
* @brief Download Lua file
* @param [in] fileName Lua file name to download, e.g. "test.lua"
* @param [in] savePath Local path to save file, e.g. "D://Down/"
* @return Error code
*/
errno_t LuaDownLoad(std::string fileName, std::string savePath);
```

2.1.10.12 Delete Lua File

New in version C++SDK-v2.1.2.0.

```
/**
* @brief Delete Lua file
* @param [in] fileName Lua file name to delete, e.g. "test.lua"
* @return Error code
*/
errno_t LuaDelete(std::string fileName);
```

2.1.10.13 Get All Current Lua File Names

New in version C++SDK-v2.1.2.0.

```
/**
* @brief Get all current Lua file names
* @param [out] luaNames List of Lua file names
* @return Error code
*/
errno_t GetLuaList(std::list<std::string>* luaNames);
```

2.1.10.14 Upload Lua File

New in version C++SDK-v2.1.2.0.

```
/**
* @brief Upload Lua file
* @param [in] filePath Local Lua file path name
* @return Error code
*/
errno_t LuaUpload(std::string filePath);
```

2.1.10.15 Robot LUA File Upload/Download Code Example

```
int TestLUAUpDownLoad(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  list<std::string> luaNames;
  rtn = robot.GetLuaList(&luaNames);
  std::cout << "res is: " << rtn << std::endl;
  std::cout << "size is: " << luaNames.size() << std::endl;
  for (auto it = luaNames.begin(); it != luaNames.end(); it++)
  {
    std::cout << it->c_str() << std::endl;
  }
  rtn = robot.LuaDownLoad("test.lua", "D://zDOWN/");
  printf("LuaDownLoad rtn is %d\n", rtn);
  rtn = robot.LuaUpload("D://zUP/airlab.lua");
  printf("LuaUpload rtn is %d\n", rtn);
  rtn = robot.LuaDelete("test.lua");
  printf("LuaDelete rtn is %d\n", rtn);
  robot.CloseRPC();
  return 0;
}
```

2.1.11 Robot Peripheral Devices

2.1.11.1 Configure Gripper

```
/**
* @brief  Configure gripper
* @param  [in] company  Gripper manufacturer (to be determined)
* @param  [in] device  Device number (currently unused, default 0)
* @param  [in] softvesion  Software version (currently unused, default 0)
* @param  [in] bus  Device bus location at end (currently unused, default 0)
* @return  Error code
*/
errno_t  SetGripperConfig(int company, int device, int softvesion, int bus);
```

2.1.11.2 Get Gripper Configuration

```
/**
* @brief  Get gripper configuration
* @param  [in] company  Gripper manufacturer (to be determined)
* @param  [in] device  Device number (currently unused, default 0)
* @param  [in] softvesion  Software version (currently unused, default 0)
* @param  [in] bus  Device bus location at end (currently unused, default 0)
* @return  Error code
*/
errno_t  GetGripperConfig(int *company, int *device, int *softvesion, int *bus);
```

2.1.11.3 Activate Gripper

```
/**
* @brief  Activate gripper
* @param  [in] index  Gripper number
* @param  [in] act  0-reset, 1-activate
* @return  Error code
*/
errno_t  ActGripper(int index, uint8_t act);
```

2.1.11.4 Control Gripper

Changed in version C++SDK-v2.1.5.0.

```
/**
     * @brief  Control gripper
     * @param  [in] index  Gripper number
     * @param  [in] pos  Position percentage [0~100]
     * @param  [in] vel  Speed percentage [0~100]
     * @param  [in] force  Torque percentage [0~100]
     * @param  [in] max_time  Maximum wait time [0~30000], unit ms
     * @param  [in] block  0-blocking, 1-non-blocking
     * @param  [in] type  Gripper type, 0-parallel gripper; 1-rotary gripper
```

(continues on next page)

(continued from previous page)

```
     * @param  [in] rotNum  Rotation turns
     * @param  [in] rotVel  Rotation speed percentage [0-100]
     * @param  [in] rotTorque  Rotation torque percentage [0-100]
     * @return  Error code
     */
    errno_t MoveGripper(int index, int pos, int vel, int force, int max_time, uint8_t␣
↪block, int type, double rotNum, int rotVel, int rotTorque);
```

2.1.11.5 Get Gripper Motion Status

```
/**
 * @brief  Get gripper motion status
 * @param  [out] fault  0-no error, 1-error
 * @param  [out] staus  0-motion not completed, 1-motion completed
 * @return  Error code
 */
errno_t  GetGripperMotionDone(uint16_t *fault, uint8_t *status);
```

2.1.11.6 Get Gripper Activation Status

```
/**
 * @brief  Get gripper activation status
 * @param  [out] fault  0-no error, 1-error
 * @param  [out] status  bit0~bit15 correspond to gripper numbers 0~15, bit=0 not␣
↪activated, bit=1 activated
 * @return  Error code
 */
errno_t  GetGripperActivateStatus(uint16_t *fault, uint16_t *status);
```

2.1.11.7 Get Gripper Position

```
/**
 * @brief  Get gripper position
 * @param  [out] fault  0-no error, 1-error
 * @param  [out] position  Position percentage, range 0~100%
 * @return  Error code
 */
errno_t  GetGripperCurPosition(uint16_t *fault, uint8_t *position);
```

2.1.11.8 Get Gripper Speed

```
/**
 * @brief  Get gripper speed
 * @param  [out] fault  0-no error, 1-error
 * @param  [out] speed  Speed percentage, range 0~100%
 * @return  Error code
 */
errno_t  GetGripperCurSpeed(uint16_t *fault, int8_t *speed);
```

2.1.11.9 Get Gripper Current

```
/**
 * @brief  Get gripper current
 * @param  [out] fault  0-no error, 1-error
 * @param  [out] current  Current percentage, range 0~100%
 * @return  Error code
 */
errno_t  GetGripperCurCurrent(uint16_t *fault, int8_t *current);
```

2.1.11.10 Get Gripper Voltage

```
/**
 * @brief  Get gripper voltage
 * @param  [out] fault  0-no error, 1-error
 * @param  [out] voltage  Voltage, unit 0.1V
 * @return  Error code
 */
errno_t  GetGripperVoltage(uint16_t *fault, int *voltage);
```

2.1.11.11 Get Gripper Temperature

```
/**
 * @brief  Get gripper temperature
 * @param  [out] fault  0-no error, 1-error
 * @param  [out] temp  Temperature, unit ℃
 * @return  Error code
 */
errno_t  GetGripperTemp(uint16_t *fault, int *temp);
```

2.1.11.12 Calculate Pre-Pick Point - Vision

```
/**
 * @brief  Calculate pre-pick point - vision
 * @param  [in] desc_pos  Pick point Cartesian pose
 * @param  [in] zlength  Z-axis offset
 * @param  [in] zangle   Rotation offset around Z-axis
 * @return  Error code
 */
errno_t  ComputePrePick(DescPose *desc_pos, double zlength, double zangle, DescPose *pre_
↪pos);
```

2.1.11.13 Calculate Retreat Point - Vision

```
/**
 * @brief  Calculate retreat point - vision
 * @param  [in] desc_pos  Pick point Cartesian pose
 * @param  [in] zlength  Z-axis offset
 * @param  [in] zangle   Rotation offset around Z-axis
 * @return  Error code
 */
errno_t  ComputePostPick(DescPose *desc_pos, double zlength, double zangle, DescPose␣
↪*post_pos);
```

2.1.11.14 Robot Gripper Operation Code Example

```
int TestGripper(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int company = 4;
  int device = 0;
  int softversion = 0;
  int bus = 2;
  int index = 2;
  int act = 0;
  int max_time = 30000;
  uint8_t block = 0;
  uint8_t status;
  uint16_t fault;
  uint16_t active_status = 0;
  uint8_t current_pos = 0;
```

(continues on next page)

(continued from previous page)

```
  int8_t current = 0;
  int voltage = 0;
  int temp = 0;
  int8_t speed = 0;
  robot.SetGripperConfig(company, device, softversion, bus);
  std::this_thread::sleep_for(std::chrono::milliseconds(1000));
  robot.GetGripperConfig(&company, &device, &softversion, &bus);
  printf("gripper config:%d,%d,%d,%d\n", company, device, softversion, bus);
  robot.ActGripper(index, act);
  std::this_thread::sleep_for(std::chrono::milliseconds(1000));
  act = 1;
  robot.ActGripper(index, act);
  std::this_thread::sleep_for(std::chrono::milliseconds(1000));
  robot.MoveGripper(index, 100, 50, 50, max_time, block, 0, 0, 0, 0);
  std::this_thread::sleep_for(std::chrono::milliseconds(1000));
  robot.MoveGripper(index, 0, 50, 0, max_time, block, 0, 0, 0, 0);
  robot.GetGripperMotionDone(&fault, &status);
  printf("motion status:%u,%u\n", fault, status);
  robot.GetGripperActivateStatus(&fault, &active_status);
  printf("gripper active fault is: %u, status is: %u\n", fault, active_status);
  robot.GetGripperCurPosition(&fault, &current_pos);
  printf("fault is:%u, current position is: %u\n", fault, current_pos);
  robot.GetGripperCurCurrent(&fault, &current);
  printf("fault is:%u, current current is: %d\n", fault, current);
  robot.GetGripperVoltage(&fault, &voltage);
  printf("fault is:%u, current voltage is: %d \n", fault, voltage);
  robot.GetGripperTemp(&fault, &temp);
  printf("fault is:%u, current temperature is: %d\n", fault, temp);
  robot.GetGripperCurSpeed(&fault, &speed);
  printf("fault is:%u, current speed is: %d\n", fault, speed);
  int retval = 0;
  DescPose prepick_pose = {};
  DescPose postpick_pose = {};
  DescPose p1Desc(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
  DescPose p2Desc(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
  retval = robot.ComputePrePick(&p1Desc, 10, 0, &prepick_pose);
  printf("ComputePrePick retval is: %d\n", retval);
  printf("xyz is: %f, %f, %f; rpy is: %f, %f, %f\n", prepick_pose.tran.x, prepick_pose.
↪tran.y, prepick_pose.tran.z, prepick_pose.rpy.rx, prepick_pose.rpy.ry, prepick_pose.
↪rpy.rz);
  retval = robot.ComputePostPick(&p2Desc, -10, 0, &postpick_pose);
  printf("ComputePostPick retval is: %d\n", retval);
  printf("xyz is: %f, %f, %f; rpy is: %f, %f, %f\n", postpick_pose.tran.x, postpick_pose.
↪tran.y, postpick_pose.tran.z, postpick_pose.rpy.rx, postpick_pose.rpy.ry, postpick_
↪pose.rpy.rz);
  robot.CloseRPC();
  return 0;
}
```

2.1.11.15 Get Rotation Count of Rotary Gripper

New in version Version: 3.7.6

```
/**
     * @brief  Get rotation count of rotary gripper
     * @param  [out] fault  0-no error, 1-error
     * @param  [out] num  Rotation count
     * @return  Error code
     */
    errno_t GetGripperRotNum(uint16_t* fault, double* num);
```

2.1.11.16 Get Rotation Speed of Rotary Gripper

New in version V3.7.6.

```
/**
     * @brief  Get rotation speed of rotary gripper
     * @param  [out] fault  0-no error, 1-error
     * @param  [out] speed  Rotation speed percentage
     * @return  Error code
     */
    errno_t GetGripperRotSpeed(uint16_t* fault, int* speed);
```

2.1.11.17 Get Rotation Torque of Rotary Gripper

New in version V3.7.6.

```
/**
     * @brief  Get rotation torque of rotary gripper
     * @param  [out] fault  0-no error, 1-error
     * @param  [out] torque  Rotation torque percentage
     * @return  Error code
     */
    errno_t GetGripperRotTorque(uint16_t* fault, int* torque);
```

2.1.11.18 Rotary Gripper Status Code Example

```
int TestRotGripperState(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
```

(continues on next page)

(continued from previous page)

```
  uint16_t fault = 0;
  double rotNum = 0.0;
  int rotSpeed = 0;
  int rotTorque = 0;
  robot.GetGripperRotNum(&fault, &rotNum);
  robot.GetGripperRotSpeed(&fault, &rotSpeed);
  robot.GetGripperRotTorque(&fault, &rotTorque);
  printf("gripper rot num : %lf, gripper rotSpeed : %d, gripper rotTorque : %d\n",␣
↪rotNum, rotSpeed, rotTorque);
  robot.CloseRPC();
  return 0;
}
```

2.1.11.19 Conveyor Start/Stop

```
/**
* @brief Conveyor start/stop
* @param [in] status State, 1-start, 0-stop
* @return Error code
*/
errno_t ConveyorStartEnd(uint8_t status);
```

2.1.11.20 Record IO Detection Point

```
/**
* @brief Record IO detection point
* @return Error code
*/
errno_t ConveyorPointIORecord();
```

2.1.11.21 Record Point A

```
/**
* @brief Record point A
* @return Error code
*/
errno_t ConveyorPointARecord();
```

2.1.11.22 Record Reference Point

```
/**
* @brief Record reference point
* @return Error code
*/
errno_t ConveyorRefPointRecord();
```

2.1.11.23 Record Point B

```
/**
* @brief Record point B
* @return Error code
*/
errno_t ConveyorPointBRecord();
```

2.1.11.24 Conveyor Workpiece IO Detection

```
/**
* @brief Conveyor workpiece IO detection
* @param [in] max_t Maximum detection time, unit ms
* @return Error code
*/
errno_t ConveyorIODetect(int max_t);
```

2.1.11.25 Get Current Object Position

```
/**
* @brief Get current object position
* @param [in] mode
* @return Error code
*/
errno_t ConveyorGetTrackData(int mode);
```

2.1.11.26 Conveyor Tracking Start

```
/**
* @brief Conveyor tracking start
* @param [in] status State, 1-start, 0-stop
* @return Error code
*/
errno_t ConveyorTrackStart(uint8_t status);
```

2.1.11.27 Conveyor Tracking Stop

```
/**
* @brief Conveyor tracking stop
* @return Error code
*/
errno_t ConveyorTrackEnd();
```

2.1.11.28 Conveyor Parameter Configuration

Changed in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Conveyor parameter configuration
* @param [in] para[0] Encoder channel 1~2
* @param [in] para[1] Pulses per encoder revolution
* @param [in] para[2] Conveyor travel distance per encoder revolution
* @param [in] para[3] Workpiece coordinate system number (0 for tracking capture and TPD␣
↪tracking)
* @param [in] para[4] Vision configuration 0-not configured 1-configured
* @param [in] para[5] Speed ratio for conveyor tracking capture (1-100), default 1 for␣
↪other options
* @param [in] followType Tracking motion type, 0-tracking motion; 1-chase inspection␣
↪motion
* @param [in] startDis For chase capture: tracking start distance (-1: auto calculate),␣
↪unit mm, default 0
* @param [in] endDis For chase capture: tracking end distance, unit mm, default 100
* @return Error code
*/
errno_t ConveyorSetParam(float para[6], int followType = 0, int startDis = 0, int endDis␣
↪= 100);
```

2.1.11.29 Conveyor Capture Point Compensation

Changed in version C++SDK-v2.1.2.0.

```
    /**
     * @brief Conveyor capture point compensation
     * @param [in] cmp Compensation position double[3]{x, y, z}
     * @return Error code
     */
errno_t ConveyorCatchPointComp(double cmp[3]);
```

2.1.11.30 Conveyor Linear Motion

```
/**
* @brief Conveyor linear motion
* @param [in] status State, 1-start, 0-stop
* @return Error code
*/
errno_t TrackMoveL(char name[32], int tool, int wobj, float vel, float acc, float ovl,␣
↪float blendR, uint8_t flag, uint8_t type);
```

2.1.11.31 Conveyor Communication Input Detection

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Conveyor communication input detection
* @param [in] timeout Wait timeout time (ms)
* @return Error code
*/
errno_t ConveyorComDetect(int timeout);
```

2.1.11.32 Conveyor Communication Input Detection Trigger

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Conveyor communication input detection trigger
* @return Error code
*/
errno_t ConveyorComDetectTrigger();
```

2.1.11.33 Robot Conveyor Operation Example Program

```
int TestConveyor(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int retval = 0;
  retval = robot.ConveyorStartEnd(1);
  printf("ConveyorStartEnd retval is: %d\n", retval);
  retval = robot.ConveyorPointIORecord();
```

(continues on next page)

(continued from previous page)

```
  printf("ConveyorPointIORecord retval is: %d\n", retval);
  retval = robot.ConveyorPointARecord();
  printf("ConveyorPointARecord retval is: %d\n", retval);
  retval = robot.ConveyorRefPointRecord();
  printf("ConveyorRefPointRecord retval is: %d\n", retval);
  retval = robot.ConveyorPointBRecord();
  printf("ConveyorPointBRecord retval is: %d\n", retval);
  retval = robot.ConveyorStartEnd(0);
  printf("ConveyorStartEnd retval is: %d\n", retval);
  retval = 0;
  float param[6] = { 1,10000,200,0,0,20 };
  retval = robot.ConveyorSetParam(param);
  printf("ConveyorSetParam retval is: %d\n", retval);
  double cmp[3] = { 0.0, 0.0, 0.0 };
  retval = robot.ConveyorCatchPointComp(cmp);
  printf("ConveyorCatchPointComp retval is: %d\n", retval);
  int index = 1;
  int max_time = 30000;
  uint8_t block = 0;
  retval = 0;
  DescPose p1Desc(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
  DescPose p2Desc(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
  retval = robot.MoveCart(&p1Desc, 1, 0, 100.0, 100.0, 100.0, -1.0, -1);
  printf("MoveCart retval is: %d\n", retval);
  retval = robot.WaitMs(1);
  printf("WaitMs retval is: %d\n", retval);
  retval = robot.ConveyorIODetect(10000);
  printf("ConveyorIODetect retval is: %d\n", retval);
  retval = robot.ConveyorGetTrackData(1);
  printf("ConveyorGetTrackData retval is: %d\n", retval);
  retval = robot.ConveyorTrackStart(1);
  printf("ConveyorTrackStart retval is: %d\n", retval);
  retval = robot.TrackMoveL("cvrCatchPoint", 1, 0, 100, 100, 100, -1.0, 0, 0);
  printf("TrackMoveL retval is: %d\n", retval);
  retval = robot.MoveGripper(index, 51, 40, 30, max_time, block, 0, 0, 0, 0);
  printf("MoveGripper retval is: %d\n", retval);
  retval = robot.TrackMoveL("cvrRaisePoint", 1, 0, 100, 100, 100, -1.0, 0, 0);
  printf("TrackMoveL retval is: %d\n", retval);
  retval = robot.ConveyorTrackEnd();
  printf("ConveyorTrackEnd retval is: %d\n", retval);
  robot.MoveCart(&p2Desc, 1, 0, 100.0, 100.0, 100.0, -1.0, -1);
  retval = robot.MoveGripper(index, 100, 40, 10, max_time, block, 0, 0, 0, 0);
  printf("MoveGripper retval is: %d\n", retval);
  rtn = robot->ConveyorComDetect(1000 * 10);
  printf("ConveyorComDetect rtn is: %d\n", rtn);
  robot.Sleep(2000);
  rtn = robot->ConveyorComDetectTrigger();
  printf("ConveyorComDetectTrigger rtn is: %d\n", rtn);
  robot.CloseRPC();
  return 0;
}
```

2.1.11.34 End Sensor Configuration

New in version C++SDK-v2.1.5.0.

```
/**
* @brief End sensor configuration
* @param [in] idCompany Manufacturer, 18-JUNKONG; 25-HUIDE
* @param [in] idDevice Type, 0-JUNKONG/RYR6T.V1.0
* @param [in] idSoftware Software version, 0-J1.0/HuiDe1.0 (not yet available)
* @param [in] idBus Mounting position, 1-End port 1; 2-End port 2...8-End port 8 (not yet available)
* @return Error code
*/
errno_t AxleSensorConfig(int idCompany, int idDevice, int idSoftware, int idBus);
```

2.1.11.35 Get End Sensor Configuration

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Get end sensor configuration
 * @param [out] idCompany Manufacturer, 18-JUNKONG; 25-HUIDE
 * @param [out] idDevice Type, 0-JUNKONG/RYR6T.V1.0
 * @return Error code
 */
errno_t AxleSensorConfigGet(int& idCompany, int& idDevice);
```

2.1.11.36 End Sensor Activation

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief End sensor activation
 * @param [in] actFlag 0-Reset; 1-Activate
 * @return Error code
 */
errno_t AxleSensorActivate(int actFlag);
```

2.1.11.37 End Sensor Register Write

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief End sensor register write
 * @param [in] devAddr Device address number 0-255
 * @param [in] regHAddr Register address high 8 bits
 * @param [in] regLAddr Register address low 8 bits
 * @param [in] regNum Register count 0-255
 * @param [in] data1 Write register value 1
 * @param [in] data2 Write register value 2
```

(continues on next page)

(continued from previous page)

```
 * @param [in] isNoBlock 0-Blocking; 1-Non-blocking
 * @return Error code
 */
errno_t AxleSensorRegWrite(int devAddr, int regHAddr, int regLAddr, int regNum, int␣
↪data1, int data2, int isNoBlock);
```

2.1.11.38 End Sensor Code Example

```
int TestAxleSensor(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.AxleSensorConfig(18, 0, 0, 1);
  int company = -1;
  int type = -1;
  robot.AxleSensorConfigGet(company, type);
  printf("company is %d, type is %d\n", company, type);
  rtn = robot.AxleSensorActivate(1);
  printf("AxleSensorActivate rtn is %d\n", rtn);
  robot.Sleep(1000);
  rtn = robot.AxleSensorRegWrite(1, 4, 6, 1, 0, 0, 0);
  printf("AxleSensorRegWrite rtn is %d\n", rtn);
  robot.CloseRPC();
  return 0;
}
```

2.1.11.39 Get Robot Peripheral Protocol

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Get robot peripheral protocol
* @param [out] protocol Robot peripheral protocol number 4096-Extended axis control card;
↪ 4097-ModbusSlave; 4098-ModbusMaster
* @return Error code
*/
errno_t GetExDevProtocol(int *protocol);
```

2.1.11.40 Set Robot Peripheral Protocol

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set robot peripheral protocol
* @param [in] protocol Robot peripheral protocol number 4096-Extended axis control card; 4097-ModbusSlave; 4098-ModbusMaster
* @return Error code
*/
errno_t SetExDevProtocol(int protocol);
```

2.1.11.41 Set Robot Peripheral Protocol Example

New in version C++SDK-v2.1.3.0.

```
int TestExDevProtocol(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int protocol = 4096;
  rtn = robot.SetExDevProtocol(protocol);
  std::cout << "SetExDevProtocol rtn " << rtn << std::endl;
  rtn = robot.GetExDevProtocol(&protocol);
  std::cout << "GetExDevProtocol rtn " << rtn << " protocol is: " << protocol << std::endl;
  robot.CloseRPC();
  return 0;
}
```

2.1.11.42 Get End Communication Parameters

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get end communication parameters
* @param param End communication parameters
* @return  Error code
*/
errno_t GetAxleCommunicationParam(AxleComParam* param);
```

2.1.11.43 Set End Communication Parameters

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set end communication parameters
* @param param  End communication parameters
* @return  Error code
*/
errno_t SetAxleCommunicationParam(AxleComParam param);
```

2.1.11.44 Set End File Transfer Type

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set end file transfer type
* @param type 1-MCU upgrade file; 2-LUA file
* @return  Error code
*/
errno_t SetAxleFileType(int type);
```

2.1.11.45 Set Enable End LUA Execution

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set enable end LUA execution
* @param enable 0-Disable; 1-Enable
* @return  Error code
*/
errno_t SetAxleLuaEnable(int enable);
```

2.1.11.46 End LUA File Error Recovery

New in version C++SDK-v2.1.5.0.

```
/**
* @brief End LUA file error recovery
* @param status 0-No recovery; 1-Recovery
* @return  Error code
*/
errno_t SetRecoverAxleLuaErr(int status);
```

2.1.11.47 Get End LUA Execution Enable Status

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get end LUA execution enable status
* @param status status[0]: 0-Not enabled; 1-Enabled
* @return  Error code
*/
errno_t GetAxleLuaEnableStatus(int status[]);
```

2.1.11.48 Set End LUA End Device Enable Type

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set end LUA end device enable type
* @param forceSensorEnable Force sensor enable status, 0-Disable; 1-Enable
* @param gripperEnable Gripper enable status, 0-Disable; 1-Enable
* @param IOEnable IO device enable status, 0-Disable; 1-Enable
* @return  Error code
*/
errno_t SetAxleLuaEnableDeviceType(int forceSensorEnable, int gripperEnable, int␣
↪IOEnable);
```

2.1.11.49 Get End LUA End Device Enable Type

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get end LUA end device enable type
* @param enable enable[0]:forceSensorEnable Force sensor enable status, 0-Disable; 1-
↪Enable
* @param enable enable[1]:gripperEnable Gripper enable status, 0-Disable; 1-Enable
* @param enable enable[2]:IOEnable IO device enable status, 0-Disable; 1-Enable
* @return  Error code
*/
errno_t GetAxleLuaEnableDeviceType(int* forceSensorEnable, int* gripperEnable, int*␣
↪IOEnable);
```

2.1.11.50 Get Currently Configured End Devices

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get currently configured end devices
* @param forceSensorEnable Force sensor enable device number 0-Not enabled; 1-Enabled
* @param gripperEnable Gripper enable device number, 0-Disable; 1-Enable
* @param IODeviceEnable IO device enable device number, 0-Disable; 1-Enable
* @return  Error code
```

(continues on next page)

(continued from previous page)

```
*/
errno_t GetAxleLuaEnableDevice(int forceSensorEnable[], int gripperEnable[], int␣
↪IODeviceEnable[]);
```

2.1.11.51 Set Enable Gripper Action Control Function

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set enable gripper action control function
* @param id Gripper device number
* @param func func[0]-Gripper enable; func[1]-Gripper initialization; 2-Position setting;
↪ 3-Speed setting; 4-Torque setting; 6-Read gripper status; 7-Read initialization␣
↪status; 8-Read fault code; 9-Read position; 10-Read speed; 11-Read torque
* @return  Error code
*/
errno_t SetAxleLuaGripperFunc(int id, int func[]);
```

2.1.11.52 Get Enable Gripper Action Control Function

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get enable gripper action control function
* @param id Gripper device number
* @param func func[0]-Gripper enable; func[1]-Gripper initialization; 2-Position setting;
↪ 3-Speed setting; 4-Torque setting; 6-Read gripper status; 7-Read initialization␣
↪status; 8-Read fault code; 9-Read position; 10-Read speed; 11-Read torque
* @return  Error code
*/
errno_t GetAxleLuaGripperFunc(int id, int func[]);
```

2.1.11.53 Robot Ethercat Slave File Write

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Robot Ethercat slave file write
* @param type Slave file type, 1-Upgrade slave file; 2-Upgrade slave configuration file
* @param slaveID Slave number
* @param fileName Upload file name
* @return  Error code
*/
errno_t SlaveFileWrite(int type, int slaveID, std::string fileName);
```

2.1.11.54 Upload End Lua Open Protocol File

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Upload end Lua open protocol file
* @param filePath Local lua file path name ".../AXLE_LUA_End_DaHuan.lua"
* @return Error code
*/
errno_t AxleLuaUpload(std::string filePath);
```

2.1.11.55 Robot Ethercat Slave Enter Boot Mode

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Robot Ethercat slave enter boot mode
* @return  Error code
*/
errno_t SetSysServoBootMode();
```

2.1.11.56 Robot End LUA File Operation Code Example

```
int TestAxleLua(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.AxleLuaUpload("D://zUP/AXLE_LUA_End_DaHuan.lua");
  AxleComParam param(7, 8, 1, 0, 5, 3, 1);
  robot.SetAxleCommunicationParam(param);
  AxleComParam getParam;
  robot.GetAxleCommunicationParam(&getParam);
  printf("GetAxleCommunicationParam param is %d %d %d %d %d %d %d\n", getParam.baudRate,␣
↪getParam.dataBit, getParam.stopBit, getParam.verify, getParam.timeout, getParam.
↪timeoutTimes, getParam.period);
  robot.SetAxleLuaEnable(1);
  int luaEnableStatus = 0;
  robot.GetAxleLuaEnableStatus(&luaEnableStatus);
  robot.SetAxleLuaEnableDeviceType(0, 1, 0);
  int forceEnable = 0;
  int gripperEnable = 0;
  int ioEnable = 0;
  robot.GetAxleLuaEnableDeviceType(&forceEnable, &gripperEnable, &ioEnable);
```

(continues on next page)

(continued from previous page)

```
  printf("GetAxleLuaEnableDeviceType param is %d %d %d\n", forceEnable, gripperEnable, ioEnable);
  int func[16] = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
  robot.SetAxleLuaGripperFunc(1, func);
  int getFunc[16] = { 0 };
  robot.GetAxleLuaGripperFunc(1, getFunc);
  int getforceEnable[16] = { 0 };
  int getgripperEnable[16] = { 0 };
  int getioEnable[16] = { 0 };
  robot.GetAxleLuaEnableDevice(getforceEnable, getgripperEnable, getioEnable);
  printf("\ngetforceEnable status : ");
  for (int i = 0; i < 16; i++)
  {
    printf("%d,", getforceEnable[i]);
  }
  printf("\ngetgripperEnable status : ");
  for (int i = 0; i < 16; i++)
  {
    printf("%d,", getgripperEnable[i]);
  }
  printf("\ngetioEnable status : ");
  for (int i = 0; i < 16; i++)
  {
    printf("%d,", getioEnable[i]);
  }
  printf("\n");
  robot.ActGripper(1, 0);
  robot.Sleep(2000);
  robot.ActGripper(1, 1);
  robot.Sleep(2000);
  robot.MoveGripper(1, 90, 10, 100, 50000, 0, 0, 0, 0, 0);
  int pos = 0;
  while (true)
  {
    robot.GetRobotRealTimeState(&pkg);
    printf("gripper pos is %u\n", pkg.gripper_position);
    robot.Sleep(100);
  }
  robot.CloseRPC();
  return 0;
}
```

2.1.11.57 Get SmartTool Button Status

```
/**
* @brief Get SmartTool button status
* @param [out] state SmartTool handle button status; (bit0:0-Communication normal; 1-Communication lost; bit1-Undo; bit2-Clear program;
bit3-A button; bit4-B button; bit5-C button; bit6-D button; bit7-E button; bit8-IO button; bit9-Manual/Auto; bit10-Start)
* @return Error code
*/
errno_t GetSmarttoolBtnState(int& state);
```

2.1.11.58 SmartTool Button Code Example

```
int main(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;

  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  robot.SetReConnectParam(true, 30000, 500);

  while (true)
  {
    int btn = 0;
    robot.GetSmarttoolBtnState(btn);
    cout << "smarttool " << std::bitset<sizeof(btn) * 8>(btn) << endl;

    Sleep(100);
  }
}
```

2.1.11.59 Control Array Suction Cup

```
/**
* @brief Control Array Suction Cup
* @param [in] slaveID Slave Station ID
* @param [in] len Length
* @param [in] ctrlValue Control Value
* @return Error Code
*/
errno_t FRRobot::SetSuckerCtrl(uint8_t slaveID, uint8_t len, uint8_t ctrlValue[20]);
```

2.1.11.60 Get Array Suction Cup Status

```
/**
* @brief Get Array Suction Cup Status
* @param [in] slaveID Slave Station ID
* @param [out] state Adsorption Status 0-Release Object 1-Workpiece Detected, Adsorption Successful 2-No Object Adsorbed 3-Object Detached
* @param [out] pressValue Current Vacuum Level Unit kpa
* @param [out] error Current Error Code of the Suction Cup
* @return Error Code
*/
    errno_t FRRobot::GetSuckerState(uint8_t slaveID, uint8_t* state, int* pressValue, int* error);
```

2.1.11.61 Wait for Suction Cup Status

```
/**
* @brief Wait for Suction Cup Status
* @param [in] slaveID Slave Station ID
* @param [in] state Adsorption Status 0-Release Object 1-Workpiece Detected, Adsorption Successful 2-No Object Adsorbed 3-Object Detached
* @param [in] ms Maximum Wait Time (ms)
* @return Error Code
*/
errno_t FRRobot::WaitSuckerState(uint8_t slaveID, uint8_t state, int ms);
```

2.1.11.62 Array Suction Cup Control Command Code Example

```
void testSucker()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    uint8_t ctrl[20];
    uint8_t state;
    int pressValue; // Corrected typo from pressVlaue
    int error;

    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return;
    }
    robot.SetReConnectParam(true, 30000, 500);
    //Upload and load the open protocol file
    robot.OpenLuaUpload("E://Project/Peripheral SDK/CtrlDev_sucker.lua");
    robot.Sleep(2000);
    robot.SetCtrlOpenLUAName(1, "CtrlDev_sucker.lua");
    robot.UnloadCtrlOpenLUA(1);
```

(continues on next page)

(continued from previous page)

```
    robot.LoadCtrlOpenLUA(1);
    robot.Sleep(1000);

    //Control suction cup in broadcast mode, adsorb at maximum capacity
    ctrl[0] = 1;
    robot.SetSuckerCtrl(0, 1, ctrl);

    //Loop to monitor the status of suction cup 1 and suction cup 12
    for (int i = 0; i < 100; i++)
    {
        robot.GetSuckerState(1, &state, &pressValue, &error);
        printf("sucker1 state is %d, pressValue is %d, error num is %d\n", state,␣
↪pressValue, error);
        robot.GetSuckerState(12, &state, &pressValue, &error);
        printf("sucker12 state is %d, pressValue is %d, error num is %d\n", state,␣
↪pressValue, error);
        robot.Sleep(100);
    }

    //Wait for suction cup 1 to reach the state of having adsorbed an object, wait time␣
↪100ms
    int ret = robot.WaitSuckerState(1, 1, 100);
    printf("WaitSuckerState result is  %d\n", ret);

    //Turn off suction cup 1 and 12 in unicast mode
    ctrl[0] = 3;
    robot.SetSuckerCtrl(1, 1, ctrl);
    robot.SetSuckerCtrl(12, 1, ctrl);

    robot.CloseRPC();
}
```

2.1.11.63 Upload Peripheral Open Protocol LUA File

```
    /**
     * @brief Upload Lua File
     * @param [in] filePath Local lua file path name
     * @return Error Code
     */
errno_t OpenLuaUpload(std::string filePath);
```

2.1.11.64 Get Slave Board Parameters

```
/**
* @brief  Get Slave Board Parameters
* @param  [out] type  0-Ethercat1-CClink, 3-Ethercat, 4-EIP
* @param  [out] version  Protocol Version
* @param  [out] connState  0-Not Connected 1-Connected
* @return  Error Code
*/
errno_t GetFieldBusConfig(uint8_t* type, uint8_t* version, uint8_t* connState);
```

2.1.11.65 Write Slave DO

```
/**
* @brief  Write Slave DO
* @param  [in] DOIndex  DO Number
* @param  [in] writeNum  Number to Write // Corrected typo from wirteNum
* @param  [in] status[8] Value to Write, maximum 8
* @return  Error Code
*/
errno_t FieldBusSlaveWriteDO(uint8_t DOIndex, uint8_t writeNum, uint8_t status[8]);
```

2.1.11.66 Write Slave AO

```
/**
* @brief  Write Slave AO
* @param  [in] AOIndex  AO Number
* @param  [in] writeNum  Number to Write // Corrected typo from wirteNum
* @param  [in] status[8] Value to Write, maximum 8
* @return  Error Code
*/
errno_t FieldBusSlaveWriteAO(uint8_t AOIndex, uint8_t writeNum, int status[8]);
```

2.1.11.67 Read Slave DI

```
/**
* @brief  Read Slave DI
* @param  [in] DIIndex  DI Number // Corrected parameter name from DOIndex
* @param  [in] readNum  Number to Read // Corrected typo from readeNum
* @param  [out] status[8] Value Read, maximum 8
* @return  Error Code
*/
errno_t FieldBusSlaveReadDI(uint8_t DIIndex, uint8_t readNum, uint8_t status[8]);
```

2.1.11.68 Read Slave AI

```
/**
* @brief  Read Slave AI
* @param  [in] AIIndex  AI Number // Corrected parameter name from AOIndex
* @param  [in] readNum  Number to Read // Corrected typo from readeNum
* @param  [out] status[8] Value Read, maximum 8
* @return  Error Code
*/
errno_t FieldBusSlaveReadAI(uint8_t AIIndex, uint8_t readNum, int status[8]);
```

2.1.11.69 Wait for Extended DI Input

```
/**
* @brief Wait for Extended DI Input
* @param [in] DIIndex DI Number
* @param [in] status 0-Low Level; 1-High Level
* @param [in] waitMs Maximum Wait Time (ms)
* @return Error Code
*/
errno_t FRRobot::FieldBusSlaveWaitDI(uint8_t DIIndex, bool status, int waitMs);
```

2.1.11.70 Wait for Extended AI Input

```
/**
* @brief Wait for Extended AI Input
* @param [in] AIIndex AI Number
* @param [in] waitType 0-Greater Than; 1-Less Than
* @param [in] value AI Value
* @param [in] waitMs Maximum Wait Time (ms)
* @return Error Code
*/
errno_t FRRobot::FieldBusSlaveWaitAI(uint8_t AIIndex, uint8_t waitType, double value,␣
↪int waitMs);
```

2.1.11.71 Slave Mode Related Interface Command Code Example

```
void testFieldBusBoard()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    uint8_t type = 0, version = 0, connState = 0;
    uint8_t ctrl[8];
    int ctrlAO[8];
    static uint8_t DI[8];
    static int AI[8];

    robot.LoggerInit();
```

(continues on next page)

(continued from previous page)

```
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return;
    }
    robot.SetReConnectParam(true, 30000, 500);
    //Upload and load the open protocol file
    robot.OpenLuaUpload("E://Project/Peripheral SDK/CtrlDev_field.lua");
    robot.Sleep(2000);
    robot.SetCtrlOpenLUAName(3, "CtrlDev_field.lua");
    robot.UnloadCtrlOpenLUA(3);
    robot.LoadCtrlOpenLUA(3);
    robot.Sleep(8000);

    //Get the protocol type, software version, and connection status with the PLC of the␣
↪slave board
    robot.GetFieldBusConfig(&type, &version, &connState);
    printf("type is %d, version is %d,connState is %d\n", type, version, connState);

    //Write DO0 = 1, DO1 = 0, DO2 = 1
    ctrl[0] = 1; // Corrected value from 0 to 1 based on comment (DO0=1)
    ctrl[1] = 0;
    ctrl[2] = 1;
    robot.FieldBusSlaveWriteDO(0, 3, ctrl);

    //Write AO2 = 0x1005 (value from ctrlAO[0])
    ctrlAO[0] = 0x1005;
    robot.FieldBusSlaveWriteAO(2, 1, ctrlAO);

    //Loop to monitor DI0~DI3 AI0~AI2
    for (int i = 0; i < 100; i++)
    {
        robot.FieldBusSlaveReadDI(0, 4, DI);
        printf("DI0 is %d, DI1 is %d,DI2 is %d,DI3 is %d\n", DI[0], DI[1], DI[2], DI[3]);
        robot.FieldBusSlaveReadAI(0, 3, AI);
        printf("AI0 is %d, AI1 is %d,AI2 is %d\n", AI[0], AI[1], AI[2]);
        robot.Sleep(10);
    }

    //Wait for DI0 to become 1, wait time 100ms, and print the result
    int ret = robot.FieldBusSlaveWaitDI(0, 1, 100);
    printf("FieldBusSlaveWaitDI result is  %d\n", ret);

    //Wait for AI0 to be greater than 400, wait time 100ms, and print the result
    ret = robot.FieldBusSlaveWaitAI(0,0,400.00,100);
    printf("FieldBusSlaveWaitAI result is  %d\n", ret);

    robot.CloseRPC();
}
```

2.1.11.72 Laser Peripheral On/Off

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Peripheral On/Off Function
 * @param [in] OnOff 0- Off 1- On
 * @param [in] weldId Weld ID, default is 0
 * @return Error code
 */
errno_t LaserTrackingLaserOnOff(int OnOff,int weldId);
```

2.1.11.73 Laser Tracking Start/Stop

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Tracking Start/Stop Function
 * @param [in] OnOff 0- Stop 1- Start
 * @param [in] coordId Laser peripheral tool coordinate system ID
 * @return Error code
 */
 errno_t LaserTrackingTrackOnOff(int OnOff, int coordId);
```

2.1.11.74 Laser Seam Search Start - Fixed Direction

New in version C++SDK-v3.8.6.

```
/**
* @brief Laser Seam Search - Fixed Direction
* @param [in] direction 0- X+ 1- X- 2- Y+ 3- Y- 4- Z+ 5- Z-
* @param [in] vel Speed, unit %
* @param [in] distance Maximum search distance, unit mm
* @param [in] timeout Search timeout time, unit ms
* @param [in] posSensorNum Laser calibrated tool coordinate number
* @return Error code
*/
errno_t LaserTrackingSearchStart_xyz(int direction, int vel, int distance, int timeout,␣
↪int posSensorNum);
```

2.1.11.75 Laser Seam Search Start - Arbitrary Point Direction

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Seam Search - Arbitrary Direction
 * @param [in] directionPoint Search input point's xyz coordinates
 * @param [in] vel Speed, unit %
 * @param [in] distance Maximum search distance, unit mm
 * @param [in] timeout Search timeout time, unit ms
```

(continues on next page)

(continued from previous page)

```
 * @param [in] posSensorNum Laser calibrated tool coordinate number
 * @return Error code
 */
errno_t LaserTrackingSearchStart_point(DescTran directionPoint, int vel, int distance,int timeout, int posSensorNum);
```

2.1.11.76 Laser Seam Search Stop

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Seam Search Stop
 * @return Error code
 */
errno_t LaserTrackingSearchStop();
```

2.1.11.77 Laser Network Parameter Configuration

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Network Parameter Configuration
 * @param [in] ip Laser peripheral IP address
 * @param [in] port Laser peripheral port number
 * @return Error code
 */
errno_t LaserTrackingSensorConfig(std::string ip, int port);
```

2.1.11.78 Laser Peripheral Sampling Period Configuration

New in version C++SDK-v3.8.6.

```
/**
* @brief Laser Peripheral Sampling Period Configuration
* @param [in] period Laser peripheral sampling period, unit ms
* @return Error code
*/
errno_t LaserTrackingSensorSamplePeriod(int period);
```

2.1.11.79 Laser Peripheral Driver Load

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Peripheral Driver Load
 * @param [in] type Laser peripheral driver protocol type 101- Ruiniu 102- Chuangxiang103- Quanshi 104- Tongzhou 105- Aotai
 * @return Error code
```

(continues on next page)

(continued from previous page)

```
 */
errno_t LoadPosSensorDriver(int type);
```

2.1.11.80 Laser Peripheral Driver Unload

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Peripheral Driver Unload
 * @return Error code
 */
errno_t UnLoadPosSensorDriver();
```

2.1.11.81 Laser Seam Trajectory Recording

New in version C++SDK-v3.8.6.

```
/**
* @brief Laser Seam Trajectory Recording
* @param [in] status 0- Stop recording 1- Real-time tracking  2- Start recording
* @param [in] delayTime Delay time, unit ms
* @return Error code
*/
errno_t LaserSensorRecord1(int status, int delayTime);
```

2.1.11.82 Laser Seam Trajectory Replay

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Seam Trajectory Replay
 * @param [in] delayTime Delay time, unit ms
 * @param [in] speed Speed, unit %
 * @return Error code
 */
errno_t LaserSensorReplay(int delayTime, double speed);
```

2.1.11.83 Laser Tracking Replay

New in version C++SDK-v3.8.6.

```
/**
 * @brief Laser Tracking Replay
 * @return Error code
 */
errno_t MoveLTR();
```

2.1.11.84 Laser Seam Trajectory Recording and Replay

New in version C++SDK-v3.8.6.

```
/**
* @brief Laser Seam Trajectory Recording and Replay
* @param [in] delayMode Mode 0-Delay Time 1-Delay Distance
* @param [in] delayTime Delay time in milliseconds (ms)
* @param [in] delayDisExAxisNum Extended Axis Number
* @param [in] delayDis Delay distance in millimeters (mm)
* @param [in] sensitivePara Compensation Sensitivity Coefficient
* @param [in] trackMode Fixed-point Tracking Type. 0-Extended Axis Asynchronous Motion; 1-Robot
* @param [in] triggerMode Fixed-point Tracking Trigger Method. 0-Tracking Duration; 1-IO
* @param [in] runTime Robot Fixed-point Tracking Duration in seconds (s)
* @param [in] speed Speed in percentage (%)
* @return Error Code
*/
errno_t LaserSensorRecordandReplay(int delayMode, int delayTime, int delayDisExAxisNum, double delayDis, double sensitivePara, int trackMode, int triggerMode, double runTime, double speed);
```

2.1.11.85 Move to Laser Recorded Start Point

New in version C++SDK-v3.8.6.

```
/**
* @brief Move to Laser Recorded Start Point
* @param [in] moveType 0- moveJ 1- moveL
* @param [in] ovl Speed, unit %
* @return Error code
*/
errno_t MoveToLaserRecordStart(int moveType, double ovl);
```

2.1.11.86 Move to Laser Recorded End Point

New in version C++SDK-v3.8.6.

```
/**
* @brief Move to Laser Recorded End Point
* @param [in] moveType 0- moveJ 1- moveL
* @param [in] ovl Speed, unit %
* @return Error code
*/
errno_t MoveToLaserRecordEnd(int moveType, double ovl);
```

2.1.11.87 Move to Laser Sensor Seam Search Point

New in version C++SDK-v3.8.6.

```
/**
* @brief Move to Laser Sensor Seam Search Point
* @param [in] moveFlag Motion type: 0- PTP; 1- LIN
* @param [in] ovl Speed scaling factor, 0-100
* @param [in] dataFlag Seam buffer data selection: 0- Execute planned data; 1- Execute recorded data
* @param [in] plateType Plate type: 0- Corrugated plate; 1- Corrugated cardboard; 2- Fence plate; 3- Oil drum; 4- Corrugated shell steel
* @param [in] trackOffectType Laser sensor offset type: 0- No offset; 1- Base coordinate system offset; 2- Tool coordinate system offset; 3- Laser sensor raw data offset
* @param [in] offset Offset amount
* @return Error code
*/
errno_t MoveToLaserSeamPos(int moveFlag, double ovl, int dataFlag, int plateType, int trackOffectType, DescPose offset);
```

2.1.11.88 Get Laser Sensor Seam Search Point Coordinates

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Laser Sensor Seam Search Point Coordinate Information
* @param [in] trackOffectType Laser sensor offset type: 0- No offset; 1- Base coordinate system offset; 2- Tool coordinate system offset; 3- Laser sensor raw data offset
* @param [in] offset Offset amount
* @param [out] jPos Joint position [°]
* @param [out] descPos Cartesian position [mm]
* @param [out] tool Tool coordinate system
* @param [out] user Work object coordinate system
* @param [out] exaxis Extended axis position [mm]
* @return Error code
*/
errno_t GetLaserSeamPos(int trackOffectType, DescPose offset, JointPos& jPos, DescPose& descPos, int& tool, int& user, ExaxisPos& exaxis);
```

2.1.11.89 Laser Peripheral Sensor Parameter Configuration and Debugging Code Example

New in version C++SDK-v3.8.6.

```
void testLaserConfig()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    uint8_t ctrl[20];
    uint8_t state;
    int pressVlaue;
    int error;
```

(continues on next page)

(continued from previous page)

```
    robot.CloseRPC();
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return;
    }
    robot.SetReConnectParam(true, 30000, 500);
    //Set IP address and port number
    robot.LaserTrackingSensorConfig("192.168.58.20", 5020);
    //Set sampling period
    robot.LaserTrackingSensorSamplePeriod(20);
    //Load driver
    robot.LoadPosSensorDriver(101);
    //Turn off laser peripheral
    robot.LaserTrackingLaserOnOff(0,0);
    robot.Sleep(3000);
    //Turn on laser peripheral
    robot.LaserTrackingLaserOnOff(1, 0);
    robot.CloseRPC();
}
```

2.1.11.90 Laser Trajectory Scanning and Trajectory Replay Code Example

New in version C++SDK-v3.8.6.

```
void testLaserRecordAndReplay()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    uint8_t ctrl[20];
    uint8_t state;
    int pressVlaue;
    int error;
    robot.CloseRPC();
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return;
    }
    robot.SetReConnectParam(true, 30000, 500);

    //Upload and load open protocol file
    robot.OpenLuaUpload("E://openlua/CtrlDev_laser_ruiniu-0117.lua");
    robot.Sleep(2000);
    robot.SetCtrlOpenLUAName(0, "CtrlDev_laser_ruiniu-0117.lua");
    robot.UnloadCtrlOpenLUA(0);
    robot.LoadCtrlOpenLUA(0);
```

(continues on next page)

(continued from previous page)

```
    robot.Sleep(8000);
    int cnt = 1;
    while(cnt<31)
    {
        //Move to scan start point
        JointPos startjointPos(56.205, -117.951, 141.872, -118.149, -94.217, -122.176);
        DescPose startdescPose(-97.552, -282.855, 26.675, 174.182, -1.338, -91.707);
        ExaxisPos exaxisPos(0, 0, 0, 0);
        DescPose offdese(0, 0, 0, 0, 0, 0);
        robot.MoveL(&startjointPos, &startdescPose, 1, 0, 100, 100, 100, -1, &exaxisPos,␣
↪0, 0, &offdese, 1, 1);
        //Start trajectory recording
        robot.LaserSensorRecord1(2, 10);
        //Move to the end point to be recorded
        JointPos endjointPos(68.809, -87.100, 121.120, -127.233, -95.038, -109.555);
        DescPose enddescPose(-103.555, -464.234, 13.076, 174.179, -1.344, -91.709);
        robot.MoveL(&endjointPos, &enddescPose, 1, 0, 30, 100, 100, -1, &exaxisPos, 0, 0,
↪ &offdese, 1, 1);
        //Stop recording
        robot.LaserSensorRecord1(0, 10);
        //Move to the recorded seam start point
        robot.MoveToLaserRecordStart(1, 30);
        //Start trajectory replay
        robot.LaserSensorReplay(10, 100);
        robot.MoveLTR();
        //Stop trajectory replay
        robot.LaserSensorRecord1(0, 10);
        printf("Laser scanning + trajectory replay stability test, %dth time\n", cnt);
        cnt++;
    }
    robot.CloseRPC();
}
```

2.1.11.91 Laser Seam Search and Real-time Tracking Code Example

New in version C++SDK-v3.8.6.

```
void testLasertrack()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    uint8_t ctrl[20];
    uint8_t state;
    int pressVlaue;
    int error;
    robot.CloseRPC();
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");

    if (rtn != 0)
```

(continues on next page)

(continued from previous page)

```
    {
        return;
    }
    robot.SetReConnectParam(true, 30000, 500);

    //Upload and load open protocol file
    robot.OpenLuaUpload("E://openlua/CtrlDev_laser_ruiniu-0117.lua");
    robot.Sleep(2000);
    robot.SetCtrlOpenLUAName(0, "CtrlDev_laser_ruiniu-0117.lua");
    robot.UnloadCtrlOpenLUA(0);
    robot.LoadCtrlOpenLUA(0);
    robot.Sleep(8000);
    int cnt = 1;
    while (cnt < 2)
    {
        //Move to the starting point for seam search
        JointPos startjointPos(58.337, -119.628, 146.037, -116.358, -92.224, -117.654);
        DescPose startdescPose(-53.375, -255.363, 0.919, 178.054, 1.077, -94.026);
        ExaxisPos exaxisPos(0, 0, 0, 0);
        DescPose offdese(0, 0, 0, 0, 0, 0);
        DescTran directionPoint;
        robot.MoveL(&startjointPos, &startdescPose, 1, 0, 100, 100, 100, -1, &exaxisPos,␣
↪0, 0, &offdese, 1, 1);

        //Start search along the -Y direction
        int ret = robot.LaserTrackingSearchStart_xyz(3, 100, 300, 1000, 2);
        robot.LaserTrackingSearchStop();
        //If search is successful
        if (ret == 0)
        {
            //Move to the search point
            robot.MoveToLaserSeamPos(1, 30, 0, 0, 0, offdese);
            //Start laser tracking along the search point
            robot.LaserTrackingTrackOnOff(1, 2);
            JointPos endjointPos(70.580, -90.918, 126.593, -125.154, -92.162, -105.403);
            DescPose enddescPose(-53.375, -419.020, 0.920, 178.054, 1.076, -94.026);
            robot.MoveL(&endjointPos, &enddescPose, 1, 0, 20, 100, 100, -1, &exaxisPos,␣
↪0, 0, &offdese, 1, 1);
            //Stop tracking
            robot.LaserTrackingTrackOnOff(0, 2);

        }
        cnt++;
    }
    robot.CloseRPC();
}
```

2.1.11.92 Extended Axis and Robot Synchronized Laser Tracking Code Example

New in version C++SDK-v3.8.6.

```
void testLasertrackandExitAxis()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    uint8_t ctrl[20];
    uint8_t state;
    int pressVlaue;
    int error;
    robot.CloseRPC();
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");

    if (rtn != 0)
    {
        return;
    }
    robot.SetReConnectParam(true, 30000, 500);

    ExaxisPos startexaxisPos = { 0,0,0,0 };
    ExaxisPos seamexaxisPos = { -10,0,0,0 };
    ExaxisPos endexaxisPos = { -30, 0, 0, 0 };
    DescPose offdese = { 0, 0, 0, 0, 0, 0 };
    JointPos seamjointPos(0, 0, 0, 0, 0, 0);
    DescPose seamdescPose(0, 0, 0, 0, 0, 0);

    int cnt = 1;
    while (cnt < 31)
    {
        //Move to the starting point for seam search
        JointPos startjointPos(58.337, -119.628, 146.037, -116.358, -92.224, -117.654);
        DescPose startdescPose(-53.375, -255.363, 0.919, 178.054, 1.077, -94.026);
        robot.ExtAxisSyncMoveJ(startjointPos, startdescPose, 1, 0, 100, 100, 100,␣
↪startexaxisPos, -1, 0, offdese);

        //Start search along the -Y direction
        int ret = robot.LaserTrackingSearchStart_xyz(3, 100, 300, 1000, 2);
        robot.LaserTrackingSearchStop();
        int tool = 0;
        int user = 0;
        robot.GetLaserSeamPos(0, offdese, seamjointPos, seamdescPose, tool, user,␣
↪startexaxisPos);
        printf("%f, %f, %f,%f, %f, %f,%f, %f, %f,%f, %f, %f\n", seamjointPos.jPos[0],␣
↪seamjointPos.jPos[1], seamjointPos.jPos[2], seamjointPos.jPos[3], seamjointPos.jPos[4],
↪ seamjointPos.jPos[5], seamdescPose.tran.x, seamdescPose.tran.y, seamdescPose.tran.z,␣
↪seamdescPose.rpy.rx, seamdescPose.rpy.ry, seamdescPose.rpy.rz);

        //If search is successful
        if (ret == 0)
```

(continues on next page)

(continued from previous page)

```
        {
            //Robot and extended axis synchronously move to the search point
            robot.ExtAxisSyncMoveJ(seamjointPos, seamdescPose, 1, 0, 100, 100, 100,␣
↪seamexaxisPos, -1, 0, offdese);

            //Start laser tracking along the search point and synchronously move with␣
↪the extended axis
            robot.LaserTrackingTrackOnOff(1, 2);
            JointPos endjointPos(70.580, -90.918, 126.593, -125.154, -92.162, -105.403);
            DescPose enddescPose(-53.375, -419.020, 0.920, 178.054, 1.076, -94.026);
            robot.ExtAxisSyncMoveL(endjointPos, enddescPose, 1, 0, 20, 100, 100, -1,␣
↪endexaxisPos, 0, offdese);;
            //Stop tracking
            robot.LaserTrackingTrackOnOff(0, 2);
        }
        cnt++;
        printf("Extended axis and robot synchronized laser tracking, %dth time\n", cnt);
    }
    robot.CloseRPC();
}
```

2.1.12 Force control

2.1.12.1 Force sensor configuration

```
/**
* @brief force sensor configuration
* @param [in] company force sensor manufacturer, 17-Kunwei Technology, 19-Aerospace 11th␣
↪Institute, 20-ATI Sensor, 21-Zhongke Midian, 22-Weihang Minxin, 23-NBIT, 24-
↪Xinjingcheng (XJC), 26-NSR
* @param [in] device device number, Kunwei (0-KWR75B), Aerospace 11th Institute (0-MCS6A-
↪200-4), ATI (0-AXIA80-M8), Zhongke Midian (0-MST2010), Weihang Minxin (0-WHC6L-YB-10A),
↪ NBIT (0-XLH93003ACS), Xinjingcheng XJC (0-XJC-6F-D82), NSR (0-NSR-FTSensorA)
* @param [in] softvesion Software version number, not used yet, default is 0
* @param [in] bus device is hung at the end bus position, not used yet, default is 0
* @return error code
*/
errno_t  FT_SetConfig(int company, int device, int softvesion, int bus);
```

2.1.12.2 Get the force sensor configuration

```
/**
* @brief  Get the force sensor configuration
* @param  [in] company  Force sensor manufacturer, to be determined
* @param  [in] device  Device number, not used yet. The default value is 0
* @param  [in] softvesion  Software version. The value is not used. The default value is␣
↪0
* @param  [in] bus The device is attached to the terminal bus and is not in use. The␣
↪default value is 0
```

(continues on next page)

(continued from previous page)

```
* @return  Error code
*/
errno_t  FT_GetConfig(int *company, int *device, int *softvesion, int *bus);
```

2.1.12.3 Force Sensor Activation

```
/**
* @brief  Force sensor activation
* @param  [in] act  0- reset, 1- activate
* @return  Error code
*/
errno_t  FT_Activate(uint8_t act);
```

2.1.12.4 Force sensor calibration

```
/**
* @brief  Force sensor calibration
* @param  [in] act  0- zero removal, 1- zero correction
* @return  Error code
*/
errno_t  FT_SetZero(uint8_t act);
```

2.1.12.5 Set Force Sensor Reference Coordinate System

```
/**
* @brief  Set force sensor reference coordinate system
* @param  [in] ref  0-Tool coordinate system, 1-Base coordinate system
* @return  Error code
*/
errno_t  FT_SetRCS(uint8_t ref);
```

2.1.12.6 Set Payload Weight Under Force Sensor

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set payload weight under force sensor
 * @param [in] weight  Payload weight in kg
 * @return  Error code
 */
errno_t SetForceSensorPayload(double weight);
```

2.1.12.7 Set Payload Center of Gravity Under Force Sensor

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set payload center of gravity under force sensor
 * @param [in] x  Payload COG x in mm
 * @param [in] y  Payload COG y in mm
 * @param [in] z  Payload COG z in mm
 * @return  Error code
 */
errno_t SetForceSensorPayloadCog(double x, double y, double z);
```

2.1.12.8 Get Payload Weight Under Force Sensor

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Get payload weight under force sensor
 * @param [in] weight  Payload weight in kg
 * @return  Error code
 */
errno_t GetForceSensorPayload(double& weight);
```

2.1.12.9 Get Payload Center of Gravity Under Force Sensor

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Get payload center of gravity under force sensor
 * @param [out] x  Payload COG x in mm
 * @param [out] y  Payload COG y in mm
 * @param [out] z  Payload COG z in mm
 * @return  Error code
 */
errno_t GetForceSensorPayloadCog(double& x, double& y, double& z);
```

2.1.12.10 Force Sensor Auto Zero Calibration

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Force sensor auto zero calibration
 * @param [out] weight  Sensor mass in kg
 * @param [out] pos  Sensor COG in mm
 * @return  Error code
 */
errno_t ForceSensorAutoComputeLoad(double& weight, DescTran& pos);
```

2.1.12.11 Get Force/Torque Data in Reference Coordinate System

```
/**
* @brief  Get force/torque data in reference coordinate system
* @param  [out] ft  Force/torque, fx, fy, fz, tx, ty, tz
* @return  Error code
*/
errno_t  FT_GetForceTorqueRCS(ForceTorque *ft);
```

2.1.12.12 Get Raw Force/Torque Data from Force Sensor

```
/**
* @brief  Get raw force/torque data from force sensor
* @param  [out] ft  Force/torque, fx, fy, fz, tx, ty, tz
* @return  Error code
*/
errno_t  FT_GetForceTorqueOrigin(ForceTorque *ft);
```

2.1.12.13 Force Sensor Configuration and Auto Zero Calibration Code Example

```
int TestFTInit(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int company = 24;
  int device = 0;
  int softversion = 0;
  int bus = 1;
  int index = 1;
  robot.FT_SetConfig(company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_GetConfig(&company, &device, &softversion, &bus);
  printf("FT config:%d,%d,%d,%d\n", company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_Activate(0);
  robot.Sleep(1000);
  robot.FT_Activate(1);
  robot.Sleep(1000);
  robot.Sleep(1000);
  robot.FT_SetZero(0);
  robot.Sleep(1000);
  ForceTorque ft;
```

(continues on next page)

(continued from previous page)

```
  memset(&ft, 0, sizeof(ForceTorque));
  robot.FT_GetForceTorqueOrigin(0, &ft);
  printf("ft origin:%f,%f,%f,%f,%f,%f\n", ft.fx, ft.fy, ft.fz, ft.tx, ft.ty, ft.tz);
  robot.FT_SetZero(1);
  robot.Sleep(1000);
  DescPose ftCoord = {};
  robot.FT_SetRCS(0, ftCoord);
  robot.SetForceSensorPayload(0.824);
  robot.SetForceSensorPayloadCog(0.778, 2.554, 48.765);
  double weight = 0;
  double x = 0, y = 0, z = 0;
  robot.GetForceSensorPayload(weight);
  robot.GetForceSensorPayloadCog(x, y, z);
  printf("the FT load is %lf, %lf %lf %lf\n", weight, x, y, z);
  robot.SetForceSensorPayload(0);
  robot.SetForceSensorPayloadCog(0, 0, 0);
  double computeWeight = 0;
  DescTran tran = {};
  robot.ForceSensorAutoComputeLoad(weight, tran);
  cout << "the result is weight " << weight << " pos is " << tran.x << " " << tran.y <<
↪" " << tran.z << endl;
  robot.CloseRPC();
  return 0;
}
```

2.1.12.14 Payload Weight Identification Record

```
/**
* @brief  Payload weight identification record
* @param  [in] id  Sensor coordinate system number, range [1~14]
* @return  Error code
*/
errno_t  FT_PdIdenRecord(int id);
```

2.1.12.15 Payload Weight Identification Calculation

```
/**
* @brief  Payload weight identification calculation
* @param  [out] weight  Payload weight in kg
* @return  Error code
*/
errno_t  FT_PdIdenCompute(float *weight);
```

2.1.12.16 Payload COG Identification Record

```
/**
* @brief  Payload COG identification record
* @param  [in] id  Sensor coordinate system number, range [1~14]
* @param  [in] index  Point number, range [1~3]
* @return  Error code
*/
errno_t  FT_PdCogIdenRecord(int id, int index);
```

2.1.12.17 Payload COG Identification Calculation

```
/**
* @brief  Payload COG identification calculation
* @param  [out] cog  Payload COG in mm
* @return  Error code
*/
errno_t  FT_PdCogIdenCompute(DescTran *cog);
```

2.1.12.18 Force Sensor Payload Identification Code Example

```
int TestFTLoadCompute(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int company = 24;
  int device = 0;
  int softversion = 0;
  int bus = 1;
  int index = 1;
  robot.FT_SetConfig(company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_GetConfig(&company, &device, &softversion, &bus);
  printf("FT config:%d,%d,%d,%d\n", company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_Activate(0);
  robot.Sleep(1000);
  robot.FT_Activate(1);
  robot.Sleep(1000);
  robot.Sleep(1000);
  robot.FT_SetZero(0);
  robot.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
  ForceTorque ft;
  memset(&ft, 0, sizeof(ForceTorque));
  robot.FT_GetForceTorqueOrigin(0, &ft);
  printf("ft origin:%f,%f,%f,%f,%f,%f\n", ft.fx, ft.fy, ft.fz, ft.tx, ft.ty, ft.tz);
  robot.FT_SetZero(1);
  robot.Sleep(1000);
  DescPose tcoord = {};
  tcoord.tran.z = 35.0;
  robot.SetToolCoord(10, &tcoord, 1, 0, 0, 0);
  robot.FT_PdIdenRecord(10);
  robot.Sleep(1000);
  float weight = 0.0;
  robot.FT_PdIdenCompute(&weight);
  printf("payload weight:%f\n", weight);
  DescPose desc_p1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
  DescPose desc_p2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
  DescPose desc_p3(-327.622, 402.230, 320.402, -178.067, 2.127, -46.207);
  robot.MoveCart(&desc_p1, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.Sleep(1000);
  robot.FT_PdCogIdenRecord(10, 1);
  robot.MoveCart(&desc_p2, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.Sleep(1000);
  robot.FT_PdCogIdenRecord(10, 2);
  robot.MoveCart(&desc_p3, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.Sleep(1000);
  robot.FT_PdCogIdenRecord(10, 3);
  robot.Sleep(1000);
  DescTran cog;
  memset(&cog, 0, sizeof(DescTran));
  robot.FT_PdCogIdenCompute(&cog);
  printf("cog:%f,%f,%f\n", cog.x, cog.y, cog.z);
  robot.CloseRPC();
  return 0;
}
```

2.1.12.19 Collision Guard

```
/**
* @brief  Collision guard
* @param  [in] flag 0-Turn off collision guard, 1-Turn on collision guard
* @param  [in] sensor_id  Force sensor number
* @param  [in] select  Select whether to detect collision in six degrees of freedom, 0-
↪Do not detect, 1-Detect
* @param  [in] ft  Collision force/torque, fx, fy, fz, tx, ty, tz
* @param  [in] max_threshold  Maximum threshold
* @param  [in] min_threshold  Minimum threshold
* @note   Force/torque detection range: (ft-min_threshold, ft+max_threshold)
* @return  Error code
*/
errno_t  FT_Guard(uint8_t flag, int sensor_id, uint8_t select[6], ForceTorque *ft, float␣
↪max_threshold[6], float min_threshold[6]);
```

2.1.12.20 Collision Guard Code Example

```
int TestFTGuard(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int company = 24;
  int device = 0;
  int softversion = 0;
  int bus = 1;
  int index = 1;
  robot.FT_SetConfig(company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_GetConfig(&company, &device, &softversion, &bus);
  printf("FT config:%d,%d,%d,%d\n", company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_Activate(0);
  robot.Sleep(1000);
  robot.FT_Activate(1);
  robot.Sleep(1000);
  robot.Sleep(1000);
  robot.FT_SetZero(0);
  robot.Sleep(1000);
  uint8_t sensor_id = 1;
  uint8_t select[6] = { 1,1,1,1,1,1 };
  float max_threshold[6] = { 10.0,10.0,10.0,10.0,10.0,10.0 };
  float min_threshold[6] = { 5.0,5.0,5.0,5.0,5.0,5.0 };
  ForceTorque ft;
  DescPose desc_p1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
  DescPose desc_p2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
  DescPose desc_p3(-327.622, 402.230, 320.402, -178.067, 2.127, -46.207);
  robot.FT_Guard(1, sensor_id, select, &ft, max_threshold, min_threshold);
  robot.MoveCart(&desc_p1, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.MoveCart(&desc_p2, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.MoveCart(&desc_p3, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.FT_Guard(0, sensor_id, select, &ft, max_threshold, min_threshold);
  robot.CloseRPC();
  return 0;
}
```

2.1.12.21 Constant force control

```
/**
* @brief Constant Force Control
* @param [in] flag 0-disable constant force control, 1-enable constant force control
* @param [in] sensor_id Force sensor ID
* @param [in] select Select whether to detect collisions in six degrees of freedom, 0-disable detection, 1-enable detection
* @param [in] ft Collision force/torque: fx, fy, fz, tx, ty, tz
* @param [in] ft_pid Force PID parameters, torque PID parameters
* @param [in] adj_sign Adaptive start/stop control, 0-disable, 1-enable
* @param [in] ILC_sign ILC start/stop control, 0-stop, 1-training, 2-practical operation
* @param [in] max_dis Maximum adjustment distance, unit: mm
* @param [in] max_ang Maximum adjustment angle, unit: deg
* @param [in] M Mass parameter for rx, ry [0.1-10], default 2
* @param [in] B Damping parameter for rx, ry [0.1-50], default 8
* @param [in] threshold Activation threshold for rx, ry [0-10], default 0.2
* @param [in] adjustCoeff Torque adjustment coefficient for rx, ry [0-1], default 1
* @param [in] polishRadio Polishing radius, unit: mm
* @param [in] filter_Sign Filter enable flag 0-off; 1-on, default off
* @param [in] posAdapt_sign Posture compliance enable flag 0-off; 1-on, default off
* @param [in] isNoBlock Blocking flag, 0-blocking; 1-non-blocking
* @return Error code
*/
errno_t FT_Control(uint8_t flag, int sensor_id, uint8_t select[6], ForceTorque* ft, float ft_pid[6], uint8_t adj_sign,
uint8_t ILC_sign, float max_dis, float max_ang, double M[2], double B[2], double threshold[2], double adjustCoeff[2], double polishRadio = 0.0, int filter_Sign = 0, int posAdapt_sign = 0, int isNoBlock = 0);
```

2.1.12.22 Example of constant force control code with damping

```
int TestFTControlWithAdjustCoeff(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  uint8_t sensor_id = 10;
  uint8_t select[6] = { 0,0,1,0,0,0 };
  float ft_pid[6] = { 0.0008, 0.0, 0.0, 0.0, 0.0, 0.0 };
  uint8_t adj_sign = 0;
  uint8_t ILC_sign = 0;
  float max_dis = 1000.0;
  float max_ang = 20;
```

(continues on next page)

(continued from previous page)

```
  ForceTorque ft = {0.0};
  ExaxisPos epos(0, 0, 0, 0);
  JointPos j1(80.765, -98.795, 106.548, -97.734, -89.999, 94.842);
  JointPos j2(43.067, -84.429, 92.620, -98.175, -90.011, 57.144);
  DescPose desc_p1(5.009, -547.463, 262.053, -179.999, -0.019, 75.923);
  DescPose desc_p2(-347.966, -547.463, 262.048, -180.000, -0.019, 75.923);
  DescPose offset_pos(0, 0, 0, 0, 0, 0);
  double M[2] = { 2.0, 2.0 };
  double B[2] = { 15.0, 15.0 };
  double threshold[2] = {1.0, 1.0};
  double adjustCoeff[2] = {1.0, 0.8};
  double polishRadio = 0.0;
  int filter_Sign = 0;
  int posAdapt_sign = 1;
  int isNoBlock;
  ft.fz = -10.0;
  while(true)
  {
    rtn = robot.FT_Control(1, sensor_id, select, &ft, ft_pid, adj_sign, ILC_sign, max_
↪dis, max_ang, M, B, threshold, adjustCoeff, 0, 0, 1, 0);
    printf("FT_Control start rtn is %d\n", rtn);
    robot.MoveL(&j1, &desc_p1, 1, 0, 100, 100, 100, -1, 0, &epos, 0, 0, &offset_pos, 200.
↪0, 0);
    robot.MoveL(&j2, &desc_p2, 1, 0, 100, 100, 100, -1, 0, &epos, 0, 0, &offset_pos, 200.
↪0, 0);
    rtn = robot.FT_Control(0, sensor_id, select, &ft, ft_pid, adj_sign, ILC_sign, max_
↪dis, max_ang, M, B, threshold, adjustCoeff, 0, 0, 1, 0);
    printf("FT_Control end rtn is %d\n", rtn);
  }
  robot.CloseRPC();
  return 0;
}
```

2.1.12.23 Spiral Search

```
/**
* @brief  Spiral search
* @param  [in] rcs  Reference coordinate system, 0-Tool coordinate system, 1-Base␣
↪coordinate system
* @param  [in] dr  Radius feed per circle
* @param  [in] ft  Force/torque threshold, fx, fy, fz, tx, ty, tz, range [0~100]
* @param  [in] max_t_ms  Maximum search time in ms
* @param  [in] max_vel  Maximum linear velocity in mm/s
* @return  Error code
*/
errno_t  FT_SpiralSearch(int rcs, float dr, float ft, float max_t_ms, float max_vel);
```

2.1.12.24 Rotary Insertion

```
/**
* @brief  Rotary insertion
* @param  [in] rcs  Reference coordinate system, 0-Tool coordinate system, 1-Base coordinate system
* @param  [in] angVelRot  Rotational angular velocity in deg/s
* @param  [in] ft  Force/torque threshold, fx, fy, fz, tx, ty, tz, range [0~100]
* @param  [in] max_angle  Maximum rotation angle in deg
* @param  [in] orn  Force/torque direction, 1-Along z-axis, 2-Around z-axis
* @param  [in] max_angAcc  Maximum rotational acceleration in deg/s^2, not currently used, default is 0
* @param  [in] rotorn  Rotation direction, 1-Clockwise, 2-Counterclockwise
* @param  [in] strategy Processing strategy for undetected force/torque, 0-Error; 1-Warning, continue motion
* @return  Error code
*/
errno_t FT_RotInsertion(int rcs, float angVelRot, float ft, float max_angle, uint8_t orn, float max_angAcc, uint8_t rotorn, int strategy = 0);
```

2.1.12.25 Example code for force sensor rotation insertion

```
int TestMove(void)
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(1);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return -1;
    }
    robot.SetReConnectParam(true, 30000, 500);
    JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3(-29.777, -84.536, 109.275, -114.075, -86.655, 74.257);
    JointPos j4(-31.154, -95.317, 94.276, -88.079, -89.740, 74.256);
    DescPose desc_pos1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose desc_pos3(-487.434, 154.362, 308.576, 176.600, 0.268, -14.061);
    DescPose desc_pos4(-443.165, 147.881, 480.951, 179.511, -0.775, -15.409);
    DescPose offset_pos(0, 0, 0, 0, 0, 0);
    ExaxisPos epos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float oacc = 100.0;
    float blendT = 0.0;
```

(continues on next page)

(continued from previous page)

```
    float blendR = 0.0;
    uint8_t flag = 0;
    uint8_t search = 0;
    int blendMode = 0;
    int velAccMode = 0;
    robot.SetSpeed(20);
    rtn = robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", rtn);
    rtn = robot.MoveL(&j2, &desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, &epos, search, flag, &offset_pos, oacc, velAccMode);
    printf("movel errcode:%d\n", rtn);
    rtn = robot.MoveC(&j3, &desc_pos3, tool, user, vel, acc, &epos, flag, &offset_pos, &j4, &desc_pos4, tool, user, vel, acc, &epos, flag, &offset_pos, ovl, blendR, oacc, velAccMode);
    printf("movec errcode:%d\n", rtn);
    rtn = robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", rtn);
    rtn = robot.Circle(&j3, &desc_pos3, tool, user, vel, acc, &epos, &j1, &desc_pos1, tool, user, vel, acc, &epos, ovl, flag, &offset_pos, oacc, -1, velAccMode);
    printf("circle errcode:%d\n", rtn);
    rtn = robot.MoveCart(&desc_pos4, tool, user, vel, acc, ovl, blendT, -1);
    printf("MoveCart errcode:%d\n", rtn);
    rtn = robot.MoveJ(&j1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", rtn);
    rtn = robot.MoveL(&desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, &epos, search, flag, &offset_pos, -1, velAccMode);
    printf("movel errcode:%d\n", rtn);
    rtn = robot.MoveC(&desc_pos3, tool, user, vel, acc, &epos, flag, &offset_pos, &desc_pos4, tool, user, vel, acc, &epos, flag, &offset_pos, ovl, blendR, -1, velAccMode);
    printf("movec errcode:%d\n", rtn);
    rtn = robot.MoveJ(&j2, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", rtn);
    rtn = robot.Circle(&desc_pos3, tool, user, vel, acc, &epos, &desc_pos1, tool, user, vel, acc, &epos, ovl, flag, &offset_pos, oacc, blendR, -1, velAccMode);
    printf("circle errcode:%d\n", rtn);
    robot.CloseRPC();
    return 0;
}
```

2.1.12.26 Linear Insertion

```
/**
* @brief  Linear insertion
* @param  [in] rcs  Reference coordinate system, 0-Tool coordinate system, 1-Base coordinate system
* @param  [in] ft  Force/torque threshold, fx, fy, fz, tx, ty, tz, range [0~100]
* @param  [in] lin_v  Linear velocity in mm/s
* @param  [in] lin_a  Linear acceleration in mm/s^2, not currently used
```

(continues on next page)

(continued from previous page)

```
* @param  [in] max_dis  Maximum insertion distance in mm
* @param  [in] linorn  Insertion direction, 0-Negative direction, 1-Positive direction
* @return  Error code
*/
errno_t  FT_LinInsertion(int rcs, float ft, float lin_v, float lin_a, float max_dis,␣
↪uint8_t linorn);
```

2.1.12.27 Spiral Search, Linear Insertion and Other Instruction Code Examples

```
int TestFTSearch(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int company = 24;
  int device = 0;
  int softversion = 0;
  int bus = 1;
  int index = 1;
  robot.FT_SetConfig(company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_GetConfig(&company, &device, &softversion, &bus);
  printf("FT config:%d,%d,%d,%d\n", company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_Activate(0);
  robot.Sleep(1000);
  robot.FT_Activate(1);
  robot.Sleep(1000);
  robot.Sleep(1000);
  robot.FT_SetZero(0);
  robot.Sleep(1000);
  uint8_t status = 1;
  int sensor_num = 1;
  float gain[6] = { 0.0001,0.0,0.0,0.0,0.0,0.0 };
  uint8_t adj_sign = 0;
  uint8_t ILC_sign = 0;
  float max_dis = 100.0;
  float max_ang = 5.0;
  ForceTorque ft;
  memset(&ft, 0, sizeof(ForceTorque));
  int rcs = 0;
  float dr = 0.7;
  float fFinish = 1.0;
```

(continues on next page)

(continued from previous page)

```
  float t = 60000.0;
  float vmax = 3.0;
  float force_goal = 20.0;
  float lin_v = 0.0;
  float lin_a = 0.0;
  float disMax = 100.0;
  uint8_t linorn = 1;
  float angVelRot = 2.0;
  float forceInsertion = 1.0;
  int angleMax = 45;
  uint8_t orn = 1;
  float angAccmax = 0.0;
  uint8_t rotorn = 1;
  uint8_t select1[6] = { 0,0,1,1,1,0 };
  ft.fz = -10.0;
  robot.FT_Control(status, sensor_num, select1, &ft, gain, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  rtn = robot.FT_SpiralSearch(rcs, dr, fFinish, t, vmax);
  printf("FT_SpiralSearch rtn is %d\n", rtn);
  status = 0;
  robot.FT_Control(status, sensor_num, select1, &ft, gain, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  uint8_t select2[6] = { 1,1,1,0,0,0 };
  gain[0] = 0.00005;
  ft.fz = -30.0;
  status = 1;
  robot.FT_Control(status, sensor_num, select2, &ft, gain, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  rtn = robot.FT_LinInsertion(rcs, force_goal, lin_v, lin_a, disMax, linorn);
  printf("FT_LinInsertion rtn is %d\n", rtn);
  status = 0;
  robot.FT_Control(status, sensor_num, select2, &ft, gain, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  uint8_t select3[6] = { 0,0,1,1,1,0 };
  ft.fz = -10.0;
  gain[0] = 0.0001;
  status = 1;
  robot.FT_Control(status, sensor_num, select3, &ft, gain, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  rtn = robot.FT_RotInsertion(rcs, angVelRot, forceInsertion, angleMax, orn, angAccmax,␣
↪rotorn);
  printf("FT_RotInsertion rtn is %d\n", rtn);
  status = 0;
  robot.FT_Control(status, sensor_num, select3, &ft, gain, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  uint8_t select4[6] = { 1,1,1,0,0,0 };
  ft.fz = -30.0;
  status = 1;
  robot.FT_Control(status, sensor_num, select4, &ft, gain, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  rtn = robot.FT_LinInsertion(rcs, force_goal, lin_v, lin_a, disMax, linorn);
  printf("FT_LinInsertion rtn is %d\n", rtn);
```

(continues on next page)

(continued from previous page)

```
  status = 0;
  robot.FT_Control(status, sensor_num, select4, &ft, gain, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  robot.CloseRPC();
  return 0;
}
```

2.1.12.28 Surface Localization

```
/**
* @brief  Surface localization
* @param  [in] rcs  Reference coordinate system, 0-Tool coordinate system, 1-Base␣
↪coordinate system
* @param  [in] dir  Movement direction, 1-Positive direction, 2-Negative direction
* @param  [in] axis  Movement axis, 1-x axis, 2-y axis, 3-z axis
* @param  [in] lin_v  Search linear velocity in mm/s
* @param  [in] lin_a  Search linear acceleration in mm/s^2, not currently used, default␣
↪is 0
* @param  [in] max_dis  Maximum search distance in mm
* @param  [in] ft  Action termination force/torque threshold, fx, fy, fz, tx, ty, tz
* @return  Error code
*/
errno_t  FT_FindSurface(int rcs, uint8_t dir, uint8_t axis, float lin_v, float lin_a,␣
↪float max_dis, float ft);
```

2.1.12.29 Calculate Middle Plane Position - Start

```
/**
* @brief  Calculate middle plane position - start
* @return  Error code
*/
errno_t  FT_CalCenterStart();
```

2.1.12.30 Calculate Middle Plane Position - End

```
/**
* @brief  Calculate middle plane position - end
* @param  [out] pos  Middle plane pose
* @return  Error code
*/
errno_t  FT_CalCenterEnd(DescPose *pos);
```

2.1.12.31 Surface Localization Code Example

```
int TestSurface(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int company = 24;
  int device = 0;
  int softversion = 0;
  int bus = 1;
  int index = 1;
  robot.FT_SetConfig(company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_GetConfig(&company, &device, &softversion, &bus);
  printf("FT config:%d,%d,%d,%d\n", company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_Activate(0);
  robot.Sleep(1000);
  robot.FT_Activate(1);
  robot.Sleep(1000);
  robot.Sleep(1000);
  robot.FT_SetZero(0);
  robot.Sleep(1000);
  int rcs = 0;
  uint8_t dir = 1;
  uint8_t axis = 1;
  float lin_v = 3.0;
  float lin_a = 0.0;
  float maxdis = 50.0;
  float ft_goal = 2.0;
  DescPose desc_pos(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
  DescPose xcenter(0, 0, 0, 0, 0, 0);
  DescPose ycenter(0, 0, 0, 0, 0, 0);
  ForceTorque ft;
  memset(&ft, 0, sizeof(ForceTorque));
  ft.fx = -2.0;
  robot.MoveCart(&desc_pos, 9, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.FT_CalCenterStart();
  robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal);
  robot.MoveCart(&desc_pos, 9, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.WaitMs(1000);
  dir = 2;
  robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal);
  robot.FT_CalCenterEnd(&xcenter);
  printf("xcenter:%f,%f,%f,%f,%f,%f\n", xcenter.tran.x, xcenter.tran.y, xcenter.tran.z,␣
```

(continues on next page)

(continued from previous page)

```
↪xcenter.rpy.rx, xcenter.rpy.ry, xcenter.rpy.rz);
  robot.MoveCart(&xcenter, 9, 0, 60.0, 50.0, 50.0, -1.0, -1);
  robot.FT_CalCenterStart();
  dir = 1;
  axis = 2;
  lin_v = 6.0;
  maxdis = 150.0;
  robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal);
  robot.MoveCart(&desc_pos, 9, 0, 100.0, 100.0, 100.0, -1.0, -1);
  robot.WaitMs(1000);
  dir = 2;
  robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal);
  robot.FT_CalCenterEnd(&ycenter);
  printf("ycenter:%f,%f,%f,%f,%f,%f\n", ycenter.tran.x, ycenter.tran.y, ycenter.tran.z,␣
↪ycenter.rpy.rx, ycenter.rpy.ry, ycenter.rpy.rz);
  robot.MoveCart(&ycenter, 9, 0, 60.0, 50.0, 50.0, 0.0, -1);
  robot.CloseRPC();
  return 0;
}
```

2.1.12.32 Compliance Control Activation

```
/**
* @brief  Compliance control activation
* @param  [in] p  Position adjustment coefficient or compliance coefficient
* @param  [in] force  Compliance activation force threshold in N
* @return  Error code
*/
errno_t  FT_ComplianceStart(float p, float force);
```

2.1.12.33 Compliance Control Deactivation

```
/**
* @brief  Compliance control deactivation
* @return  Error code
*/
errno_t  FT_ComplianceStop();
```

2.1.12.34 Compliance Control Code Example

```
int TestCompliance(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
```

(continues on next page)

(continued from previous page)

```
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int company = 24;
  int device = 0;
  int softversion = 0;
  int bus = 1;
  int index = 1;
  robot.FT_SetConfig(company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_GetConfig(&company, &device, &softversion, &bus);
  printf("FT config:%d,%d,%d,%d\n", company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_Activate(0);
  robot.Sleep(1000);
  robot.FT_Activate(1);
  robot.Sleep(1000);
  robot.Sleep(1000);
  robot.FT_SetZero(0);
  robot.Sleep(1000);
  uint8_t flag = 1;
  int sensor_id = 1;
  uint8_t select[6] = { 1,1,1,0,0,0 };
  float ft_pid[6] = { 0.0005,0.0,0.0,0.0,0.0,0.0 };
  uint8_t adj_sign = 0;
  uint8_t ILC_sign = 0;
  float max_dis = 100.0;
  float max_ang = 0.0;
  ForceTorque ft;
  DescPose offset_pos(0, 0, 0, 0, 0, 0);
  ExaxisPos epos(0, 0, 0, 0);
  memset(&ft, 0, sizeof(ForceTorque));
  JointPos j1(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
  JointPos j2(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
  DescPose desc_p1(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
  DescPose desc_p2(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
  ft.fx = -10.0;
  ft.fy = -10.0;
  ft.fz = -10.0;
  robot.FT_Control(flag, sensor_id, select, &ft, ft_pid, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  float p = 0.00005;
  float force = 30.0;
  rtn = robot.FT_ComplianceStart(p, force);
  printf("FT_ComplianceStart rtn is %d\n", rtn);
  int count = 15;
  while (count)
  {
    robot.MoveL(&j1, &desc_p1, 0, 0, 100.0, 180.0, 100.0, -1.0, &epos, 0, 1, &offset_
↪pos);
```

(continues on next page)

(continued from previous page)

```
    robot.MoveL(&j2, &desc_p2, 0, 0, 100.0, 180.0, 100.0, -1.0, &epos, 0, 0, &offset_
↪pos);
    count -= 1;
  }
  robot.FT_ComplianceStop();
  printf("FT_ComplianceStop rtn is %d\n", rtn);
  flag = 0;
  robot.FT_Control(flag, sensor_id, select, &ft, ft_pid, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
  robot.CloseRPC();
  return 0;
}
```

2.1.12.35 Payload Identification Initialization

```
/**
* @brief Payload identification initialization
* @return Error code
*/
errno_t LoadIdentifyDynFilterInit();
```

2.1.12.36 Payload Identification Initialization

```
/**
* @brief Payload identification initialization
* @return Error code
*/
errno_t LoadIdentifyDynVarInit();
```

2.1.12.37 Payload Identification Main Program

```
/**
* @brief Payload identification main program
* @param [in] joint_torque  Joint torque
* @param [in] joint_pos  Joint position
* @param [in] t  Sampling period
* @return Error code
*/
errno_t LoadIdentifyMain(double joint_torque[6], double joint_pos[6], double t);
```

2.1.12.38 Get Payload Identification Result

```
/**
* @brief Get payload identification result
* @param [in] gain
* @param [out] weight  Payload weight
* @param [out] cog  Payload COG
* @return Error code
*/
errno_t LoadIdentifyGetResult(double gain[12], double *weight, DescTran *cog);
```

2.1.12.39 Robot Payload Identification Code Example

```
int TestIdentify(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int retval = 0;
  retval = robot.LoadIdentifyDynFilterInit();
  printf("LoadIdentifyDynFilterInit retval is: %d \n", retval);
  retval = robot.LoadIdentifyDynVarInit();
  printf("LoadIdentifyDynVarInit retval is: %d \n", retval);
  JointPos posJ = {};
  DescPose posDec = {};
  float joint_toq[6] = { 0.0 };
  robot.GetActualJointPosDegree(0, &posJ);
  posJ.jPos[1] = posJ.jPos[1] + 10;
  robot.GetJointTorques(0, joint_toq);
  joint_toq[1] = joint_toq[1] + 2;
  double tmpTorque[6] = { 0.0 };
  for (int i = 0; i < 6; i++)
  {
    tmpTorque[i] = joint_toq[i];
  }
  retval = robot.LoadIdentifyMain(tmpTorque, posJ.jPos, 1);
  printf("LoadIdentifyMain retval is: %d \n", retval);
  double gain[12] = { 0,0.05,0,0,0,0,0,0.02,0,0,0,0 };
  double weight = 0;
  DescTran load_pos;
  memset(&load_pos, 0, sizeof(DescTran));
  retval = robot.LoadIdentifyGetResult(gain, &weight, &load_pos);
  printf("LoadIdentifyGetResult retval is: %d ; weight is %f cog is %f %f %f \n", retval,
↪ weight, load_pos.x, load_pos.y, load_pos.z);
```

(continues on next page)

(continued from previous page)

```
  robot.CloseRPC();
  return 0;
}
```

2.1.12.40 Force Sensor Assisted Drag

Changed in version C++SDK-v2.2.0-3.8.0.

```
/**
      * @brief  Force sensor assisted drag
      * @param  [in] status  Control status, 0-Off; 1-On
      * @param  [in] asaptiveFlag  Adaptive activation flag, 0-Off; 1-On
      * @param  [in] interfereDragFlag  Interference area drag flag, 0-Off; 1-On
      * @param  [in] ingularityConstraintsFlag  Singularity strategy, 0-Avoid; 1-Cross
      * @param  [in] forceCollisionFlag Robot collision detection sign for auxiliary␣
↪dragging; 0-off; 1-on
      * @param  [in] M  Inertia coefficient
      * @param  [in] B  Damping coefficient
      * @param  [in] K  Stiffness coefficient
      * @param  [in] F  Drag six-dimensional force threshold
      * @param  [in] Fmax  Maximum drag force limit
      * @param  [in] Vmax  Maximum joint velocity limit
  * @return  error code
  */
  errno_t EndForceDragControl(int status, int asaptiveFlag, int interfereDragFlag, int␣
↪ingularityConstraintsFlag, int forceCollisionFlag, std::vector<double> M, std::vector
↪<double> B, std::vector<double> K, std::vector<double> F, double Fmax, double Vmax);
```

2.1.12.41 Get Force Sensor Drag Switch Status

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Get force sensor drag switch status
 * @param [out] dragState  Force sensor assisted drag control status, 0-Off; 1-On
 * @param [out] sixDimensionalDragState  Six-dimensional force assisted drag control␣
↪status, 0-Off; 1-On
 * @return Error code
 */
errno_t GetForceAndTorqueDragState(int& dragState, int& sixDimensionalDragState);
```

2.1.12.42 Force Sensor Auto Activation After Error Clear

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Auto activate force sensor after error clear
 * @param [in] status  Control status, 0-Off; 1-On
 * @return Error code
 */
errno_t SetForceSensorDragAutoFlag(int status);
```

2.1.12.43 Force Sensor Assisted Drag Code Example

```
int TestEndForceDragCtrl(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.SetForceSensorDragAutoFlag(1);
  vector <double> M = { 15.0, 15.0, 15.0, 0.5, 0.5, 0.1 };
  vector <double> B = { 150.0, 150.0, 150.0, 5.0, 5.0, 1.0 };
  vector <double> K = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
  vector <double> F = { 10.0, 10.0, 10.0, 1.0, 1.0, 1.0 };
  robot.EndForceDragControl(1, 0, 0, 0, 1, M, B, K, F, 50, 100);
  robot.Sleep(5000);
  int dragState = 0;
  int sixDimensionalDragState = 0;
  robot.GetForceAndTorqueDragState(dragState, sixDimensionalDragState);
  printf("the drag state is %d %d \n", dragState, sixDimensionalDragState);
  robot.EndForceDragControl(0, 0, 0, 0, 1, M, B, K, F, 50, 100);
  robot.CloseRPC();
  return 0;
}
```

2.1.12.44 Set Six-Dimensional Force and Joint Impedance Hybrid Drag Switch and Parameters

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set six-dimensional force and joint impedance hybrid drag switch and parameters
 * @param [in] status  Control status, 0-Off; 1-On
 * @param [in] impedanceFlag  Impedance activation flag, 0-Off; 1-On
 * @param [in] lamdeDain  Drag gain
 * @param [in] KGain  Stiffness gain
```

(continues on next page)

(continued from previous page)

```
 * @param [in] BGain  Damping gain
 * @param [in] dragMaxTcpVel  Maximum end linear velocity limit for drag
 * @param [in] dragMaxTcpOriVel  Maximum end angular velocity limit for drag
 * @return Error code
 */
errno_t ForceAndJointImpedanceStartStop(int status, int impedanceFlag, std::vector
↪<double> lamdeDain, std::vector<double> KGain, std::vector<double> BGain, double␣
↪dragMaxTcpVel, double dragMaxTcpOriVel);
```

2.1.12.45 Six-Dimensional Force and Joint Impedance Hybrid Drag Code Example

New in version C++SDK-v2.1.5.0.

```
int TestForceAndJointImpedance(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.DragTeachSwitch(1);
  vector <double> lamdeDain = { 3.0, 2.0, 2.0, 2.0, 2.0, 3.0 };
  vector <double> KGain = { 0, 0, 0, 0, 0, 0 };
  vector <double> BGain = { 150, 150, 150, 5.0, 5.0, 1.0 };
  rtn = robot.ForceAndJointImpedanceStartStop(1, 0, lamdeDain, KGain, BGain, 1000, 180);
  printf("ForceAndJointImpedanceStartStop rtn is %d\n", rtn);
  robot.Sleep(5000);
  robot.DragTeachSwitch(0);
  rtn = robot.ForceAndJointImpedanceStartStop(0, 0, lamdeDain, KGain, BGain, 1000, 180);
  printf("ForceAndJointImpedanceStartStop rtn is %d\n", rtn);
  robot.CloseRPC();
  return 0;
}
```

2.1.12.46 Impedance Start/Stop Control

New in version C++SDK-v3.8.6.

```
/**
* @brief Impedance Start/Stop Control
* @param [in] status 0: Stop; 1- Start
* @param [in] workSpace 0- Joint Space; 1- Cartesian Space
* @param [in] forceThreshold Trigger force threshold (N)
* @param [in] m Mass parameter
* @param [in] b Damping parameter
```

(continues on next page)

(continued from previous page)

```
* @param [in] k Stiffness parameter
* @param [in] maxV Maximum linear velocity (mm/s)
* @param [in] maxVA Maximum linear acceleration (mm/s^2)
* @param [in] maxW Maximum angular velocity (°/s)
* @param [in] maxWA Maximum angular acceleration (°/s^2)
* @return Error code
*/
errno_t ImpedanceControlStartStop(int status, int workSpace, double forceThreshold[6],
↪double m[6], double b[6], double k[6], double maxV, double maxVA, double maxW, double
↪maxWA);
```

2.1.12.47 Robot Impedance Start/Stop Control Code Example

New in version C++SDK-v3.8.6.

```
int TestImpedanceControl()
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  uint8_t ctrl[20];
  uint8_t state;
  int pressVlaue;
  int error;
  robot.CloseRPC();
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return 0;
  }
  robot.SetReConnectParam(true, 30000, 500);
  JointPos j1(102.622, -135.990, 120.769, -73.950, -90.848, 35.507);
  JointPos j2(93.674, -80.062, 82.947, -92.199, -90.967, 26.559);
  DescPose desc_pos1(136.552, -149.799, 449.532, 179.817, -1.172, 157.123);
  DescPose desc_pos2(136.540, -561.048, 449.542, 179.819, -1.172, 157.122);
  DescPose offset_pos(0, 0, 0, 0, 0, 0);
  ExaxisPos epos(0, 0, 0, 0);
  int tool = 0;
  int user = 0;
  float vel = 100.0;
  float acc = 200.0;
  float ovl = 100.0;
  float blendT = -1.0;
  float blendR = -1.0;
  uint8_t flag = 0;
  uint8_t search = 0;
  robot.SetSpeed(20);
  int company = 17;
  int device = 0;
  int softversion = 0;
```

(continues on next page)

(continued from previous page)

```
  int bus = 1;
  robot.FT_SetConfig(company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_GetConfig(&company, &device, &softversion, &bus);
  printf("FT config:%d,%d,%d,%d\n", company, device, softversion, bus);
  robot.Sleep(1000);
  robot.FT_Activate(0);
  robot.Sleep(1000);
  robot.FT_Activate(1);
  robot.Sleep(1000);
  robot.Sleep(1000);
  robot.FT_SetZero(0);
  robot.Sleep(1000);
  robot.FT_SetZero(1);
  robot.Sleep(1000);
  double forceThreshold[6] = { 30,30,30,5,5,5 };
  double m[6] = { 0.1,0.1,0.1,0.02,0.02,0.02 };
  double b[6] = { 1,1,1,0.08,0.08,0.08 };
  double k[6] = { 0,0,0,0,0,0 };
  rtn = robot.ImpedanceControlStartStop(1, 1, forceThreshold, m, b, k, 1000, 500, 100,␣
↪100);
  printf("ImpedanceControlStartStop errcode:%d\n", rtn);
  rtn = robot.MoveL(&desc_pos1, tool, user, vel, acc, ovl, blendR, 0, &epos, search,␣
↪flag, &offset_pos, -1, 1);
  rtn = robot.MoveL(&desc_pos2, tool, user, vel, acc, ovl, blendR, 0, &epos, search,␣
↪flag, &offset_pos, -1, 1);
  rtn = robot.MoveL(&desc_pos1, tool, user, vel, acc, ovl, blendR, 0, &epos, search,␣
↪flag, &offset_pos, -1, 1);
  rtn = robot.MoveL(&desc_pos2, tool, user, vel, acc, ovl, blendR, 0, &epos, search,␣
↪flag, &offset_pos, -1, 1);
  printf("movel errcode:%d\n", rtn);
  robot.ImpedanceControlStartStop(0, 1, forceThreshold, m, b, k, 1000, 500, 100, 100);
  robot.CloseRPC();
  return 0;
}
```

2.1.12.48 Enable torque compensation function and compensation coefficient

```
/**
* @brief Enable torque compensation function and compensation coefficient
* @param [in] status Switch, 0-Disable; 1-Enable
* @param [in] torqueCoeff J1-J6 torque compensation coefficient [0-1]
* @return Error code
*/
errno_t SerCoderCompenParams(int status, double torqueCoeff[6]);
```

2.1.13 Extended Axis

2.1.13.1 Set 485 Extended Axis Parameters

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set 485 extended axis parameters
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] servoCompany Servo driver manufacturer, 1-Dynatek
* @param [in] servoModel Servo driver model, 1-FD100-750C
* @param [in] servoSoftVersion Servo driver software version, 1-V1.0
* @param [in] servoResolution Encoder resolution
* @param [in] axisMechTransRatio Mechanical transmission ratio
* @return Error code
*/
errno_t AuxServoSetParam(int servoId, int servoCompany, int servoModel, int␣
↪servoSoftVersion, int servoResolution, double axisMechTransRatio);
```

2.1.13.2 Get 485 Extended Axis Configuration Parameters

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Get 485 extended axis configuration parameters
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [out] servoCompany Servo driver manufacturer, 1-Dynatek
* @param [out] servoModel Servo driver model, 1-FD100-750C
* @param [out] servoSoftVersion Servo driver software version, 1-V1.0
* @param [out] servoResolution Encoder resolution
* @param [out] axisMechTransRatio Mechanical transmission ratio
* @return Error code
*/
errno_t AuxServoGetParam(int servoId, int* servoCompany, int* servoModel, int*␣
↪servoSoftVersion, int* servoResolution, double* axisMechTransRatio);
```

2.1.13.3 Set 485 Extended Axis Enable/Disable

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set 485 extended axis enable/disable
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] status Enable status, 0-disable, 1-enable
* @return Error code
*/
errno_t AuxServoEnable(int servoId, int status);
```

2.1.13.4 Set 485 Extended Axis Control Mode

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set 485 extended axis control mode
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] mode Control mode, 0-position mode, 1-velocity mode
* @return Error code
*/
errno_t AuxServoSetControlMode(int servoId, int mode);
```

2.1.13.5 Set 485 Extended Axis Target Position (Position Mode)

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set 485 extended axis target position (position mode)
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] pos Target position, mm or °
* @param [in] speed Target speed, mm/s or °/s
* @return Error code
*/
errno_t AuxServoSetTargetPos(int servoId, double pos, double speed);
```

2.1.13.6 Set 485 Extended Axis Target Torque (Torque Mode) - Temporarily Unavailable

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set 485 extended axis target torque (torque mode)
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] torque Target torque, Nm
* @return Error code
*/
errno_t AuxServoSetTargetTorque(int servoId, double torque);
```

2.1.13.7 Set 485 Extended Axis Homing

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set 485 extended axis homing
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] mode Homing mode, 0-current position homing; 1-limit homing
* @param [in] searchVel Homing speed, mm/s or °/s
* @param [in] latchVel Latch speed, mm/s or °/s
* @return Error code
*/
errno_t AuxServoHoming(int servoId, int mode, double searchVel, double latchVel);
```

2.1.13.8 Clear 485 Extended Axis Error Information

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Clear 485 extended axis error information
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @return Error code
*/
errno_t AuxServoClearError(int servoId);
```

2.1.13.9 Get 485 Extended Axis Servo Status

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Get 485 extended axis servo status
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [out] servoErrCode Servo driver error code
* @param [out] servoState Servo driver state [Decimal converted to binary, bit0-bit5:␣
↪Servo enable-Servo running-Positive limit triggered-Negative limit triggered-
↪Positioning complete-Homing complete]
* @param [out] servoPos Current servo position, mm or °
* @param [out] servoSpeed Current servo speed, mm/s or °/s
* @param [out] servoTorque Current servo torque, Nm
* @return Error code
*/
errno_t AuxServoGetStatus(int servoId, int* servoErrCode, int* servoState, double*␣
↪servoPos, double* servoSpeed, double* servoTorque);
```

2.1.13.10 Set 485 Extended Axis Target Speed (Velocity Mode)

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set 485 extended axis target speed (velocity mode)
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] speed Target speed, mm/s or °/s
* @return Error code
*/
errno_t AuxServoSetTargetSpeed(int servoId, double speed);
```

2.1.13.11 Set Status Feedback 485 Extended Axis Data Axis Number

New in version C++SDK-v2.1.3.0.

```
/**
* @brief Set status feedback 485 extended axis data axis number
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @return Error code
*/
errno_t AuxServosetStatusID(int servoId);
```

2.1.13.12 Set 485 Extended Axis Motion Acceleration/Deceleration

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set 485 extended axis motion acceleration/deceleration
* @param [in] acc 485 extended axis motion acceleration
* @param [in] dec 485 extended axis motion deceleration
* @return Error code
*/
errno_t AuxServoSetAcc(double acc, double dec);
```

2.1.13.13 Set 485 Extended Axis Emergency Stop Acceleration/Deceleration

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set 485 extended axis emergency stop acceleration/deceleration
* @param [in] acc 485 extended axis emergency stop acceleration
* @param [in] dec 485 extended axis emergency stop deceleration
* @return Error code
*/
errno_t AuxServoSetEmergencyStopAcc(double acc, double dec);
```

2.1.13.14 Get 485 Extended Axis Motion Acceleration/Deceleration

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get 485 extended axis motion acceleration/deceleration
* @param [out] acc 485 extended axis motion acceleration
* @param [out] dec 485 extended axis motion deceleration
* @return Error code
*/
errno_t AuxServoGetAcc(double& acc, double& dec);
```

2.1.13.15 Get 485 Extended Axis Emergency Stop Acceleration/Deceleration

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get 485 extended axis emergency stop acceleration/deceleration
* @param [out] acc 485 extended axis emergency stop acceleration
* @param [out] dec 485 extended axis emergency stop deceleration
* @return Error code
*/
errno_t AuxServoGetEmergencyStopAcc(double& acc, double& dec);
```

2.1.13.16 Extended Axis Control Code Example

New in version C++SDK-v2.1.5.0.

```
int Test485Auxservo(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  int retval = robot.AuxServoSetParam(1, 1, 1, 1, 131072, 15.45);
  std::cout << "AuxServoSetParam is: " << retval << std::endl;
  int servoCompany;
  int servoModel;
  int servoSoftVersion;
  int servoResolution;
  double axisMechTransRatio;
  retval = robot.AuxServoGetParam(1, &servoCompany, &servoModel, &servoSoftVersion, &
↪servoResolution, &axisMechTransRatio);
  std::cout << "servoCompany " << servoCompany << "\n"
    << "servoModel " << servoModel << "\n"
    << "servoSoftVersion " << servoSoftVersion << "\n"
    << "servoResolution " << servoResolution << "\n"
    << "axisMechTransRatio " << axisMechTransRatio << "\n"
    << std::endl;
  retval = robot.AuxServoSetParam(1, 10, 11, 12, 13, 14);
  std::cout << "AuxServoSetParam is: " << retval << std::endl;
  retval = robot.AuxServoGetParam(1, &servoCompany, &servoModel, &servoSoftVersion, &
↪servoResolution, &axisMechTransRatio);
  std::cout << "servoCompany " << servoCompany << "\n"
    << "servoModel " << servoModel << "\n"
    << "servoSoftVersion " << servoSoftVersion << "\n"
    << "servoResolution " << servoResolution << "\n"
    << "axisMechTransRatio " << axisMechTransRatio << "\n"
```

(continues on next page)

(continued from previous page)

```
    << std::endl;
  retval = robot.AuxServoSetParam(1, 1, 1, 1, 131072, 36);
  std::cout << "AuxServoSetParam is: " << retval << std::endl;
  robot.Sleep(3000);
  robot.AuxServoSetAcc(3000, 3000);
  robot.AuxServoSetEmergencyStopAcc(5000, 5000);
  robot.Sleep(1000);
  double emagacc = 0, acc = 0;
  double emagdec = 0, dec = 0;
  robot.AuxServoGetEmergencyStopAcc(emagacc, emagdec);
  printf("emergency acc is %f dec is %f \n", emagacc, emagdec);
  robot.AuxServoGetAcc(acc, dec);
  printf("acc is %f dec is %f \n", acc, dec);
  robot.AuxServoSetControlMode(1, 0);
  robot.Sleep(2000);
  retval = robot.AuxServoEnable(1, 0);
  std::cout << "AuxServoEnable disenable " << retval << std::endl;
  robot.Sleep(1000);
  int servoerrcode = 0;
  int servoErrCode;
  int servoState;
  double servoPos;
  double servoSpeed;
  double servoTorque;
  retval = robot.AuxServoGetStatus(1, &servoErrCode, &servoState, &servoPos, &servoSpeed,
↪ &servoTorque);
  std::cout << "AuxServoGetStatus servoState " << servoState << std::endl;
  robot.Sleep(1000);;
  retval = robot.AuxServoEnable(1, 1);
  std::cout << "AuxServoEnable enable " << retval << std::endl;
  robot.Sleep(1000);
  retval = robot.AuxServoGetStatus(1, &servoErrCode, &servoState, &servoPos, &servoSpeed,
↪ &servoTorque);
  std::cout << "AuxServoGetStatus servoState " << servoState << std::endl;
  robot.Sleep(1000);
  retval = robot.AuxServoHoming(1, 1, 5, 1);
  std::cout << "AuxServoHoming " << retval << std::endl;
  robot.Sleep(3000);
  retval = robot.AuxServoSetTargetPos(1, 200, 30);
  std::cout << "AuxServoSetTargetPos " << retval << std::endl;
  robot.Sleep(1000);
  retval = robot.AuxServoGetStatus(1, &servoErrCode, &servoState, &servoPos, &servoSpeed,
↪ &servoTorque);
  std::cout << "AuxServoGetStatus servoSpeed " << servoSpeed << std::endl;
  robot.Sleep(8000);
  robot.AuxServoSetControlMode(1, 1);
  robot.Sleep(2000);
  robot.AuxServoEnable(1, 0);
  robot.Sleep(1000);
  robot.AuxServoEnable(1, 1);
  robot.Sleep(1000);
  robot.AuxServoSetTargetSpeed(1, 100, 80);
```

(continues on next page)

(continued from previous page)

```
  robot.Sleep(5000);
  robot.AuxServoSetTargetSpeed(1, 0, 80);
  robot.CloseRPC();
  return 0;
}
```

2.1.13.17 UDP Extended Axis Communication Parameter Configuration

```
/**
* @brief UDP extended axis communication parameter configuration
* @param [in] ip PLC IP address
* @param [in] port  Port number
* @param [in] period    Communication cycle (ms, default is 2, do not modify this␣
↪parameter)
* @param [in] lossPkgTime   Packet loss detection time (ms)
* @param [in] lossPkgNum    Packet loss count
* @param [in] disconnectTime    Communication disconnection confirmation duration
* @param [in] reconnectEnable   Communication disconnection auto-reconnect enable, 0-
↪disable, 1-enable
* @param [in] reconnectPeriod   Reconnect cycle interval (ms)
* @param [in] reconnectNum  Reconnect count
* @param [in] selfConnect Auto-establish connection after power restart; 0-do not␣
↪establish; 1-establish
* @return Error code
*/
errno_t ExtDevSetUDPComParam(std::string ip, int port, int period, int lossPkgTime, int␣
↪lossPkgNum, int disconnectTime, int reconnectEnable, int reconnectPeriod, int␣
↪reconnectNum, int selfConnect = 1);
```

2.1.13.18 Get UDP Extended Axis Communication Parameter Configuration

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Get UDP extended axis communication parameters
* @param [out] ip PLC IP address
* @param [out] port Port number
* @param [out] period Communication cycle (ms, default is 2, do not modify this␣
↪parameter)
* @param [out] lossPkgTime Packet loss detection time (ms)
* @param [out] lossPkgNum Packet loss count
* @param [out] disconnectTime Communication disconnection confirmation duration
* @param [out] reconnectEnable Communication disconnection auto-reconnect enable, 0-
↪disable, 1-enable
* @param [out] reconnectPeriod Reconnect cycle interval (ms)
* @param [out] reconnectNum Reconnect count
* @return Error code
*/
errno_t ExtDevGetUDPComParam(std::string& ip, int& port, int& period, int& lossPkgTime,␣
```

(continues on next page)

(continued from previous page)

```
↪int& lossPkgNum, int& disconnectTime, int& reconnectEnable, int& reconnectPeriod, int&␣
↪reconnectNum);
```

2.1.13.19 Load UDP Communication

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Load UDP communication
* @return Error code
*/
errno_t ExtDevLoadUDPDriver();
```

2.1.13.20 Unload UDP Communication

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Unload UDP communication
* @return Error code
*/
errno_t ExtDevUnloadUDPDriver();
```

2.1.13.21 UDP Extended Axis Communication Exception Disconnection Recovery

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis communication exception disconnection recovery
* @return Error code
*/
errno_t ExtDevUDPClientComReset();
```

2.1.13.22 UDP Extended Axis Communication Exception Disconnection Close Communication

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis communication exception disconnection close communication
* @return Error code
*/
errno_t ExtDevUDPClientComClose();
```

2.1.13.23 UDP Extended Axis Parameter Configuration

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis parameter configuration
* @param [in] axisID Axis number
* @param [in] axisType Extended axis type, 0-translation; 1-rotation
* @param [in] axisDirection Extended axis direction, 0-forward; 1-reverse
* @param [in] axisMax Extended axis maximum position, mm
* @param [in] axisMin Extended axis minimum position, mm
* @param [in] axisVel Speed, mm/s
* @param [in] axisAcc Acceleration, mm/s^2
* @param [in] axisLead Lead, mm
* @param [in] encResolution Encoder resolution
* @param [in] axisOffect Weld starting point extended axis offset
* @param [in] axisCompany Driver manufacturer, 1-Hiwin; 2-Inovance; 3-Panasonic
* @param [in] axisModel Driver model, 1-Hiwin-SV-XD3EA040L-E, 2-Hiwin-SV-X2EA150A-A, 1-Inovance-SV620PT5R4I, 1-Panasonic-MADLN15SG, 2-Panasonic-MSDLN25SG, 3-Panasonic-MCDLN35SG
* @param [in] axisEncType Encoder type, 0-incremental; 1-absolute
* @return Error code
*/
errno_t ExtAxisParamConfig(int axisID, int axisType, int axisDirection, double axisMax, double axisMin, double axisVel, double axisAcc, double axisLead, long encResolution, double axisOffect, int axisCompany, int axisModel, int axisEncType);
```

2.1.13.24 Set Extended Axis Installation Position

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Set extended axis installation position
* @param [in] installType 0-Robot installed on external axis, 1-Robot installed outside external axis
* @return Error code
*/
errno_t SetRobotPosToAxis(int installType);
```

2.1.13.25 Set Extended Axis System DH Parameter Configuration

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Set extended axis system DH parameter configuration
* @param [in] axisConfig External axis configuration, 0-single degree of freedom linear slide, 1-two degree of freedom L-type positioner, 2-three degree of freedom, 3-four degree of freedom, 4-single degree of freedom positioner
* @param [in] axisDHd1 External axis DH parameter d1, mm
* @param [in] axisDHd2 External axis DH parameter d2, mm
* @param [in] axisDHd3 External axis DH parameter d3, mm
```

(continues on next page)

(continued from previous page)

```
* @param [in] axisDHd4 External axis DH parameter d4, mm
* @param [in] axisDHa1 External axis DH parameter a1, mm
* @param [in] axisDHa2 External axis DH parameter a2, mm
* @param [in] axisDHa3 External axis DH parameter a3, mm
* @param [in] axisDHa4 External axis DH parameter a4, mm
* @return Error code
*/
errno_t SetAxisDHParaConfig(int axisConfig, double axisDHd1, double axisDHd2, double␣
↪axisDHd3, double axisDHd4, double axisDHa1, double axisDHa2, double axisDHa3, double␣
↪axisDHa4);
```

2.1.13.26 UDP Extended Axis Enable

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis enable
* @param [in] axisID Axis number [1-4]
* @param [in] status 0-disable; 1-enable
* @return Error code
*/
errno_t ExtAxisServoOn(int axisID, int status);
```

2.1.13.27 UDP Extended Axis Homing

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis homing
* @param [in] axisID Axis number [1-4]
* @param [in] mode Homing method, 0-current position homing, 1-negative limit homing, 2-
↪positive limit homing
* @param [in] searchVel Homing speed (mm/s)
* @param [in] latchVel Homing latch speed (mm/s)
* @return Error code
*/
errno_t ExtAxisSetHoming(int axisID, int mode, double searchVel, double latchVel);
```

2.1.13.28 UDP Extended Axis Jog Start

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis jog start
* @param [in] axisID Axis number [1-4]
* @param [in] direction Rotation direction, 0-reverse; 1-forward
* @param [in] vel Speed (mm/s)
* @param [in] acc Acceleration (mm/s^2)
* @param [in] maxDistance Maximum jog distance
```

(continues on next page)

(continued from previous page)

```
* @return Error code
*/
errno_t ExtAxisStartJog(int axisID, int direction, double vel, double acc, double␣
↪maxDistance);
```

2.1.13.29 UDP Extended Axis Jog Stop

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis jog stop
* @param [in] axisID Axis number [1-4]
* @return Error code
*/
errno_t ExtAxisStopJog(int axisID);
```

2.1.13.30 UDP Extended Axis Configuration and Jog Code Example

```
int TestUDPAxis(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  rtn = robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100, 5, 1);
  cout << "ExtDevSetUDPComParam rtn is " << rtn << endl;
  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;
  rtn = robot.ExtDevGetUDPComParam(ip, port, period, lossPkgTime, lossPkgNum,␣
↪disconnectTime, reconnectEnable, reconnectPeriod, reconnectNum);
  string patam = "\nip " + ip + "\nport " + to_string(port) + "\nperiod " + to_
↪string(period) + "\nlossPkgTime " + to_string(lossPkgTime) + "\nlossPkgNum " + to_
↪string(lossPkgNum) + "\ndisConntime " + to_string(disconnectTime) + "\nreconnecable "␣
↪+ to_string(reconnectEnable) + "\nreconnperiod " + to_string(reconnectPeriod) + "\
↪nreconnnun " + to_string(reconnectNum);
  cout << "ExtDevGetUDPComParam rtn is " << rtn << patam << endl;
  robot.ExtDevLoadUDPDriver();
  rtn = robot.ExtAxisServoOn(1, 1);
  cout << "ExtAxisServoOn axis id 1 rtn is " << rtn << endl;
  rtn = robot.ExtAxisServoOn(2, 1);
  cout << "ExtAxisServoOn axis id 2 rtn is " << rtn << endl;
  robot.Sleep(2000);
```

(continues on next page)

(continued from previous page)

```
  robot.ExtAxisSetHoming(1, 0, 10, 2);
  robot.Sleep(2000);
  rtn = robot.ExtAxisSetHoming(2, 0, 10, 2);
  cout << "ExtAxisSetHoming rtnn is " << rtn << endl;
  robot.Sleep(4000);
  rtn = robot.SetRobotPosToAxis(1);
  cout << "SetRobotPosToAxis rtn is " << rtn << endl;
  rtn = robot.SetAxisDHParaConfig(10, 20, 0, 0, 0, 0, 0, 0, 0);
  cout << "SetAxisDHParaConfig rtn is " << rtn << endl;
  rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905, 262144, 200, 1,
↪ 0, 0);
  cout << "ExtAxisParamConfig axis 1 rtn is " << rtn << endl;
  rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444, 262144, 200, 1,
↪ 0, 0);
  cout << "ExtAxisParamConfig axis 1 rtn is " << rtn << endl;
  robot.Sleep(1000 * 3);
  robot.ExtAxisStartJog(1, 0, 10, 10, 30);
  robot.Sleep(1000 * 1);
  robot.ExtAxisStopJog(1);
  robot.Sleep(1000 * 3);
  robot.ExtAxisServoOn(1, 0);
  robot.Sleep(1000 * 3);
  robot.ExtAxisStartJog(2, 0, 10, 10, 30);
  robot.Sleep(1000 * 1);
  robot.ExtAxisStopJog(2);
  robot.Sleep(1000 * 3);
  robot.ExtAxisServoOn(2, 0);
  robot.ExtDevUnloadUDPDriver();
  robot.CloseRPC();
  return 0;
}
```

2.1.13.31 Set Extended Axis Coordinate System Reference Point - Four-Point Method

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Set extended axis coordinate system reference point - four-point method
* @param [in] pointNum Point number [1-4]
* @return Error code
*/
errno_t ExtAxisSetRefPoint(int pointNum);
```

2.1.13.32 Calculate Extended Axis Coordinate System - Four-Point Method

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Calculate extended axis coordinate system - four-point method
* @param [out] coord Coordinate system value
* @return Error code
*/
errno_t ExtAxisComputeECoordSys(DescPose& coord);
```

2.1.13.33 Positioner Coordinate System Reference Point Setting

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Positioner coordinate system reference point setting
* @param [in] pointNum Point number [1-4]
* @return Error code
*/
errno_t PositionorSetRefPoint(int pointNum);
```

2.1.13.34 Positioner Coordinate System Calculation - Four-Point Method

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Positioner coordinate system calculation - four-point method
* @param [out] coord Coordinate system value
* @return Error code
*/
errno_t PositionorComputeECoordSys(DescPose& coord);
```

2.1.13.35 Set Calibration Reference Point Pose in Extended Axis End Coordinate System

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Set calibration reference point pose in extended axis end coordinate system
* @param [in] pos Pose value
* @return Error code
*/
errno_t SetRefPointInExAxisEnd(DescPose pos);
```

2.1.13.36 Apply Extended Axis Coordinate System

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Apply extended axis coordinate system
* @param [in] applyAxisId Extended axis number, bit0-bit3 corresponds to extended axis number 1-4, e.g., to apply extended axis 1 and 3, use 0b 0000 0101; which is 5
* @param [in] axisCoordNum Extended axis coordinate system number
* @param [in] coord Coordinate system value
* @param [in] calibFlag Calibration flag, 0-no, 1-yes
* @return Error code
*/
errno_t ExtAxisActiveECoordSys(int applyAxisId, int axisCoordNum, DescPose coord, int calibFlag);
```

2.1.13.37 Get Extended Axis Coordinate System

```
/**
* @brief Get extended axis coordinate system
* @param [out] coord Extended axis coordinate system
* @return Error code
*/
errno_t ExtAxisGetCoord(DescPose& coord);
```

2.1.13.38 Extended Axis Coordinate System Calibration Code Example

```
int TestUDPAxisCalib(void)
{
   ROBOT_STATE_PKG pkg = {};
   FRRobot robot;
   robot.LoggerInit();
   robot.SetLoggerLevel(1);
   int rtn = robot.RPC("192.168.58.2");
   if (rtn != 0)
   {
      return -1;
   }
   robot.SetReConnectParam(true, 30000, 500);
   rtn = robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100, 5, 1);
   cout << "ExtDevSetUDPComParam rtn is " << rtn << endl;
   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;
   rtn = robot.ExtDevGetUDPComParam(ip, port, period, lossPkgTime, lossPkgNum, disconnectTime, reconnectEnable, reconnectPeriod, reconnectNum);
   string patam = "\nip " + ip + "\nport " + to_string(port) + "\nperiod " + to_string(period) + "\nlossPkgTime " + to_string(lossPkgTime) + "\nlossPkgNum " + to_string(lossPkgNum) + "\ndisConntime " + to_string(disconnectTime) + "\nreconnecable " + to_string(reconnectEnable) + "\nreconnperiod " + to_string(reconnectPeriod) + "\
```

(continues on next page)

(continued from previous page)

```
↪nreconnnun " + to_string(reconnectNum);
   cout << "ExtDevGetUDPComParam rtn is " << rtn << patam << endl;
   robot.ExtDevLoadUDPDriver();
   rtn = robot.ExtAxisServoOn(1, 1);
   cout << "ExtAxisServoOn axis id 1 rtn is " << rtn << endl;
   rtn = robot.ExtAxisServoOn(2, 1);
   cout << "ExtAxisServoOn axis id 2 rtn is " << rtn << endl;
   robot.Sleep(2000);
   robot.ExtAxisSetHoming(1, 0, 10, 2);
   robot.Sleep(2000);
   rtn = robot.ExtAxisSetHoming(2, 0, 10, 2);
   cout << "ExtAxisSetHoming rtnn is " << rtn << endl;
   robot.Sleep(4000);
   rtn = robot.SetRobotPosToAxis(1);
   cout << "SetRobotPosToAxis rtn is " << rtn << endl;
   rtn = robot.SetAxisDHParaConfig(1, 128.5, 206.4, 0, 0, 0, 0, 0, 0);
   cout << "SetAxisDHParaConfig rtn is " << rtn << endl;
   rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905, 262144, 200,␣
↪1, 0, 0);
   cout << "ExtAxisParamConfig axis 1 rtn is " << rtn << endl;
   rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444, 262144, 200,␣
↪1, 0, 0);
   cout << "ExtAxisParamConfig axis 1 rtn is " << rtn << endl;
   DescPose toolCoord(0, 0, 210, 0, 0, 0);
   robot.SetToolCoord(1, &toolCoord, 0, 0, 1, 0);
   JointPos jSafe(115.193, -96.149, 92.489, -87.068, -89.15, -83.488);
   JointPos j1(117.559, -92.624, 100.329, -96.909, -94.057, -83.488);
   JointPos j2(112.239, -90.096, 99.282, -95.909, -89.824, -83.488);
   JointPos j3(110.839, -83.473, 93.166, -89.22, -90.499, -83.487);
   JointPos j4(107.935, -83.572, 95.424, -92.873, -87.933, -83.488);
   DescPose descSafe = {};
   DescPose desc1 = {};
   DescPose desc2 = {};
   DescPose desc3 = {};
   DescPose desc4 = {};
   ExaxisPos exaxisPos = { 0, 0, 0, 0 };
   DescPose offdese = { 0, 0, 0, 0, 0, 0 };
   robot.GetForwardKin(&jSafe, &descSafe);
   robot.MoveJ(&jSafe, &descSafe, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
   robot.Sleep(2000);
   robot.GetForwardKin(&j1, &desc1);
   robot.MoveJ(&j1, &desc1, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
   robot.Sleep(2000);
   DescPose actualTCPPos = {};
   robot.GetActualTCPPose(0, &actualTCPPos);
   robot.SetRefPointInExAxisEnd(actualTCPPos);
   rtn = robot.PositionorSetRefPoint(1);
   cout << "PositionorSetRefPoint 1 rtn is " << rtn << endl;
   robot.Sleep(2000);
   robot.MoveJ(&jSafe, &descSafe, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
   robot.ExtAxisStartJog(1, 0, 50, 50, 10);
   robot.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.GetForwardKin(&j2, &desc2);
    rtn = robot.MoveJ(&j2, &desc2, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    rtn = robot.PositionorSetRefPoint(2);
    cout << "PositionorSetRefPoint 2 rtn is " << rtn << endl;
    robot.Sleep(2000);
    robot.MoveJ(&jSafe, &descSafe, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.ExtAxisStartJog(1, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.GetForwardKin(&j3, &desc3);
    robot.MoveJ(&j3, &desc3, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    rtn = robot.PositionorSetRefPoint(3);
    cout << "PositionorSetRefPoint 3 rtn is " << rtn << endl;
    robot.Sleep(2000);
    robot.MoveJ(&jSafe, &descSafe, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    robot.ExtAxisStartJog(1, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.GetForwardKin(&j4, &desc4);
    robot.MoveJ(&j4, &desc4, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    rtn = robot.PositionorSetRefPoint(4);
    cout << "PositionorSetRefPoint 4 rtn is " << rtn << endl;
    robot.Sleep(2000);
    DescPose axisCoord = {};
    robot.PositionorComputeECoordSys(axisCoord);
    robot.MoveJ(&jSafe, &descSafe, 1, 0, 100, 100, 100, &exaxisPos, -1, 0, &offdese);
    printf("PositionorComputeECoordSys rtn is %f %f %f %f %f %f\n", axisCoord.tran.x,␣
↪axisCoord.tran.y, axisCoord.tran.z, axisCoord.rpy.rx, axisCoord.rpy.ry, axisCoord.rpy.
↪rz);
    rtn = robot.ExtAxisActiveECoordSys(3, 1, axisCoord, 1);
    cout << "ExtAxisActiveECoordSys rtn is " << rtn << endl;
DescPose getCoord(0, 0, 0, 0, 0, 0);
rtn = robot.ExtAxisGetCoord(getCoord);
printf("ExtAxisGetCoord rtn is %f %f %f %f %f %f\n", getCoord.tran.x, getCoord.tran.y,␣
↪getCoord.tran.z, getCoord.rpy.rx, getCoord.rpy.ry, getCoord.rpy.rz);
robot.CloseRPC();
return 0;
}
```

2.1.13.39 UDP Extended Axis Motion

New in version C++SDK-v2.2.4.0.

```
/**
* @brief UDP extended axis motion
* @param [in] pos Target position
* @param [in] ovl Speed percentage
* @param [in] blend Smoothing parameter (mm or ms); -1: Wait for motion completion
* @return Error code
*/
    errno_t ExtAxisMove(ExaxisPos pos, double ovl, double blend = -1);
```

2.1.13.40 UDP Extended Axis Motion Code Example

```
int TestUDPAxisCalib(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  ExaxisPos axisPos;
  axisPos.ePos[0] = 20;
  axisPos.ePos[1] = 0;
  axisPos.ePos[2] = 0;
  axisPos.ePos[3] = 0;
  robot.ExtAxisMove(axisPos, 50);
  robot.CloseRPC();
  return 0;
}
```

2.1.13.41 UDP Extended Axis and Robot Joint Motion Synchronous Motion

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis and robot joint motion synchronous motion
* @param [in] joint_pos Target joint position, unit deg
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Speed percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], temporarily unavailable
* @param [in] ovl Speed scaling factor, range [0~100]
* @param [in] epos Extended axis position, unit mm
* @param [in] blendT [-1.0]-Motion complete (blocking), [0~500.0]-Smoothing time (non-
```

(continues on next page)

(continued from previous page)

```
↪blocking), unit ms
* @param [in] offset_flag 0-No offset, 1-Base coordinate/workpiece coordinate offset, 2-
↪Tool coordinate offset
* @param [in] offset_pos Pose offset
* @return Error code
*/
errno_t ExtAxisSyncMoveJ(JointPos joint_pos, DescPose desc_pos, int tool, int user,␣
↪float vel, float acc, float ovl, ExaxisPos epos, float blendT, byte offset_flag,␣
↪DescPose offset_pos);
```

2.1.13.42 UDP Extended Axis and Robot Joint Motion Synchronization (Automatic Forward Kinematics Calculation)

```
/**
* @brief UDP Extended Axis and Robot Joint Motion Synchronization (Automatic Forward␣
↪Kinematics Calculation)
* @param [in] joint_pos Target joint position, unit: deg
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], temporarily not available
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] epos Extended axis position, unit: mm
* @param [in] blendT [-1.0]-Move to position (blocking), [0~500.0]-Smoothing time (non-
↪blocking), unit: ms
* @param [in] offset_flag 0-No offset, 1-Offset in base coordinate system/workpiece␣
↪coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos Pose offset value
* @return Error code
*/
errno_t ExtAxisSyncMoveJ(JointPos joint_pos, int tool, int user, float vel, float acc,␣
↪float ovl, ExaxisPos epos, float blendT, uint8_t offset_flag, DescPose offset_pos);
```

2.1.13.43 UDP Extended Axis and Robot Joint Motion Synchronous Motion Code Example

```
int testSyncMoveJ()
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  //1. Calibrate and apply robot tool coordinate system. You can use the four-point or␣
↪six-point method for tool coordinate system calibration and application. The␣
```

(continues on next page)

(continued from previous page)

```
↪interfaces involved in tool coordinate system calibration are as follows:
  //  int SetToolPoint(int point_num); //Set tool reference point - six-point method
  //  int ComputeTool(ref DescPose tcp_pose); //Calculate tool coordinate system
  //  int SetTcp4RefPoint(int point_num);  //Set tool reference point - four-point method
  //  int ComputeTcp4(ref DescPose tcp_pose);  //Calculate tool coordinate system - four-
↪point method
  //  int SetToolCoord(int id, DescPose coord, int type, int install); //Set application␣
↪tool coordinate system
  //  int SetToolList(int id, DescPose coord, int type, int install);  //Set application␣
↪tool coordinate system list
  //2. Set UDP communication parameters and load UDP communication
  robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10);
  robot.ExtDevLoadUDPDriver();
  //3. Set extended axis parameters, including extended axis type, extended axis driver␣
↪parameters, extended axis DH parameters
  robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner and␣
↪DH parameters
  robot.SetRobotPosToAxis(1); //Extended axis installation position
  robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //Servo␣
↪driver parameters. This example is for a single-axis positioner, so only one driver␣
↪parameter needs to be set. If you choose an extended axis type with multiple axes, you␣
↪need to set driver parameters for each axis.
  //4. Set the selected axis enable and homing
  robot.ExtAxisServoOn(1, 0);
  robot.ExtAxisSetHoming(1, 0, 20, 3);
  //5. Perform extended axis coordinate system calibration and application
  DescPose pos = {/* Enter your calibration point coordinates */ };
  robot.SetRefPointInExAxisEnd(pos);
  robot.PositionorSetRefPoint(1); /* You need to call this interface 4 times to complete␣
↪the calibration of the extended axis using four different position points */
  DescPose coord = {};
  robot.PositionorComputeECoordSys(coord); //Calculate extended axis calibration result
  robot.ExtAxisActiveECoordSys(1, 1, coord, 1); //Apply the calibration result to the␣
↪extended axis coordinate system
  //6. Calibrate the workpiece coordinate system on the extended axis. You need to use␣
↪the following interfaces:
  //int SetWObjCoordPoint(int point_num);
  //int ComputeWObjCoord(int method, ref DescPose wobj_pose);
  //int SetWObjCoord(int id, DescPose coord);
  //int SetWObjList(int id, DescPose coord);
  //7. Record your synchronous joint motion starting point
  DescPose startdescPose = {/*Enter your coordinates*/ };
  JointPos startjointPos = {/*Enter your coordinates*/ };
  ExaxisPos startexaxisPos = {/* Enter your extended axis starting point coordinates */ }
↪;
  //8. Record your synchronous joint motion end point coordinates
  DescPose enddescPose = {/*Enter your coordinates*/ };
  JointPos endjointPos = {/*Enter your coordinates*/ };
  ExaxisPos endexaxisPos = {/* Enter your extended axis end point coordinates */ };
  //9. Write synchronous motion program
  //Move to starting point, assuming the applied tool coordinate system and workpiece␣
↪coordinate system are both 1
```

(continues on next page)

(continued from previous page)

```
  robot.ExtAxisMove(startexaxisPos, 20);
  DescPose offdese = { 0, 0, 0, 0, 0, 0 };
  robot.MoveJ(&startjointPos, &startdescPose, 1, 1, 100, 100, 100, &startexaxisPos, 0, 0, &offdese);
  //Start synchronous motion
  robot.ExtAxisSyncMoveJ(endjointPos, enddescPose, 1, 1, 100, 100, 100, endexaxisPos, -1, 0, offdese);
  robot.MoveJ(&startjointPos, 1, 1, 100, 100, 100, &startexaxisPos, 0, 0, &offdese);
  // Start synchronous motion
  robot.ExtAxisSyncMoveJ(endjointPos, 1, 1, 100, 100, 100, endexaxisPos, -1, 0, offdese);
  robot.CloseRPC();
}
```

2.1.13.44 UDP Extended Axis and Robot Linear Motion Synchronous Motion

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis and robot linear motion synchronous motion
* @param [in] joint_pos Target joint position, unit deg
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Speed percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], temporarily unavailable
* @param [in] ovl Speed scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Smoothing radius (non-blocking), unit mm
* @param [in] epos Extended axis position, unit mm
* @param [in] offset_flag 0-No offset, 1-Base coordinate/workpiece coordinate offset, 2-Tool coordinate offset
* @param [in] offset_pos Pose offset
* @return Error code
*/
errno_t ExtAxisSyncMoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, float vel, float acc, float ovl, float blendR, ExaxisPos epos, int offset_flag, DescPose offset_pos);
```

2.1.13.45 UDP Extended Axis and Robot Linear Motion Synchronization (Automatic Inverse Kinematics Calculation)

```
/**
* @brief UDP Extended Axis and Robot Linear Motion Synchronization (Automatic Inverse Kinematics Calculation)
* @param [in] desc_pos  Target Cartesian pose
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], temporarily not available
```

(continues on next page)

(continued from previous page)

```
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Move to position (blocking), [0~1000.0]-Smoothing radius␣
↪(non-blocking), unit: mm
* @param [in] epos Extended axis position, unit: mm
* @param [in] offset_flag 0-No offset, 1-Offset in base coordinate system/workpiece␣
↪coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos Pose offset value
* @param [in] config Inverse kinematics joint space configuration, [-1]-Calculate based␣
↪on current joint position, [0~7]-Solve based on specific joint space configuration
* @return Error code
*/
errno_t ExtAxisSyncMoveL(DescPose desc_pos, int tool, int user, float vel, float acc,␣
↪float ovl, float blendR, ExaxisPos epos, uint8_t offset_flag, DescPose offset_pos, int␣
↪config = -1);
```

2.1.13.46 UDP Extended Axis and Robot Linear Motion Synchronous Motion Code Example

```
int testSyncMoveL()
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  //1. Calibrate and apply robot tool coordinate system. You can use the four-point or␣
↪six-point method for tool coordinate system calibration and application. The␣
↪interfaces involved in tool coordinate system calibration are as follows:
  //  int SetToolPoint(int point_num); //Set tool reference point - six-point method
  //  int ComputeTool(ref DescPose tcp_pose); //Calculate tool coordinate system
  //  int SetTcp4RefPoint(int point_num);  //Set tool reference point - four-point method
  //  int ComputeTcp4(ref DescPose tcp_pose);  //Calculate tool coordinate system - four-
↪point method
  //  int SetToolCoord(int id, DescPose coord, int type, int install); //Set application␣
↪tool coordinate system
  //  int SetToolList(int id, DescPose coord, int type, int install);  //Set application␣
↪tool coordinate system list
  //2. Set UDP communication parameters and load UDP communication
  robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10);
  robot.ExtDevLoadUDPDriver();
  //3. Set extended axis parameters, including extended axis type, extended axis driver␣
↪parameters, extended axis DH parameters
  robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner and␣
↪DH parameters
  robot.SetRobotPosToAxis(1); //Extended axis installation position
  robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //Servo␣
```

(continues on next page)

(continued from previous page)

```
↪driver parameters. This example is for a single-axis positioner, so only one driver␣
↪parameter needs to be set. If you choose an extended axis type with multiple axes, you␣
↪need to set driver parameters for each axis.
  //4. Set the selected axis enable and homing
  robot.ExtAxisServoOn(1, 0);
  robot.ExtAxisSetHoming(1, 0, 20, 3);
  //5. Perform extended axis coordinate system calibration and application
  DescPose pos = {/* Enter your calibration point coordinates */ };
  robot.SetRefPointInExAxisEnd(pos);
  robot.PositionorSetRefPoint(1); /* You need to call this interface 4 times to complete␣
↪the calibration of the extended axis using four different position points */
  DescPose coord = {};
  robot.PositionorComputeECoordSys(coord); //Calculate extended axis calibration result
  robot.ExtAxisActiveECoordSys(1, 1, coord, 1); //Apply the calibration result to the␣
↪extended axis coordinate system
  //6. Calibrate the workpiece coordinate system on the extended axis. You need to use␣
↪the following interfaces:
  //int SetWObjCoordPoint(int point_num);
  //int ComputeWObjCoord(int method, ref DescPose wobj_pose);
  //int SetWObjCoord(int id, DescPose coord);
  //int SetWObjList(int id, DescPose coord);
  //7. Record your synchronous linear motion starting point
  DescPose startdescPose = {/*Enter your coordinates*/ };
  JointPos startjointPos = {/*Enter your coordinates*/ };
  ExaxisPos startexaxisPos = {/* Enter your extended axis starting point coordinates */ }
↪;
  //8. Record your synchronous linear motion end point coordinates
  DescPose enddescPose = {/*Enter your coordinates*/ };
  JointPos endjointPos = {/*Enter your coordinates*/ };
  ExaxisPos endexaxisPos = {/* Enter your extended axis end point coordinates */ };
  //9. Write synchronous motion program
  //Move to starting point, assuming the applied tool coordinate system and workpiece␣
↪coordinate system are both 1
  robot.ExtAxisMove(startexaxisPos, 20);
  DescPose offdese = { 0, 0, 0, 0, 0, 0 };
  robot.MoveJ(&startjointPos, &startdescPose, 1, 1, 100, 100, 100, &startexaxisPos, 0, 0,
↪ &offdese);
  //Start synchronous motion
  robot.ExtAxisSyncMoveL(endjointPos, enddescPose, 1, 1, 100, 100, 100, 0, endexaxisPos,␣
↪0, offdese);
  robot.MoveJ(&startjointPos, 1, 1, 100, 100, 100, &startexaxisPos, 0, 0, &offdese);
  // Start synchronous motion
  robot.ExtAxisSyncMoveL(enddescPose, 1, 1, 100, 100, 100, 0, endexaxisPos, 0, offdese);
  robot.CloseRPC();
}
```

2.1.13.47 UDP Extended Axis and Robot Arc Motion Synchronous Motion

New in version C++SDK-v2.1.4.0.

```
/**
* @brief UDP extended axis and robot arc motion synchronous motion
* @param [in] joint_pos_p Path point joint position, unit deg
* @param [in] desc_pos_p Path point Cartesian pose
* @param [in] ptool Tool coordinate number, range [0~14]
* @param [in] puser Workpiece coordinate number, range [0~14]
* @param [in] pvel Speed percentage, range [0~100]
* @param [in] pacc Acceleration percentage, range [0~100], temporarily unavailable
* @param [in] epos_p Intermediate point extended axis position, unit mm
* @param [in] poffset_flag 0-No offset, 1-Base coordinate/workpiece coordinate offset, 2-Tool coordinate offset
* @param [in] offset_pos_p Pose offset
* @param [in] joint_pos_t Target point joint position, unit deg
* @param [in] desc_pos_t Target point Cartesian pose
* @param [in] ttool Tool coordinate number, range [0~14]
* @param [in] tuser Workpiece coordinate number, range [0~14]
* @param [in] tvel Speed percentage, range [0~100]
* @param [in] tacc Acceleration percentage, range [0~100], temporarily unavailable
* @param [in] epos_t Extended axis position, unit mm
* @param [in] toffset_flag 0-No offset, 1-Base coordinate/workpiece coordinate offset, 2-Tool coordinate offset
* @param [in] offset_pos_t Pose offset
* @param [in] ovl Speed scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Smoothing radius (non-blocking), unit mm
* @return Error code
*/
errno_t ExtAxisSyncMoveC(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser, float pvel, float pacc, ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p, JointPos joint_pos_t, DescPose desc_pos_t, int ttool, int tuser, float tvel, float tacc, ExaxisPos epos_t, int toffset_flag, DescPose offset_pos_t, float ovl, float blendR);
```

2.1.13.48 UDP Extended Axis and Robot Circular Motion Synchronization (Automatic Inverse Kinematics Calculation)

```
/**
* @brief UDP Extended Axis and Robot Circular Motion Synchronization (Automatic Inverse Kinematics Calculation)
* @param [in] desc_pos_p  Path point Cartesian pose
* @param [in] ptool Tool coordinate number, range [0~14]
* @param [in] puser Workpiece coordinate number, range [0~14]
* @param [in] pvel Velocity percentage, range [0~100]
* @param [in] pacc Acceleration percentage, range [0~100], temporarily not available
* @param [in] epos_p Extended axis position, unit: mm
* @param [in] poffset_flag 0-No offset, 1-Offset in base coordinate system/workpiece coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos_p Pose offset value
```

(continues on next page)

(continued from previous page)

```
* @param [in] desc_pos_t  Target point Cartesian pose
* @param [in] ttool Tool coordinate number, range [0~14]
* @param [in] tuser Workpiece coordinate number, range [0~14]
* @param [in] tvel Velocity percentage, range [0~100]
* @param [in] tacc Acceleration percentage, range [0~100], temporarily not available
* @param [in] epos_t Extended axis position, unit: mm
* @param [in] toffset_flag 0-No offset, 1-Offset in base coordinate system/workpiece coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos_t Pose offset value
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Move to position (blocking), [0~1000.0]-Smoothing radius (non-blocking), unit: mm
* @param [in] config Inverse kinematics joint space configuration, [-1]-Calculate based on current joint position, [0~7]-Solve based on specific joint space configuration
* @return Error code
*/
errno_t ExtAxisSyncMoveC(DescPose desc_pos_p, int ptool, int puser, float pvel, float pacc, ExaxisPos epos_p, uint8_t poffset_flag, DescPose offset_pos_p, DescPose desc_pos_t, int ttool, int tuser, float tvel, float tacc, ExaxisPos epos_t, uint8_t toffset_flag, DescPose offset_pos_t, float ovl, float blendR, int config = -1);
```

2.1.13.49 UDP Extended Axis and Robot Arc Motion Synchronous Motion Code Example

```
int testSyncMoveC()
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  //1. Calibrate and apply robot tool coordinate system. You can use the four-point or six-point method for tool coordinate system calibration and application. The interfaces involved in tool coordinate system calibration are as follows:
  //  int SetToolPoint(int point_num); //Set tool reference point - six-point method
  //  int ComputeTool(ref DescPose tcp_pose); //Calculate tool coordinate system
  //  int SetTcp4RefPoint(int point_num);  //Set tool reference point - four-point method
  //  int ComputeTcp4(ref DescPose tcp_pose);  //Calculate tool coordinate system - four-point method
  //  int SetToolCoord(int id, DescPose coord, int type, int install); //Set application tool coordinate system
  //  int SetToolList(int id, DescPose coord, int type, int install);  //Set application tool coordinate system list
  //2. Set UDP communication parameters and load UDP communication
  robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10);
  robot.ExtDevLoadUDPDriver();
```

(continues on next page)

(continued from previous page)

```
  //3. Set extended axis parameters, including extended axis type, extended axis driver parameters, extended axis DH parameters
  robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner and DH parameters
  robot.SetRobotPosToAxis(1); //Extended axis installation position
  robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //Servo driver parameters. This example is for a single-axis positioner, so only one driver parameter needs to be set. If you choose an extended axis type with multiple axes, you need to set driver parameters for each axis.
  //4. Set the selected axis enable and homing
  robot.ExtAxisServoOn(1, 0);
  robot.ExtAxisSetHoming(1, 0, 20, 3);
  //5. Perform extended axis coordinate system calibration and application
  DescPose pos = {/* Enter your calibration point coordinates */ };
  robot.SetRefPointInExAxisEnd(pos);
  robot.PositionorSetRefPoint(1); /* You need to call this interface 4 times to complete the calibration of the extended axis using four different position points */
  DescPose coord = {};
  robot.PositionorComputeECoordSys(coord); //Calculate extended axis calibration result
  robot.ExtAxisActiveECoordSys(1, 1, coord, 1); //Apply the calibration result to the extended axis coordinate system
  //6. Calibrate the workpiece coordinate system on the extended axis. You need to use the following interfaces:
  //int SetWObjCoordPoint(int point_num);
  //int ComputeWObjCoord(int method, ref DescPose wobj_pose);
  //int SetWObjCoord(int id, DescPose coord);
  //int SetWObjList(int id, DescPose coord);
  //7. Record your synchronous arc motion starting point
  DescPose startdescPose = {/*Enter your coordinates*/ };
  JointPos startjointPos = {/*Enter your coordinates*/ };
  ExaxisPos startexaxisPos = {/* Enter your extended axis starting point coordinates */ };
  //8. Record your synchronous arc motion end point coordinates
  DescPose enddescPose = {/*Enter your coordinates*/ };
  JointPos endjointPos = {/*Enter your coordinates*/ };
  ExaxisPos endexaxisPos = {/* Enter your extended axis end point coordinates */ };
  //9. Record your synchronous arc motion intermediate point coordinates
  DescPose middescPose = {/*Enter your coordinates*/ };
  JointPos midjointPos = {/*Enter your coordinates*/ };
  ExaxisPos midexaxisPos = {/* Enter the extended axis coordinates at the robot arc intermediate point */ };
  //10. Write synchronous motion program
  //Move to starting point, assuming the applied tool coordinate system and workpiece coordinate system are both 1
  robot.ExtAxisMove(startexaxisPos, 20);
  DescPose offdese = { 0, 0, 0, 0, 0, 0 };
  robot.MoveJ(&startjointPos, &startdescPose, 1, 1, 100, 100, 100, &startexaxisPos, 0, 0, &offdese);
  //Start synchronous motion
  robot.ExtAxisSyncMoveC(midjointPos, middescPose, 1, 1, 100, 100, midexaxisPos, 0, offdese, endjointPos, enddescPose, 1, 1, 100, 100, endexaxisPos, 0, offdese, 100, 0);
  robot.MoveJ(&startjointPos, 1, 1, 100, 100, 100, &startexaxisPos, 0, 0, &offdese);
```

(continues on next page)

(continued from previous page)

```
  // Start synchronous motion
  robot.ExtAxisSyncMoveC(middescPose, 1, 1, 100, 100, midexaxisPos, 0, offdese,␣
↪enddescPose, 1, 1, 100, 100, endexaxisPos, 0, offdese, 100, 0);
  robot.CloseRPC();
}
```

2.1.13.50 Set Extended DO

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Set extended DO
* @param [in] DONum DO number
* @param [in] bOpen Switch, true-on; false-off
* @param [in] smooth Whether to smooth
* @param [in] block Whether to block
* @return Error code
*/
errno_t SetAuxDO(int DONum, bool bOpen, bool smooth, bool block);
```

2.1.13.51 Set Extended AO

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Set extended AO
* @param [in] AONum AO number
* @param [in] value Analog value [0-4095]
* @param [in] block Whether to block
* @return Error code
*/
errno_t SetAuxAO(int AONum, double value, bool block);
```

2.1.13.52 Set Extended DI Input Filter Time

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Set extended DI input filter time
* @param [in] filterTime Filter time (ms)
* @return Error code
*/
errno_t SetAuxDIFilterTime(int filterTime);
```

2.1.13.53 Set Extended AI Input Filter Time

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Set extended AI input filter time
* @param [in] filterTime Filter time (ms)
* @return Error code
*/
errno_t SetAuxAIFilterTime(int filterTime);
```

2.1.13.54 Wait for Extended DI Input

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Wait for extended DI input
* @param [in] DINum DI number
* @param [in] bOpen Switch, 0-off; 1-on
* @param [in] time Maximum wait time (ms)
* @param [in] errorAlarm Whether to continue motion
* @return Error code
*/
errno_t WaitAuxDI(int DINum, bool bOpen, int time, bool errorAlarm);
```

2.1.13.55 Wait for Extended AI Input

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Wait for extended AI input
* @param [in] AINum AI number
* @param [in] sign 0-Greater than; 1-Less than
* @param [in] value AI value
* @param [in] time Maximum wait time (ms)
* @param [in] errorAlarm Whether to continue motion
* @return Error code
*/
errno_t WaitAuxAI(int AINum, int sign, int value, int time, bool errorAlarm);
```

2.1.13.56 Get Extended DI Value

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Get extended DI value
* @param [in] DINum DI number
* @param [in] isNoBlock Whether to block
* @param [out] isOpen 0-off; 1-on
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
errno_t GetAuxDI(int DINum, bool isNoBlock, bool& isOpen);
```

2.1.13.57 Get Extended AI Value

New in version C++SDK-v2.1.4.0.

```
/**
* @brief Get extended AI value
* @param [in] AINum AI number
* @param [in] isNoBlock Whether to block
* @param [in] value Input value
* @return Error code
*/
errno_t GetAuxAI(int AINum, bool isNoBlock, int& value);
```

2.1.13.58 Extended IO Code Example

```
int TestAuxDOAO(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  for (int i = 0; i < 128; i++)
  {
    robot.SetAuxDO(i, true, false, true);
    Sleep(100);
  }
  for (int i = 0; i < 128; i++)
  {
    robot.SetAuxDO(i, false, false, true);
    Sleep(100);
  }
  for (int i = 0; i < 409; i++)
  {
    robot.SetAuxAO(0, i * 10, true);
    robot.SetAuxAO(1, 4095 - i * 10, true);
    robot.SetAuxAO(2, i * 10, true);
    robot.SetAuxAO(3, 4095 - i * 10, true);
    Sleep(10);
  }
  robot.SetAuxDIFilterTime(10);
```

(continues on next page)

(continued from previous page)

```
  robot.SetAuxAIFilterTime(0, 10);
  for (int i = 0; i < 20; i++)
  {
    bool curValue = false;
    int rtn = robot.GetAuxDI(i, false, curValue);
    cout << "DI" << i << "  " << curValue << endl;
  }
  int curValue = -1;
  for (int i = 0; i < 4; i++)
  {
    rtn = robot.GetAuxAI(i, true, curValue);
  }
  robot.WaitAuxDI(1, false, 1000, false);
  robot.WaitAuxAI(1, 1, 132, 1000, false);
  robot.CloseRPC();
  return 0;
}
```

2.1.13.59 Mobile Device Enable

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Mobile device enable
* @param enable false-disable; true-enable
* @return Error code
*/
errno_t TractorEnable(bool enable);
```

2.1.13.60 Mobile Device Homing

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Mobile device homing
* @return Error code
*/
errno_t TractorHoming();
```

2.1.13.61 Mobile Device Linear Motion

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Mobile device linear motion
* @param distance Linear motion distance (mm)
* @param vel Linear motion speed percentage (0-100)
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
errno_t TractorMoveL(double distance, double vel);
```

2.1.13.62 Mobile Device Arc Motion

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Mobile device arc motion
* @param radio Arc motion radius (mm)
* @param angle Arc motion angle (°)
* @param vel Linear motion speed percentage (0-100)
* @return Error code
*/
errno_t TractorMoveC(double radio, double angle, double vel);
```

2.1.13.63 Mobile Device Stop Motion

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Mobile device stop motion
* @return Error code
*/
errno_t TractorStop();
```

2.1.13.64 Mobile Device Code Example

```
int TestTractor(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.ExtDevSetUDPComParam("192.168.58.2", 2021, 2, 50, 5, 50, 1, 50, 10, 1);
  robot.ExtDevLoadUDPDriver();
  rtn = robot.ExtAxisServoOn(1, 1);
  rtn = robot.ExtAxisServoOn(2, 1);
  robot.Sleep(2000);
  robot.ExtAxisSetHoming(1, 0, 10, 2);
  robot.Sleep(2000);
  rtn = robot.ExtAxisSetHoming(2, 0, 10, 2);
```

(continues on next page)

(continued from previous page)

```
  robot.Sleep(4000);
  robot.ExtAxisParamConfig(1, 0, 0, 50000, -50000, 1000, 1000, 6.280, 16384, 200, 0, 0,␣
↪0);
  robot.ExtAxisParamConfig(2, 0, 0, 50000, -50000, 1000, 1000, 6.280, 16384, 200, 0, 0,␣
↪0);
  robot.SetAxisDHParaConfig(5, 0, 0, 0, 0, 0, 0, 0, 0);
  robot.TractorEnable(false);
  robot.Sleep(2000);
  robot.TractorEnable(true);
  robot.Sleep(2000);
  robot.TractorHoming();
  robot.Sleep(2000);
  robot.TractorMoveL(100, 2);
  robot.Sleep(5000);
  robot.TractorStop();
  robot.TractorMoveL(-100, 20);
  robot.Sleep(5000);
  robot.TractorMoveC(300, 90, 20);
  robot.Sleep(10000);
  robot.TractorMoveC(300, -90, 20);
  robot.Sleep(1);
  robot.CloseRPC();
  return 0;
}
```

2.1.14 Robot Welding

2.1.14.1 Set Welding Process Curve Parameters

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Set welding process curve parameters
 * @param [in] id Welding process ID (1-99)
 * @param [in] startCurrent Arc starting current (A)
 * @param [in] startVoltage Arc starting voltage (V)
 * @param [in] startTime Arc starting time (ms)
 * @param [in] weldCurrent Welding current (A)
 * @param [in] weldVoltage Welding voltage (V)
 * @param [in] endCurrent Arc ending current (A)
 * @param [in] endVoltage Arc ending voltage (V)
 * @param [in] endTime Arc ending time (ms)
 * @return Error code
 */
errno_t WeldingSetProcessParam(int id, double startCurrent, double startVoltage, double␣
↪startTime, double weldCurrent, double weldVoltage, double endCurrent, double␣
↪endVoltage, double endTime);
```

2.1.14.2 Get Welding Process Curve Parameters

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Get welding process curve parameters
 * @param [in] id Welding process ID (1-99)
 * @param [out] startCurrent Arc starting current (A)
 * @param [out] startVoltage Arc starting voltage (V)
 * @param [out] startTime Arc starting time (ms)
 * @param [out] weldCurrent Welding current (A)
 * @param [out] weldVoltage Welding voltage (V)
 * @param [out] endCurrent Arc ending current (A)
 * @param [out] endVoltage Arc ending voltage (V)
 * @param [out] endTime Arc ending time (ms)
 * @return Error code
 */
errno_t WeldingGetProcessParam(int id, double& startCurrent, double& startVoltage,␣
↪double& startTime, double& weldCurrent, double& weldVoltage, double& endCurrent,␣
↪double& endVoltage, double& endTime);
```

2.1.14.3 Set Welding Current to Analog Output Relationship

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Set welding current to analog output relationship
* @param [in] currentMin Current value at left point of linear relationship (A)
* @param [in] currentMax Current value at right point of linear relationship (A)
* @param [in] outputVoltageMin Analog output voltage at left point (V)
* @param [in] outputVoltageMax Analog output voltage at right point (V)
* @return Error code
*/
errno_t WeldingSetCurrentRelation(double currentMin, double currentMax, double␣
↪outputVoltageMin, double outputVoltageMax);
```

2.1.14.4 Set Welding Voltage to Analog Output Relationship

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Set welding voltage to analog output relationship
* @param [in] weldVoltageMin Welding voltage at left point of linear relationship (A)
* @param [in] weldVoltageMax Welding voltage at right point of linear relationship (A)
* @param [in] outputVoltageMin Analog output voltage at left point (V)
* @param [in] outputVoltageMax Analog output voltage at right point (V)
* @return Error code
*/
errno_t WeldingSetVoltageRelation(double weldVoltageMin, double weldVoltageMax, double␣
↪outputVoltageMin, double outputVoltageMax);
```

2.1.14.5 Get Welding Current to Analog Output Relationship

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Get welding current to analog output relationship
* @param [out] currentMin Current value at left point of linear relationship (A)
* @param [out] currentMax Current value at right point of linear relationship (A)
* @param [out] outputVoltageMin Analog output voltage at left point (V)
* @param [out] outputVoltageMax Analog output voltage at right point (V)
* @return Error code
*/
errno_t WeldingGetCurrentRelation(double *currentMin, double *currentMax, double␣
↪*outputVoltageMin, double *outputVoltageMax);
```

2.1.14.6 Get Welding Voltage to Analog Output Relationship

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Get welding voltage to analog output relationship
* @param [out] weldVoltageMin Welding voltage at left point of linear relationship (A)
* @param [out] weldVoltageMax Welding voltage at right point of linear relationship (A)
* @param [out] outputVoltageMin Analog output voltage at left point (V)
* @param [out] outputVoltageMax Analog output voltage at right point (V)
* @return Error code
*/
errno_t WeldingGetVoltageRelation(double *weldVoltageMin, double *weldVoltageMax, double␣
↪*outputVoltageMin, double *outputVoltageMax);
```

2.1.14.7 Set Welding Current

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Set welding current
* @param [in] ioType Control IO type 0-Control box IO; 1-Extension IO
* @param [in] current Welding current value (A)
* @param [in] AOIndex Welding current control box analog output port (0-1)
* @param [in] blend Whether to smooth 0-Not smooth; 1-Smooth
* @return Error code
*/
errno_t WeldingSetCurrent(int ioType, double current, int AOIndex, int blend);
```

2.1.14.8 Set Welding Voltage

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Set welding voltage
* @param [in] ioType Control IO type 0-Control box IO; 1-Extension IO
* @param [in] voltage Welding voltage value (V)
* @param [in] AOIndex Welding voltage control box analog output port (0-1)
* @param [in] blend Whether to smooth 0-Not smooth; 1-Smooth
* @return Error code
*/
errno_t WeldingSetVoltage(int ioType, double voltage, int AOIndex, int blend);
```

2.1.14.9 Set Weaving Parameters

```
/**
 * @brief Set weaving parameters
 * @param [in] weaveNum Weaving parameter configuration ID
 * @param [in] weaveType Weaving type:
 *        0-Horizontal triangular wave weaving;
 *        1-Vertical L-shaped triangular wave weaving;
 *        2-Clockwise circular weaving;
 *        3-Counter-clockwise circular weaving;
 *        4-Horizontal sine wave weaving;
 *        5-Vertical L-shaped sine wave weaving;
 *        6-Vertical triangular wave weaving;
 *        7-Vertical sine wave weaving
 * @param [in] weaveFrequency Weaving frequency (Hz)
 * @param [in] weaveIncStayTime Wait mode:
 *        0-Period excludes wait time;
 *        1-Period includes wait time
 * @param [in] weaveRange Weaving amplitude (mm)
 * @param [in] weaveLeftRange Left chord length for vertical triangular weaving (mm)
 * @param [in] weaveRightRange Right chord length for vertical triangular weaving (mm)
 * @param [in] additionalStayTime Vertical triangular weaving apex dwell time (mm)
 * @param [in] weaveLeftStayTime Left dwell time (ms)
 * @param [in] weaveRightStayTime Right dwell time (ms)
 * @param [in] weaveCircleRadio Circular weaving-callback ratio (0-100%)
 * @param [in] weaveStationary Position wait during weaving:
 *        0-Position continues moving during wait time;
 *        1-Position remains static during wait time
 * @param [in] weaveYawAngle Weaving direction azimuth angle (rotation around Z-axis),␣
↪in °
 * @param [in] weaveRotAngle Weaving direction tilt angle (deflection around X-axis), in␣
↪°
 * @return Error code
 */
errno_t 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 = 0);
```

2.1.14.10 Welding Parameter Setup Code Example

```
int TestSetWeldParam(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.WeldingSetProcessParam(1, 177, 27, 1000, 178, 28, 176, 26, 1000);
  robot.WeldingSetProcessParam(2, 188, 28, 555, 199, 29, 133, 23, 333);
  double startCurrent = 0;
  double startVoltage = 0;
  double startTime = 0;
  double weldCurrent = 0;
  double weldVoltage = 0;
  double endCurrent = 0;
  double endVoltage = 0;
  double endTime = 0;
  robot.WeldingGetProcessParam(1, startCurrent, startVoltage, startTime, weldCurrent,␣
↪weldVoltage, endCurrent, endVoltage, endTime);
  cout << "the Num 1 process param is " << startCurrent << " " << startVoltage << " " <<␣
↪startTime << " " << weldCurrent << " " << weldVoltage << " " << endCurrent << " " <<␣
↪endVoltage << " " << endTime << endl;
  robot.WeldingGetProcessParam(2, startCurrent, startVoltage, startTime, weldCurrent,␣
↪weldVoltage, endCurrent, endVoltage, endTime);
  cout << "the Num 2 process param is " << startCurrent << " " << startVoltage << " " <<␣
↪startTime << " " << weldCurrent << " " << weldVoltage << " " << endCurrent << " " <<␣
↪endVoltage << " " << endTime << endl;
  rtn = robot.WeldingSetCurrentRelation(0, 400, 0, 10, 0);
  cout << "WeldingSetCurrentRelation rtn is: " << rtn << endl;
  rtn = robot.WeldingSetVoltageRelation(0, 40, 0, 10, 1);
  cout << "WeldingSetVoltageRelation rtn is: " << rtn << endl;
  double current_min = 0;
  double current_max = 0;
  double vol_min = 0;
  double vol_max = 0;
  double output_vmin = 0;
  double output_vmax = 0;
  int curIndex = 0;
  int volIndex = 0;
  rtn = robot.WeldingGetCurrentRelation(&current_min, &current_max, &output_vmin, &
↪output_vmax, &curIndex);
  cout << "WeldingGetCurrentRelation rtn is: " << rtn << endl;
  cout << "current min " << current_min << " current max " << current_max << " output␣
↪vol min " << output_vmin << " output vol max " << output_vmax << endl;
  rtn = robot.WeldingGetVoltageRelation(&vol_min, &vol_max, &output_vmin, &output_vmax, &
↪volIndex);
```

(continues on next page)

(continued from previous page)

```
  cout << "WeldingGetVoltageRelation rtn is: " << rtn << endl;
  cout << "vol min " << vol_min << " vol max " << vol_max << " output vol min " <<␣
↪output_vmin << " output vol max " << output_vmax << endl;
  rtn = robot.WeldingSetCurrent(1, 100, 0, 0);
  cout << "WeldingSetCurrent rtn is: " << rtn << endl;
  this_thread::sleep_for(chrono::seconds(3));
  rtn = robot.WeldingSetVoltage(1, 10, 0, 0);
  cout << "WeldingSetVoltage rtn is: " << rtn << endl;
  rtn = robot.WeaveSetPara(0, 0, 2.000000, 0, 10.000000, 0.000000, 0.000000, 0, 0, 0, 0,␣
↪0, 60.000000);
  cout << "rtn is: " << rtn << endl;
  robot.WeaveOnlineSetPara(0, 0, 1, 0, 20, 0, 0, 0, 0);
  rtn = robot.WeldingSetCheckArcInterruptionParam(1, 200);
  printf("WeldingSetCheckArcInterruptionParam  %d\n", rtn);
  rtn = robot.WeldingSetReWeldAfterBreakOffParam(1, 5.7, 98.2, 0);
  printf("WeldingSetReWeldAfterBreakOffParam  %d\n", rtn);
  int enable = 0;
  double length = 0;
  double velocity = 0;
  int moveType = 0;
  int checkEnable = 0;
  int arcInterruptTimeLength = 0;
  rtn = robot.WeldingGetCheckArcInterruptionParam(&checkEnable, &arcInterruptTimeLength);
  printf("WeldingGetCheckArcInterruptionParam checkEnable %d  arcInterruptTimeLength %d\n
↪", checkEnable, arcInterruptTimeLength);
  rtn = robot.WeldingGetReWeldAfterBreakOffParam(&enable, &length, &velocity, &moveType);
  printf("WeldingGetReWeldAfterBreakOffParam enable = %d, length = %lf, velocity = %lf,␣
↪moveType = %d\n", enable, length, velocity, moveType);
  robot.SetWeldMachineCtrlModeExtDoNum(17);
  for (int i = 0; i < 5; i++)
  {
    robot.SetWeldMachineCtrlMode(0);
    robot.Sleep(1000);
    robot.SetWeldMachineCtrlMode(1);
    robot.Sleep(1000);
  }
  robot.CloseRPC();
  return 0;
}
```

2.1.14.11 Instantaneously Set Weaving Parameters

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Instantly set weaving parameters
* @param [in] weaveNum Weaving parameter configuration number
* @param [in] weaveType Weaving type 0-Flat triangular wave; 1-Vertical L-shaped␣
↪triangular wave; 2-Clockwise circular; 3-Counter-clockwise circular; 4-Flat sine wave;␣
↪5-Vertical L-shaped sine wave; 6-Vertical triangular wave; 7-Vertical sine wave
* @param [in] weaveFrequency Weaving frequency (Hz)
```

(continues on next page)

(continued from previous page)

```
* @param [in] weaveIncStayTime Wait mode 0-Cycle excludes wait time; 1-Cycle includes␣
↪wait time
* @param [in] weaveRange Weaving amplitude (mm)
* @param [in] weaveLeftStayTime Left dwell time (ms)
* @param [in] weaveRightStayTime Right dwell time (ms)
* @param [in] weaveCircleRadio Circular weaving - callback ratio (0-100%)
* @param [in] weaveStationary Weaving position wait, 0-Position continues moving during␣
↪wait; 1-Position remains stationary during wait
* @return Error code
*/
errno_t WeaveOnlineSetPara(int weaveNum, int weaveType, double weaveFrequency, int␣
↪weaveIncStayTime, double weaveRange, int weaveLeftStayTime, int weaveRightStayTime,␣
↪int weaveCircleRadio, int weaveStationary);
```

2.1.14.12 Set Robot Welding Arc Unexpected Interruption Detection Parameters

New in version C++SDK-v2.1.8-3.7.8.

```
/**
     * @brief Set robot welding arc unexpected interruption detection parameters
     * @param [in] checkEnable Enable detection; 0-Disable; 1-Enable
     * @param [in] arcInterruptTimeLength Arc interruption confirmation duration (ms)
     * @return Error code
*/
    errno_t WeldingSetCheckArcInterruptionParam(int checkEnable, int␣
↪arcInterruptTimeLength);
```

2.1.14.13 Get Robot Welding Arc Unexpected Interruption Detection Parameters

New in version C++SDK-v2.1.8-3.7.8.

```
/**
     * @brief Get robot welding arc unexpected interruption detection parameters
     * @param [out] checkEnable Whether detection is enabled; 0-Disabled; 1-Enabled
     * @param [out] arcInterruptTimeLength Arc interruption confirmation duration (ms)
     * @return Error code
*/
    errno_t WeldingGetCheckArcInterruptionParam(int* checkEnable, int*␣
↪arcInterruptTimeLength);
```

2.1.14.14 Set Robot Welding Interruption Recovery Parameters

New in version C++SDK-v2.1.8-3.7.8.

```
/**
     * @brief Set robot welding interruption recovery parameters
     * @param [in] enable Whether to enable welding interruption recovery
     * @param [in] length Weld overlap distance (mm)
     * @param [in] velocity Robot return to re-arc point speed percentage (0-100)
     * @param [in] moveType Robot movement to re-arc point method; 0-LIN; 1-PTP
     * @return Error code
*/
    errno_t WeldingSetReWeldAfterBreakOffParam(int enable, double length, double␣
↪velocity, int moveType);
```

2.1.14.15 Get Robot Welding Interruption Recovery Parameters

New in version C++SDK-v2.1.8-3.7.8.

```
/**
     * @brief Get robot welding interruption recovery parameters
     * @param [out] enable Whether welding interruption recovery is enabled
     * @param [out] length Weld overlap distance (mm)
     * @param [out] velocity Robot return to re-arc point speed percentage (0-100)
     * @param [out] moveType Robot movement to re-arc point method; 0-LIN; 1-PTP
     * @return Error code
*/
    errno_t WeldingGetReWeldAfterBreakOffParam(int* enable, double* length, double*␣
↪velocity, int* moveType);
```

2.1.14.16 Set Welder Control Mode Extended DO Port

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set welder control mode extended DO port
* @param DONum Welder control mode DO port (0-127)
* @return Error code
*/
errno_t SetWeldMachineCtrlModeExtDoNum(int DONum);
```

2.1.14.17 Set Welder Control Mode

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set welder control mode
* @param mode Welder control mode; 0-Unary
* @return Error code
*/
errno_t SetWeldMachineCtrlMode(int mode);
```

2.1.14.18 Welding Start

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Welding start
* @param [in] ioType IO type 0-Controller IO; 1-Extended IO
* @param [in] arcNum Welder configuration file number
* @param [in] timeout Arc ignition timeout
* @return Error code
*/
errno_t ARCStart(int ioType, int arcNum, int timeout);
```

2.1.14.19 Welding End

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Welding end
* @param [in] ioType IO type 0-Controller IO; 1-Extended IO
* @param [in] arcNum Welder configuration file number
* @param [in] timeout Arc extinguishing timeout
* @return Error code
*/
errno_t ARCEnd(int ioType, int arcNum, int timeout);
```

2.1.14.20 Weaving Start

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Weaving start
* @param [in] weaveNum Weaving parameter configuration number
* @return Error code
*/
errno_t WeaveStart(int weaveNum);
```

2.1.14.21 Weaving End

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Weaving end
* @param [in] weaveNum Weaving parameter configuration number
* @return Error code
*/
errno_t WeaveEnd(int weaveNum);
```

2.1.14.22 Forward Wire Feeding

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Forward wire feeding
* @param [in] ioType IO type 0-Controller IO; 1-Extended IO
* @param [in] wireFeed Wire feed control 0-Stop feeding; 1-Feed wire
* @return Error code
*/
errno_t SetForwardWireFeed(int ioType, int wireFeed);
```

2.1.14.23 Reverse Wire Feeding

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Reverse wire feeding
* @param [in] ioType IO type 0-Controller IO; 1-Extended IO
* @param [in] wireFeed Wire feed control 0-Stop feeding; 1-Feed wire
* @return Error code
*/
errno_t SetReverseWireFeed(int ioType, int wireFeed);
```

2.1.14.24 Gas Feeding

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Gas feeding
* @param [in] ioType IO type 0-Controller IO; 1-Extended IO
* @param [in] airControl Gas feed control 0-Stop feeding; 1-Feed gas
* @return Error code
*/
errno_t SetAspirated(int ioType, int airControl);
```

2.1.14.25 Set Robot to Resume Welding After Interruption

New in version C++SDK-v2.1.8-3.7.8.

```
/**
     * @brief Set robot to resume welding after interruption
     * @return Error code
*/
    errno_t WeldingStartReWeldAfterBreakOff();
```

2.1.14.26 Set Robot to Abort Welding After Interruption

New in version C++SDK-v2.1.8-3.7.8.

```
/**
     * @brief Set robot to abort welding after interruption
     * @return Error code
     */
    errno_t WeldingAbortWeldAfterBreakOff();
```

2.1.14.27 Robot Welding Control Code Example

```
int TestWelding(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.SetForwardWireFeed(0, 1);
  robot.Sleep(1000);
  robot.SetForwardWireFeed(0, 0);
  robot.SetReverseWireFeed(0, 1);
  robot.Sleep(1000);
  robot.SetReverseWireFeed(0, 0);
  robot.SetAspirated(0, 1);
  robot.Sleep(1000);
  robot.SetAspirated(0, 0);
  robot.WeldingSetCurrent(1, 230, 0, 0);
  robot.WeldingSetVoltage(1, 24, 0, 1);
  DescPose p1Desc(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
  JointPos p1Joint(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);
  DescPose p2Desc(-333.302, -435.580, 449.866, -174.997, 2.017, 109.815);
  JointPos p2Joint(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);
  ExaxisPos exaxisPos(0, 0, 0, 0);
  DescPose offdese(0, 0, 0, 0, 0, 0);
  robot.MoveJ(&p1Joint, &p1Desc, 13, 0, 20, 100, 100, &exaxisPos, -1, 0, &offdese);
  robot.ARCStart(1, 0, 10000);
  robot.WeaveStart(0);
  robot.MoveL(&p2Joint, &p2Desc, 13, 0, 20, 100, 100, -1, 0, &exaxisPos, 0, 0, &offdese);
  robot.ARCEnd(1, 0, 10000);
  robot.WeaveEnd(0);
  robot.WeldingStartReWeldAfterBreakOff();
  robot.WeldingAbortWeldAfterBreakOff();
  robot.CloseRPC();
  return 0;
}
```

2.1.14.28 Segment Welding Start

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Segment welding start
* @param [in] startDesePos Starting point Cartesian position
* @param [in] endDesePos Ending point Cartesian pose
* @param [in] startJPos Starting point joint pose
* @param [in] endJPos Ending point joint pose
* @param [in] weldLength Welding segment length (mm)
* @param [in] noWeldLength Non-welding segment length (mm)
* @param [in] weldIOType Welding IO type (0-Control box IO; 1-Extended IO)
* @param [in] arcNum Welder configuration file number
* @param [in] weldTimeout Arc ignition/extinguishing timeout
* @param [in] isWeave Whether to weave
* @param [in] weaveNum Weaving parameter configuration number
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Speed percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], not currently open
* @param [in] ovl Speed scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Move to position (blocking), [0~1000.0]-Smoothing radius␣
↪(non-blocking), unit mm
* @param [in] epos Extended axis position, unit mm
* @param [in] search 0-No wire search, 1-Wire search
* @param [in] offset_flag 0-No offset, 1-Offset in base/workpiece coordinate system, 2-
↪Offset in tool coordinate system
* @param [in] offset_pos Pose offset
* @return Error code
*/
errno_t SegmentWeldStart(DescPose *startDesePos, DescPose *endDesePos, JointPos␣
↪*startJPos, JointPos *endJPos, double weldLength, double noWeldLength, int weldIOType,␣
↪int arcNum, int weldTimeout, bool isWeave, int weaveNum, int tool, int user, float vel,
↪ float acc, float ovl, float blendR, ExaxisPos *epos, uint8_t search, uint8_t offset_
↪flag, DescPose *offset_pos);
```

2.1.14.29 Robot Segment Welding Code Example

Changed in version C++SDK-v2.1.2.0.

```
int TestSegWeld(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
```

(continues on next page)

(continued from previous page)

```
  robot.SetReConnectParam(true, 30000, 500);
  robot.WeldingSetCurrent(1, 230, 0, 0);
  robot.WeldingSetVoltage(1, 24, 0, 1);
  DescPose p1Desc(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
  JointPos p1Joint(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);
  DescPose p2Desc(-333.302, -435.580, 449.866, -174.997, 2.017, 109.815);
  JointPos p2Joint(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);
  ExaxisPos exaxisPos(0, 0, 0, 0);
  DescPose offdese(0, 0, 0, 0, 0, 0);
  rtn = robot.SegmentWeldStart(&p1Desc, &p2Desc, &p1Joint, &p2Joint, 20, 20, 0, 0, 5000,␣
↪0, 0, 0, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &offdese);
  printf("SegmentWeldStart rtn is %d\n", rtn);
  robot.CloseRPC();
  return 0;
}
```

2.1.14.30 Simulation Weaving Start

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Simulation weaving start
 * @param [in] weaveNum Weaving parameter number
 * @return Error code
 */
errno_t WeaveStartSim(int weaveNum);
```

2.1.14.31 Simulation Weaving End

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Simulation weaving end
 * @param [in] weaveNum Weaving parameter number
 * @return Error code
 */
errno_t WeaveEndSim(int weaveNum);
```

2.1.14.32 Start Trajectory Detection Warning (No Movement)

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Start trajectory detection warning (no movement)
 * @param [in] weaveNum  Weaving parameter number
 * @return Error code
 */
errno_t WeaveInspectStart(int weaveNum);
```

2.1.14.33 End Trajectory Detection Warning (No Movement)

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief End trajectory detection warning (no movement)
 * @param [in] weaveNum  Weaving parameter number
 * @return Error code
 */
errno_t WeaveInspectEnd(int weaveNum);
```

2.1.14.34 Weaving Gradual Change Start

```
/**
 * @brief Weaving gradual change start
 * @param [in] weaveChangeFlag 1-Change weaving parameters; 2-Change weaving parameters
↪+ welding speed
 * @param [in] weaveNum Weaving number
 * @param [in] velStart Welding start speed (cm/min)
 * @param [in] velEnd Welding end speed (cm/min)
 * @return Error code
 */
 errno_t WeaveChangeStart(int weaveChangeFlag, int weaveNum, double velStart, double
↪velEnd);
```

2.1.14.35 Robot Weaving Gradual Change Welding Code Example

```
int TestWeave(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  DescPose p1Desc(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
  JointPos p1Joint(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);
  DescPose p2Desc(-333.302, -435.580, 449.866, -174.997, 2.017, 109.815);
  JointPos p2Joint(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);
  ExaxisPos exaxisPos(0, 0, 0, 0);
  DescPose offdese(0, 0, 0, 0, 0, 0);
  robot.MoveJ(&p1Joint, &p1Desc, 13, 0, 20, 100, 100, &exaxisPos, -1, 0, &offdese);
  robot.WeaveStartSim(0);
  robot.MoveL(&p2Joint, &p2Desc, 13, 0, 20, 100, 100, -1, 0, &exaxisPos, 0, 0, &offdese);
  robot.WeaveEndSim(0);
  robot.MoveJ(&p1Joint, &p1Desc, 13, 0, 20, 100, 100, &exaxisPos, -1, 0, &offdese);
```

(continues on next page)

(continued from previous page)

```
  robot.WeaveInspectStart(0);
  robot.MoveL(&p2Joint, &p2Desc, 13, 0, 20, 100, 100, -1, 0, &exaxisPos, 0, 0, &offdese);
  robot.WeaveInspectEnd(0);
  robot.WeldingSetVoltage(1, 19, 0, 0);
  robot.WeldingSetCurrent(1, 190, 0, 0);
  robot.MoveL(&p1Joint, &p1Desc, 1, 1, 100, 100, 50, -1, &exaxisPos, 0, 0, &offdese);
  robot.ARCStart(1, 0, 10000);
  robot.ArcWeldTraceControl(1, 0, 1, 0.06, 5, 5, 60, 1, 0.06, 5, 5, 80, 0, 0, 4, 1, 10,␣
↪0, 0);
  robot.WeaveStart(0);
  robot.WeaveChangeStart(1, 0, 50, 30);
  robot.MoveL(&p2Joint, &p2Desc, 1, 1, 100, 100, 1, -1, &exaxisPos, 0, 0, &offdese);
  robot.WeaveChangeEnd();
  robot.WeaveEnd(0);
  robot.ArcWeldTraceControl(0, 0, 1, 0.06, 5, 5, 60, 1, 0.06, 5, 5, 80, 0, 0, 4, 1, 10,␣
↪0, 0);
  robot.ARCEnd(1, 0, 10000);
  robot.CloseRPC();
  return 0;
}
```

2.1.14.36 Weaving Gradual Change End

New in version C++SDK-v2.2.0-3.8.0.

```
    /**
     * @brief  Weaving gradual change end
     * @return  Error code
     */
errno_t WeaveChangeEnd();
```

2.1.14.37 Extended IO-Configure Welder Gas Detection Signal

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Extended IO-Configure welder gas detection signal
 * @param [in] DONum Gas detection signal extended DO number
 * @return Error code
 */
errno_t SetAirControlExtDoNum(int DONum);
```

2.1.14.38 Extended IO-Configure Welder Arc Start Signal

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Extended IO-Configure welder arc start signal
 * @param [in] DONum Welder arc start signal extended DO number
 * @return Error code
 */
errno_t SetArcStartExtDoNum(int DONum);
```

2.1.14.39 Extended IO-Configure Welder Reverse Wire Feed Signal

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Extended IO-Configure welder reverse wire feed signal
 * @param [in] DONum Reverse wire feed signal extended DO number
 * @return Error code
 */
errno_t SetWireReverseFeedExtDoNum(int DONum);
```

2.1.14.40 Extended IO-Configure Welder Forward Wire Feed Signal

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Extended IO-Configure welder forward wire feed signal
 * @param [in] DONum Forward wire feed signal extended DO number
 * @return Error code
 */
errno_t SetWireForwardFeedExtDoNum(int DONum);
```

2.1.14.41 Extended IO-Configure Welder Arc Success Signal

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Extended IO-Configure welder arc success signal
 * @param [in] DINum Arc success signal extended DI number
 * @return Error code
 */
errno_t SetArcDoneExtDiNum(int DINum);
```

2.1.14.42 Extended IO-Configure Welder Ready Signal

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Extended IO-Configure welder ready signal
 * @param [in] DINum Welder ready signal extended DI number
 * @return Error code
 */
errno_t SetWeldReadyExtDiNum(int DINum);
```

2.1.14.43 Extended IO-Configure Welding Interruption Recovery Signal

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief Extended IO-Configure welding interruption recovery signal
 * @param [in] reWeldDINum Resume welding after interruption signal extended DI number
 * @param [in] abortWeldDINum Abort welding after interruption signal extended DI number
 * @return Error code
 */
errno_t SetExtDIWeldBreakOffRecover(int reWeldDINum, int abortWeldDINum);
```

2.1.14.44 Set Extended IO Welding Signal Code Example

```
int TestExtDIConfig(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.SetArcStartExtDoNum(10);
  robot.SetAirControlExtDoNum(20);
  robot.SetWireForwardFeedExtDoNum(30);
  robot.SetWireReverseFeedExtDoNum(40);
  robot.SetWeldReadyExtDiNum(50);
  robot.SetArcDoneExtDiNum(60);
  robot.SetExtDIWeldBreakOffRecover(70, 80);
  robot.SetWireSearchExtDIONum(0, 1);
  robot.CloseRPC();
  return 0;
}
```

2.1.14.45 Arc Tracking Control

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief  Arc tracking control
     * @param  [in] flag Switch, 0-Off; 1-On
     * @param  [in] dalayTime Lag time, unit ms
     * @param  [in] isLeftRight Left-right deviation compensation
     * @param  [in] klr Left-right adjustment coefficient (sensitivity);
     * @param  [in] tStartLr Left-right start compensation time cyc
     * @param  [in] stepMaxLr Left-right maximum compensation per cycle mm
     * @param  [in] sumMaxLr Left-right total maximum compensation mm
     * @param  [in] isUpLow Up-down deviation compensation
     * @param  [in] kud Up-down adjustment coefficient (sensitivity);
     * @param  [in] tStartUd Up-down start compensation time cyc
     * @param  [in] stepMaxUd Up-down maximum compensation per cycle mm
     * @param  [in] sumMaxUd Up-down total maximum compensation
     * @param  [in] axisSelect Up-down coordinate system selection, 0-Weaving; 1-Tool; 2-Base
     * @param  [in] referenceType Up-down reference current setting method, 0-Feedback; 1-Constant
     * @param  [in] referSampleStartUd Up-down reference current sampling start count (feedback);, cyc
     * @param  [in] referSampleCountUd Up-down reference current sampling cycle count (feedback);, cyc
     * @param  [in] referenceCurrent Up-down reference current mA
     * @param  [in] offsetType Offset tracking type, 0-No offset; 1-Sampling; 2-Percentage
     * @param  [in] offsetParameter Offset parameter; Sampling (offset sampling start time, default one cycle); Percentage (offset percentage (-100 ~ 100))
     * @return  Error code
     */
    errno_t ArcWeldTraceControl(int flag, double delaytime, int isLeftRight, double klr, double tStartLr, double stepMaxLr, double sumMaxLr, int isUpLow, double kud, double tStartUd, double stepMaxUd, double sumMaxUd, int axisSelect, int referenceType, double referSampleStartUd, double referSampleCountUd, double referenceCurrent, int offsetType = 0, int offsetParameter = 0);
```

2.1.14.46 Set Arc Tracking Input Signal Port

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief  Set arc tracking input signal port
 * @param  [in] channel Arc tracking AI channel selection,[0-3]
 * @return  Error code
 */
errno_t ArcWeldTraceExtAIChannelConfig(int channel);
```

2.1.14.47 Arc Tracking + Multi-layer Multi-pass Compensation Start

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Arc tracking + multi-layer multi-pass compensation start
* @return Error code
*/
errno_t ArcWeldTraceReplayStart();
```

2.1.14.48 Arc Tracking + Multi-layer Multi-pass Compensation End

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Arc tracking + multi-layer multi-pass compensation end
* @return Error code
*/
errno_t ArcWeldTraceReplayEnd();
```

2.1.14.49 Offset Coordinate Transformation - Multi-layer Multi-pass Welding

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Offset coordinate transformation - multi-layer multi-pass welding
* @return Error code
*/
errno_t MultilayerOffsetTrsfToBase(DescTran point0, DescTran pointX, DescTran pointZ,␣
↪double dx, double dy, double db, DescPose& offset);
```

2.1.14.50 Multi-layer Multi-pass Welding Arc Tracking Code Example

```
int TestArcWeldTrace(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  JointPos mulitilineorigin1_joint(-24.090, -63.501, 84.288, -111.940, -93.426, 57.669);
  DescPose mulitilineorigin1_desc(-677.559, 190.951, -1.205, 1.144, -41.482, -82.577);
  DescTran mulitilineX1_desc;
  mulitilineX1_desc.x = -677.556;
  mulitilineX1_desc.y = 211.949;
```

(continues on next page)

(continued from previous page)

```
  mulitilineX1_desc.z = -1.206;
  DescTran mulitilineZ1_desc;
  mulitilineZ1_desc.x = -677.564;
  mulitilineZ1_desc.y = 190.956;
  mulitilineZ1_desc.z = 19.817;
  JointPos mulitilinesafe_joint(-25.734, -63.778, 81.502, -108.975, -93.392, 56.021);
  DescPose mulitilinesafe_desc(-677.561, 211.950, 19.812, 1.144, -41.482, -82.577);
  JointPos mulitilineorigin2_joint(-29.743, -75.623, 101.241, -116.354, -94.928, 55.735);
  DescPose mulitilineorigin2_desc(-563.961, 215.359, -0.681, 2.845, -40.476, -87.443);
  DescTran mulitilineX2_desc;
  mulitilineX2_desc.x = -563.965;
  mulitilineX2_desc.y = 220.355;
  mulitilineX2_desc.z = -0.680;
  DescTran mulitilineZ2_desc;
  mulitilineZ2_desc.x = -563.968;
  mulitilineZ2_desc.y = 215.362;
  mulitilineZ2_desc.z = 4.331;
  ExaxisPos epos(0, 0, 0, 0);
  DescPose offset(0, 0, 0, 0, 0, 0);
  robot.Sleep(10);
  int error = robot.MoveJ(&mulitilinesafe_joint, &mulitilinesafe_desc, 13, 0, 10, 100,␣
↪100, &epos, -1, 0, &offset);
  printf("MoveJ return: %d\n", error);
  error = robot.MoveL(&mulitilineorigin1_joint, &mulitilineorigin1_desc, 13, 0, 10, 100,␣
↪100, -1, &epos, 0, 0, &offset, 0, 100);
  printf("MoveL return: %d\n", error);
  error = robot.MoveJ(&mulitilinesafe_joint, &mulitilinesafe_desc, 13, 0, 10, 100, 100, &
↪epos, -1, 0, &offset);
  printf("MoveJ return: %d\n", error);
  error = robot.MoveL(&mulitilineorigin2_joint, &mulitilineorigin2_desc, 13, 0, 10, 100,␣
↪100, -1, &epos, 0, 0, &offset, 0, 100);
  printf("MoveL return: %d\n", error);
  error = robot.MoveJ(&mulitilinesafe_joint, &mulitilinesafe_desc, 13, 0, 10, 100, 100, &
↪epos, -1, 0, &offset);
  printf("MoveJ return: %d\n", error);
  error = robot.MoveL(&mulitilineorigin1_joint, &mulitilineorigin1_desc, 13, 0, 10, 100,␣
↪100, -1, &epos, 0, 0, &offset, 0, 100);
  printf("MoveL return: %d\n", error);
  error = robot.ARCStart(1, 0, 3000);
  printf("ARCStart return: %d\n", error);
  error = robot.WeaveStart(0);
  printf("WeaveStart return: %d\n", error);
  error = robot.ArcWeldTraceControl(1, 0, 1, 0.06, 5, 5, 50, 1, 0.06, 5, 5, 55, 0, 0, 4,␣
↪1, 10);
  printf("ArcWeldTraceControl return: %d\n", error);
  error = robot.MoveL(&mulitilineorigin2_joint, &mulitilineorigin2_desc, 13, 0, 1, 100,␣
↪100, -1, &epos, 0, 0, &offset, 0, 100);
  printf("MoveL return: %d\n", error);
  error = robot.ArcWeldTraceControl(0, 0, 1, 0.06, 5, 5, 50, 1, 0.06, 5, 5, 55, 0, 0, 4,␣
↪1, 10);
  printf("ArcWeldTraceControl return: %d\n", error);
  error = robot.WeaveEnd(0);
```

(continues on next page)

(continued from previous page)

```
  printf("WeaveEnd return: %d\n", error);
  error = robot.ARCEnd(1, 0, 10000);
  printf("ARCEnd return: %d\n", error);
  error = robot.MoveJ(&mulitilinesafe_joint, &mulitilinesafe_desc, 13, 0, 10, 100, 100, &epos, -1, 0, &offset);
  printf("MoveJ return: %d\n", error);
  error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin1_desc.tran, mulitilineX1_desc, mulitilineZ1_desc, 10.0, 0.0, 0.0, offset);
  printf("MultilayerOffsetTrsfToBase return: %d offect is %f %f %f \n", error, offset.tran.x, offset.tran.y, offset.tran.z);
  error = robot.MoveL(&mulitilineorigin1_joint, &mulitilineorigin1_desc, 13, 0, 10, 100, 100, -1, &epos, 0, 1, &offset, 0, 100);
  printf("MoveL return: %d\n", error);
  error = robot.ARCStart(1, 0, 3000);
  printf("ARCStart return: %d\n", error);
  error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin2_desc.tran, mulitilineX2_desc, mulitilineZ2_desc, 10, 0, 0, offset);
  printf("MultilayerOffsetTrsfToBase return: %d offect is %f %f %f \n", error, offset.tran.x, offset.tran.y, offset.tran.z);
  error = robot.ArcWeldTraceReplayStart();
  printf("ArcWeldTraceReplayStart return: %d\n", error);
  error = robot.MoveL(&mulitilineorigin2_joint, &mulitilineorigin2_desc, 13, 0, 2, 100, 100, -1, &epos, 0, 1, &offset, 0, 100);
  printf("MoveL return: %d\n", error);
  error = robot.ArcWeldTraceReplayEnd();
  printf("ArcWeldTraceReplayEnd return: %d\n", error);
  error = robot.ARCEnd(1, 0, 10000);
  printf("ARCEnd return: %d\n", error);
  error = robot.MoveJ(&mulitilinesafe_joint, &mulitilinesafe_desc, 13, 0, 10, 100, 100, &epos, -1, 0, &offset);
  printf("MoveJ return: %d\n", error);
  error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin1_desc.tran, mulitilineX1_desc, mulitilineZ1_desc, 0, 10, 0, offset);
  printf("MultilayerOffsetTrsfToBase return: %d offect is %f %f %f \n", error, offset.tran.x, offset.tran.y, offset.tran.z);
  error = robot.MoveL(&mulitilineorigin1_joint, &mulitilineorigin1_desc, 13, 0, 10, 100, 100, -1, &epos, 0, 1, &offset, 0, 100);
  printf("MoveL return: %d\n", error);
  error = robot.ARCStart(1, 0, 3000);
  printf("ARCStart return: %d\n", error);
  error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin2_desc.tran, mulitilineX2_desc, mulitilineZ2_desc, 0, 10, 0, offset);
  printf("MultilayerOffsetTrsfToBase return: %d offect is %f %f %f \n", error, offset.tran.x, offset.tran.y, offset.tran.z);
  error = robot.ArcWeldTraceReplayStart();
  printf("MoveJ return: %d\n", error);
  error = robot.MoveL(&mulitilineorigin2_joint, &mulitilineorigin2_desc, 13, 0, 2, 100, 100, -1, &epos, 0, 1, &offset, 0, 100);
  printf("MoveL return: %d\n", error);
  error = robot.ArcWeldTraceReplayEnd();
  printf("ArcWeldTraceReplayEnd return: %d\n", error);
  error = robot.ARCEnd(1, 0, 3000);
```

(continues on next page)

(continued from previous page)

```
  printf("ARCEnd return: %d\n", error);
  error = robot.MoveJ(&mulitilinesafe_joint, &mulitilinesafe_desc, 13, 0, 10, 100, 100, &epos, -1, 0, &offset);
  printf("MoveJ return: %d\n", error);
  robot.CloseRPC();
  return 0;
}
```

2.1.14.51 Arc Tracking Welder Current Feedback AI Channel Selection

```
/**
 * @brief Arc tracking welder current feedback AI channel selection
 * @param [in]  channel Channel; 0-Extended AI0; 1-Extended AI1; 2-Extended AI2; 3-Extended AI3; 4-Control box AI0; 5-Control box AI1
 * @return Error code
 */
errno_t ArcWeldTraceAIChannelCurrent(int channel);
```

2.1.14.52 Arc Tracking Welder Voltage Feedback AI Channel Selection

```
/**
 * @brief Arc tracking welder voltage feedback AI channel selection
 * @param [in]  channel Channel; 0-Extended AI0; 1-Extended AI1; 2-Extended AI2; 3-Extended AI3; 4-Control box AI0; 5-Control box AI1
 * @return Error code
 */
errno_t ArcWeldTraceAIChannelVoltage(int channel);
```

2.1.14.53 Arc Tracking Welder Current Feedback Conversion Parameters

```
/**
 * @brief Arc tracking welder current feedback conversion parameters
 * @param [in] AILow AI channel lower limit, default 0V, range [0-10V]
 * @param [in] AIHigh AI channel upper limit, default 10V, range [0-10V]
 * @param [in] currentLow AI channel lower limit corresponding welder current value, default 0V, range [0-200V]
 * @param [in] currentHigh AI channel upper limit corresponding welder current value, default 100V, range [0-200V]
 * @return Error code
 */
errno_t ArcWeldTraceCurrentPara(float AILow, float AIHigh, float currentLow, float currentHigh);
```

2.1.14.54 Arc Tracking Welder Voltage Feedback Conversion Parameters

```
/**
* @brief Arc tracking welder voltage feedback conversion parameters
* @param [in] AILow AI channel lower limit, default 0V, range [0-10V]
* @param [in] AIHigh AI channel upper limit, default 10V, range [0-10V]
* @param [in] voltageLow AI channel lower limit corresponding welder voltage value, default 0V, range [0-200V]
* @param [in] voltageHigh AI channel upper limit corresponding welder voltage value, default 100V, range [0-200V]
* @return Error code
*/
errno_t ArcWeldTraceVoltagePara(float AILow, float AIHigh, float voltageLow, float voltageHigh);
```

2.1.14.55 Arc Tracking Code Example

```
int WeldTraceControlWithCtrlBoxAI(FRRobot* robot)
{
  DescPose startdescPose = { -473.86, 257.879, -20.849, -37.317, -42.021, 2.543 };
  JointPos startjointPos = { -43.487, -76.526, 95.568, -104.445, -89.356, 3.72 };

  DescPose enddescPose = { -499.844, 141.225, 7.72, -34.856, -40.17, 13.13 };
  JointPos endjointPos = { -31.305, -82.998, 99.401, -104.426, -89.35, 3.696 };

  DescPose safedescPose = { -504.043, 275.181, 40.908, -28.002, -42.025, -14.044 };
  JointPos safejointPos = { -39.078, -76.732, 87.227, -99.47, -94.301, 18.714 };

  ExaxisPos exaxisPos = { 0, 0, 0, 0 };
  DescPose offdese = { 0, 0, 0, 0, 0, 0 };

  robot->WeldingSetCurrentRelation(0, 495, 1, 10, 0);
  robot->WeldingSetVoltageRelation(10, 45, 1, 10, 1);

  robot->WeldingSetVoltage(0, 25, 1, 0);// ----Set voltage
  robot->WeldingSetCurrent(0, 260, 0, 0);// ----Set current

  int rtn = robot->ArcWeldTraceAIChannelCurrent(4);
  cout << "ArcWeldTraceAIChannelCurrent rtn is " << rtn << endl;
  rtn = robot->ArcWeldTraceAIChannelVoltage(5);
  cout << "ArcWeldTraceAIChannelVoltage rtn is " << rtn << endl;
  rtn = robot->ArcWeldTraceCurrentPara(0, 5, 0, 500);
  cout << "ArcWeldTraceCurrentPara rtn is " << rtn << endl;
  rtn = robot->ArcWeldTraceVoltagePara(1.018, 10, 0, 50);
  cout << "ArcWeldTraceVoltagePara rtn is " << rtn << endl;
  robot->MoveJ(&safejointPos, &safedescPose, 1, 0, 5, 100, 100, &exaxisPos, -1, 0, &offdese);
  robot->MoveJ(&startjointPos, &startdescPose, 1, 0, 5, 100, 100, &exaxisPos, -1, 0, &offdese);
  rtn = robot->ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
```

(continues on next page)

(continued from previous page)

```
  cout << "ArcWeldTraceControl rtn is " << rtn << endl;
  robot->ARCStart(0, 0, 10000);
  robot->WeaveStart(0);
  robot->MoveL(&endjointPos, &enddescPose, 1, 0, 100, 100, 2, -1, &exaxisPos, 0, 0, &
↪offdese);
  robot->ARCEnd(0, 0, 10000);
  robot->WeaveEnd(0);
  robot->ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1,␣
↪10, 0, 0);
  return 0;
}
```

2.1.14.56 Set Wire Search Extended IO Port

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set wire search extended IO port
* @param searchDoneDINum  Wire search success DO port (0-127)
* @param searchStartDONum  Wire search start/stop control DO port (0-127)
* @return Error code
*/
errno_t SetWireSearchExtDIONum(int searchDoneDINum, int searchStartDONum);
```

2.1.14.57 Example Program

New in version C++SDK-v2.1.5.0.

```
void TestUDPWireSearch(FRRobot* robot)
{
robot->ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 50, 5, 50, 1, 50, 10);
robot->ExtDevLoadUDPDriver();

robot->SetWireSearchExtDIONum(0, 0);

int rtn0, rtn1, rtn2 = 0;
ExaxisPos exaxisPos = { 0.0, 0.0, 0.0, 0.0 };
DescPose offdese = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };

DescPose descStart = { -158.767, -510.596, 271.709, -179.427, -0.745, -137.349 };
JointPos jointStart = { 61.667, -79.848, 108.639, -119.682, -89.700, -70.985 };

DescPose descEnd = { 0.332, -516.427, 270.688, 178.165, 0.017, -119.989 };
JointPos jointEnd = { 79.021, -81.839, 110.752, -118.298, -91.729, -70.981 };

robot->MoveL(&jointStart, &descStart, 1, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese);
robot->MoveL(&jointEnd, &descEnd, 1, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &offdese);

DescPose descREF0A = { -66.106, -560.746, 270.381, 176.479, -0.126, -126.745 };
```

(continues on next page)

(continued from previous page)

```
JointPos jointREF0A = { 73.531, -75.588, 102.941, -116.250, -93.347, -69.689 };

DescPose descREF0B = { -66.109, -528.440, 270.407, 176.479, -0.129, -126.744 };
JointPos jointREF0B = { 72.534, -79.625, 108.046, -117.379, -93.366, -70.687 };

DescPose descREF1A = { 72.975, -473.242, 270.399, 176.479, -0.129, -126.744 };
JointPos jointREF1A = { 87.169, -86.509, 115.710, -117.341, -92.993, -56.034 };

DescPose descREF1B = { 31.355, -473.238, 270.405, 176.480, -0.130, -126.745 };
JointPos jointREF1B = { 82.117, -87.146, 116.470, -117.737, -93.145, -61.090 };

rtn0 = robot->WireSearchStart(0, 10, 100, 0, 10, 100, 0);
robot->MoveL(&jointREF0A, &descREF0A, 1, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese); //Start point
robot->MoveL(&jointREF0B, &descREF0B, 1, 0, 10, 100, 100, -1, &exaxisPos, 1, 0, &
↪offdese); //Location point
rtn1 = robot->WireSearchWait("REF0");
rtn2 = robot->WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

rtn0 = robot->WireSearchStart(0, 10, 100, 0, 10, 100, 0);
robot->MoveL(&jointREF1A, &descREF1A, 1, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese); //Start point
robot->MoveL(&jointREF1B, &descREF1B, 1, 0, 10, 100, 100, -1, &exaxisPos, 1, 0, &
↪offdese); //Location point
rtn1 = robot->WireSearchWait("REF1");
rtn2 = robot->WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

rtn0 = robot->WireSearchStart(0, 10, 100, 0, 10, 100, 0);
robot->MoveL(&jointREF0A, &descREF0A, 1, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese); //Start point
robot->MoveL(&jointREF0B, &descREF0B, 1, 0, 10, 100, 100, -1, &exaxisPos, 1, 0, &
↪offdese); //Location point
rtn1 = robot->WireSearchWait("RES0");
rtn2 = robot->WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

rtn0 = robot->WireSearchStart(0, 10, 100, 0, 10, 100, 0);
robot->MoveL(&jointREF1A, &descREF1A, 1, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese); //Start point
robot->MoveL(&jointREF1B, &descREF1B, 1, 0, 10, 100, 100, -1, &exaxisPos, 1, 0, &
↪offdese); //Location point
rtn1 = robot->WireSearchWait("RES1");
rtn2 = robot->WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

vector <string> varNameRef = { "REF0", "REF1", "#", "#", "#", "#" };
vector <string> varNameRes = { "RES0", "RES1", "#", "#", "#", "#" };
int offectFlag = 0;
DescPose offectPos = { 0, 0, 0, 0, 0, 0 };
rtn0 = robot->GetWireSearchOffset(0, 0, varNameRef, varNameRes, offectFlag, offectPos);
robot->PointsOffsetEnable(0, &offectPos);
robot->MoveL(&jointStart, &descStart, 1, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese);
robot->MoveL(&jointEnd, &descEnd, 1, 0, 100, 100, 100, -1, &exaxisPos, 0, 0, &offdese);
```

(continues on next page)

(continued from previous page)

```
robot->PointsOffsetDisable();
}
```

2.1.14.58 Wire Search Start

New in version C++SDK-v2.1.5.0.

```
/**
* @brief  Wire search start
* @param  [in] refPos  1-Reference point 0-Contact point
* @param  [in] searchVel   Search speed %
* @param  [in] searchDis  Search distance mm
* @param  [in] autoBackFlag Auto return flag, 0-No auto; -Auto
* @param  [in] autoBackVel  Auto return speed %
* @param  [in] autoBackDis  Auto return distance mm
* @param  [in] offectFlag  1-Search with offset; 0-Teach point search
* @return  Error code
*/
 errno_t WireSearchStart(int refPos, float searchVel, int searchDis, int autoBackFlag,␣
↪float autoBackVel, int autoBackDis, int offectFlag);
```

2.1.14.59 Wire Search End

New in version C++SDK-v2.1.5.0.

```
 /**
  * @brief  Wire search end
  * @param  [in] refPos  1-Reference point 2-Contact point
  * @param  [in] searchVel   Search speed %
  * @param  [in] searchDis  Search distance mm
  * @param  [in] autoBackFlag Auto return flag, 0-No auto; -Auto
  * @param  [in] autoBackVel  Auto return speed %
  * @param  [in] autoBackDis  Auto return distance mm
  * @param  [in] offectFlag  1-Search with offset; 2-Teach point search
  * @return  Error code
  */
 errno_t WireSearchEnd(int refPos, float searchVel, int searchDis, int autoBackFlag,␣
↪float autoBackVel, int autoBackDis, int offectFlag);
```

2.1.14.60 Calculate Wire Search Offset

New in version C++SDK-v2.1.5.0.

```
 /**
  * @brief  Calculate wire search offset
  * @param  [in] seamType  Weld seam type
  * @param  [in] method   Calculation method
  * @param  [in] varNameRef Reference point 1-6, "#" means no point variable
  * @param  [in] varNameRes Contact point 1-6, "#" means no point variable
```

(continues on next page)

(continued from previous page)

```
 * @param  [out] offectFlag 0-Offset directly added to command point; 1-Offset requires coordinate transformation of command point
 * @param  [out] offect Offset pose [x, y, z, a, b, c]
 * @return  Error code
 */
errno_t GetWireSearchOffset(int seamType, int method, std::vector<std::string> varNameRef, std::vector<std::string> varNameRes, int& offectFlag, DescPose& offect);
```

2.1.14.61 Wait for Wire Search Completion

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief  Wait for wire search completion
 * @return  Error code
 */
errno_t WireSearchWait(std::string varName);
```

2.1.14.62 Write Wire Search Contact Point to Database

New in version C++SDK-v2.1.5.0.

```
/**
 * @brief  Write wire search contact point to database
 * @param  [in] varName  Contact point name "RES0" ~ "RES99"
 * @param  [in] pos  Contact point data [x, y, x, a, b, c]
 * @return  Error code
 */
errno_t SetPointToDatabase(std::string varName, DescPose pos);
```

2.1.14.63 Robot Wire Search Code Example

New in version C++SDK-v2.1.5.0.

```
int TestWireSearch(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  DescPose toolCoord(0, 0, 200, 0, 0, 0);
  robot.SetToolCoord(1, &toolCoord, 0, 0, 1, 0);
  DescPose wobjCoord(0, 0, 0, 0, 0, 0);
```

(continues on next page)

(continued from previous page)

```
  robot.SetWObjCoord(1, &wobjCoord, 0);
  int rtn0, rtn1, rtn2 = 0;
  ExaxisPos exaxisPos = { 0, 0, 0, 0 };
  DescPose offdese = { 0, 0, 0, 0, 0, 0 };
  DescPose descStart = { 216.543, 445.175, 93.465, 179.683, 1.757, -112.641 };
  JointPos jointStart = { -128.345, -86.660, 114.679, -119.625, -89.219, 74.303 };
  DescPose descEnd = { 111.143, 523.384, 87.659, 179.703, 1.835, -97.750 };
  JointPos jointEnd = { -113.454, -81.060, 109.328, -119.954, -89.218, 74.302 };
  robot.MoveL(&jointStart, &descStart, 1, 1, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese);
  robot.MoveL(&jointEnd, &descEnd, 1, 1, 100, 100, 100, -1, &exaxisPos, 0, 0, &offdese);
  DescPose descREF0A = { 142.135, 367.604, 86.523, 179.728, 1.922, -111.089 };
  JointPos jointREF0A = { -126.794, -100.834, 128.922, -119.864, -89.218, 74.302 };
  DescPose descREF0B = { 254.633, 463.125, 72.604, 179.845, 2.341, -114.704 };
  JointPos jointREF0B = { -130.413, -81.093, 112.044, -123.163, -89.217, 74.303 };
  DescPose descREF1A = { 92.556, 485.259, 47.476, -179.932, 3.130, -97.512 };
  JointPos jointREF1A = { -113.231, -83.815, 119.877, -129.092, -89.217, 74.303 };
  DescPose descREF1B = { 203.103, 583.836, 63.909, 179.991, 2.854, -103.372 };
  JointPos jointREF1B = { -119.088, -69.676, 98.692, -121.761, -89.219, 74.303 };
  rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
  robot.MoveL(&jointREF0A, &descREF0A, 1, 1, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese); //Start point
  robot.MoveL(&jointREF0B, &descREF0B, 1, 1, 100, 100, 100, -1, &exaxisPos, 1, 0, &
↪offdese); //Direction point
  rtn1 = robot.WireSearchWait("REF0");
  rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);
  rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
  robot.MoveL(&jointREF1A, &descREF1A, 1, 1, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese); //Start point
  robot.MoveL(&jointREF1B, &descREF1B, 1, 1, 100, 100, 100, -1, &exaxisPos, 1, 0, &
↪offdese); //Direction point
  rtn1 = robot.WireSearchWait("REF1");
  rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);
  rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
  robot.MoveL(&jointREF0A, &descREF0A, 1, 1, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese); //Start point
  robot.MoveL(&jointREF0B, &descREF0B, 1, 1, 100, 100, 100, -1, &exaxisPos, 1, 0, &
↪offdese); //Direction point
  rtn1 = robot.WireSearchWait("RES0");
  rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);
  rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
  robot.MoveL(&jointREF1A, &descREF1A, 1, 1, 100, 100, 100, -1, &exaxisPos, 0, 0, &
↪offdese); //Start point
  robot.MoveL(&jointREF1B, &descREF1B, 1, 1, 100, 100, 100, -1, &exaxisPos, 1, 0, &
↪offdese); //Direction point
  rtn1 = robot.WireSearchWait("RES1");
  rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);
  vector <string> varNameRef = { "REF0", "REF1", "#", "#", "#", "#" };
  vector <string> varNameRes = { "RES0", "RES1", "#", "#", "#", "#" };
  int offectFlag = 0;
  DescPose offectPos = { 0, 0, 0, 0, 0, 0 };
  rtn0 = robot.GetWireSearchOffset(0, 0, varNameRef, varNameRes, offectFlag, offectPos);
```

(continues on next page)

(continued from previous page)

```
 robot.PointsOffsetEnable(0, &offectPos);
 robot.MoveL(&jointStart, &descStart, 1, 1, 100, 100, 100, -1, &exaxisPos, 0, 0, &offdese);
 robot.MoveL(&jointEnd, &descEnd, 1, 1, 100, 100, 100, -1, &exaxisPos, 1, 0, &offdese);
 robot.PointsOffsetDisable();
 robot.CloseRPC();
 return 0;
```

2.1.14.64 Set Welding Voltage Gradual Change Start

```
/**
* @brief Set welding voltage gradual change start
* @param [in] IOType Control type; 0-Control box IO; 1-Digital communication protocol (UDP);2-Digital communication protocol (ModbusTCP)
* @param [in] voltageStart Starting welding voltage (V)
* @param [in] voltageEnd Ending welding voltage (V)
* @param [in] AOIndex Control box AO port number (0-1)
* @param [in] blend Whether to smooth 0-No smoothing; 1-Smooth
* @return Error code
*/
errno_t WeldingSetVoltageGradualChangeStart(int IOType, double voltageStart, double voltageEnd, int AOIndex, int blend);
```

2.1.14.65 Set Welding Voltage Gradual Change End

```
/**
 * @brief Set welding voltage gradual change end
 * @return Error code
 */
errno_t WeldingSetVoltageGradualChangeEnd();
```

2.1.14.66 Set Welding Current Gradual Change Start

```
/**
 * @brief Set welding current gradual change start
 * @param [in] IOType Control type; 0-Control box IO; 1-Digital communication protocol (UDP);2-Digital communication protocol (ModbusTCP)
 * @param [in] voltageStart Starting welding current (A)
 * @param [in] voltageEnd Ending welding current (A)
 * @param [in] AOIndex Control box AO port number (0-1)
 * @param [in] blend Whether to smooth 0-No smoothing; 1-Smooth
 * @return Error code
 */
errno_t WeldingSetCurrentGradualChangeStart(int IOType, double currentStart, double currentEnd, int AOIndex, int blend);
```

2.1.14.67 Set Welding Current Gradual Change End

```
/**
 * @brief Set welding current gradual change end
 * @return Error code
 */
errno_t WeldingSetCurrentGradualChangeEnd();
```

2.1.14.68 Robot Welding Current Voltage Gradual Change Code Example

```
int WeldparamChange(FRRobot* robot)
{
  DescPose startdescPose = { -484.707, 276.996, -14.013, -37.657, -40.508, -1.548 };
  JointPos startjointPos = { -45.421, -75.673, 93.627, -104.302, -87.938, 6.005 };

  DescPose enddescPose = { -508.767, 137.109, -13.966, -37.639, -40.508, -1.559 };
  JointPos endjointPos = { -32.768, -80.947, 100.254, -106.201, -87.201, 18.648 };

  DescPose safedescPose = { -484.709, 294.436, 13.621, -37.660, -40.508, -1.545 };
  JointPos safejointPos = { -46.604, -75.410, 89.109, -100.003, -88.012, 4.823 };
  ExaxisPos exaxisPos = { 0, 0, 0, 0 };
  DescPose offdese = { 0, 0, 0, 0, 0, 0 };

  robot->WeldingSetCurrentRelation(0, 495, 1, 10, 0);
  robot->WeldingSetVoltageRelation(10, 45, 1, 10, 1);
  robot->MoveJ(&safejointPos, &safedescPose, 1, 0, 5, 100, 100, &exaxisPos, -1, 0, &
↪offdese);
  int rtn = robot->WeldingSetCurrentGradualChangeStart(0, 260, 220, 0, 0);
  cout << "WeldingSetCurrentGradualChangeStart rtn is " << rtn << endl;
  rtn = robot->WeldingSetVoltageGradualChangeStart(0, 25, 22, 1, 0);
  cout << "WeldingSetVoltageGradualChangeStart rtn is " << rtn << endl;
  rtn = robot->ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4,
↪ 1, 10, 0, 0);
  cout << "ArcWeldTraceControl rtn is " << rtn << endl;
  robot->MoveJ(&startjointPos, &startdescPose, 1, 0, 5, 100, 100, &exaxisPos, -1, 0, &
↪offdese);

  robot->ARCStart(0, 0, 10000);
  robot->WeaveStart(0);
  robot->WeaveChangeStart(2, 1, 24, 36);
  robot->MoveL(&endjointPos, &enddescPose, 1, 0, 100, 100, 2, -1, &exaxisPos, 0, 0, &
↪offdese);
  robot->ARCEnd(0, 0, 10000);
  robot->WeaveChangeEnd();
  robot->WeaveEnd(0);
  robot->ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1,␣
↪10, 0, 0);
  robot->WeldingSetCurrentGradualChangeEnd();
  robot->WeldingSetVoltageGradualChangeEnd();
  return 0;
}
```

2.1.14.69 Set Custom Weave Parameters

New in version C++SDK-v3.8.6.

```
/**
* @brief Set Custom Weave Parameters
* @param [in] id Custom weave ID: 0-2
* @param [in] pointNum Number of weave points 0-10
* @param [in] point Movement endpoint data x,y,z
* @param [in] stayTime Weave dwell time (ms)
* @param [in] frequency Weave frequency (Hz)
* @param [in] incStayType Wait mode: 0- Cycle does not include wait time; 1- Cycle␣
↪includes wait time
* @param [in] stationary Weave position wait: 0- Continue motion during wait time; 1-␣
↪Position stationary during wait time
* @return Error code
*/
errno_t CustomWeaveSetPara(int id, int pointNum, DescTran point[10], double stayTime[10],
↪ double frequency, int incStayType, int stationary);
```

2.1.14.70 Get Custom Weave Parameters

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Custom Weave Parameters
* @param [in] id Custom weave ID: 0-2
* @param [out] pointNum Number of weave points 0-10
* @param [out] point Movement endpoint data x,y,z
* @param [out] stayTime Weave dwell time (ms)
* @param [out] frequency Weave frequency (Hz)
* @param [out] incStayType Wait mode: 0- Cycle does not include wait time; 1- Cycle␣
↪includes wait time
* @param [out] stationary Weave position wait: 0- Continue motion during wait time; 1-␣
↪Position stationary during wait time
* @return Error code
*/
errno_t CustomWeaveGetPara(int id, int& pointNum, DescTran point[10], double␣
↪stayTime[10], double& frequency, int& incStayType, int& stationary);
```

2.1.14.71 Custom Weave Parameters Code Example

New in version C++SDK-v3.8.6.

```
int TestCustomWeaveSetPara()
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
```

(continues on next page)

(continued from previous page)

```
  if (rtn != 0)
  {
    return 0;
  }
  robot.SetReConnectParam(true, 30000, 500);
  DescTran point[10] = {};
  point[0].x = -3;
  point[0].y = -3;
  point[0].z = 0;
  point[1].x = -6;
  point[1].y = 0;
  point[1].z = 0;
  point[2].x = -3;
  point[2].y = 3;
  point[2].z = 0;
  point[3].x = 0;
  point[3].y = 0;
  point[3].z = 0;
  double stayTime[10] = { 0,0,0,0,0,0,0,0,0,0 };
  rtn = robot.CustomWeaveSetPara(2, 4, point, stayTime, 1.000, 0, 0);
  printf("CustomWeaveSetPara rtn is %d\n", rtn);
  robot.Sleep(1000);
  int pointNum = 0;
  double frequency;
  int incStayType;
  int stationary;
  robot.CustomWeaveGetPara(2, pointNum, point, stayTime, frequency, incStayType,␣
↪stationary);
  printf("pointNum is %d\n", pointNum);
  for (int i = 0; i < pointNum; i++)
  {
    printf("point %d, point x y z %f %f %f\n", i, point[i].x, point[i].y, point[i].z);
  }
  printf("fre is %f, stay is %d %d \n", frequency, incStayType, stationary);
  robot.WeaveSetPara(0, 9, 1.000000, 1, 5.000000, 6.000000, 5.000000, 50, 100, 100, 0, 1,
↪ 0.000000, 0.000000);
  DescPose desc_p1 = { -288.650, 367.807, 288.404, 0.000, -0.001, 0.001 };
  DescPose desc_p2 = { -431.714, 367.815, 288.415, 0.001, 0.001, 0.000 };
  DescPose desc_p3 = { -348.666, 427.798, 288.404, -0.000, -0.000, 0.001 };
  JointPos j1 = { 140.656, -84.560, -91.707, -93.734, 90.000, 50.655 };
  JointPos j2 = { 149.873, -98.298, -77.599, -94.103, 90.000, 59.873 };
  JointPos j3 = { 139.773, -96.173, -80.014, -93.814, 90.000, 49.772 };
  ExaxisPos epos = {};
  DescPose offset_pos = {};
  robot.MoveJ(&j1, &desc_p1, 3, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
  robot.WeaveStart(0);
  robot.Circle(&j3, &desc_p3, 3, 0, 100, 100, &epos, &j2, &desc_p2, 3, 0, 100, 100, &
↪epos, 10, -1, &offset_pos);
  robot.WeaveEnd(0);
  robot.MoveJ(&j1, &desc_p1, 3, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
  robot.WeaveStart(0);
  robot.MoveC(&j3, &desc_p3, 3, 0, 100, 100, &epos, 0, &offset_pos, &j2, &desc_p2, 3, 0,␣
```

(continues on next page)

(continued from previous page)

```
↪100, 100, &epos, 0, &offset_pos, 10, -1);
  robot.WeaveEnd(0);
  robot.MoveJ(&j1, &desc_p1, 3, 0, 100, 100, 100, &epos, -1, 0, &offset_pos);
  robot.WeaveStart(0);
  robot.MoveL(&j2, &desc_p2, 3, 0, 100, 100, 10, -1, &epos, 0, 0, &offset_pos, 0, 100);
  robot.WeaveEnd(0);
  robot.CloseRPC();
}
```

2.1.15 Other Interfaces

2.1.15.1 Get SSH Public Key

```
/**
* @brief Get SSH public key
* @param [out] keygen Public key
* @return Error code
*/
errno_t GetSSHKeygen(char keygen[1024]);
```

2.1.15.2 Send SCP Command

```
/**
* @brief Send SCP command
* @param [in] mode 0-Upload (host->controller), 1-Download (controller->host)
* @param [in] sshname Host username
* @param [in] sship Host IP address
* @param [in] usr_file_url Host file path
* @param [in] robot_file_url Robot controller file path
* @return Error code
*/
errno_t SetSSHScpCmd(int mode, char sshname[32], char sship[32], char usr_file_url[128],␣
↪char robot_file_url[128]);
```

2.1.15.3 Calculate MD5 Value of Specified File

```
/**
* @brief Calculate MD5 value of specified file
* @param [in] file_path File path including filename, default Traj folder path: "/fruser/
↪traj/", e.g. "/fruser/traj/trajHelix_aima_1.txt"
* @param [out] md5 File MD5 value
* @return Error code
*/
errno_t ComputeFileMD5(char file_path[256], char md5[256]);
```

2.1.15.4 Robot SSH/MD5 Command Code Example

```
int TestSSHMd5(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  char file_path[256] = "/fruser/airlab.lua";
  char md5[256] = { 0 };
  uint8_t emerg_state = 0;
  uint8_t si0_state = 0;
  uint8_t si1_state = 0;
  int sdk_com_state = 0;
  char ssh_keygen[1024] = { 0 };
  int retval = robot.GetSSHKeygen(ssh_keygen);
  printf("GetSSHKeygen retval is: %d\n", retval);
  printf("ssh key is: %s \n", ssh_keygen);
  char ssh_name[32] = "fr";
  char ssh_ip[32] = "192.168.58.45";
  char ssh_route[128] = "/home/fr";
  char ssh_robot_url[128] = "/root/robot/dhpara.config";
  retval = robot.SetSSHScpCmd(1, ssh_name, ssh_ip, ssh_route, ssh_robot_url);
  printf("SetSSHScpCmd retval is: %d\n", retval);
  printf("robot url is: %s\n", ssh_robot_url);
  robot.ComputeFileMD5(file_path, md5);
  printf("md5 is: %s \n", md5);
  robot.CloseRPC();
  return 0;
}
```

2.1.15.5 Set Robot Port 20004 Feedback Cycle

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Set robot port 20004 feedback cycle
* @param [in] period Robot port 20004 feedback cycle (ms)
* @return Error code
*/
errno_t SetRobotRealtimeStateSamplePeriod(int period);
```

2.1.15.6 Get Robot Port 20004 Feedback Cycle

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get robot port 20004 feedback cycle
* @param [out] period Robot port 20004 feedback cycle (ms)
* @return Error code
*/
errno_t GetRobotRealtimeStateSamplePeriod(int& period);
```

2.1.15.7 Robot Port 20004 State Feedback Cycle Configuration Example

```
int TestRealtimePeriod(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.SetRobotRealtimeStateSamplePeriod(10);
  int getPeriod = 0;
  robot.GetRobotRealtimeStateSamplePeriod(getPeriod);
  cout << "period is " << getPeriod << endl;
  robot.Sleep(1000);
  robot.CloseRPC();
  return 0;
}
```

2.1.15.8 Robot Software Upgrade

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Robot software upgrade
* @param [in] filePath Full path of software upgrade package
* @param [in] block Whether to block until upgrade completes (true: block; false: non-
↪block)
* @return Error code
*/
errno_t SoftwareUpgrade(std::string filePath, bool block);
```

2.1.15.9 Get Robot Software Upgrade Status

New in version C++SDK-v2.1.5.0.

```
/**
* @brief Get robot software upgrade status
* @param [out] state Robot software upgrade status (0-Idle or uploading upgrade package; 1~100: Upgrade completion percentage; -1: Upgrade failed; -2: Verification failed; -3: Version verification failed; -4: Decompression failed; -5: User configuration upgrade failed; -6: Peripheral configuration upgrade failed; -7: Extended axis configuration upgrade failed; -8: Robot configuration upgrade failed; -9: DH parameter configuration upgrade failed)
* @return Error code
*/
errno_t GetSoftwareUpgradeState(int &state);
```

2.1.15.10 Robot Software Upgrade Code Example

```
int TestUpgrade(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(3);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  robot.SoftwareUpgrade("D://zUP/QNX/software.tar.gz", false);
  while (true)
  {
    int curState = -1;
    robot.GetSoftwareUpgradeState(curState);
    printf("upgrade state is %d\n", curState);
    robot.Sleep(300);
  }
  robot.CloseRPC();
  return 0;
}
```

2.1.15.11 Download Point Table Database

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Download point table database
* @param [in] pointTableName Point table name to download    pointTable1.db
* @param [in] saveFilePath Storage path for downloaded point table   C://test/
* @return Error code
*/
errno_t PointTableDownLoad(const std::string &pointTableName, const std::string &
↪saveFilePath);
```

2.1.15.12 Upload Point Table Database

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Upload point table database
* @param [in] pointTableFilePath Full path name of point table to upload   C://test/
↪pointTable1.db
* @return Error code
*/
errno_t PointTableUpLoad(const std::string &pointTableFilePath);
```

2.1.15.13 Update Lua File for Point Table

New in version C++SDK-v2.1.1.0.

```
/**
* @brief Update Lua file for point table
* @param [in] pointTableName Point table name to switch to   "pointTable1.db", when␣
↪empty (""), means updating Lua program to initial program without applied point table
* @param [in] luaFileName Lua file name to update   "testPointTable.lua"
* @param [out] errorStr Error message for switching point table
* @return Error code
*/
errno_t PointTableUpdateLua(const std::string &pointTableName, const std::string &
↪luaFileName);
```

2.1.15.14 Robot Point Table Operation Code Example

```
int TestPointTable(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
```

(continues on next page)

(continued from previous page)

```
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  string save_path = "D://zDOWN/";
  string point_table_name = "point_table_FR5.db";
  rtn = robot.PointTableDownLoad(point_table_name, save_path);
  cout << "download : " << point_table_name << " fail: " << rtn << endl;
  string upload_path = "D://zUP/point_table_FR5.db";
  rtn = robot.PointTableUpLoad(upload_path);
  cout << "retval is: " << rtn << endl;
  string point_tablename = "point_table_FR5.db";
  string lua_name = "airlab.lua";
  rtn = robot.PointTableUpdateLua(point_tablename, lua_name);
  cout << "retval is: " << rtn << endl;
  robot.CloseRPC();
  return 0;
}
```

2.1.15.15 Controller Log Download

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Controller log download
* @param [in] savePath Save file path "D://zDown/"
* @return Error code
*/
errno_t RbLogDownload(std::string savePath);
```

2.1.15.16 All Data Source Download

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief All data source download
* @param [in] savePath Save file path "D://zDown/"
* @return Error code
*/
errno_t AllDataSourceDownload(std::string savePath);
```

2.1.15.17 Data Backup Package Download

New in version C++SDK-v2.2.1-3.8.1.

```
/**
* @brief Data backup package download
* @param [in] savePath Save file path "D://zDown/"
* @return Error code
*/
errno_t DataPackageDownload(std::string savePath);
```

2.1.15.18 Download Controller Data Code Example

```
int TestDownLoadRobotData(void)
{
  ROBOT_STATE_PKG pkg = {};
  FRRobot robot;
  robot.LoggerInit();
  robot.SetLoggerLevel(1);
  int rtn = robot.RPC("192.168.58.2");
  if (rtn != 0)
  {
    return -1;
  }
  robot.SetReConnectParam(true, 30000, 500);
  rtn = robot.RbLogDownload("D://zDOWN/");
  cout << "RbLogDownload rtn is " << rtn << endl;
  rtn = robot.AllDataSourceDownload("D://zDOWN/");
  cout << "AllDataSourceDownload rtn is " << rtn << endl;
  rtn = robot.DataPackageDownload("D://zDOWN/");
  cout << "DataPackageDownload rtn is " << rtn << endl;
  robot.CloseRPC();
  return 0;
}
```

2.1.15.19 Set Joint Firmware Upgrade

```
/**
* @brief Set joint firmware upgrade
* @param [in] type Upgrade file type: 1-Firmware upgrade (requires boot mode); 2-Slave␣
↪config file upgrade (requires robot disable)
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
errno_t SetJointFirmwareUpgrade(int type, std::string path);
```

2.1.15.20 Set Controller Firmware Upgrade

```
/**
* @brief Set controller firmware upgrade
* @param [in] type Upgrade file type: 1-Firmware upgrade (requires boot mode); 2-Slave␣
↪config file upgrade (requires robot disable)
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
errno_t SetCtrlFirmwareUpgrade(int type, std::string path);
```

2.1.15.21 Set End-Effector Firmware Upgrade

```
/**
* @brief Set end-effector firmware upgrade
* @param [in] type Upgrade file type: 1-Firmware upgrade (requires boot mode); 2-Slave␣
↪config file upgrade (requires robot disable)
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
errno_t SetEndFirmwareUpgrade(int type, std::string path);
```

2.1.15.22 Joint Full Parameter Configuration Upgrade

```
/**
* @brief Joint full parameter configuration upgrade (requires robot disable)
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
errno_t JointAllParamUpgrade(std::string path);
```

2.1.15.23 Robot Slave Firmware Upgrade Code Example

```
int TestFirmWareUpgrade()
{
ROBOT_STATE_PKG pkg = {};
FRRobot robot;
robot.LoggerInit();
robot.SetLoggerLevel(1);
int rtn = robot.RPC("192.168.58.2");
if (rtn != 0)
{
return -1;
}
robot.SetReConnectParam(true, 30000, 500);
robot.RobotEnable(0);
robot.Sleep(200);
rtn = robot.JointAllParamUpgrade("D://zUP/upgrade/jointallparameters.db");
```

(continues on next page)

(continued from previous page)

```
printf("robot JointAllParamUpgrade rtn is %d\n", rtn);
rtn = robot.SetCtrlFirmwareUpgrade(2, "D://zUP/upgrade/FAIR_Cobot_Cbd_Asix_V2.0.bin");
printf("robot SetCtrlFirmwareUpgrade config param rtn is %d\n", rtn);
rtn = robot.SetEndFirmwareUpgrade(2, "D://zUP/upgrade/FAIR_Cobot_Axle_Asix_V2.4.bin");
printf("robot SetEndFirmwareUpgrade config param rtn is %d\n", rtn);
robot.SetSysServoBootMode();
rtn = robot.SetCtrlFirmwareUpgrade(1, "D://zUP/upgrade/FR_CTRL_PRIMCU_FV201212_MAIN_U4_
↪T01_20250428(MT).bin");
printf("robot SetCtrlFirmwareUpgrade rtn is %d\n", rtn);
rtn = robot.SetEndFirmwareUpgrade(1, "D://zUP/upgrade/FR_END_FV201009_MAIN_U1_T01_
↪20250428.bin");
printf("robot SetEndFirmwareUpgrade rtn is %d\n", rtn);
rtn = robot.SetJointFirmwareUpgrade(1, "D://zUP/upgrade/FR_SERVO_FV504214_MAIN_U7_T07_
↪20250519.bin");
printf("robot SetJointFirmwareUpgrade rtn is %d\n", rtn);
robot.CloseRPC();
return 0;
}
```

2.1.15.24 Robot Operating System Upgrade (LA Control Box)

New in version C++SDK-v3.8.6.

```
/**
* @brief Robot Operating System Upgrade (LA Control Box)
* @param [in] filePath Full path of the operating system upgrade package
* @return Error code
*/
errno_t KernelUpgrade(std::string filePath);
```

2.1.15.25 Get Robot Operating System Upgrade Result (LA Control Box)

New in version C++SDK-v3.8.6.

```
/**
* @brief Get Robot Operating System Upgrade Result (LA Control Box)
* @param [out] result Upgrade result: 0: Success; -1: Failure
* @return Error code
*/
errno_t GetKernelUpgradeResult(int& result);
```

2.1.15.26 Robot MCU log generation

```
/**
* @brief robot MCU log generation
* @return error code
*/
Errno RobotMCULogCollect();
```

2.1.16 Appendix

2.1.16.1 Source Code Download

On the Fairino documentation site (https://fairino-doc-en.readthedocs.io/latest/), locate the "Resources Download" section, click the "CPP SDK" button, then click "FAIRINO CPP SDK" on the right page and wait for the browser to complete the download.

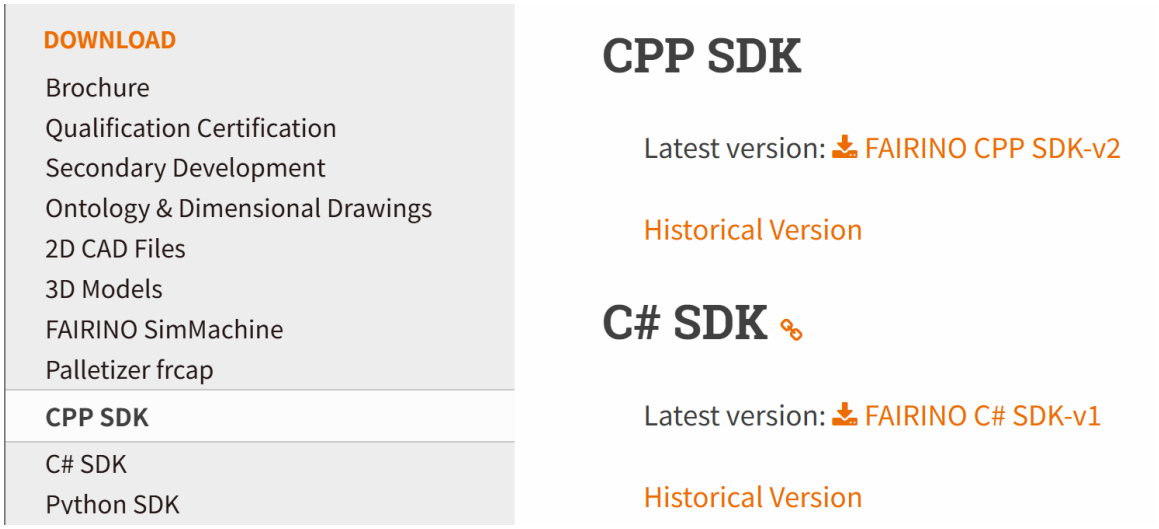


Figure 15.1-1 C++SDK Source Code Download

After extracting the package, the directory structure is as shown below, including:

- windows: Header files and library files (.lib and .dll) compiled for common environments like VS2015~VS2019, including both Debug and Release modes
- linux: Header files and library files (.so) for common environments like gcc, rk3399, rk3568
- libfairino: C++SDK source code

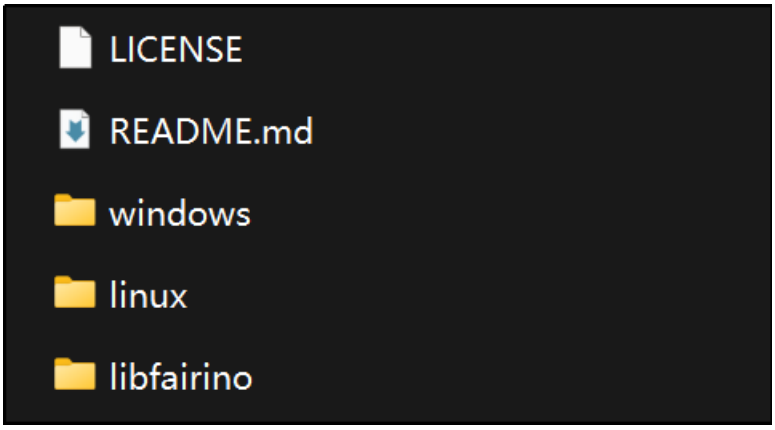


Figure 15.1-2 C++SDK Source Directory

2.1.16.2 Windows Platform Compilation

Open Visual Studio, click “Continue without code” at bottom right;

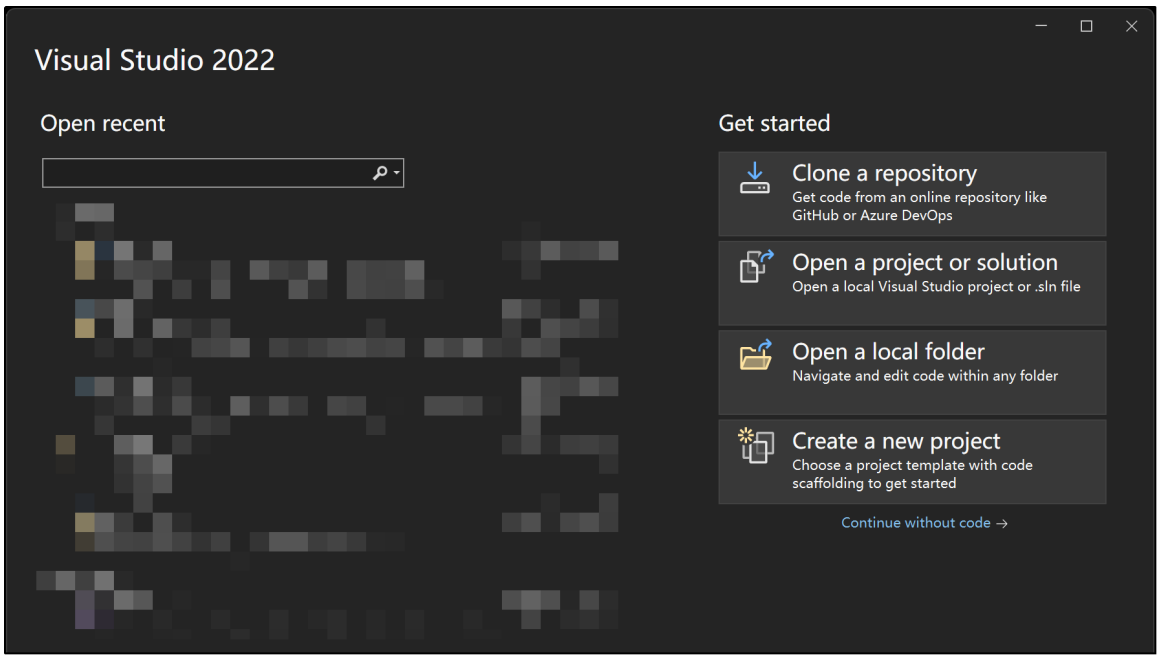


Figure 15.2-1 Opening Visual Studio

Navigate to “File” → “Open” → “CMake”, select the “CMakeLists.txt” file from the downloaded C++SDK source code (libfairinoCMakeLists.txt). Visual Studio will automatically load the project based on CMakeLists.txt definitions.

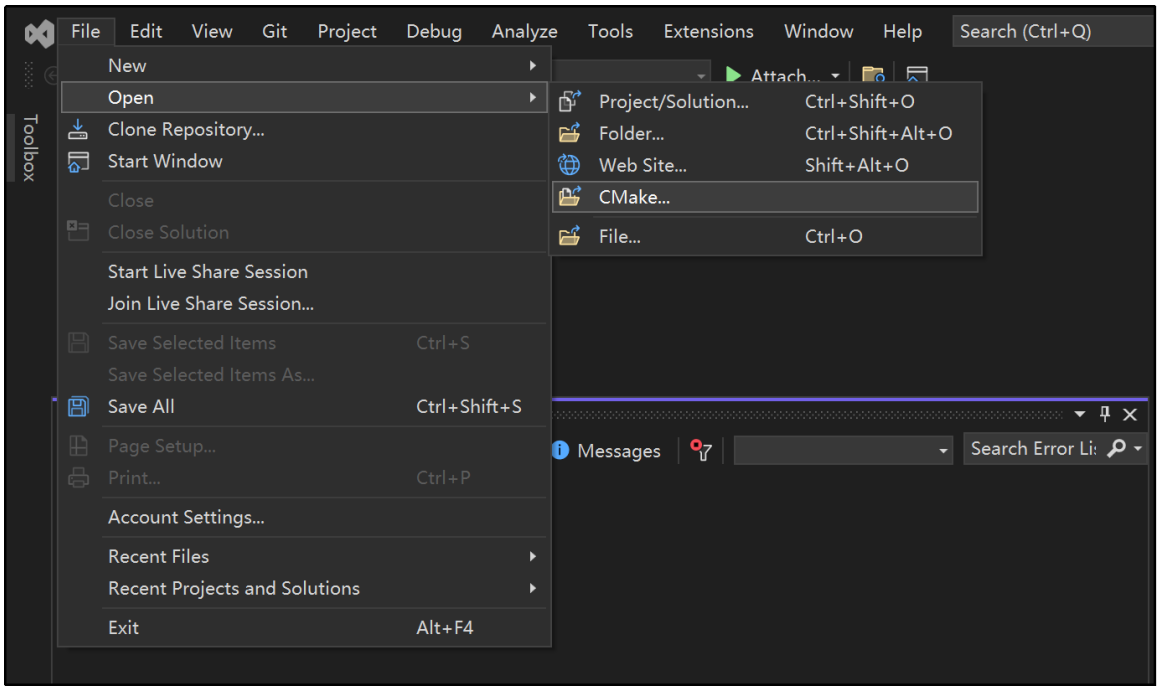


Figure 15.2-2 Opening CMake Project

Select build platform ("x64-Debug" or "x64-Release" etc.) and set startup item to "fairino.dll".

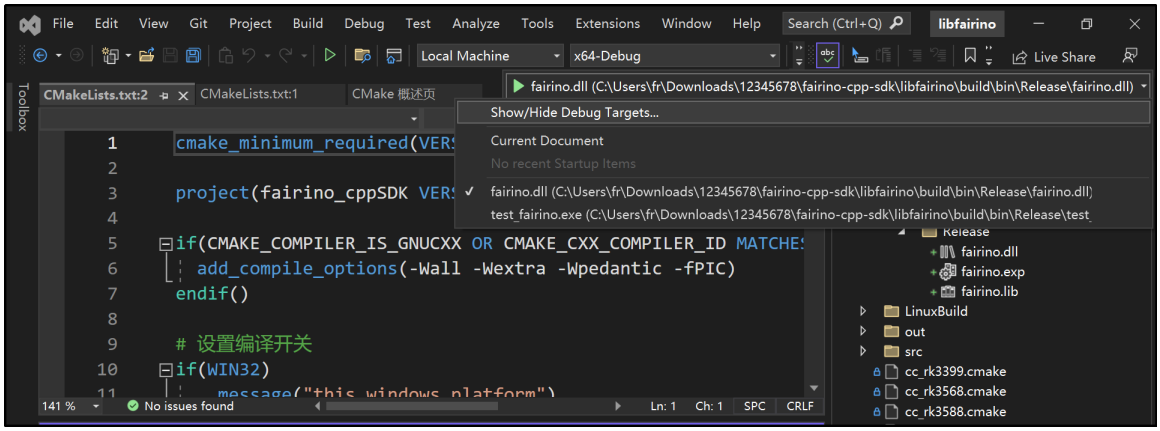


Figure 15.2-3 Selecting Startup Item

Navigate to "Build" → "Rebuild fairino.dll" in the menu bar to start compilation.

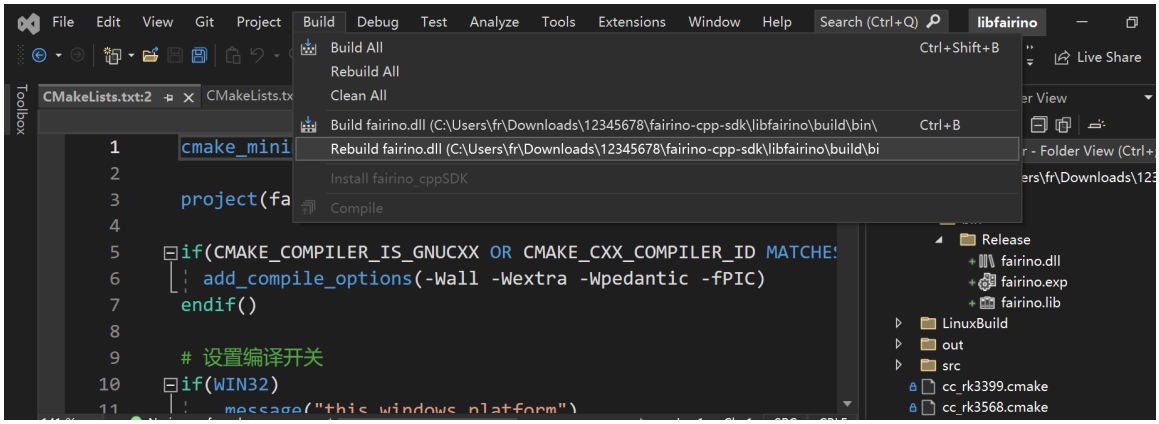


Figure 15.2-4 Building fairino.dll

Locate the compiled fairino.dll and fairino.lib files in the “build” folder under project directory.

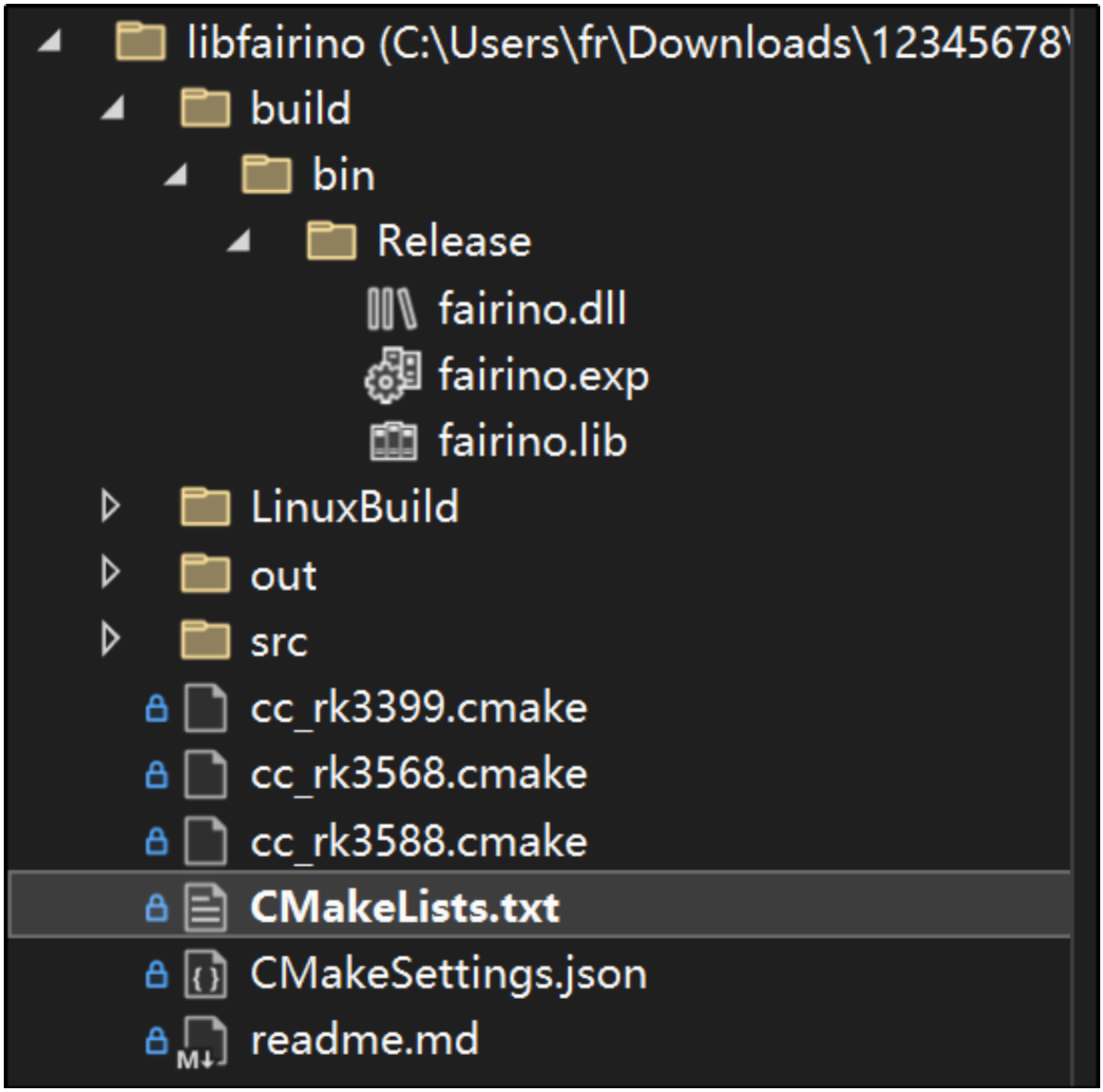


Figure 15.2-5 Locating fairino.lib and fairino.dll

To use the C++SDK: First copy the three header files (“robot.h”, “robot_error.h”, “robot_type.h”) from libfairinosrcincludeRobot-CNto your project directory. Add fairino.lib to link libraries, then place fairino.dll in the executable directory.

2.1.16.3 Linux Platform Compilation

Before compiling on Linux, ensure gcc, g++ compilers and cmake build system (v3.10+) are installed.

The “buildGcc.sh” script in libfairinolinuxBuildcontains commands for: “cmake..”, “make”, and copying final header/library files to linuxBuild. Execute this script to compile the C++SDK.

Open terminal, navigate to libfairinolinuxBuild, execute: “sh buildGcc.sh” and wait for completion.

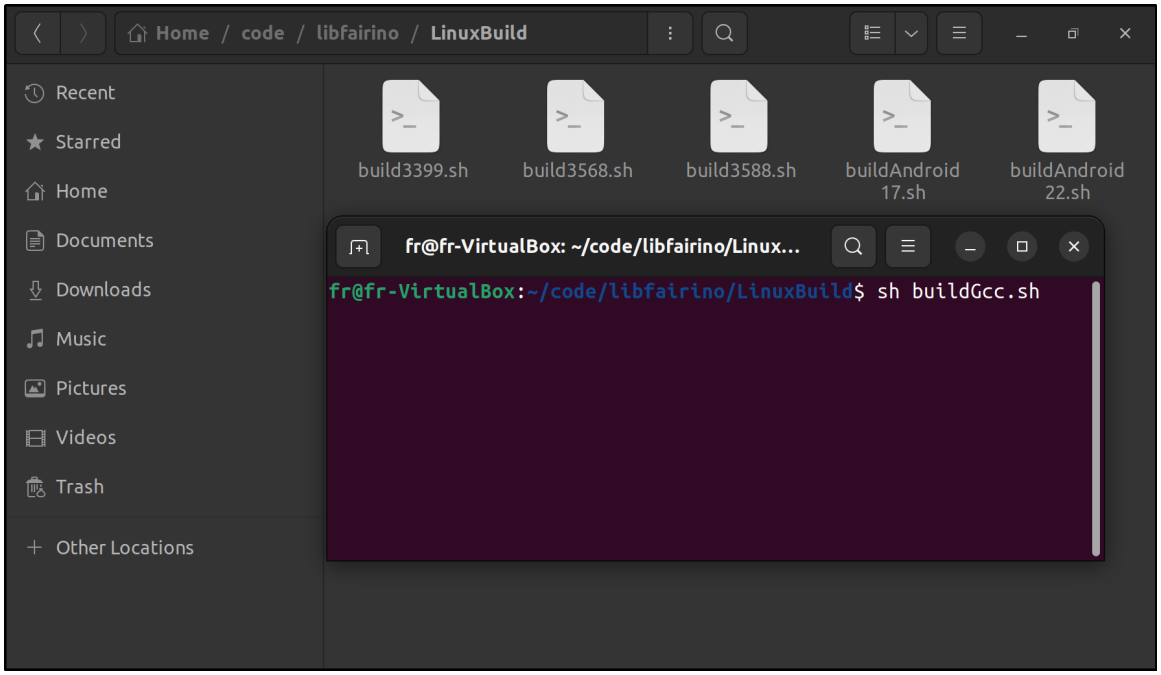


Figure 15.3-1 Executing Build Script

After compilation, the includeand libdirectories under libfairinolinuxBuildwill contain required header and library files.

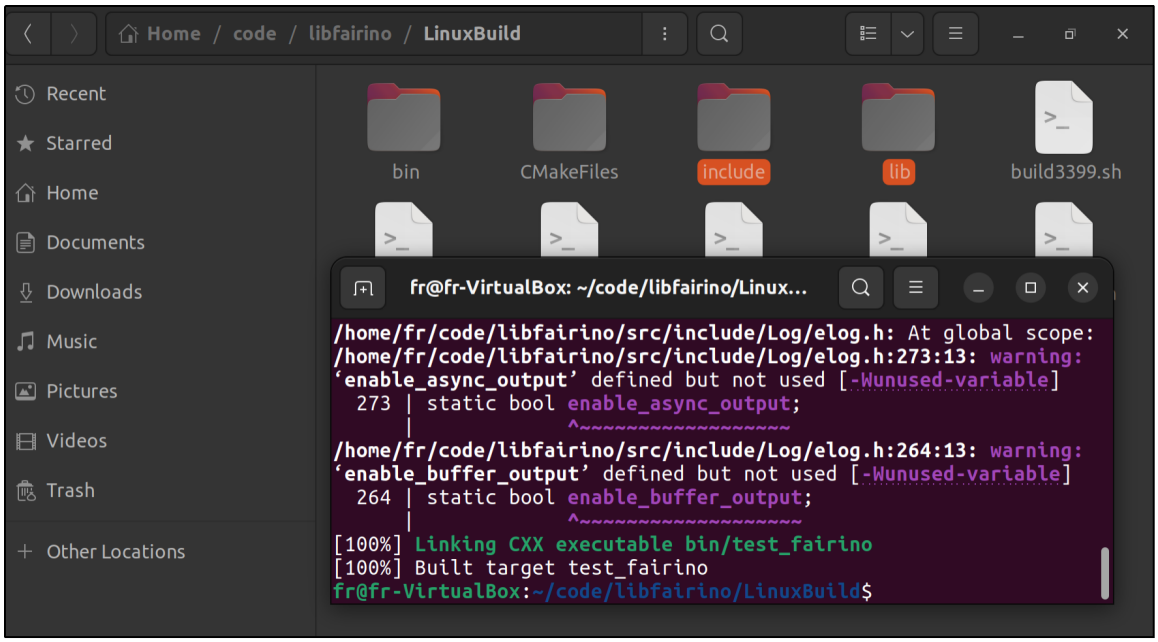


Figure 15.3-2 Compilation Results

2.2 C#

This document is the C# version of the secondary development interface document.

Important: Description of robot parameter units: robot position in millimeters (mm) and attitude in degrees (°).

Important:

1) All code examples in the documentation default to the robot being powered on and enabled unless otherwise specified;
2) All code examples in the documentation default to no interference in the robot's workspace;
3) Please use the data of the robot in the field for actual use testing.
4) Before using this SDK, you need to find the "xmlrpcnet" package through NuGet and add it to the project reference;

2.2.1 Version Update Description

Version	Date	Update Description
V3.9.3	2026-02-11	1.ServoCart() interface added extended axis parameters 2.SetOutputResetCtlBoxDO() interface added parameter for whether to reload DO state before reset after pause resume 3.SetOutputResetCtlBoxAO() interface added parameter for whether to reload DO state before reset after pause resume 4.SetOutputResetAxleDO() interface added parameter for whether to reload DO state before reset after pause resume 5.SetOutputResetAxleAO() interface added parameter for whether to reload DO state before reset after pause resume 6.SetOutputResetExtDO() interface added parameter for whether to reload DO state before reset after pause resume 7.SetOutputResetExtAO() interface added parameter for whether to reload DO state before reset after pause resume 8.SetOutputResetSmartToolDO() interface added parameter for whether to reload DO state before reset after pause resume 9.Added GetInverseKinExaxis() inverse kinematics solution interface including extended axis position
V3.9.2	2026-01-26	1. Added a processing strategy parameter for undetected force/torque to the FT_RotInsertion() interface 2. Added robot fixed-point tracking related parameters to the LaserSensorRecordandReplay() interface 3. Added the MoveStationary() interface 4. Added the TCPComputeRPY() interface 5. Added the TCPComputeXYZ() interface 6. Added the TCPRecordFlangePosStart() interface 7. Added the TCPRecordFlangePosEnd() interface 8. Added the TCPGetRecordFlangePos() interface 9. Added the PhotoelectricSensorTCPCalibration() interface
V3.9.1	2025-12-25	1. Added oacc speed scaling factor parameter / physical acceleration parameter to the MoveL() interface; 2. Added oacc speed scaling factor parameter / physical acceleration parameter to the MoveC() interface; 3. Optimized parameter descriptions for physical speed and physical acceleration in the Circle() interface; 4. Added FT_Control() overloaded function with rx, ry activation thresholds and torque adjustment coefficient parameters; 5. Added SerCoderCompenParams() interface;
V3.9.0	2025-11-26	1. JointSensitivityCalibration() interface adds j1~j6 joint linearity return 2. Added JointHysteresisError() interface 3. Added JointRepeatability() interface 4. Added SetAdmittanceParams() interface 5. Added MoveToIntersectLineStart() interface

2.2.2 Data Structure Description

2.2.2.1 Joint position data type

```
/**
* @brief Joint position data type
*/
struct JointPos
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] jPos; /* Six joint positions in deg */
}
```

2.2.2.2 Cartesian space position data type

```
/**
* @brief Cartesian spatial position datatype.
*/
struct DescTran
{
    public double x; /* x-axis coordinate in mm */
    public double y; /* y-axis coordinate in mm */
    public double z; /* z-axis coordinate in mm */
}
```

2.2.2.3 Euler Angle Attitude data type

```
/**
* @brief Euler Angle Attitude data type.
*/
struct Rpy
{
public double rx; /* Angle of rotation around fixed axis X in deg */
public double ry; /* Angle of rotation around fixed axis Y in degrees */
public double rz; /* Angle of rotation about fixed axis Z in degrees */
}
```

2.2.2.4 Cartesian space position data type

```
/**
*@brief Cartesian space position type.
*/
struct DescPose
{
    public DescTran tran; /* Cartesian space position */
    public Rpy rpy; /* Cartesian space pose */
}
```

2.2.2.5 Extended axis position data type

```
/**
* @brief Extended axis position datatype.
*/
struct ExaxisPos
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public double[] ePos; /* Four extended axis positions in mm */
}
```

2.2.2.6 Torque sensor data type

```
/**
* @brief Force components and moment components of a force sensor.
*/
struct ForceTorque
{ public double fx; /* Force component along x-axis, in N */
    public double fx; /* Force component along x-axis in N */
    public double fy; /* Force component along y-axis in N */
    public double fz; /* Component of force along z-axis in N */
    public double tx; /* Component of moment around x-axis, unit Nm */
    public double ty; /* Component of moment around y-axis, in Nm */
    public double tz; /* Moment component around z-axis, in Nm */
}
```

2.2.2.7 Helix parameter data type

```
/**
* @brief Spiral parameter data types
*/
public struct SpiralParam
{
    public int circle_num;           /* Number of spiral turns */
    public float circle_angle;       /* Spiral inclination angle */
    public float rad_init;           /* Initial spiral radius, unit: mm */
    public float rad_add;            /* Radius increment */
    public float rotaxis_add;        /* Rotation axis direction increment */
    public uint rot_direction;       /* Rotation direction, 0-clockwise, 1-
↪counterclockwise */
    public int velAccMode;           // Velocity/acceleration parameter mode: 0-constant␣
↪angular velocity; 1-constant linear velocity

    public SpiralParam(int num, float angle, float initRad, float addRad, float axisAdd,␣
↪uint direction, int mode)
    {
        circle_num = num;
        circle_angle = angle;
        rad_init = initRad;
        rad_add = addRad;
```

(continues on next page)

(continued from previous page)

```
        rotaxis_add = axisAdd;
        rot_direction = direction;
        velAccMode = mode;
    }
}
```

2.2.2.8 Extended axis state type

Changed in version C#SDK-v1.0.7.

```
/**
* @brief Extended axis state type
*/
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public struct ROBOT_AUX_STATE
{
    public byte servoId; // servo drive ID number
    public int servoErrCode; //servo drive error code
    public int servoState; //servo drive state
    public double servoPos; //Servo current position
    public float servoVel; //Servo current speed
    public float servoTorque; //Servo current torque
}
```

2.2.2.9 Welding interrupt status

```
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public struct WELDING_BREAKOFF_STATE
{
    public byte breakOffState;  // Welding interrupt status
    public byte weldArcState;   // Welding arc interrupted state
}
```

2.2.2.10 Robot State Feedback Structures Types

New in version C#SDK-V1.1.4: Web-3.8.3

```
/**
* @brief Robot status feedback struct type.
*/
[StructLayout(LayoutKind.Sequential, Pack = 1)]
public struct ROBOT_STATE_PKG
{
    public UInt16 frame_head;           //Frame header 0x5A5A
    public byte frame_cnt;              //Frame count
    public UInt16 data_len;             //Data length 5
    public byte program_state;          //Program running state, 1-stop; 2-running; 3-
↪pause
```

(continues on next page)

(continued from previous page)

```
    public byte robot_state;            //Robot motion state, 1-stop; 2-running; 3-pause;
↪ 4-drag 7
    public int main_code;               //Main fault code
    public int sub_code;                //Sub fault code
    public byte robot_mode;             //Robot mode, 0-auto mode; 1-manual mode 16

    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] jt_cur_pos;                                 //Joint current position
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] tl_cur_pos;                                 //Tool current pose
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] flange_cur_pos;                             //End flange current pose
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] actual_qd;                                  //Robot current joint␣
↪velocity
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] actual_qdd;                                 //Robot current joint␣
↪acceleration 16 + 8 * 6 * 5 = 256
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
    public double[] target_TCP_CmpSpeed;                        //Robot TCP composite␣
↪command velocity
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] target_TCP_Speed;                           //Robot TCP command velocity
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
    public double[] actual_TCP_CmpSpeed;                        //Robot TCP composite actual␣
↪velocity
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] actual_TCP_Speed;                           //Robot TCP actual velocity
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] jt_cur_tor;                                 //Current torque
    public int tool;                        //Tool number
    public int user;                        //Workpiece number
    public byte cl_dgt_output_h;            //Digital output 15-8
    public byte cl_dgt_output_l;            //Digital output 7-0
    public byte tl_dgt_output_l;            //Tool digital output 7-0 (only bit0-bit1␣
↪valid)
    public byte cl_dgt_input_h;             //Digital input 15-8
    public byte cl_dgt_input_l;             //Digital input 7-0
    public byte tl_dgt_input_l;             //Tool digital input 7-0 (only bit0-bit1␣
↪valid)
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
    public UInt16[] cl_analog_input;        //Control box analog input
    public UInt16 tl_anglog_input;          //Tool analog input
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] ft_sensor_raw_data;     //Force/torque sensor raw data
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] ft_sensor_data;         //Force/torque sensor data
    public byte ft_sensor_active;           //Force/torque sensor activation state, 0-
↪reset, 1-active
    public byte EmergencyStop;              //Emergency stop flag
    public int motion_done;                 //In-position signal
    public byte gripper_motiondone;         //Gripper motion complete signal
```

(continues on next page)

(continued from previous page)

```
    public int mc_queue_len;                //Motion queue length
    public byte collisionState;             //Collision detection, 1-collision; 0-no␣
↪collision
    public int trajectory_pnum;             //Trajectory point number
    public byte safety_stop0_state;  /* Safety stop signal SI0 */
    public byte safety_stop1_state;  /* Safety stop signal SI1 */
    public byte gripper_fault_id;    /* Faulty gripper number */
    public UInt16 gripper_fault;     /* Gripper fault */
    public UInt16 gripper_active;    /* Gripper activation state */
    public byte gripper_position;    /* Gripper position */
    public byte gripper_speed;       /* Gripper speed */
    public byte gripper_current;     /* Gripper current */
    public int gripper_tmp;          /* Gripper temperature */
    public int gripper_voltage;      /* Gripper voltage */
    public ROBOT_AUX_STATE auxState; /* 485 extension axis state */
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public EXT_AXIS_STATUS[] extAxisStatus;  /* UDP extension axis state */
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
    public UInt16[] extDIState;        //Extended DI input
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
    public UInt16[] extDOState;        //Extended DO output
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public UInt16[] extAIState;        //Extended AI input
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public UInt16[] extAOState;        //Extended AO output
    public int rbtEnableState;       //Robot enable state
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] jointDriverTorque;               //Joint driver torque
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] jointDriverTemperature;          //Joint driver temperature
    public ROBOT_TIME robotTime;     //Robot system time
    public int softwareUpgradeState; //Software upgrade state 0-idle or uploading␣
↪upgrade package; 1~100: upgrade completion percentage; -1: upgrade software failed; -
↪2: verification failed; -3: version verification failed; -4: decompression failed; -5:␣
↪user configuration upgrade failed; -6: peripheral configuration upgrade failed; -7:␣
↪extension axis configuration upgrade failed; -8: robot configuration upgrade failed; -
↪9: DH parameter configuration upgrade failed
    public UInt16 endLuaErrCode;    //End Lua running state
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
    public UInt16[] cl_analog_output;  //Control box analog output
    public UInt16 tl_analog_output;     //Tool analog output
    public float gripperRotNum;           //Rotating gripper current rotation turns
    public byte gripperRotSpeed;       //Rotating gripper current rotation speed␣
↪percentage
    public byte gripperRotTorque;      //Rotating gripper current rotation torque␣
↪percentage
    public WELDING_BREAKOFF_STATE weldingBreakOffState;//Welding breakoff state

    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] jt_tgt_tor;//Joint command torque
    public int smartToolState; //SmartTool handle button state
    public float wideVoltageCtrlBoxTemp;        //Wide voltage control box temperature
```

(continues on next page)

(continued from previous page)

```
    public UInt16 wideVoltageCtrlBoxFanVel;   //Wide voltage control box fan current (mA)

    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] toolCoord;         //Tool coordinate system
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] wobjCoord;         //Workpiece coordinate system
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] extoolCoord;        //External tool coordinate system
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] exAxisCoord;          //Extension axis coordinate system
    public double load;                   //Load mass

    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
    public double[] loadCog;           //Load center of gravity
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
    public double[] lastServoTarget;//Last servoJ target position in queue

    public int servoJCmdNum;// servoJ command count
    public UInt16 check_sum;         /* Checksum */
}
```

2.2.3 Robot Basics

2.2.3.1 Instantiating the Robot

```
/**
* @brief  Robot interface class constructor
*/
Robot();
```

2.2.3.2 Establishing Communication with the Controller

```
/**
* @brief  Establish communication with the robot controller
* @param  [in] ip  Controller IP address, default is 192.168.58.2
* @return Error code
*/
int RPC(string ip);
```

2.2.3.3 Disconnect from the robot

```
/**
* @brief Disconnect from the robot controller
* @return Error code
*/
int CloseRPC();
```

2.2.3.4 Query the SDK version number

```
/**
* @brief Query the SDK version number
* @param [out] version SDK version number
* @return Error code
*/
int GetSDKVersion(ref string version);
```

2.2.3.5 Get the controller IP address

```
/**
* @brief  Get the controller IP
* @param  [out] ip  Controller IP
* @return  Error code
*/
int GetControllerIP(ref string ip);
```

2.2.3.6 Control the robot to enter or exit drag teaching mode

```
/**
* @brief  Control the robot to enter or exit drag teaching mode
* @param  [in] state 0-exit drag teaching mode, 1-enter drag teaching mode
* @return  Error code
*/
int DragTeachSwitch(byte state);
```

2.2.3.7 Check if the robot is in drag mode

```
/**
* @brief  Check if the robot is in drag teach mode
* @param  [out] state 0-not in drag teach mode, 1-in drag teach mode
* @return Error code
*/
int IsInDragTeach(ref byte state);
```

2.2.3.8 Control robot enable or disable

```
/**
* @brief Control robot enable or disable; robot is automatically enabled by default after power-on
* @param  [in] state  0-disable, 1-enable
* @return  Error code
*/
int RobotEnable(byte state);
```

2.2.3.9 Control robot manual/automatic mode switching

```
/**
* @brief Control robot manual/automatic mode switching
* @param [in] mode 0-automatic mode, 1-manual mode
* @return Error code
*/
int Mode(int mode);
```

2.2.3.10 Shut down the robot operating system

```
/**
* @brief Shut down the robot operating system
* @return Error code
*/
int ShutDownRobotOS();
```

2.2.3.11 Code example

```
private void button6_Click(object sender, EventArgse)
{
    int rtn = robot.ShutDownRobotOS();
    Console.WriteLine($"ShutDownRobotOS rtn is {rtn}");
}
```

2.2.3.12 Set robot communication reconnection parameters

```
/**
* @brief Set robot communication reconnection parameters
* @param [in] enable Whether to enable true-enable, false-disable
* @param [in] times Reconnect times
* @param [in] period Reconnect time interval (milliseconds)
*/
void SetReconnectParam(bool enable, int times, int period);
```

2.2.3.13 Code example

```
private void btnStandard_Click(object sender, EventArgs e)
{
    Robot robot = new Robot();
    robot.SetReconnectParam(true, 100, 20000);//Disconnection reconnection parameters
    robot.RPC("192.168.58.2");

    string ip = "";
    string version = "";
    byte state = 0;

    robot.GetSDKVersion(ref version);
    Console.WriteLine($"SDK version : {version}");
    robot.GetControllerIP(ref ip);
    Console.WriteLine($"controller ip : {ip}");

    robot.Mode(1);
    Thread.Sleep(1000);
    robot.DragTeachSwitch(1);
    int rtn = robot.IsInDragTeach(ref state);
    Console.WriteLine($"drag state : {state}");
    Thread.Sleep(3000);
    robot.DragTeachSwitch(0);
    Thread.Sleep(1000);
    robot.IsInDragTeach(ref state);
    Console.WriteLine($"drag state: {state}");
    Thread.Sleep(3000);
    robot.RobotEnable(0);
    Thread.Sleep(3000);
    robot.RobotEnable(1);

    robot.Mode(0);
    Thread.Sleep(1000);
    robot.Mode(1);
}
```

2.2.3.14 Initialize log parameters

```
/**
* @brief Initialize log parameters
* @param [in] logType: Output mode, DIRECT-direct output; BUFFER-buffered output; ASYNC-
↪asynchronous output
* @param [in] logLevel: Log filtering level, ERROR-error; WARNING-warning; INFO-
↪information; DEBUG-debug
* @param [in] filePath: File save path, e.g., "D://Log/"
* @param [in] saveFileNum: Number of files to save; files exceeding both the number of␣
↪files to save and the number of days to save will be deleted
* @param [in] saveDays: Number of days to save; files exceeding both the number of files␣
↪to save and the number of days to save will be deleted
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
int LoggerInit(FrLogType logType = FrLogType.DIRECT, FrLogLevel logLevel = FrLogLevel.
↪INFO, string filePath = "", int saveFileNum = 10, int saveDays = 10);
```

2.2.3.15 Set log filtering level

```
/**
* @brief Set log filtering level
* @param [in] logLevel: Log filtering level, ERROR-error; WARNING-warning; INFO-
↪information; DEBUG-debug
* @return Error code
*/
int SetLoggerLevel(FrLogLevel logLevel);
```

2.2.3.16 Get robot software version

```
/**
* @brief Get robot software version information
* @param [out] robotModel Robot model
* @param [out] webVersion Web version
* @param [out] controllerVersion Controller version
* @return Error code
*/
int GetSoftwareVersion(ref string robotModel, ref string webVersion, ref string␣
↪controllerVersion);
```

2.2.3.17 Get robot hardware version

```
/**
* @brief Get robot hardware version information
* @param [out] ctrlBoxBoardVersion Control box board hardware version
* @param [out] driver1Version Driver 1 hardware version
* @param [out] driver1Version Driver 2 hardware version
* @param [out] driver1Version Driver 3 hardware version
* @param [out] driver1Version Driver 4 hardware version
* @param [out] driver1Version Driver 5 hardware version
* @param [out] driver1Version Driver 6 hardware version
* @param [out] endBoardVersion End board hardware version
* @return Error code
*/
int GetHardwareVersion(ref string ctrlBoxBoardVersion, ref string driver1Version, ref␣
↪string driver2Version, ref string driver3Version, ref string driver4Version, ref␣
↪string driver5Version, ref string driver6Version, ref string endBoardVersion);
```

2.2.3.18 Get robot firmware version

```
/**
* @brief Get robot firmware version information
* @param [out] ctrlBoxBoardVersion Control box board firmware version
* @param [out] driver1Version Driver 1 firmware version
* @param [out] driver1Version Driver 2 firmware version
* @param [out] driver1Version Driver 3 firmware version
* @param [out] driver1Version Driver 4 firmware version
* @param [out] driver1Version Driver 5 firmware version
* @param [out] driver1Version Driver 6 firmware version
* @param [out] endBoardVersion End board firmware version
* @return Error code
*/
int GetFirmwareVersion(ref string ctrlBoxBoardVersion, ref string driver1Version, ref␣
↪string driver2Version, ref string driver3Version, ref string driver4Version, ref␣
↪string driver5Version, ref string driver6Version, ref string endBoardVersion);
```

2.2.3.19 Code example

```
private void btnGetVersions_Click(object sender, EventArgs e)
{
    string[] ver = new string[20];
    int rtn = 0;
    rtn = robot.GetSoftwareVersion(ref ver[0], ref ver[1], ref ver[2]);
    rtn = robot.GetHardwareVersion(ref ver[3], ref ver[4], ref ver[5], ref ver[6], ref␣
↪ver[7], ref ver[8], ref ver[9], ref ver[10]);
    rtn = robot.GetFirmwareVersion(ref ver[11], ref ver[12], ref ver[13], ref ver[14],␣
↪ref ver[15], ref ver[16], ref ver[17], ref ver[18]);
    Console.WriteLine($"robotmodel  is: {ver[0]}");
    Console.WriteLine($"webVersion  is: {ver[1]}");
    Console.WriteLine($"controllerVersion  is: {ver[2]}");
    Console.WriteLine($"Hard ctrlBox Version  is: {ver[3]}");
    Console.WriteLine($"Hard driver1 Version  is: {ver[4]}");
    Console.WriteLine($"Hard driver2 Version is: {ver[5]}");
    Console.WriteLine($"Hard driver3 Version is: {ver[6]}");
    Console.WriteLine($"Hard driver4 Version is: {ver[7]}");
    Console.WriteLine($"Hard driver5 Version is: {ver[8]}");
    Console.WriteLine($"Hard driver6 Version is: {ver[9]}");
    Console.WriteLine($"Hard end Version is: {ver[10]}");
    Console.WriteLine($"Firm ctrlBox Version is: {ver[11]}");
    Console.WriteLine($"Firm driver1 Version is: {ver[12]}");
    Console.WriteLine($"Firm driver2 Version is: {ver[13]}");
    Console.WriteLine($"Firm driver3 Version is: {ver[14]}");
    Console.WriteLine($"Firm driver4 Version is: {ver[15]}");
    Console.WriteLine($"Firm driver5 Version is: {ver[16]}");
    Console.WriteLine($"Firm driver6 Version is: {ver[17]}");
    Console.WriteLine($"Firm end Version is: {ver[18]}");
}
```

2.2.4 Robot Motion

2.2.4.1 jog point motion

```
/**
* @brief jog pointing
* @param [in] refType Type of pointing: 0-joint pointing, 2-pointing in base coordinate system, 4-pointing in tool coordinate system, 8-pointing in artifact coordinate system.
* @param [in] nb 1-joint 1 (or x-axis), 2-joint 2 (or y-axis), 3-joint 3 (or z-axis), 4-joint 4 (or rotate around x-axis), 5-joint 5 (or rotate around y-axis), 6-joint 6 (or rotate around z-axis)
* @param [in] dir 0-negative direction, 1-positive direction
* @param [in] vel velocity percentage, [0~100]
* @param [in] acc acceleration percentage, [0~100]
* @param [in] max_dis Maximum angle of a single tap, in [°] or distance, in [mm]
* @return Error code
*/
int StartJOG(byte refType, byte nb, byte dir, float vel, float acc, float max_dis);
```

2.2.4.2 jog nudging deceleration stop

```
/**
* @brief jog pointwise deceleration stops.
* @param [in] ref 1-joint-point deceleration stop, 3-point deceleration stop in base coordinate system, 5-point deceleration stop in tool coordinate system, 9-point deceleration stop in workpiece coordinate system
* @return Error code
*/
int StopJOG(byte stopType).
```

2.2.4.3 jog pointing stops immediately

```
/**
* @brief jog tapping stops immediately.
* @return Error code
*/
int ImmStopJOG();
```

2.2.4.4 Sample Robot Tap Control Code

```
private void btnJOG_Click(object sender, EventArgs e)
{
    Robot robot = new Robot();
    robot.RPC("192.168.58.2");

    robot.SetSpeed(35);
    robot.StartJOG(0, 1, 0, 15, 20.0f, 30.0f);
    robot.StopJOG(1);
```

(continues on next page)

(continued from previous page)

```
    //robot.ImmStopJOG();
    robot.StartJOG(0, 2, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(0, 3, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(0, 4, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(0, 5, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(0, 6, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();

    robot.StartJOG(2, 1, 0, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.StopJOG(3);
    //robot.ImmStopJOG();
    robot.StartJOG(2, 2, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(2, 3, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(2, 4, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(2, 5, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(2, 6, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();

    robot.StartJOG(4, 1, 0, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.StopJOG(5);
    //robot.ImmStopJOG();
    robot.StartJOG(4, 2, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(4, 3, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(4, 4, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(4, 5, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
    robot.ImmStopJOG();
    robot.StartJOG(4, 6, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();

    robot.StartJOG(8, 1, 0, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.StopJOG(9);
    //robot.ImmStopJOG();
    robot.StartJOG(8, 2, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(8, 3, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(8, 4, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(8, 5, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
    robot.StartJOG(8, 6, 1, 15, 20.0f, 30.0f);
    Thread.Sleep(1000);
    robot.ImmStopJOG();
}
```

2.2.4.5 Joint space motion

```
/**
* @brief Joint space motion.
* @param [in] joint_pos Target joint position in degrees.
* @param [in] desc_pos Target Cartesian position.
* @param [in] tool Tool coordinate number, range [0~14].
* @param [in] user Workpiece coordinate number, range [0~14].
* @param [in] vel velocity percentage, range [0~100] * @param [in] acc.
* @param [in] acc Acceleration percentage, range [0~100], not available yet.
* @param [in] ovl Velocity scaling factor, range [0~100].
* @param [in] epos Extended axis position in mm.
* @param [in] blendT [-1.0]-motion in place (blocking), [0~500.0]-smoothing time (non-
↪blocking) in ms
* @param [in] offset_flag 0-no offset, 1-offset in base/work coordinate system, 2-offset␣
↪in tool coordinate system
* @param [in] offset_pos Bit position offset
* @return Error code
*/
int MoveJ(JointPos joint_pos, DescPose desc_pos, int tool, int user, float vel, float␣
↪acc, float ovl, ExaxisPos epos, float blendT, byte offset_flag, DescPose offset_pos);
```

2.2.4.6 Joint space motion (automatic forward kinematics calculation)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Joint space motion (automatic forward kinematics calculation)
* @param [in] joint_pos  Target joint position, unit deg
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] epos  Extended axis position, unit mm
* @param [in] blendT [-1.0]-move to position (blocking), [0~500.0]-smoothing time (non-blocking), unit ms
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @return Error code
*/
int MoveJ(JointPos joint_pos, int tool, int user, double vel, double acc, double ovl, ExaxisPos epos, double blendT, int offset_flag, DescPose offset_pos)
```

2.2.4.7 Linear motion in Cartesian space

```
/**
* @brief Cartesian space linear motion
* @param [in] joint_pos Target joint position, unit deg
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Speed percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], currently not available
* @param [in] ovl Speed scaling factor [0~100]/Physical speed (mm/s)
* @param [in] blendR [-1.0]-Motion to position (blocking), [0~1000.0]-Blend radius (non-blocking), unit mm
* @param [in] blendMode Transition mode; 0-Inscribed transition; 1-Corner transition
* @param [in] epos Extended axis position, unit mm
* @param [in] search 0-No wire seam tracking, 1-Wire seam tracking
* @param [in] offset_flag 0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param [in] offset_pos Pose offset
* @param [in] oacc Acceleration scaling factor [0-100]/Physical acceleration (mm/s^2)
* @param [in] velAccParamMode Speed/acceleration parameter mode; 0-Percentage; 1-Physical speed (mm/s) and acceleration (mm/s^2)
* @param [in] overSpeedStrategy Overspeed handling strategy, 1-Standard; 2-Error and stop on overspeed; 3-Adaptive speed reduction, default is 0
* @param [in] speedPercent Allowable speed reduction threshold percentage [0-100], default 10%
* @return Error code
*/
public int MoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, float vel,
```

(continues on next page)

(continued from previous page)

```
↪float acc, float ovl, float blendR, int blendMode, ExaxisPos epos, int search, int␣
↪offset_flag, DescPose offset_pos, float oacc, int velAccParamMode, int␣
↪overSpeedStrategy = 0, int speedPercent = 10)
```

2.2.4.8 Cartesian space linear motion (automatic inverse kinematics calculation)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Cartesian space linear motion (automatic inverse kinematics calculation)
* @param [in] desc_pos  Target cartesian pose
* @param [in] tool  Tool coordinate number, range [1~15]
* @param [in] user  Workpiece coordinate number, range [1~15]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius␣
↪(non-blocking), unit mm
* @param [in] blendMode Transition mode; 0-tangent transition; 1-corner transition
* @param [in] epos  Extended axis position, unit mm
* @param [in] search  0-no wire search, 1-wire search
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-
↪offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based␣
↪on current joint position, [0~7]-solve according to specific joint space configuration
* @param [in] overSpeedStrategy  Overspeed handling strategy, 1-standard; 2-stop on␣
↪error when overspeed; 3-adaptive speed reduction, default is 0
* @param [in] speedPercent  Allowed speed reduction threshold percentage [0-100],␣
↪default 10%
* @return Error code
*/
int MoveL(DescPose desc_pos, int tool, int user, double vel, double acc, double ovl,␣
↪double blendR, int blendMode, ExaxisPos epos, int search, int offset_flag, DescPose␣
↪offset_pos, int config, int overSpeedStrategy, int speedPercent)
```

2.2.4.9 Cartesian Space Linear Motion (Added velAccParamMode parameter for velocity and acceleration modes)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Cartesian Space Linear Motion (Added velAccParamMode parameter for velocity␣
↪and acceleration modes)
* @param  [in] joint_pos  Target joint position, unit deg
* @param  [in] desc_pos   Target Cartesian pose
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Velocity percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not yet open
```

(continues on next page)

(continued from previous page)

```
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-blocking), unit mm
* @param  [in] epos  Extended axis position, unit mm
* @param  [in] search  0-No wire search, 1-Wire search
* @param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
* @param  [in] overSpeedStrategy  Overspeed handling strategy, 1-Standard; 2-Stop with error on overspeed; 3-Adaptive deceleration, default is 0
* @param  [in] speedPercent  Allowed deceleration threshold percentage [0-100], default 10%
* @return  Error code
*/
public int MoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel, double acc, double ovl, double blendR, ExaxisPos epos, int search, int offset_flag, DescPose offset_pos, int velAccParamMode, int overSpeedStrategy, int speedPercent)
```

2.2.4.10 Cartesian Space Linear Motion (Overload Function 1, Added blendMode)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Cartesian Space Linear Motion (Overload Function 1, Added blendMode)
* @param  [in] joint_pos  Target joint position, unit deg
* @param  [in] desc_pos  Target Cartesian pose
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Velocity percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not yet open
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-blocking), unit mm
* @param  [in] blendMode Transition mode; 0-Tangent transition; 1-Corner transition
* @param  [in] epos  Extended axis position, unit mm
* @param  [in] search  0-No wire search, 1-Wire search
* @param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
* @param  [in] overSpeedStrategy  Overspeed handling strategy, 1-Standard; 2-Stop with error on overspeed; 3-Adaptive deceleration, default is 0
* @param  [in] speedPercent  Allowed deceleration threshold percentage [0-100], default 10%
* @return  Error code
*/
public int MoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel, double acc, double ovl, double blendR, int blendMode, ExaxisPos epos, int search, int
```

(continues on next page)

(continued from previous page)

```
↪offset_flag, DescPose offset_pos, int velAccParamMode, int overSpeedStrategy, int␣
↪speedPercent)
```

2.2.4.11 Cartesian Space Linear Motion (Overload Function 2, No Joint Position Input Required)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Cartesian Space Linear Motion (Overload Function 2, No Joint Position Input␣
↪Required)
* @param  [in] desc_pos   Target Cartesian pose
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Velocity percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not yet open
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-
↪blocking), unit mm
* @param  [in] blendMode Transition mode; 0-Tangent transition; 1-Corner transition
* @param  [in] epos  Extended axis position, unit mm
* @param  [in] search  0-No wire search, 1-Wire search
* @param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-
↪Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset
* @param  [in] config Inverse kinematic joint space configuration, [-1]-Reference␣
↪current joint position for calculation, [0~7]-Solve based on specific joint space␣
↪configuration
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-
↪Physical velocity (mm/s) and acceleration (mm/s^2)
* @param  [in] overSpeedStrategy  Overspeed handling strategy, 1-Standard; 2-Stop with␣
↪error on overspeed; 3-Adaptive deceleration, default is 0
* @param  [in] speedPercent  Allowed deceleration threshold percentage [0-100], default␣
↪10%
* @return  Error code
*/
public int MoveL(DescPose desc_pos, int tool, int user, double vel, double acc, double␣
↪ovl, double blendR, int blendMode, ExaxisPos epos, int search, int offset_flag,␣
↪DescPose offset_pos, int config, int velAccParamMode, int overSpeedStrategy, int␣
↪speedPercent)
```

2.2.4.12 Circular motion in Cartesian space

```
/**
* @brief Cartesian circular motion.
* @param [in] joint_pos_p Pathpoint joint position in deg.
* @param [in] desc_pos_p Path point Cartesian position.
* @param [in] ptool tool coordinate number, range [0~14].
* @param [in] puser workpiece coordinate number, range [0~14]
* @param [in] pvel speed percentage, range [0~100]
```

(continues on next page)

(continued from previous page)

```
* @param [in] pacc Acceleration percentage, range [0~100], not available yet.
* @param [in] epos_p Extended axis position, in mm
* @param [in] poffset_flag 0-no offset, 1-offset in base/work coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos_p Positional offset
* @param [in] joint_pos_t Joint position of target point, in deg.
* @param [in] desc_pos_t Cartesian position of target point.
* @param [in] ttool tool coordinate number, range [0~14].
* @param [in] tuser Workpiece coordinate number, range [0~14].
* @param [in] tvel Velocity percentage, range [0~100]
* @param [in] tacc Acceleration percentage, range [0~100], not available yet.
* @param [in] epos_t Extended axis position in mm.
* @param [in] toffset_flag 0-no offset, 1-offset in base/work coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos_t Bit position offset
* @param [in] ovl velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-motion in place (blocking), [0~1000.0]-smoothing radius (non-blocking) in mm
* @param  [in] oacc Acceleration scaling factor [0-100]/Physical acceleration (mm/s^2)
* @param  [in] velAccParamMode Speed/acceleration parameter mode; 0-Percentage; 1-Physical speed (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int MoveC(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser, float pvel, float pacc,ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p,JointPos joint_pos_t, DescPose desc_pos_t, int ttool, int tuser, float tvel, float tacc, ExaxisPos epos_t, int toffset_flag, DescPose offset_pos_t,float ovl, float blendR, float oacc, int velAccParamMode)
```

2.2.4.13 Cartesian space circular motion (automatic inverse kinematics calculation)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Cartesian space circular motion (automatic inverse kinematics calculation)
* @param [in] desc_pos_p  Path point cartesian pose
* @param [in] ptool  Tool coordinate number, range [1~15]
* @param [in] puser  Workpiece coordinate number, range [1~15]
* @param [in] pvel  Velocity percentage, range [0~100]
* @param [in] pacc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos_p  Extended axis position, unit mm
* @param [in] poffset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos_p  Pose offset
* @param [in] desc_pos_t  Target point cartesian pose
* @param [in] ttool  Tool coordinate number, range [1~15]
* @param [in] tuser  Workpiece coordinate number, range [1~15]
* @param [in] tvel  Velocity percentage, range [0~100]
* @param [in] tacc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos_t  Extended axis position, unit mm
* @param [in] toffset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-
```

(continues on next page)

(continued from previous page)

```
↪offset in tool coordinate system
* @param [in] offset_pos_t  Pose offset
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius␣
↪(non-blocking), unit mm
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based␣
↪on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
int MoveC(DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc, ExaxisPos␣
↪epos_p, int poffset_flag, DescPose offset_pos_p, DescPose desc_pos_t, int ttool, int␣
↪tuser, double tvel, double tacc, ExaxisPos epos_t, int toffset_flag, DescPose offset_
↪pos_t, double ovl, double blendR, int config)
```

2.2.4.14 Cartesian Space Arc Motion (Added velAccParamMode parameter for velocity and acceleration modes)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Cartesian Space Arc Motion (Added velAccParamMode parameter for velocity and␣
↪acceleration modes)
* @param  [in] joint_pos_p  Path point joint position, unit deg
* @param  [in] desc_pos_p   Path point Cartesian pose
* @param  [in] ptool  Tool coordinate number, range [1~15]
* @param  [in] puser  Workpiece coordinate number, range [1~15]
* @param  [in] pvel  Velocity percentage, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_p  Extended axis position, unit mm
* @param  [in] poffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system,␣
↪2-Offset in tool coordinate system
* @param  [in] offset_pos_p  Pose offset
* @param  [in] joint_pos_t  Target point joint position, unit deg
* @param  [in] desc_pos_t   Target point Cartesian pose
* @param  [in] ttool  Tool coordinate number, range [1~15]
* @param  [in] tuser  Workpiece coordinate number, range [1~15]
* @param  [in] tvel  Velocity percentage, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_t  Extended axis position, unit mm
* @param  [in] toffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system,␣
↪2-Offset in tool coordinate system
* @param  [in] offset_pos_t  Pose offset
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-
↪blocking), unit mm
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-
↪Physical velocity (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int MoveC(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser, double␣
↪pvel, double pacc, ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p, JointPos␣
```

(continues on next page)

(continued from previous page)

```
↪joint_pos_t, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc,␣
↪ExaxisPos epos_t, int toffset_flag, DescPose offset_pos_t, double ovl, double blendR,␣
↪int velAccParamMode)
```

2.2.4.15 Cartesian Space Arc Motion (Overload Function 1, No Joint Position Input Required)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Cartesian Space Arc Motion (Overload Function 1, No Joint Position Input␣
↪Required)
* @param  [in] desc_pos_p   Path point Cartesian pose
* @param  [in] ptool  Tool coordinate number, range [1~15]
* @param  [in] puser  Workpiece coordinate number, range [1~15]
* @param  [in] pvel  Velocity percentage, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_p  Extended axis position, unit mm
* @param  [in] poffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system,␣
↪2-Offset in tool coordinate system
* @param  [in] offset_pos_p  Pose offset
* @param  [in] desc_pos_t   Target point Cartesian pose
* @param  [in] ttool  Tool coordinate number, range [1~15]
* @param  [in] tuser  Workpiece coordinate number, range [1~15]
* @param  [in] tvel  Velocity percentage, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_t  Extended axis position, unit mm
* @param  [in] toffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system,␣
↪2-Offset in tool coordinate system
* @param  [in] offset_pos_t  Pose offset
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-
↪blocking), unit mm
* @param  [in] config Inverse kinematic joint space configuration, [-1]-Reference␣
↪current joint position for calculation, [0~7]-Solve based on specific joint space␣
↪configuration
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-
↪Physical velocity (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int MoveC(DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc,␣
↪ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p, DescPose desc_pos_t, int␣
↪ttool, int tuser, double tvel, double tacc, ExaxisPos epos_t, int toffset_flag,␣
↪DescPose offset_pos_t, double ovl, double blendR, int config, int velAccParamMode)
```

2.2.4.16 Point-to-point motion in Cartesian space

```
/**
* @brief Cartesian space point-to-point motion.
* @param [in] desc_pos Cartesian position of target in base coordinate system.
* @param [in] tool tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14].
* @param [in] vel velocity percentage, range [0~100] * @param [in] acc [in] tool␣
↪coordinate number, range [0~14]
* @param [in] acc Acceleration percentage, range [0~100], not available yet.
* @param [in] ovl velocity scaling factor, range [0~100]
* @param [in] blendT [-1.0]-motion in place (blocking), [0~500.0]-smoothing time (non-
↪blocking) in ms
* @param [in] config Joint space configuration, [-1]-Refer to current joint position, [0~
↪7]-Refer to specific joint space configuration, default is -1.
* @return Error code
*/
int MoveCart(DescPose desc_pos, int tool, int user, float vel, float acc, float ovl,␣
↪float blendT, int config);
```

2.2.4.17 Whole circle motion in Cartesian space

New in version C#SDK-V1.1.4: Web-3.8.3

```
/**
* @brief Whole circle motion in Cartesian space.
* @param [in] joint_pos_p path point 1 joint position in deg.
* @param [in] desc_pos_p path point 1 Cartesian position.
* @param [in] ptool tool coordinate number, range [0~14].
* @param [in] puser workpiece coordinate number, range [0~14]
* @param [in] pvel velocity percentage, range [0~100]
* @param [in] pacc Acceleration percentage, range [0~100], not available yet.
* @param [in] epos_p extended axis position in mm
* @param [in] joint_pos_t Path point 2 joint position in deg.
* @param [in] desc_pos_t Path point 2 Cartesian position in deg.
* @param [in] ttool tool coordinate number, range [0~14].
* @param [in] tuser Workpiece coordinate number, range [0~14]
* @param [in] tvel Velocity percentage, range [0~100]
* @param [in] tacc Acceleration percentage, range [0~100], not available yet.
* @param [in] epos_t Extended axis position in mm.
* @param [in] ovl  Speed scaling factor [0~100]/Physical speed (mm/s)
* @param [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-
↪Offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @param [in] oacc Acceleration scaling factor [0-100]/Physical acceleration (mm/s^2)
* @param [in] blendR -1: Blocking; 0~1000: Blend radius
* @param [in] velAccParamMode Speed/acceleration parameter mode; 0-Percentage; 1-
↪Physical speed (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int Circle(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser, float␣
↪pvel, float pacc,ExaxisPos epos_p, JointPos joint_pos_t, DescPose desc_pos_t, int␣
```

(continues on next page)

(continued from previous page)

```
↪ttool, int tuser,float tvel, float tacc, ExaxisPos epos_t, float ovl, int offset_flag,
↪DescPose offset_pos, double oacc, double blendR, int velAccParamMode)
```

2.2.4.18 Cartesian space full circle motion (automatic inverse kinematics calculation)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Cartesian space full circle motion (automatic inverse kinematics calculation)
* @param [in] desc_pos_p  Path point 1 cartesian pose
* @param [in] ptool  Tool coordinate number, range [0~14]
* @param [in] puser  Workpiece coordinate number, range [0~14]
* @param [in] pvel  Velocity percentage, range [0~100]
* @param [in] pacc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos_p  Extended axis position, unit mm
* @param [in] desc_pos_t  Path point 2 cartesian pose
* @param [in] ttool  Tool coordinate number, range [0~14]
* @param [in] tuser  Workpiece coordinate number, range [0~14]
* @param [in] tvel  Velocity percentage, range [0~100]
* @param [in] tacc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos_t  Extended axis position, unit mm
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-
↪offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @param [in] oacc Acceleration percentage
* @param [in] blendR -1: blocking; 0~1000: smoothing radius
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based␣
↪on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
int Circle(DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc,␣
↪ExaxisPos epos_p, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc,␣
↪ExaxisPos epos_t, double ovl, int offset_flag, DescPose offset_pos, double oacc,␣
↪double blendR,int config)
```

2.2.4.19 Cartesian Space Full Circle Motion (Added velAccParamMode parameter for velocity and acceleration modes)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
*@brief  Cartesian Space Full Circle Motion (Added velAccParamMode parameter for␣
↪velocity and acceleration modes)
*@param  [in] joint_pos_p  Path point 1 joint position, unit deg
*@param  [in] desc_pos_p  Path point 1 Cartesian pose
*@param  [in] ptool  Tool coordinate number, range [1~15]
*@param  [in] puser  Workpiece coordinate number, range [1~15]
*@param  [in] pvel  Velocity percentage, range [0~100]
*@param  [in] pacc  Acceleration percentage, range [0~100], not yet open
```

(continues on next page)

(continued from previous page)

```
*@param  [in] epos_p  Extended axis position, unit mm
*@param  [in] joint_pos_t  Path point 2 joint position, unit deg
*@param  [in] desc_pos_t   Path point 2 Cartesian pose
*@param  [in] ttool  Tool coordinate number, range [1~15]
*@param  [in] tuser  Workpiece coordinate number, range [1~15]
*@param  [in] tvel  Velocity percentage, range [0~100]
*@param  [in] tacc  Acceleration percentage, range [0~100], not yet open
*@param  [in] epos_t  Extended axis position, unit mm
*@param  [in] ovl  Velocity scaling factor, range [0~100]
*@param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
*@param  [in] offset_pos  Pose offset
*@param  [in] oacc Acceleration percentage
*@param  [in] blendR -1: Blocking; 0~1000: Blending radius
*@param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
*@return  Error code
*/
public int Circle(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser,double pvel, double pacc, ExaxisPos epos_p, JointPos joint_pos_t, DescPose desc_pos_t,int ttool, int tuser, double tvel, double tacc, ExaxisPos epos_t, double ovl, intoffset_flag, DescPose offset_pos, double oacc, double blendR, int velAccParamMode)
```

2.2.4.20 Cartesian Space Full Circle Motion (Overload Function 1, No Joint Position Input Required)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Cartesian Space Full Circle Motion (Overload Function 1, No Joint PositionInput Required)
* @param  [in] desc_pos_p   Path point 1 Cartesian pose
* @param  [in] ptool  Tool coordinate number, range [0~14]
* @param  [in] puser  Workpiece coordinate number, range [0~14]
* @param  [in] pvel  Velocity percentage, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_p  Extended axis position, unit mm
* @param  [in] desc_pos_t   Path point 2 Cartesian pose
* @param  [in] ttool  Tool coordinate number, range [0~14]
* @param  [in] tuser  Workpiece coordinate number, range [0~14]
* @param  [in] tvel  Velocity percentage, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_t  Extended axis position, unit mm
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset
* @param  [in] oacc Acceleration percentage
* @param  [in] blendR -1: Blocking; 0~1000: Blending radius
* @param  [in] config Inverse kinematic joint space configuration, [-1]-Reference current joint position for calculation, [0~7]-Solve based on specific joint space configuration
```

(continues on next page)

(continued from previous page)

```
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int Circle(DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc, ExaxisPos epos_p, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc, ExaxisPos epos_t, double ovl, int offset_flag, DescPose offset_pos, double oacc, double blendR, int config, int velAccParamMode)
```

2.2.4.21 Sample Code for Whole Circle Motion in Cartesian Space

New in version C#SDK-V1.1.4: Web-3.8.3

```
private void btnMovetest_Click(object sender, EventArgs e)
{
    int rtn = 0;
    DescPose middescPoseCir1 = new DescPose(-435.414, -342.926, 309.205, -171.382, -4.513, 171.520);
    JointPos midjointPosCir1 = new JointPos(26.804, -79.866, 106.642, -125.433, -85.562, -54.721);
    DescPose enddescPoseCir1 = new DescPose(-524.862, -217.402, 308.459, -171.425, -4.810, 156.088);
    JointPos endjointPosCir1 = new JointPos(11.399, -78.055, 104.603, -125.421, -85.770, -54.721);

    DescPose middescPoseCir2 = new DescPose(-482.691, -587.899, 318.594, -171.001, -4.999, -172.996);
    JointPos midjointPosCir2 = new JointPos(42.314, -53.600, 67.296, -112.969, -85.533, -54.721);
    DescPose enddescPoseCir2 = new DescPose(-403.942, -489.061, 317.038, -163.189, -10.425, -175.627);
    JointPos endjointPosCir2 = new JointPos(39.959, -70.616, 96.679, -134.243, -82.276, -54.721);

    DescPose middescPoseMoveC = new DescPose(-435.414, -342.926, 309.205, -171.382, -4.513, 171.520);
    JointPos midjointPosMoveC = new JointPos(26.804, -79.866, 106.642, -125.433, -85.562, -54.721);
    DescPose enddescPoseMoveC = new DescPose(-524.862, -217.402, 308.459, -171.425, -4.810, 156.088);
    JointPos endjointPosmoveC = new JointPos(11.399, -78.055, 104.603, -125.421, -85.770, -54.721);

    DescPose middescPoseCir3 = new DescPose(-435.414, -342.926, 309.205, -171.382, -4.513, 171.520);
    JointPos midjointPosCir3 = new JointPos(26.804, -79.866, 106.642, -125.433, -85.562, -54.721);
    DescPose enddescPoseCir3 = new DescPose(-569.505, -405.378, 357.596, -172.862, -10.939, 171.108);
    JointPos endjointPosCir3 = new JointPos(27.138, -63.750, 78.586, -117.861, -90.588, -54.721);
```

(continues on next page)

(continued from previous page)

```

    DescPose middescPoseCir4 = new DescPose(-482.691, -587.899, 318.594, -171.001, -4.
↪999, -172.996);
    JointPos midjointPosCir4 = new JointPos(42.314, -53.600, 67.296, -112.969, -85.533, -
↪54.721);
    DescPose enddescPoseCir4 = new DescPose(-569.505, -405.378, 357.596, -172.862, -10.
↪939, 171.108);
    JointPos endjointPosCir4 = new JointPos(27.138, -63.750, 78.586, -117.861, -90.588, -
↪54.721);

    DescPose startdescPose = new DescPose(-569.505, -405.378, 357.596, -172.862, -10.939,
↪ 171.108);
    JointPos startjointPos = new JointPos(27.138, -63.750, 78.586, -117.861, -90.588, -
↪54.721);

    DescPose linedescPose = new DescPose(-403.942, -489.061, 317.038, -163.189, -10.425,␣
↪-175.627);
    JointPos linejointPos = new JointPos(39.959, -70.616, 96.679, -134.243, -82.276, -54.
↪721);


    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);


    robot.MoveJ(startjointPos, startdescPose, 3, 0, 100, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    rtn = robot.Circle(midjointPosCir1, middescPoseCir1, 3, 0, 100, 100, exaxisPos,␣
↪endjointPosCir1, enddescPoseCir1, 3, 0, 100, 100, exaxisPos, 100, -1, offdese, 100,␣
↪20);
    Console.WriteLine("Circle1" + rtn);
    rtn = robot.Circle(midjointPosCir2, middescPoseCir2, 3, 0, 100, 100, exaxisPos,␣
↪endjointPosCir2, enddescPoseCir2, 3, 0, 100, 100, exaxisPos, 100, -1, offdese, 100,␣
↪20);
    Console.WriteLine("Circle2" + rtn);

    robot.MoveC(midjointPosMoveC, middescPoseMoveC, 3, 0, 100, 100, exaxisPos, 0,␣
↪offdese, endjointPosmoveC, enddescPoseMoveC, 3, 0, 100, 100, exaxisPos, 0, offdese,␣
↪100, 20);
    rtn = robot.Circle(midjointPosCir3, middescPoseCir3, 3, 0, 100, 100, exaxisPos,␣
↪endjointPosCir3, enddescPoseCir3, 3, 0, 100, 100, exaxisPos, 100, -1, offdese, 100,␣
↪20);
    Console.WriteLine("Circle3" + rtn);
    rtn = robot.MoveL(linejointPos, linedescPose, 3, 0, 100, 100, 100, -1, 0, exaxisPos,␣
↪0, 0, offdese);
    Console.WriteLine("MoveL " + rtn);
    rtn = robot.Circle(midjointPosCir4, middescPoseCir4, 3, 0, 100, 100, exaxisPos,␣
↪endjointPosCir4, enddescPoseCir4, 3, 0, 100, 100, exaxisPos, 100, -1, offdese, 100,␣
↪20);
    Console.WriteLine("Circle4" + rtn);
}
```

2.2.4.22 Sample code for basic robot motion instructions

New in version C#SDK-V1.1.7: Web-3.8.5

```
public void TestMove()
    int rtn;
    JointPos j1 = new JointPos(-11.904f, -99.669f, 117.473f, -108.616f, -91.726f, 74.256f);
    JointPos j2 = new JointPos(-45.615f, -106.172f, 124.296f, -107.151f, -91.282f, 74.255f);
    JointPos j3 = new JointPos(-29.777f, -84.536f, 109.275f, -114.075f, -86.655f, 74.257f);
    JointPos j4 = new JointPos(-31.154f, -95.317f, 94.276f, -88.079f, -89.740f, 74.256f);
    DescPose desc_pos1 = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f,3.833f);
    DescPose desc_pos2 = new DescPose(-321.222f, 185.189f, 335.520f, -179.030f, -1.284f,-29.869f);
    DescPose desc_pos3 = new DescPose(-487.434f, 154.362f, 308.576f, 176.600f, 0.268f, -14.061f);
    DescPose desc_pos4 = new DescPose(-443.165f, 147.881f, 480.951f, 179.511f, -0.775f, -15.409f);
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 100.0f;
    float ovl = 100.0f;
    float oacc = 100.0f;
    float blendT = 0.0f;
    float blendR = 0.0f;
    byte flag = 0;
    byte search = 0;
    int blendMode = 0;
    int velAccMode = 0;
    robot.SetSpeed(20);
    rtn = robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag,offset_pos);
    Console.WriteLine($"movej errcode:{rtn}");
    rtn = robot.MoveL(j2, desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, epos,search, flag, offset_pos, oacc, velAccMode,0,10);
    Console.WriteLine($"movel errcode:{rtn}");
    rtn = robot.MoveC(j3, desc_pos3, tool, user, vel, acc, epos, flag, offset_pos,j4,desc_pos4, tool, user, vel, acc, epos, flag, offset_pos, ovl, blendR, oacc,velAccMode);
    Console.WriteLine($"movec errcode:{rtn}");
    rtn = robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag,offset_pos);
    Console.WriteLine($"movej errcode:{rtn}");
    rtn = robot.Circle(j3, desc_pos3, tool, user, vel, acc, epos,j1, desc_pos1, tool,user, vel, acc, epos,ovl, flag, offset_pos, oacc, -1, velAccMode);
    Console.WriteLine($"circle errcode:{rtn}");
    rtn = robot.MoveCart(desc_pos4, tool, user, vel, acc, ovl, blendT, -1);
```

(continues on next page)

(continued from previous page)

```
    Console.WriteLine($"MoveCart errcode:{rtn}");
    rtn = robot.MoveJ(j1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos);
    Console.WriteLine($"movej errcode:{rtn}");
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, epos,␣
↪search, flag, offset_pos, -1, velAccMode);
    Console.WriteLine($"movel errcode:{rtn}");
    rtn = robot.MoveC(desc_pos3, tool, user, vel, acc, epos, flag, offset_pos,desc_pos4,␣
↪tool, user, vel, acc, epos, flag, offset_pos,ovl, blendR, -1, velAccMode);
    Console.WriteLine($"movec errcode:{rtn}");
    rtn = robot.MoveJ(j2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos);
    Console.WriteLine($"movej errcode:{rtn}");
    rtn = robot.Circle(desc_pos3, tool, user, vel, acc, epos, desc_pos1, tool, user, vel,
↪ acc, epos,ovl, flag, offset_pos, oacc, blendR, -1, velAccMode);
    Console.WriteLine($"circle errcode:{rtn}");
}
```

2.2.4.23 Spiral motion in Cartesian space

```
/**
* @brief Cartesian spiral motion.
* @param [in] joint_pos target joint position in deg
* @param [in] desc_pos target cartesian position in deg
* @param [in] tool Tool coordinate number, range [0~14].
* @param [in] user Workpiece coordinate number, range [0~14].
* @param [in] vel velocity percentage, range [0~100] * @param [in] acc [in] tool␣
↪coordinate number, range [0~14]
* @param [in] acc Acceleration percentage, range [0~100], not available yet.
* @param [in] epos extended axis position in mm
* @param [in] ovl Velocity scaling factor, range [0~100].
* @param [in] offset_flag 0-no offset, 1- offset in base/work coordinate system, 2-␣
↪offset in tool coordinate system
* @param [in] offset_pos Bit position offset
* @param [in] spiral_param spiral_param
* @return Error code
*/
int NewSpiral(JointPos joint_pos, DescPose desc_pos, int tool, int user, float vel,␣
↪float acc, ExaxisPos epos, float ovl, byte offset_flag, DescPose offset_pos,␣
↪SpiralParam spiral_param);
```

2.2.4.24 Cartesian space spiral motion (automatic inverse kinematics calculation)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Cartesian space spiral motion (automatic inverse kinematics calculation)
* @param [in] desc_pos  Target cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
```

(continues on next page)

(continued from previous page)

```
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos  Extended axis position, unit mm
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @param [in] spiral_param  Spiral parameters
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
int NewSpiral(DescPose desc_pos, int tool, int user, double vel, double acc, ExaxisPos epos, double ovl, int offset_flag, DescPose offset_pos, SpiralParam spiral_param,int config)
```

2.2.4.25 Sample code for spiral motion

```
private void btnDescSpiral_Click(object sender, EventArgs e)
{
    int rtn;
    JointPos j = new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    DescPose desc_pos = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose offset_pos1 = new DescPose(50, 0, 0, -30, 0, 0);
    DescPose offset_pos2 = new DescPose(50, 0, 0, -5, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);
    SpiralParam sp;
    sp.circle_num = 5;
    sp.circle_angle = 5.0f;
    sp.rad_init = 50.0f;
    sp.rad_add = 10.0f;
    sp.rotaxis_add = 10.0f;
    sp.rot_direction = 0;

    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 100.0f;
    float ovl = 100.0f;
    float blendT = 0.0f;
    byte flag = 2;

    robot.SetSpeed(20);

    rtn = robot.MoveJ(j, desc_pos, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos1);
    Console.WriteLine($"MoveJ errcode:{rtn}");

    rtn = robot.NewSpiral(j, desc_pos, tool, user, vel, acc, epos, ovl, flag, offset_pos2, sp);
    Console.WriteLine($"NewSpiral errcode:{rtn}");
}
```

2.2.4.26 Starting servo motion

```
/**
* @brief Servo motion start, used with ServoJ, ServoCart instructions.
* @return Error code.
*/
int ServoMoveStart();
```

2.2.4.27 ServoMoveStart

```
/**
* @brief End of servo motion, used with ServoJ, ServoCart instructions.
* @return Error code.
*/
int ServoMoveEnd().
```

2.2.4.28 Joint space servo mode movement

New in version C#SDK-V1.1.4: Web-3.8.3

```
/**
* @brief Joint space servo mode motion
* @param [in] joint_pos Target joint position in degrees.
* @param [in] acc Acceleration percentage, range [0~100], not open, default is 0.
* @param [in] vel Speed percentage, range[0~100], not open, default is 0.
* @param [in] cmdT Command send cycle, unit s, recommended range [0.001~0.0016].
* @param [in] filterT filter time, unit s, not open, default is 0
* @param [in] gain proportional amplifier for target position, not open, default is 0
* @param [in] id servoJ command ID, default is 0
* @return Error code
*/
int ServoJ(JointPos joint_pos, float acc, float vel, float cmdT, float filterT, float␣
↪gain,int id=0);
```

2.2.4.29 Joint space servo mode motion code example

New in version C#SDK-V1.1.4: Web-3.8.3

```
private void btnJointServoMove_Click(object sender, EventArgs e)
{
    JointPos j = new JointPos(0, 0, 0, 0, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);

    float vel = 0.0f;
    float acc = 0.0f;
    float cmdT = 0.008f;
    float filterT = 0.0f;
    float gain = 0.0f;
    byte flag = 0;
```

(continues on next page)

(continued from previous page)

```
    int count = 500;
    float dt = 0.1f;
    int cmdID = 0;
    int ret = robot.GetActualJointPosDegree(flag, ref j);
    if (ret == 0)
    {
        robot.ServoMoveStart();

        try
        {
            while (count > 0)
            {

                robot.ServoJ(j, epos, acc, vel, cmdT, filterT, gain, cmdID);


                j.jPos[0] += dt;
                count--;


                robot.WaitMs((int)(cmdT * 1000));
            }
        }
        finally
        {

            robot.ServoMoveEnd();
        }
    }
    else
    {
        Console.WriteLine($"GetActualJointPosDegree error code: {ret}");

    }
}
```

2.2.4.30 Start of joint torque control

```
/**
* @brief Joint torque control start
* @return Error code
*/
int ServoJTStart().
```

2.2.4.31 Joint torque control

```
/**
* @brief Joint Torque Control
* @param [in] torque Torque for joints j1~j6, unit: Nm
* @param [in] interval Command cycle, unit: s, range: [0.001~0.008]
* @param [in] checkFlag Detection strategy
*                       0-No restrictions;
*                       1-Power limit;
*                       2-Velocity limit;
*                       3-Both power and velocity limits
* @param [in] jPowerLimit Maximum joint power limit (W)
* @param [in] jVelLimit Maximum joint velocity (°/s)
* @return Error code
*/
public int ServoJT(double[] torque, double interval, int checkFlag, double[] jPowerLimit,
↪ double[] jVelLimit)
```

2.2.4.32 End of joint torque control

```
/**
* @brief End of joint torque control.
* @return Error code
*/
int ServoJTEnd();
```

2.2.4.33 Joint Torque Control Code Example

```
private void button27_Click(object sender, EventArgs e)
{
    robot.DragTeachSwitch(1);
    robot.SetPowerLimit(1, 200);
    double[] torques = { 0, 0, 0, 0, 0, 0 };
    robot.GetJointTorques(1, torques);

    int count = 100;
    robot.ServoJTStart();
    int error = 0;
    while (count > 0)
    {
        error = robot.ServoJT(torques, 0.001f);
        count--;
        Thread.Sleep(1);
    }
    error = robot.ServoJTEnd();
    robot.DragTeachSwitch(0);
}
```

2.2.4.34 Joint Torque Control Code Example with Overspeed Protection

```
public int ServoJTWithSafety()
{
    robot.ResetAllError();
    Thread.Sleep(500);

    double[] torques = new double[6] { 0, 0, 0, 0, 0, 0 };
    robot.GetJointTorques(1, torques);

    robot.ServoJTStart();
    ROBOT_STATE_PKG pkg = new ROBOT_STATE_PKG();
    robot.DragTeachSwitch(1);

    int checkFlag = 0;
    double[] jPowerLimit = new double[6] { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    //double[] jPowerLimit = new double[6] { 10.0, 10.0, 10.0, 10.0, 10.0, 10.0 };
    // double[] jVelLimit = new double[6] { 10.0, 10.0, 10.0, 10.0, 10.0, 10.0 };
    double[] jVelLimit = new double[6] {50, 50, 50, 50, 50, 50 };
    int count = 80000;
    int errorNum = 0;
    int error = 0;
    while (count > 0)
    {

        torques[2] = torques[2] + 0.01;
        error = robot.ServoJT(torques, 0.008, checkFlag, jPowerLimit, jVelLimit);

        Console.WriteLine($"ServoJT rtn is {error}");
        count = count - 1;
        Thread.Sleep(1);

        robot.GetRobotRealTimeState(ref pkg);
        Console.WriteLine($"maincode {pkg.main_code}, subcode {pkg.sub_code}");
        if (error != 0)
        {
            errorNum++;
            if (errorNum > 5)
            {
                break;
            }

        }
    }
    robot.DragTeachSwitch(0);
    error = robot.ServoJTEnd();

    return 0;
}
```

2.2.4.35 Servo mode motion in Cartesian space

```

```

2.2.4.36 Cartesian Space Servo Mode Motion

```
/**
* @brief Cartesian space servo mode motion
* @param [in] mode 0-Absolute motion (base coordinate system), 1-Incremental motion (base coordinate system), 2-Incremental motion (tool coordinate system)
* @param [in] desc_pos Target Cartesian pose or pose increment
* @param [in] exaxis Extended axis position
* @param [in] pos_gain Pose increment proportionality coefficient, only effective in incremental motion, range [0~1]
* @param [in] acc Acceleration percentage, range [0~100], temporarily not available, default 0
* @param [in] vel Velocity percentage, range [0~100], temporarily not available, default 0
* @param [in] cmdT Command transmission period, unit s, recommended range [0.001~0.016]
* @param [in] filterT Filter time, unit s, temporarily not available, default 0
* @param [in] gain Proportional amplifier for target position, temporarily not available, default 0
* @return Error code
*/
public int ServoCart(int mode, DescPose desc_pose, ExaxisPos exaxis, double[] pos_gain, double acc, double vel, double cmdT, double filterT, double gain);
```

2.2.4.37 Example of servo mode motion code in Cartesian space

```
public void TestServoCart()
{
    ROBOT_STATE_PKG pkg = new ROBOT_STATE_PKG();

    int rtn;
    DescPose desc_pos_dt = new DescPose(83.00800f, 50.525000f, 29.246f, 179.629f, -7.138f, -166.975f);
    ExaxisPos exaxis = new ExaxisPos(100.0f, 0.0f, 0.0f, 0.0f);
    double[] pos_gain = { 0.0f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f };
    int mode = 0;
    float vel = 0.0f;
    float acc = 0.0f;
    float cmdT = 0.001f;
    float filterT = 0.0f;
    float gain = 0.0f;
    byte flag = 0;
    int count = 5000;

    robot.SetSpeed(20);

    while (count > 0)
```

(continues on next page)

(continued from previous page)

```
    {
        rtn = robot.ServoCart(mode, desc_pos_dt, exaxis, pos_gain, acc, vel, cmdT,␣
↪filterT, gain);
        Console.WriteLine($"ServoCart rtn is {rtn}");
        count -= 1;
        desc_pos_dt.tran.x += 0.01f;
        exaxis.ePos[0] += 0.01f;
    }
}
```

2.2.4.38 Spline motion starts

```
/**
* @brief Spline start
* @return Error code.
*/
int SplineStart();
```

2.2.4.39 Spline motion PTP

```
/**
* @brief Joint space spline motion.
* @param [in] joint_pos Target joint position in degrees.
* @param [in] desc_pos Target Cartesian position.
* @param [in] tool Tool coordinate number, range [0~14].
* @param [in] user Workpiece coordinate number, range [0~14].
* @param [in] vel velocity percentage, range [0~100] * @param [in] acc.
* @param [in] acc Acceleration percentage, range [0~100], not available yet.
* @param [in] ovl velocity scaling factor, range [0~100].
* @return Error code
*/
int SplinePTP(JointPos joint_pos, DescPose desc_pos, int tool, int user, float vel,␣
↪float acc, float ovl);
```

2.2.4.40 Joint space spline motion (automatic forward kinematics calculation)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Joint space spline motion (automatic forward kinematics calculation)
* @param [in] joint_pos  Target joint position, unit deg
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
int SplinePTP(JointPos joint_pos, int tool, int user, double vel, double acc, double ovl)
```

2.2.4.41 End of spline motion

```
/**
* @brief End of spline motion.
* @return Error code.
*/
int SplineEnd();
```

2.2.4.42 Sample code for spline motion

```
private void btnSplineMove_Click(object sender, EventArgs e)
{
    JointPos j1 = new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2 = new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3 = new JointPos(-61.954, -84.409, 108.153, -116.316, -91.283, 74.260);
    JointPos j4 = new JointPos(-89.575, -80.276, 102.713, -116.302, -91.284, 74.267);
    DescPose desc_pos1 = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.
↪833);
    DescPose desc_pos2 = new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);
    DescPose desc_pos3 = new DescPose(-327.622, 402.230, 320.402, -178.067, 2.127, -46.
↪207);
    DescPose desc_pos4 = new DescPose(-104.066, 544.321, 327.023, -177.715, 3.371, -73.
↪818);
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 100.0f;
    float ovl = 100.0f;
    float blendT = -1.0f;
    byte flag = 0;

    robot.SetSpeed(20);

    int err = -1;
    err = robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag,␣
↪offset_pos);
    Console.WriteLine($"movej errcode:  {err}");

    robot.SplineStart();
    robot.SplinePTP(j1, desc_pos1, tool, user, vel, acc, ovl);
    robot.SplinePTP(j2, desc_pos2, tool, user, vel, acc, ovl);
    robot.SplinePTP(j3, desc_pos3, tool, user, vel, acc, ovl);
```

(continues on next page)

(continued from previous page)

```
    robot.SplinePTP(j4, desc_pos4, tool, user, vel, acc, ovl);
    robot.SplineEnd();
}
```

2.2.4.43 Starting new spline motion

Changed in version C#SDK-v1.0.6.

```
/**
* @brief New spline motion start
* @param [in] type 0-circular transition, 1-given point as path point
* @param [in] averageTime global average articulation time (ms) (10 ~ ), default 2000
* @return ErrorCode
*/
int NewSplineStart(int type, int averageTime=2000);
```

2.2.4.44 New spline instruction point

```
/**
* @brief Add spline command point.
* @param [in] joint_pos Target joint position, in deg.
* @param [in] desc_pos Target Cartesian position.
* @param [in] tool Tool coordinate number, range [0~14].
* @param [in] user Tool coordinate number, range [0~14].
* @param [in] vel Speed percentage, range [0~100].
* @param [in] acc Acceleration Percentage, range [0~100], not available yet.
* @param [in] ovl velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-motion in place (blocking), [0~1000.0]-smoothing radius (non-
↪blocking), unit mm
* @param [in] lastFlag whether it is the last point, 0-no, 1-yes
* @return error code
*/
int NewSplinePoint(JointPos joint_pos, DescPose desc_pos, int tool, int user, float vel,␣
↪float acc, float ovl, float blendR, int lastFlag);;
```

2.2.4.45 New spline command point (automatic inverse kinematics calculation)

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief New spline command point (automatic inverse kinematics calculation)
* @param [in] desc_pos  Target cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius␣
```

(continues on next page)

(continued from previous page)

```
↪(non-blocking), unit mm
* @param [in] lastFlag Whether it is the last point, 0-no, 1-yes
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based␣
↪on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
int NewSplinePoint(DescPose desc_pos, int tool, int user, double vel, double acc, double␣
↪ovl, double blendR, int lastFlag,int config)
```

2.2.4.46 End of new spline motion

```
/**
* @brief Start of new spline motion
* @return Error code.
*/
int NewSplineEnd();
```

2.2.4.47 Sample code for new spline motion

```
private void btnNewSpline_Click(object sender, EventArgs e)
{
    JointPos j1 = new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2 = new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3 = new JointPos(-61.954, -84.409, 108.153, -116.316, -91.283, 74.260);
    JointPos j4 = new JointPos(-89.575, -80.276, 102.713, -116.302, -91.284, 74.267);
    JointPos j5 = new JointPos(-95.228, -54.621, 73.691, -112.245, -91.280, 74.268);
    DescPose desc_pos1 = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.
↪833);
    DescPose desc_pos2 = new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);
    DescPose desc_pos3 = new DescPose(-327.622, 402.230, 320.402, -178.067, 2.127, -46.
↪207);
    DescPose desc_pos4 = new DescPose(-104.066, 544.321, 327.023, -177.715, 3.371, -73.
↪818);
    DescPose desc_pos5 = new DescPose(-33.421, 732.572, 275.103, -177.907, 2.709, -79.
↪482);
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 100.0f;
    float ovl = 100.0f;
    float blendT = -1.0f;
    byte flag = 0;

    robot.SetSpeed(20);

```

(continues on next page)

(continued from previous page)

```
    int err = -1;
    err = robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag,␣
↪offset_pos);
    Console.WriteLine($"movej errcode:  {err}");

    robot.NewSplineStart(1, 2000);
    robot.NewSplinePoint(j1, desc_pos1, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(j2, desc_pos2, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(j3, desc_pos3, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(j4, desc_pos4, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplinePoint(j5, desc_pos5, tool, user, vel, acc, ovl, -1, 0);
    robot.NewSplineEnd();
}
```

2.2.4.48 Terminate the motion

```
/**
* @brief Terminate the campaign.
* @return Error code
*/
int StopMotion();
```

2.2.4.49 Pause the motion.

```
/**
* @brief Pause motion
* @return Error code
*/
int PauseMotion();
```

2.2.4.50 Resumes motion.

```
/**
* @brief Resume movement
* @return Error code
*/
int ResumeMotion();
```

2.2.4.51 Motion Pause, Resume, Stop Code Example

```
private void btnMotionPause_Click(object sender, EventArgs e)
{
    int rtn;
    JointPos j1 = new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j5 = new JointPos(-95.228, -54.621, 73.691, -112.245, -91.280, 74.268);
    DescPose desc_pos1 = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.
↪833);
    DescPose desc_pos5 = new DescPose(-33.421, 732.572, 275.103, -177.907, 2.709, -79.
↪482);
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 100.0f;
    float ovl = 100.0f;
    float blendT = -1.0f;
    byte flag = 0;

    robot.SetSpeed(20);

    rtn = robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag,␣
↪offset_pos);
    rtn = robot.MoveJ(j5, desc_pos5, tool, user, vel, acc, ovl, epos, 1, flag, offset_
↪pos);
    Thread.Sleep(1000);
    robot.PauseMotion();

    Thread.Sleep(1000);
    robot.ResumeMotion();

    Thread.Sleep(1000);
    robot.StopMotion();

    Thread.Sleep(1000);

}
```

2.2.4.52 Start the overall point offset

```
/**
* @brief Start of overall point offset.
* @param [in] flag 0 - offset in base/workpiece coordinate system, 2 - offset in tool␣
↪coordinate system
* @param [in] offset_pos Positional Offset
* @return Error code
*/
int PointsOffsetEnable(int flag, DescPose offset_pos);
```

2.2.4.53 Points overall offset end

```
/**
* @brief End of point offset.
* @return Error code
*/
int PointsOffsetDisable();
```

2.2.4.54 Points Offset Code Example

```
private void btnPointOffect_Click(object sender, EventArgs e)
{
    JointPos j1, j2;
    DescPose desc_pos1, desc_pos2, offset_pos, offset_pos1;
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);

    j1 = new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    desc_pos1 = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);

    j2 = new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);

    desc_pos2 = new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);

    offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    offset_pos1 = new DescPose(50.0, 50.0, 50.0, 5.0, 5.0, 5.0);

    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 100.0f;
    float ovl = 100.0f;
    float blendT = -1.0f;
    byte flag = 0;
    int type = 0;

    robot.SetSpeed(20);

    robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    Thread.Sleep(1000);
    robot.PointsOffsetEnable(type, offset_pos1);
    Thread.Sleep(1000);
    robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    Thread.Sleep(1000);
    robot.PointsOffsetDisable();
}
```

2.2.4.55 Control box AO flytap start

New in version C#SDK-v1.0.7.

```
/**
* @brief Control box AO flyswatter start.
* @param [in] AONum control box AO number
* @param [in] maxTCPSpeed maxTCPSpeed value [1-5000mm/s], default 1000
* @param [in] maxAOPercent Percentage of AO corresponding to the maximum TCP speed value, default 100%
* @param [in] zeroZoneCmp AO percentage for dead zone compensation value, shaped, default 20%, range [0-100]
* @return Error code.
*/
int MoveAOStart(int AONum, int maxTCPSpeed, int maxAOPercent, int zeroZoneCmp);
```

2.2.4.56 Control box AO flytap stop

New in version C#SDK-v1.0.7.

```
/**
* @brief Control box AO flytap stops
* @return Error code.
*/
int MoveAOStop();
```

2.2.4.57 End AO flyswatter start

New in version C#SDK-v1.0.7.

```
/**
* @brief End AO flytap start
* @param [in] AONum The end AO number.
* @param [in] maxTCPSpeed maxTCPSpeed value [1-5000mm/s], default 1000
* @param [in] maxAOPercent Percentage of AOs corresponding to the maximum TCP speed value, default 100%
* @param [in] zeroZoneCmp AO percentage for dead zone compensation value, shaped, default 20%, range [0-100]
* @return Error code.
*/
int MoveToolAOStart(int AONum, int maxTCPSpeed, int maxAOPercent, int zeroZoneCmp);
```

2.2.4.58 End AO flytap stop

New in version C#SDK-v1.0.7.

```
/**
* @brief End AO flyswatter stops
* @return Error code
*/
int MoveToolAOStop();
```

2.2.4.59 AO flyswatter code example

```
private void btnMoveAO_Click(object sender, EventArgs e)
{
    JointPos j1 = new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2 = new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose desc_pos1 = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.
↪833);
    DescPose desc_pos2 = new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 100.0f;
    float ovl = 100.0f;
    float blendT = 0.0f;
    float blendR = 0.0f;
    byte flag = 0;
    byte search = 0;

    robot.SetSpeed(5);

    robot.MoveAOStart(0,100,100,20);
    robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveAOStop();

    robot.MoveToolAOStart(0, 100, 100, 20);
    robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveToolAOStop();
}
```

2.2.4.60 Start Ptp motion FIR filtering

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Start Ptp motion FIR filtering.
* @param [in] maxAcc Maximum Acceleration Extreme (deg/s2)
* @param [in] maxJek Uniform Joint Acuity Extreme (deg/s3)
* @return Error code.
*/
int PtpFIRPlanningStart(double maxAcc, double maxJek=1000);
```

2.2.4.61 Turn off Ptp motion FIR filtering

```
/**
* @brief Turn off Ptp motion FIR filtering.
* @return Error code
*/
int PtpFIRPlanningEnd();
```

2.2.4.62 Start LIN, ARC motion FIR filtering.

```
/**
* @brief Start LIN, ARC motion FIR filtering.
* @param [in] maxAccLin Linear Acceleration Extreme (mm/s2)
* @param [in] maxAccDeg angular acceleration extreme (deg/s2)
* @param [in] maxJerkLin Linear plus Acceleration Extreme (mm/s3)
* @param [in] maxJerkDeg Angle plus acceleration extreme (deg/s3)
* @return Error code
*/
int LinArcFIRPlanningStart(double maxAccLin, double maxAccDeg, double maxJerkLin, double␣
↪maxJerkDeg);
```

2.2.4.63 Turn off LIN, ARC motion FIR filtering

```
/**
* @brief Turn off LIN, ARC motion FIR filtering.
* @return Error code
*/
int LinArcFIRPlanningEnd();
```

2.2.4.64 FIR filtering code example

New in version C#SDK-V1.1.3: Web-3.8.2

```
private void button69_Click(object sender, EventArgs e)
{
    int rtn;
    JointPos startjointPos = new JointPos(-11.904f, -99.669f, 117.473f, -108.616f, -91.726f, 74.256f);
    JointPos midjointPos = new JointPos(-45.615f, -106.172f, 124.296f, -107.151f, -91.282f, 74.255f);
    JointPos endjointPos = new JointPos(-29.777f, -84.536f, 109.275f, -114.075f, -86.655f, 74.257f);

    DescPose startdescPose = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f, 3.833f);
    DescPose middescPose = new DescPose(-321.222f, 185.189f, 335.520f, -179.030f, -1.284f, -29.869f);
    DescPose enddescPose = new DescPose(-487.434f, 154.362f, 308.576f, 176.600f, 0.268f, -14.061f);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    rtn = robot.PtpFIRPlanningStart(1000,1000);
    Console.WriteLine("PtpFIRPlanningStart rtn is " + rtn);
    robot.MoveJ( startjointPos,  startdescPose, 0, 0, 100, 100, 100,  exaxisPos, -1, 0, offdese);
    robot.MoveJ( endjointPos,  enddescPose, 0, 0, 100, 100, 100,  exaxisPos, -1, 0, offdese);
    robot.PtpFIRPlanningEnd();
    Console.WriteLine("PtpFIRPlanningEnd rtn is " + rtn);

    robot.LinArcFIRPlanningStart(1000, 1000, 1000, 1000);
    Console.WriteLine("LinArcFIRPlanningStart rtn is " + rtn);
    robot.MoveL( startjointPos,  startdescPose, 0, 0, 100, 100, 100, -1,  exaxisPos, 0, 0,  offdese, 1, 1);
    robot.MoveC( midjointPos,  middescPose, 0, 0, 100, 100,  exaxisPos, 0,  offdese, endjointPos,  enddescPose, 0, 0, 100, 100,  exaxisPos, 0,  offdese, 100, -1);
    robot.LinArcFIRPlanningEnd();
    Console.WriteLine("LinArcFIRPlanningEnd rtn is " + rtn);
}
```

2.2.4.65 Acceleration smoothing on

```
/**
* @brief Acceleration smoothing on
* @param [in] saveFlag Whether to power down and save.
* @return ErrorCode
*/
int AccSmoothStart(bool saveFlag);
```

2.2.4.66 Acceleration SmoothStart(bool saveFlag); int AccSmoothStart

```
/**
* @brief Accelerate smooth shutdown
* @param [in] saveFlag whether to power down and save
* @return ErrorCode
*/
int AccSmoothEnd(bool saveFlag);
```

2.2.4.67 Code example

```
private void button1_Click(object sender, EventArgs e)
{

    int rtn;
    JointPos startjointPos = new JointPos(-11.904f, -99.669f, 117.473f, -108.616f, -91.726f, 74.256f);
    JointPos endjointPos = new JointPos(-45.615f, -106.172f, 124.296f, -107.151f, -91.282f, 74.255f);

    DescPose startdescPose = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f, 3.833f);
    DescPose enddescPose = new DescPose(-321.222f, 185.189f, 335.520f, -179.030f, -1.284f, -29.869f);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
    rtn = robot.AccSmoothStart(false);
    Console.WriteLine("AccSmoothStart rtn is " + rtn);
    robot.MoveJ( startjointPos,  startdescPose, 0, 0, 100, 100, 100,  exaxisPos, -1, 0, offdese);
    robot.MoveJ( endjointPos,  enddescPose, 0, 0, 100, 100, 100,  exaxisPos, -1, 0, offdese);
    rtn = robot.AccSmoothEnd(false);
    Console.WriteLine("AccSmoothEnd rtn is " + rtn);
}
```

2.2.4.68 Specify the attitude speed to turn on

```
/**
* @brief Specify that Attitude Speed is on
* @param [in] ratio Attitude Speed Percentage [0-300]
* @return Error code
*/
int AngularSpeedStart(int ratio).
```

2.2.4.69 Specifies that the attitude speed is off

```
/**
* @brief Specify attitude speed off
* @return Error code
*/
int AngularSpeedEnd();
```

2.2.4.70 Robot Specify Angular Speed Code Example

```
private void button71_Click(object sender, EventArgs e)
{
    int rtn;
    JointPos startjointPos = new JointPos(-11.904f, -99.669f, 117.473f, -108.616f, -91.726f, 74.256f);
    JointPos endjointPos = new JointPos(-45.615f, -106.172f, 124.296f, -107.151f, -91.282f, 74.255f);

    DescPose startdescPose = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f, 3.833f);
    DescPose enddescPose = new DescPose(-321.222f, 185.189f, 335.520f, -179.030f, -1.284f, -29.869f);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
    rtn = robot.AngularSpeedStart(50);
    Console.WriteLine("AngularSpeedStart rtn is " + rtn);
    robot.MoveJ( startjointPos,  startdescPose, 0, 0, 100, 100, 100,  exaxisPos, -1, 0, offdese);
    robot.MoveJ( endjointPos,  enddescPose, 0, 0, 100, 100, 100,  exaxisPos, -1, 0, offdese);
    rtn = robot.AngularSpeedEnd();
    Console.WriteLine("AngularSpeedEnd rtn is " + rtn);
}
```

2.2.4.71 Starting singular position protection

New in version C#SDK-v1.0.9.

```
/**
* @brief Starting singular bitmap protection.
* @param [in] protectMode Singular protection mode, 0: articulation mode; 1 - Cartesian mode
* @param [in] minShoulderPos shoulder singularity adjustment range (mm), default 100
* @param [in] minElbowPos Elbow singularity adjustment range (mm), default 50
* @param [in] minWristPos Wrist singularity range (°), default 10
* @return Error code
*/
int SingularAvoidStart(int protectMode, double minShoulderPos, double minElbowPos, double minWristPos);
```

2.2.4.72 Stop singular attitude protection

New in version C#SDK-v1.0.9.

```
/**
* @brief Stop odd-bit pose protection.
* @return Error code
*/
int SingularAvoidEnd();
```

2.2.4.73 Code example

New in version C#SDK-v1.0.9.

```
private void btnTestSingularAvoidEArc_Click(object sender, EventArgs e)
{
    int rtn;
    JointPos startjointPos = new JointPos(-11.904f, -99.669f, 117.473f, -108.616f, -91.726f, 74.256f);
    JointPos endjointPos = new JointPos(-45.615f, -106.172f, 124.296f, -107.151f, -91.282f, 74.255f);

    DescPose startdescPose = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f, 3.833f);
    DescPose enddescPose = new DescPose(-321.222f, 185.189f, 335.520f, -179.030f, -1.284f, -29.869f);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    rtn = robot.SingularAvoidStart(2, 10, 5, 5);
    Console.WriteLine("SingularAvoidStart rtn is " + rtn);
    robot.MoveJ( startjointPos,  startdescPose, 0, 0, 100, 100, 100,  exaxisPos, -1, 0, offdese);
    robot.MoveJ( endjointPos,  enddescPose, 0, 0, 100, 100, 100,  exaxisPos, -1, 0, offdese);
    rtn = robot.SingularAvoidEnd();
    Console.WriteLine("SingularAvoidEnd rtn is " + rtn);
}
```

2.2.4.74 Safety Stop Trigger

```
/**
* @brief Safe stop trigger signal
* @return Error code
*/
int GetSafetyCode();
```

2.2.4.75 Clear the motion command queue

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Clear the motion command queue
* @return Error code
*/
public int MotionQueueClear();
```

2.2.4.76 Move to Intersecting Line Start Point

```
/**
* @brief Move to intersecting line start point
* @param [in] mainPoint Cartesian poses of 6 taught points on the main pipe
* @param [in] mainExaxisPos Extended axis positions for 6 taught points on the main pipe
* @param [in] piecePoint Cartesian poses of 6 taught points on the branch pipe
* @param [in] pieceExaxisPos Extended axis positions for 6 taught points on the branch␣
↪pipe
* @param [in] extAxisFlag Whether to enable extended axis; 0-Disable; 1-Enable
* @param [in] exaxisPos Start point extended axis position
* @param [in] tool Tool coordinate system number
* @param [in] wobj Workpiece coordinate system number
* @param [in] vel Velocity percentage
* @param [in] acc Acceleration percentage
* @param [in] ovl Velocity scaling factor
* @param [in] oacc Acceleration scaling factor
* @param [in] moveType Motion type; 0-PTP; 1-LIN
* @param [in] moveDirection Motion direction; 0-Clockwise; 1-Counterclockwise
* @param [in] offset Offset value
* @return Error code
*/
public int MoveToIntersectLineStart(DescPose[] mainPoint, ExaxisPos[] mainExaxisPos,␣
↪DescPose[] piecePoint, ExaxisPos[] pieceExaxisPos, int extAxisFlag, ExaxisPos␣
↪exaxisPos, int tool, int wobj, double vel, double acc, double ovl, double oacc, int␣
↪moveType, int moveDirection, DescPose offset);
```

2.2.4.77 Intersecting Line Motion

```
/**
* @brief Intersecting line motion
* @param [in] mainPoint Cartesian poses of 6 taught points on the main pipe
* @param [in] mainExaxisPos Extended axis positions for 6 taught points on the main pipe
* @param [in] piecePoint Cartesian poses of 6 taught points on the branch pipe
* @param [in] pieceExaxisPos Extended axis positions for 6 taught points on the branch␣
↪pipe
* @param [in] extAxisFlag Whether to enable extended axis; 0-Disable; 1-Enable
* @param [in] exaxisPos Start point extended axis positions
* @param [in] tool Tool coordinate system number
* @param [in] wobj Workpiece coordinate system number
```

(continues on next page)

(continued from previous page)

```
* @param [in] vel Velocity percentage
* @param [in] acc Acceleration percentage
* @param [in] ovl Velocity scaling factor
* @param [in] oacc Acceleration scaling factor
* @param [in] moveDirection Motion direction; 0-Clockwise; 1-Counterclockwise
* @param [in] offset Offset value
* @return Error code
*/
public int MoveIntersectLine(DescPose[] mainPoint, ExaxisPos[] mainExaxisPos, DescPose[]␣
↪piecePoint, ExaxisPos[] pieceExaxisPos, int extAxisFlag, ExaxisPos[] exaxisPos, int␣
↪tool, int wobj, double vel, double acc, double ovl, double oacc, int moveDirection,␣
↪DescPose offset);
```

2.2.4.78 Robot Intersecting Line Motion Code Example

```
void TestIntersectLineMove()
{
    ROBOT_STATE_PKG pkg = {};
    FRRobot robot;
    robot.LoggerInit();
    robot.SetLoggerLevel(3);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn != 0)
    {
        return ;
    }
    robot.SetReConnectParam(true, 30000, 500);
    DescPose mainPoint[6] = {};
    DescPose piecePoint[6] = {};
    ExaxisPos mainExaxisPos[6] = {};
    ExaxisPos pieceExaxisPos[6] = {};
    int extAxisFlag = 1;
    ExaxisPos exaxisPos[4] = {};
    DescPose offset = { 0.0, 2.0 ,30.0, -2.0, 0.0, 0.0 };
    mainPoint[0] = {490.004, -383.194, 402.735, -9.332, -1.528, 69.594};
    mainPoint[1] = {444.950, -407.117, 389.011, -5.546, -2.196, 65.279};
    mainPoint[2] = {445.168, -463.605, 355.759, -1.544, -10.886, 57.104};
    mainPoint[3] = {507.529, -485.385, 343.013, -0.786, -4.834, 61.799};
    mainPoint[4] = {554.390, -442.647, 367.701, -4.761, -10.181, 64.925};
    mainPoint[5] = {532.552, -394.003, 396.467, -13.732, -13.592, 67.411};
    mainExaxisPos[0] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[1] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[2] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[3] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[4] = { -29.996, 0.000, 0.000, 0.000 };
    mainExaxisPos[5] = { -29.996, 0.000, 0.000, 0.000 };
    piecePoint[0] = { 505.571, -192.408, 316.759, 38.098, 37.051, 139.447 };
    piecePoint[1] = {533.837, -201.558, 332.340, 34.644, 42.339, 137.748};
    piecePoint[2] = {530.386, -225.085, 373.808, 35.431, 45.111, 137.560};
    piecePoint[3] = {485.646, -229.195, 383.778, 33.870, 45.173, 137.064};
```

(continues on next page)

(continued from previous page)

```
    piecePoint[4] = {460.551, -212.161, 354.256, 28.856, 45.602, 135.930};
    piecePoint[5] = {474.217, -197.124, 324.611, 42.469, 41.133, 148.167};
    pieceExaxisPos[0] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[1] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[2] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[3] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[4] = { -29.996, -0.000, 0.000, 0.000 };
    pieceExaxisPos[5] = { -29.996, -0.000, 0.000, 0.000 };
    exaxisPos[0] = {-29.996, -0.000, 0.000, 0.000};
    exaxisPos[1] = {-44.994, 90.000, 0.000, 0.000};
    exaxisPos[2] = {-59.992, 0.002, 0.000, 0.000};
    exaxisPos[3] = {-44.994, -89.997, 0.000, 0.000};
    int tool = 2;
    int wobj = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 12.0;
    double oacc = 12.0;
    int moveType = 1;
    int moveDirection = 1;
    rtn = robot.MoveToIntersectLineStart(mainPoint, mainExaxisPos, piecePoint,␣
↪pieceExaxisPos, extAxisFlag, exaxisPos[0], tool, wobj, vel, acc, ovl, oacc, moveType,␣
↪moveDirection, offset);
    printf("MoveToIntersectLineStart rtn is %d\n", rtn);
    rtn = robot.MoveIntersectLine(mainPoint, mainExaxisPos, piecePoint, pieceExaxisPos,␣
↪extAxisFlag, exaxisPos, tool, wobj, vel, acc, 5.0, 5.0, moveDirection, offset);
    printf("MoveIntersectLine rtn is %d\n", rtn);
    robot.CloseRPC();
    return ;
}
```

2.2.4.79 Stationary Air Motion

```
/**
* @brief Stationary Air Motion
* @return Error code
*/
public int MoveStationary()
```

2.2.4.80 Stationary Air Motion Code Example

```
public void LaserSensorRecordandReplay()
{
    int rtn = robot.LaserSensorRecordandReplay(0, 10, 1, 0, 0.1, 1, 1, 10, 100);
    Console.WriteLine($"LaserSensorRecordandReplay rtn is {rtn}");
    rtn = robot.MoveStationary();
    Console.WriteLine($"MoveStationary rtn is {rtn}");
    rtn = robot.LaserSensorRecord1(0, 10);
```

(continues on next page)

(continued from previous page)

```
    Console.WriteLine($"LaserSensorRecord1 rtn is {rtn}");
}
```

2.2.5 Robot IO

2.2.5.1 Setting the control box digital outputs

```
/**
* @brief Setting up the control box digital outputs.
* @param [in] id io number, range [0~15]
* @param [in] status 0-off, 1-on
* @param [in] smooth 0-not smooth, 1-smooth
* @param [in] block 0-Blocking, 1-Non-blocking
* @return Error code
*/
int SetDO(int id, byte status, byte smooth, byte block);
```

2.2.5.2 Set the tool digital output

```
/**
* @brief Setting the tool digital output
* @param [in] id io number, range [0~1]
* @param [in] status 0-off, 1-on
* @param [in] smooth 0-not smooth, 1-smooth
* @param [in] block 0-Blocking, 1-Non-blocking
* @return Error code
*/
int SetToolDO(int id, byte status, byte smooth, byte block);
```

2.2.5.3 Set the control box analog output

```
/**
* @brief Setting up the control box analog outputs.
* @param [in] id io number, range [0~1].
* @param [in] value Percentage of current or voltage value, range [0~100] Corresponding␣
↪to current value [0~20mA] or voltage [0~10V].
* @param [in] block 0-blocking, 1-non-blocking
* @return Error code
*/
int SetAO(int id, float value, byte block).
```

2.2.5.4 Set the tool analog output

```
/**
* @brief set tool analog output
* @param [in] id io number, range [0].
* @param [in] value Percentage of current or voltage value, range [0~100] corresponding to current value [0~20mA] or voltage [0~10V]
* @param [in] block 0-blocking, 1-non-blocking
* @return Error code
*/
int SetToolAO(int id, float value, byte block).
```

2.2.5.5 Set digital, analog output code example

```
private void button14_Click(object sender, EventArgs e)
{
    byte status = 1;
    byte smooth = 0;
    byte block = 0;
    byte di = 0, tool_di = 0;
    float ai = 0.0f, tool_ai = 0.0f;
    float value = 0.0f;


    for (int i = 0; i < 16; i++)
    {
        robot.SetDO(i, status, smooth, block);
        Thread.Sleep(300);
    }

    status = 0;

    for (int i = 0; i < 16; i++)
    {
        robot.SetDO(i, status, smooth, block);
        Thread.Sleep(300);
    }

    status = 1;

    for (int i = 0; i < 2; i++)
    {
        robot.SetToolDO(i, status, smooth, block);
        Thread.Sleep(1000);
    }

    status = 0;

    for (int i = 0; i < 2; i++)
    {
        robot.SetToolDO(i, status, smooth, block);
```

(continues on next page)

(continued from previous page)

```
        Thread.Sleep(1000);
    }

    for (int i = 0; i < 100; i++)
    {
        robot.SetAO(0, i, block);
        Thread.Sleep(30);
    }

    for (int i = 0; i < 100; i++)
    {
        robot.SetToolAO(0, i, block);
        Thread.Sleep(30);
    }

}
```

2.2.5.6 etting control box digital inputs

```
/**
* @brief Get control box digital inputs.
* @param [in] id io number, range [0~15]
* @param [in] block 0-blocking, 1-non-blocking
* @param [out] result 0-low level, 1-high level
* @return error code
*/
int GetDI(int id, byte block, ref byte level).
```

2.2.5.7 Get tool digital input

```
/**
* @brief Get instrumented digital input
* @param [in] id io number, range [0~1]
* @param [in] block 0-blocking, 1-non-blocking
* @param [out] result 0-low level, 1-high level
* @return error code
*/
int GetToolDI(int id, byte block, ref byte level);
```

2.2.5.8 Get control box analog input

```
/**
* @brief Get control box analog inputs.
* @param [in] id io number, range [0~1]
* @param [in] block 0-blocking, 1-non-blocking
* @param [out] result Input current or voltage value percentage, range [0~100]␣
↪corresponding to current value [0~20mS] or voltage [0~10V]
* @return error code
*/
int GetAI(int id, byte block, ref float persent).
```

2.2.5.9 Get tool analog input

```
/**
* @brief Get tool analog input
* @param [in] id io number, range [0]
* @param [in] block 0-blocking, 1-non-blocking
* @param [out] result Input current or voltage value percentage, range [0~100]␣
↪corresponds to current value [0~20mS] or voltage [0~10V]
* @return error code
*/
int GetToolAI(int id, byte block, ref float persent).
```

2.2.5.10 Get the robot end record button status

```
/**
* @brief Get end-of-robot record button state.
* @param [out] state Button state, 0-pressed, 1-unpressed.
* @return Error code.
*/
int GetAxlePointRecordBtnState(ref byte state).
```

2.2.5.11 Get the robot end DO output state

```
/**
* @brief Get robot end DO output state.
* @param [out] do_state DO output state, do0~do1 corresponds to bit1~bit2, start from␣
↪bit0.
* @return Error code
*/
int GetToolDO(ref byte do_state).
```

2.2.5.12 Get the DO output state of the machine controller

```
/**
* @brief Get robot controller DO output state.
* @param [out] do_state_h DO output state, co0~co7 corresponds to bit0~bit7
* @param [out] do_state_l DO output state, do0~do7 corresponds to bit0~bit7
* @return Error code
*/
int GetDO(ref int do_state_h, ref int do_state_l);
```

2.2.5.13 Get robot DI, DO state code example

```
private void button15_Click(object sender, EventArgs e)
{
    byte status = 1;
    byte smooth = 0;
    byte block = 0;
    byte di = 0, tool_di = 0;
    float ai = 0.0f, tool_ai = 0.0f;
    float value = 0.0f;

    robot.GetDI(0, block, ref di);
    Console.WriteLine($"di0: {di}");

    tool_di = (byte)robot.GetToolDI(1, block, ref tool_di);
    Console.WriteLine($"tool_di1: {tool_di}");

    robot.GetAI(0, block, ref ai);
    Console.WriteLine($"ai0: {ai}");

    tool_ai = robot.GetToolAI(0, block, ref tool_ai);
    Console.WriteLine($"tool_ai0: {tool_ai}");

    byte _button_state = 0;
    robot.GetAxlePointRecordBtnState(ref _button_state);
    Console.WriteLine($"_button_state is: {_button_state}");

    byte tool_do_state = 0;
    robot.GetToolDO(ref tool_do_state);
    Console.WriteLine($"tool DO state is: {tool_do_state}");

    int do_state_h = 0;
    int do_state_l = 0;
    robot.GetDO(ref do_state_h, ref do_state_l);
    Console.WriteLine($"DO state high is: {do_state_h}\n DO state low is: {do_state_l}");
}
```

2.2.5.14 Wait for control box digital input

```
/**
* @brief Wait for control box digital input.
* @param [in] id io number, range [0~15]
* @param [in] status 0-off, 1-on
* @param [in] max_time Maximum waiting time in ms
* @param [in] opt Timeout policy, 0-program stops and prompts for timeout, 1-ignore␣
↪timeout prompts and continue execution, 2-wait all the time.
* @return error_code
*/
int WaitDI(int id, byte status, int max_time, int opt).
```

2.2.5.15 Wait for multiple digital inputs to the control box

```
/**
* @brief Waiting for control box multiplexed digital inputs.
* @param [in] mode 0-multiplex with, 1-multiplex or
* @param [in] id io number, bit0~bit7 corresponds to DI0~DI7, bit8~bit15 corresponds to␣
↪CI0~CI7
* @param [in] status 0-off, 1-on
* @param [in] max_time Maximum wait time in ms.
* @param [in] opt Timeout policy, 0 - program stops and prompts for timeout, 1 - ignores␣
↪timeout prompts and continues execution, 2 - waits forever.
* @return error_code
*/
int WaitMultiDI(int mode, int id, byte status, int max_time, int opt).
```

2.2.5.16 Wait for tool digital input

```
/**
* @brief Waiting for instrumented digital input
* @param [in] id io number, range [0~1]
* @param [in] status 0-off, 1-on
* @param [in] max_time Maximum wait time in ms
* @param [in] opt Timeout policy, 0-program stops and prompts for timeout, 1-ignore␣
↪timeout prompts and continue execution, 2-wait all the time.
* @return error_code
*/
int WaitToolDI(int id, byte status, int max_time, int opt);
```

2.2.5.17 Wait for control box analog input

```
/**
* @brief Waiting for control box analog input.
* @param [in] id io number, range [0~1]
* @param [in] sign 0-greater than, 1-less than
* @param [in] value Input current or voltage value percentage, range [0~100]␣
↪corresponding to current value [0~20mS] or voltage [0~10V]
* @param [in] max_time Maximum wait time in ms
* @param [in] opt Policy after timeout, 0-program stops and prompts for timeout, 1-
↪ignore timeout and prompt program to continue, 2-always wait
* @return error_code
*/
int WaitAI(int id, int sign, float value, int max_time, int opt);
```

2.2.5.18 Wait for tool analog input

```
/**
* @brief Waiting for tool analog input
* @param [in] id io number, range [0]
* @param [in] sign 0-greater than, 1-less than
* @param [in] value Input current or voltage value percentage, range [0~100]␣
↪corresponding to current value [0~20mS] or voltage [0~10V]
* @param [in] max_time Maximum wait time in ms
* @param [in] opt Policy after timeout, 0-program stops and prompts for timeout, 1-
↪ignore timeout and prompt program to continue, 2-always wait
* @return error_code
*/
int WaitToolAI(int id, int sign, float value, int max_time, int opt).
```

2.2.5.19 Wait for the control box digital, analog input signal code example

```
private void btnIOTest_Click(object sender, EventArgs e)
{
    byte status = 1;
    byte smooth = 0;
    byte block = 0;
    byte di = 0, tool_di = 0;
    float ai = 0.0f, tool_ai = 0.0f;
    float value = 0.0f;

    int rtn = robot.WaitDI(0, 1, 1000, 1);
    Console.WriteLine("WaitDI over; rtn is: " + rtn);

    robot.WaitMultiDI(1, 3, 3, 1000, 1);
    Console.WriteLine("WaitMultiDI over; rtn is: " + rtn);

    robot.WaitToolDI(1, 1, 1000, 1);
    Console.WriteLine("WaitToolDI over; rtn is: " + rtn);

```

(continues on next page)

(continued from previous page)

```
    robot.WaitAI(0, 0, 50, 1000, 1);
    Console.WriteLine("WaitAI over; rtn is: " + rtn);

    robot.WaitToolAI(0, 0, 50, 1000, 1);
    Console.WriteLine("WaitToolAI over; rtn is: " + rtn);
}
```

2.2.5.20 Set Whether Control Box DO Output Resets After Stop/Pause

```
/**
* @brief Set whether control box DO output resets after stop/pause
* @param [in] resetFlag 0-Do not reset; 1-Reset
* @param [in] reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetCtlBoxDO(int resetFlag, int reloadFlag);
```

2.2.5.21 Set Whether Control Box AO Output Resets After Stop/Pause

```
/**
* @brief Set whether control box AO output resets after stop/pause
* @param [in] resetFlag 0-Do not reset; 1-Reset
* @param [in] reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetCtlBoxAO(int resetFlag, int reloadFlag);
```

2.2.5.22 Set Whether End Tool DO Output Resets After Stop/Pause

```
/**
* @brief Set whether end tool DO output resets after stop/pause
* @param [in] resetFlag 0-Do not reset; 1-Reset
* @param [in] reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetAxleDO(int resetFlag, int reloadFlag);
```

2.2.5.23 Set Whether End Tool AO Output Resets After Stop/Pause

```
/**
* @brief Set whether end tool AO output resets after stop/pause
* @param [in] resetFlag 0-Do not reset; 1-Reset
* @param [in] reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetAxleAO(int resetFlag, int reloadFlag);
```

2.2.5.24 Set Whether Extended DO Output Resets After Stop/Pause

```
/**
* @brief Set whether extended DO output resets after stop/pause
* @param [in] resetFlag 0-Do not reset; 1-Reset
* @param [in] reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetExtDO(int resetFlag, int reloadFlag);
```

2.2.5.25 Set Whether Extended AO Output Resets After Stop/Pause

```
/**
* @brief Set whether extended AO output resets after stop/pause
* @param [in] resetFlag 0-Do not reset; 1-Reset
* @param [in] reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetExtAO(int resetFlag, int reloadFlag);
```

2.2.5.26 Set Whether SmartTool Output Resets After Stop/Pause

```
/**
* @brief Set whether SmartTool output resets after stop/pause
* @param [in] resetFlag 0-Do not reset; 1-Reset
* @param [in] reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetSmartToolDO(int resetFlag, int reloadFlag);
```

2.2.5.27 Code Example for Setting Output Reset After Lua Program Stop/Pause

```
public void TestDOReset()
{
    ROBOT_STATE_PKG pkg = new ROBOT_STATE_PKG();

    for (int i = 0; i < 16; i++)
    {
        robot.SetDO(i, 1, 0, 0);
        Thread.Sleep(200);
    }

    int resetFlag = 1;
    int resumeReloadFlag = 1;
    int rtn = robot.SetOutputResetCtlBoxDO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetCtlBoxAO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetAxleDO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetAxleAO(resetFlag, resumeReloadFlag);
```

(continues on next page)

(continued from previous page)

```
    robot.SetOutputResetExtDO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetExtAO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetSmartToolDO(resetFlag, resumeReloadFlag);

    robot.ProgramLoad("/fruser/test.lua");
    robot.ProgramRun();

    Thread.Sleep(2000);
    robot.PauseMotion();
    Thread.Sleep(2000);
    robot.ResumeMotion();
    Thread.Sleep(2000);
}
```

2.2.6 Common Robot Settings

2.2.6.1 Setting the tool reference point - six-point method

```
/**
* @brief Setting the tool reference point - the six-point method.
* @param [in] point_num point_num, range [1~6].
* @return Error code
*/
int SetToolPoint(int point_num).
```

2.2.6.2 Calculating the Tool Coordinate System - Six Point Method

```
/**
* @brief Calculate the tool coordinate system
* @param [out] tcp_pose tool coordinate system
* @return Error code.
*/
int ComputeTool(ref DescPose tcp_pose).
```

2.2.6.3 Setting the tool reference point - four-point method

```
/**
* @brief Setting the tool reference point - four-point method.
* @param [in] point_num point_num, range [1~4].
* @return Error code
*/
int SetTcp4RefPoint(int point_num).
```

2.2.6.4 Calculate the tool coordinate system - four-point method

```
 /**
* @brief Calculate the tool coordinate system
* @param [out] tcp_pose tool coordinate system
* @return Error code.
*/
int ComputeTcp4(ref DescPose tcp_pose).
```

2.2.6.5 Set the tool coordinate system

```
/**
* @brief Set the tool coordinate system.
* @param [in] id Coordinate system number, range [0~14].
* @param [in] coord tool center with respect to end-flange center.
* @param [in] type 0-tool coordinate system, 1-sensor coordinate system
* @param [in] install mounting position, 0-robot end, 1-robot exterior
* param [in] toolID toolID
* @param [in] loadNum loadNumber
* @return ErrorCode
*/
int SetToolCoord(int id, DescPose coord, int type, int install,int toolID, int loadNum);
```

2.2.6.6 Calculate the tool coordinate system from the point information

```
/**
* @brief Calculate tool coordinate system from point information.
* @param [in] method Calculation method; 0 - four-point method; 1 - six-point method
* @param [in] pos Joint position group, array length is 4 for 4-point method, 6 for 6-
↪point method.
* @return Error code
*/
int ComputeToolCoordWithPoints(int method, JointPos[] pos, ref DescPose coordRtn)
```

2.2.6.7 Set the list of tool coordinate systems

```
/**
* @brief Setting up a list of tool coordinate systems.
* @param [in] id Coordinate system number, range [0~14].
* @param [in] coord Tool center point relative to end flange center position.
* @param [in] type 0-tool coordinate system, 1-sensor coordinate system
* @param [in] install mounting position, 0-robot end, 1-robot exterior
* @param [in] loadNum loadNumber
* @return Error code
*/
int SetToolList(int id, DescPose coord, int type, int install, int loadNum);
```

2.2.6.8 Get the current tool coordinate system

```
/**
 * @brief Get the current tool coordinate system.
 * @param [in] flag 0-blocking, 1-non-blocking
 * @param [out] desc_pos Tool coordinate system position.
 * @return Error code.
 */
int GetTCPOffset(byte flag, ref DescPose desc_pos);
```

2.2.6.9 Robot tool coordinate system operation code example

```
private void button18_Click(object sender, EventArgs e)
{
    DescPose p1Desc = new DescPose(186.331f, 487.913f, 209.850f, 149.030f, 0.688f, -114.347f);
    JointPos p1Joint = new JointPos(-127.876f, -75.341f, 115.417f, -122.741f, -59.820f, 74.300f);

    DescPose p2Desc = new DescPose(69.721f, 535.073f, 202.882f, -144.406f, -14.775f, -89.012f);
    JointPos p2Joint = new JointPos(-101.780f, -69.828f, 110.917f, -125.740f, -127.841f, 74.300f);

    DescPose p3Desc = new DescPose(146.861f, 578.426f, 205.598f, 175.997f, -36.178f, -93.437f);
    JointPos p3Joint = new JointPos(-112.851f, -60.191f, 86.566f, -80.676f, -97.463f, 74.300f);

    DescPose p4Desc = new DescPose(136.284f, 509.876f, 225.613f, 178.987f, 1.372f, -100.696f);
    JointPos p4Joint = new JointPos(-116.397f, -76.281f, 113.845f, -128.611f, -88.654f, 74.299f);

    DescPose p5Desc = new DescPose(138.395f, 505.972f, 298.016f, 179.134f, 2.147f, -101.110f);
    JointPos p5Joint = new JointPos(-116.814f, -82.333f, 109.162f, -118.662f, -88.585f, 74.302f);

    DescPose p6Desc = new DescPose(105.553f, 454.325f, 232.017f, -179.426f, 0.444f, -99.952f);
    JointPos p6Joint = new JointPos(-115.649f, -84.367f, 122.447f, -128.663f, -90.432f, 74.303f);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    JointPos[] posJ = new JointPos[] { p1Joint, p2Joint, p3Joint, p4Joint, p5Joint, p6Joint };
    DescPose coordRtn = new DescPose();
    int rtn = robot.ComputeToolCoordWithPoints(1, posJ, ref coordRtn);
```

(continues on next page)

(continued from previous page)

```
    Console.WriteLine($"ComputeToolCoordWithPoints    {rtn}  coord is {coordRtn.tran.x} {coordRtn.tran.y} {coordRtn.tran.z} {coordRtn.rpy.rx} {coordRtn.rpy.ry} {coordRtn.rpy.rz}");

    robot.MoveJ( p1Joint,  p1Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetToolPoint(1);
    robot.MoveJ( p2Joint,  p2Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetToolPoint(2);
    robot.MoveJ( p3Joint,  p3Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetToolPoint(3);
    robot.MoveJ( p4Joint,  p4Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetToolPoint(4);
    robot.MoveJ( p5Joint,  p5Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetToolPoint(5);
    robot.MoveJ( p6Joint,  p6Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetToolPoint(6);
    rtn = robot.ComputeTool(ref coordRtn);
    Console.WriteLine($"6 Point ComputeTool        {rtn}  coord is {coordRtn.tran.x} {coordRtn.tran.y} {coordRtn.tran.z} {coordRtn.rpy.rx} {coordRtn.rpy.ry} {coordRtn.rpy.rz}");
    robot.SetToolList(1,  coordRtn, 0, 0, 0);

    robot.MoveJ( p1Joint,  p1Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetTcp4RefPoint(1);
    robot.MoveJ( p2Joint,  p2Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetTcp4RefPoint(2);
    robot.MoveJ( p3Joint,  p3Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetTcp4RefPoint(3);
    robot.MoveJ( p4Joint,  p4Desc, 0, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetTcp4RefPoint(4);
    rtn = robot.ComputeTcp4(ref coordRtn);
    Console.WriteLine($"4 Point ComputeTool        {rtn}  coord is {coordRtn.tran.x} {coordRtn.tran.y} {coordRtn.tran.z} {coordRtn.rpy.rx} {coordRtn.rpy.ry} {coordRtn.rpy.rz}");

    robot.SetToolCoord(2, coordRtn, 0, 0, 1, 0);

    DescPose getCoord = new DescPose();
    rtn = robot.GetTCPOffset(0, ref getCoord);
    Console.WriteLine($"GetTCPOffset    {rtn}  coord is {coordRtn.tran.x} {coordRtn.tran.y} {coordRtn.tran.z} {coordRtn.rpy.rx} {coordRtn.rpy.ry} {coordRtn.rpy.rz}");
}
```

2.2.6.10 Setting the external tool coordinate reference point - three-point method

```
/**
* @brief Setting the external tool reference point - the three-point method.
* @param [in] point_num point_num, range [1~3].
* @return Error code
*/
int SetExTCPPoint(int point_num).
```

2.2.6.11 Calculating an external tool coordinate system - three-point method

```
/**
* @brief Calculate external tool coordinate system - three-point method.
* @param [out] tcp_pose external tool coordinate system
* @return Error code.
*/
int ComputeExTCF(ref DescPose tcp_pose).
```

2.2.6.12 Set the external tool coordinate system

```
/**
* @brief Set the external tool coordinate system.
* @param [in] id Coordinate system number, range [0~14].
* @param [in] etcp Tool center point relative to end-flange center position.
* @param [in] etool To be determined.
* @return Error code
*/
int SetExToolCoord(int id, DescPose etcp, DescPose etool);
```

2.2.6.13 Setting up a list of external tool coordinate systems

```
/**
* @brief Set the list of external tool coordinate systems.
* @param [in] id Coordinate system number, range [0~14].
* @param [in] etcp tool center point relative to end-flange center position
* @param [in] etool pending
* @return Error code
*/
int SetExToolList(int id, DescPose etcp, DescPose etool);
```

2.2.6.14 Calculate the workpiece coordinate system from the point information

```
/**
 * @brief Calculate the workpiece coordinate system from point information.
 * @param [in] method Calculation method; 0: origin-x-axis-z-axis 1: origin-x-axis-xy plane
 * @param [in] pos Three TCP position groups
 * @param [in] refFrame reference coordinate system
 * @return Error code
 */
int ComputeWObjCoordWithPoints(int method, DescPose[] pos, int refFrame, ref DescPose coordRtn)
```

2.2.6.15 Sample code for manipulating the robot's external tool coordinate system

```
private void button20_Click(object sender, EventArgs e)
{
    DescPose p1Desc = new DescPose(-89.606f, 779.517f, 193.516f, 178.000f, 0.476f, -92.484f);
    JointPos p1Joint = new JointPos(-108.145f, -50.137f, 85.818f, -125.599f, -87.946f, 74.329f);

    DescPose p2Desc = new DescPose(-24.656f, 850.384f, 191.361f, 177.079f, -2.058f, -95.355f);
    JointPos p2Joint = new JointPos(-111.024f, -41.538f, 69.222f, -114.913f, -87.743f, 74.329f);

    DescPose p3Desc = new DescPose(-99.813f, 766.661f, 241.878f, -176.817f, 1.917f, -91.604f);
    JointPos p3Joint = new JointPos(-107.266f, -56.116f, 85.971f, -122.560f, -92.548f, 74.331f);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    DescPose[] posTCP = new DescPose[] { p1Desc, p2Desc, p3Desc };
    DescPose coordRtn = new DescPose();

    robot.MoveJ( p1Joint,  p1Desc, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetExTCPPoint(1);
    robot.MoveJ( p2Joint,  p2Desc, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetExTCPPoint(2);
    robot.MoveJ( p3Joint,  p3Desc, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetExTCPPoint(3);
    int rtn = robot.ComputeExTCF(ref coordRtn);
    Console.WriteLine($"ComputeExTCF                    {rtn}  coord is {coordRtn.tran.x} {coordRtn.tran.y} {coordRtn.tran.z} {coordRtn.rpy.rx} {coordRtn.rpy.ry} {coordRtn.rpy.rz}");

    robot.SetExToolCoord(1,  coordRtn,  offdese);
    robot.SetExToolList(1,  coordRtn,  offdese);
}
```

2.2.6.16 Setting the reference point of the workpiece coordinate system - three-point method

```
/**
* @brief Setting the reference point of a workpiece - the three-point method.
* @param [in] point_num point_num, range [1~3].
* @return Error code
*/
int SetWObjCoordPoint(int point_num).
```

2.2.6.17 Calculate the work coordinate system

```
/**
* @brief Calculate the workpiece coordinate system.
* @param [in] method Calculation method 0: origin-x-axis-z-axis 1: origin-x-axis-xy-
↪plane
* @param [in] refFrame reference coordinate system
* @param [out] wobj_pose Workpiece coordinate system
* @return Error code
*/
int ComputeWObjCoord(int method, int refFrame, ref DescPose wobj_pose);
```

2.2.6.18 Set the workpiece coordinate system

```
/**
* @brief Set the workpiece coordinate system.
* @param [in] id Coordinate system number, range [1~15].
* @param [in] coord Workpiece coordinate system relative to end flange center.
* @param [in] refFrame The reference coordinate system.
* @return Error code
*/
int SetWObjCoord(int id, DescPose coord, int refFrame);
```

2.2.6.19 Set the list of workpiece coordinate systems

```
/**
* @brief Set the list of workpiece coordinate systems.
* @param [in] id coordinate system number, range [0~14]
* @param [in] coord Workpiece coordinate system relative to the center of the end␣
↪flange.
* @param [in] refFrame The reference coordinate system.
* @return Error code
*/
int SetWObjList(int id, DescPose coord, int refFrame);
```

2.2.6.20 Get the current workpiece coordinate system

```
/**
 * @brief Get the current workpiece coordinate system.
 * @param [in] flag 0-blocking, 1-non-blocking
 * @param [out] desc_pos Workpiece coordinate system position.
 * @return Error code.
 */
int GetWObjOffset(byte flag, ref DescPose desc_pos);
```

2.2.6.21 Robot workpiece coordinate system operation code example

```
private void button19_Click(object sender, EventArgs e)
{
    DescPose p1Desc = new DescPose(-89.606, 779.517, 193.516, 178.000, 0.476, -92.484);
    JointPos p1Joint = new JointPos(-108.145, -50.137, 85.818, -125.599, -87.946, 74.
↪329);

    DescPose p2Desc = new DescPose(-24.656, 850.384, 191.361, 177.079, -2.058, -95.355);
    JointPos p2Joint = new JointPos(-111.024, -41.538, 69.222, -114.913, -87.743, 74.
↪329);

    DescPose p3Desc = new DescPose(-99.813, 766.661, 241.878, -176.817, 1.917, -91.604);
    JointPos p3Joint = new JointPos(-107.266, -56.116, 85.971, -122.560, -92.548, 74.
↪331);

    robot.GetForwardKin(p1Joint,ref p1Desc);
    robot.GetForwardKin(p2Joint,ref p2Desc);
    robot.GetForwardKin(p3Joint, ref p3Desc);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    DescPose[] posTCP = new DescPose[] { p1Desc, p2Desc, p3Desc };
    DescPose coordRtn = new DescPose();
    int rtn = robot.ComputeWObjCoordWithPoints(1, posTCP, 0, ref coordRtn);
    Console.WriteLine($"ComputeWObjCoordWithPoints    {rtn}  coord is {coordRtn.tran.x}
↪{coordRtn.tran.y} {coordRtn.tran.z} {coordRtn.rpy.rx} {coordRtn.rpy.ry} {coordRtn.rpy.
↪rz}");

    robot.MoveJ( p1Joint,  p1Desc, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetWObjCoordPoint(1);
    robot.MoveJ( p2Joint,  p2Desc, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetWObjCoordPoint(2);
    robot.MoveJ( p3Joint,  p3Desc, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.SetWObjCoordPoint(3);
    rtn = robot.ComputeWObjCoord(1, 0, ref coordRtn);
    Console.WriteLine($"ComputeWObjCoord                    {rtn}  coord is {coordRtn.
↪tran.x} {coordRtn.tran.y} {coordRtn.tran.z} {coordRtn.rpy.rx} {coordRtn.rpy.ry}
↪{coordRtn.rpy.rz}");

```

(continues on next page)

(continued from previous page)

```
    robot.SetWObjCoord(1,  coordRtn, 0);
    robot.SetWObjList(1,  coordRtn, 0);

    DescPose getWobjDesc = new DescPose();
    rtn = robot.GetWObjOffset(0, ref getWobjDesc);
    Console.WriteLine($"GetWObjOffset                    {rtn}  coord is {coordRtn.tran.x}
↪ {coordRtn.tran.y} {coordRtn.tran.z} {coordRtn.rpy.rx} {coordRtn.rpy.ry} {coordRtn.rpy.
↪rz}");
}
```

2.2.6.22 Setting the global speed

```
/**
* @brief Set the global speed.
* @param [in] vel Speed percentage, range [0~100].
* @return Error code.
*/
int SetSpeed(int vel).
```

2.2.6.23 Set the robot acceleration

```
/**
* @brief Set robot acceleration
* @param [in] acc robot acceleration percentage
* @return Error code.
*/
int SetOaccScale(double acc)
```

2.2.6.24 Get the default robot speed

```
/**
* @brief Get the default speed of the robot.
* @param [out] vel velocity in mm/s
* @return Error code.
*/
int GetDefaultTransVel(ref double vel).
```

2.2.6.25 Set the end load weight

```
/**
* @brief Set the end load weight.
* @param [in] loadNum loadNumber
* @param [in] weight load weight in kg
* @return Error code.
*/
int SetLoadWeight(int loadNum, float weight)
```

2.2.6.26 Set the center of mass of the end load

```
/**
* @brief Set end-load center-of-mass coordinates.
* @param [in] coord center of mass in mm.
* @return Error code.
*/
int SetLoadCoord(DescTran coord).
```

2.2.6.27 Get the weight of the current load

```
/**
* @brief Get the weight of the current load
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] weight weight of the load in kg
* @return Error code.
*/
int GetTargetPayload(byte flag, ref double weight);
```

2.2.6.28 Get the center of mass of the current load

```
/**
* @brief Get the center of mass of the current load.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] cog load center of mass in mm
* @return Error code.
*/
int GetTargetPayloadCog(byte flag, ref DescTran cog);
```

2.2.6.29 Set the robot mounting method

```
/**
* @brief Set the robot installation method.
* @param [in] install Installation method, 0-front, 1-side, 2-backwards.
* @return Error code.
*/
int SetRobotInstallPos(byte install).
```

2.2.6.30 Set the robot installation angle

```
/**
* @brief Setting the robot mounting angle for free mounting.
* @param [in] yangle tilt angle
* @param [in] zangle angle of rotation.
* @return Error code.
*/
int SetRobotInstallAngle(double yangle, double zangle);
```

2.2.6.31 Get the robot installation angle

```
/**
* @brief Get the robot mounting angle.
* @param [out] yangle tilt angle
* @param [out] zangle angle of rotation.
* @return Error code.
*/
int GetRobotInstallAngle(ref double yangle, ref double zangle);
```

2.2.6.32 Sets the value of the system variable

```
/**
* @brief Setting system variable values
* @param [in] id Variable number, range [1~20].
* @param [in] value Variable number, range [1~20].
* @return Error code.
*/
int SetSysVarValue(int id, double value);
```

2.2.6.33 Get the value of a system variable

```
/**
* @brief Get the value of a system variable.
* @param [in] id System variable number, in the range [1~20].
* @param [out] value System variable value.
* @return Error code.
*/
int GetSysVarValue(int id, ref double value);
```

2.2.6.34 Sample code for common robot settings

```
private void button21_Click(object sender, EventArgs e)
{
    for (int i = 1; i < 100; i++)
    {
        robot.SetSpeed(i);
        robot.SetOaccScale(i);
        Thread.Sleep(30);
    }

    double defaultVel = 0.0f;
    robot.GetDefaultTransVel(ref defaultVel);
    Console.WriteLine($"GetDefaultTransVel is {defaultVel}");

    for (int i = 1; i < 21; i++)
    {
        robot.SetSysVarValue(i, i + 0.5f);
```

(continues on next page)

(continued from previous page)

```
        Thread.Sleep(100);
    }

    for (int i = 1; i < 21; i++)
    {
        double value = 0;
        robot.GetSysVarValue(i, ref value);
        Console.WriteLine($"sys value  {i} is :{value}");
        Thread.Sleep(100);
    }

    robot.SetLoadWeight(0, 2.5f);

    DescTran loadCoord = new DescTran();
    loadCoord.x = 3.0f;
    loadCoord.y = 4.0f;
    loadCoord.z = 5.0f;
    robot.SetLoadCoord( loadCoord);

    Thread.Sleep(1000);

    double getLoad = 0.0f;
    robot.GetTargetPayload(0, ref getLoad);

    DescTran getLoadTran = new DescTran();
    robot.GetTargetPayloadCog(0, ref getLoadTran);
    Console.WriteLine($"get load is {getLoad}; get load cog is {getLoadTran.x}
↪{getLoadTran.y} {getLoadTran.z}");

    robot.SetRobotInstallPos(0);
    robot.SetRobotInstallAngle(15.0f, 25.0f);

    double anglex = 0.0f;
    double angley = 0.0f;
    robot.GetRobotInstallAngle(ref anglex, ref angley);
    Console.WriteLine($"GetRobotInstallAngle x:  {anglex};  y:  {angley}");
}
```

2.2.6.35 Joint friction compensation switch

```
/**
* @brief Joint friction compensation switch.
* @param [in] state 0-off, 1-on
* @return Error code.
*/
int FrictionCompensationOnOff(byte state).
```

2.2.6.36 Sets the joint friction compensation factor - proper

```
/**
* @brief Set the joint friction compensation coefficients-front-loading
* @param [in] coeff Six joint compensation coefficients, range [0~1].
* @return error code
*/
int SetFrictionValue_level(double[] coeff).
```

2.2.6.37 Set joint friction compensation coefficients - side mount

```
/**
* @brief Set joint friction compensation coefficients-side-loading
* @param [in] coeff Six joint compensation coefficients, range [0~1].
* @return error code
*/
int SetFrictionValue_wall(double[] coeff).
```

2.2.6.38 Set joint friction compensation coefficients - inverted

```
/**
* @brief Set joint friction compensation coefficients-flip-flop
* @param [in] coeff Six joint compensation coefficients, range [0~1].
* @return Error code
*/
int SetFrictionValue_ceiling(double[] coeff).
```

2.2.6.39 Set joint friction compensation coefficients - free mounting

```
/**
* @brief Set joint friction compensation coefficients - free-mounting
* @param [in] coeff Six joint compensation coefficients, range [0 to 1].
* @return Error code.
*/
int SetFrictionValue_freedom(double[] coeff);
```

2.2.6.40 Robot set joint friction compensation code example

```
private void btnRobotSafetySet_Click(object sender, EventArgs e)
{
    double[] lcoeff = { 0.9f, 0.9f, 0.9f, 0.9f, 0.9f, 0.9f };
    double[] wcoeff = { 0.4f, 0.4f, 0.4f, 0.4f, 0.4f, 0.4f };
    double[] ccoeff = { 0.6f, 0.6f, 0.6f, 0.6f, 0.6f, 0.6f };
    double[] fcoeff = { 0.5f, 0.5f, 0.5f, 0.5f, 0.5f, 0.5f };

    int rtn = robot.FrictionCompensationOnOff(1);
    Console.WriteLine($"FrictionCompensationOnOff rtn is{rtn}");
```

(continues on next page)

(continued from previous page)

```

    Console.WriteLine($"FrictionCompensationOnOff rtn is{rtn}");
    Console.WriteLine($"SetFrictionValue_level rtn is {rtn}");

    Console.WriteLine($"SetFrictionValue_level rtn is {rtn}");
    Console.WriteLine($"SetFrictionValue_wall rtn is {rtn}");

    rtn = robot.SetFrictionValue_ceiling(ccoeff);
    Console.WriteLine($"SetFrictionValue_ceiling rtn is {rtn}");

    rtn = robot.SetFrictionValue_freedom(fcoeff);
    Console.WriteLine($"SetFrictionValue_freedom rtn is {rtn}");
}
```

2.2.6.41 Querying the robot error code

```
/**
* @brief Query robot error code
* @param [out] maincode mainerrorcode
* @param [out] subcode suberrorcode
* @return errorcode
*/
int GetRobotErrorCode(ref int maincode, ref int subcode);
```

2.2.6.42 Error status clearing

```
/**
* @brief Error state clearing
* @return Error code
*/
int ResetAllError();
```

2.2.6.43 Example of getting robot fault status and clearing error code

```
private void btnRobotSafetySet_Click(object sender, EventArgs e)
{
    int maincode=0, subcode=0;
    robot.GetRobotErrorCode(ref maincode, ref subcode);
    Console.WriteLine($"robot maincode is{maincode}; subcode is {subcode}" );

    robot.ResetAllError();

    Thread.Sleep(1000);

    robot.GetRobotErrorCode(ref maincode, ref subcode);
    Console.WriteLine($"robot maincode is{maincode}; subcode is{subcode}");
}
```

2.2.6.44 Setting the parameters for monitoring the temperature and fan speed of the wide voltage control box

New in version C#SDK-V1.1.4: Web-3.8.3

```
/**
 * @brief Setting the parameters for monitoring the temperature and fan speed of the wide voltage control box.
 * @param [in] enable 0-does not enable monitoring; 1-enables monitoring
 * @param [in] period Monitoring period (s), range 1-100.
 * @return Error code
 */
int SetWideBoxTempFanMonitorParam(int enable, int period);
```

2.2.6.45 Get the parameters for monitoring the temperature and fan speed of the wide-voltage control box.

New in version C#SDK-V1.1.4: Web-3.8.3

```
/**
 * @brief Get the parameters for monitoring the temperature and fan speed of the wide voltage control box.
 * @param [out] enable 0-not enable monitoring; 1-enable monitoring
 * @param [out] period Monitoring period (s), range 1-100.
 * @return Error code
 */
int GetWideBoxTempFanMonitorParam(ref int enable, ref int period);
```

2.2.6.46 Code example

New in version C#SDK-V1.1.4: Web-3.8.3

```
private void button46_Click(object sender, EventArgs e)
{
    var pkg = new ROBOT_STATE_PKG();
    robot.SetWideBoxTempFanMonitorParam(1, 2);
    int enable = 0;
    int period = 0;
    robot.GetWideBoxTempFanMonitorParam(ref enable, ref period);
    Console.WriteLine($"GetWideBoxTempFanMonitorParam enable is {enable}   period is {period}");
    for (int i = 0; i < 100; i++)
    {
        robot.GetRobotRealTimeState(ref pkg);
        Console.WriteLine($"robot ctrl box temp is {pkg.wideVoltageCtrlBoxTemp}, fan current is {pkg.wideVoltageCtrlBoxFanVel}");
        Thread.Sleep(100);
    }
    int rtn = robot.SetWideBoxTempFanMonitorParam(0, 2);
    Console.WriteLine($"SetWideBoxTempFanMonitorParam rtn is {rtn}");
    enable = 0;
```

(continues on next page)

(continued from previous page)

```
    period = 0;
    robot.GetWideBoxTempFanMonitorParam(ref enable, ref period);
    Console.WriteLine($"GetWideBoxTempFanMonitorParam enable is {enable}   period is
↪{period}");
    for (int i = 0; i < 100; i++)
    {
        robot.GetRobotRealTimeState(ref pkg);
        Console.WriteLine($" robot ctrl box temp is {pkg.wideVoltageCtrlBoxTemp}, fan␣
↪current is {pkg.wideVoltageCtrlBoxFanVel}");
        Thread.Sleep(100);
    }
}
```

2.2.6.47 Set the focus calibration point

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Set the focus calibration point
* @param [in] pointNum Calibration point number of the focus 1-8
* @param [in] point Coordinates of the calibration point
* @return Error code
*/
int SetFocusCalibPoint(int pointNum, DescPose point);
```

2.2.6.48 Set the focus coordinates

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Set the focus coordinates
* @param [in] pos Focus coordinates XYZ
* @return Error code
*/
int SetFocusPosition(DescTran pos);
```

2.2.6.49 Enable focus following

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Enable focus following
* @param [in] kp Proportional parameter, default 50.0
* @param [in] kpredict Feedforward parameter, default 19.0
* @param [in] aMax Maximum angular acceleration limit, default 1440°/s^2
* @param [in] vMax Maximum angular velocity limit, default 180°/s
* @param [in] type Lock X-axis direction (0 - reference input vector; 1 - horizontal; 2 -
↪ vertical)
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
int FocusStart(double kp, double kpredict, double aMax, double vMax, int type);
```

2.2.6.50 Stop focus following

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Stop focus following
* @return Error code
*/
int FocusEnd();
```

2.2.6.51 Focus Follow Code Example

New in version C#SDK-V1.1.5: Web-3.8.4

```
private void button81_Click(object sender, EventArgs e)
{
    DescPose p1Desc=new DescPose(186.331, 487.913, 209.850, 149.030, 0.688, -114.347);
    JointPos p1Joint = new JointPos(-127.876, -75.341, 115.417, -122.741, -59.820, 74.300);
    DescPose p2Desc = new DescPose(69.721, 535.073, 202.882, -144.406, -14.775, -89.012);
    JointPos p2Joint = new JointPos(-101.780, -69.828, 110.917, -125.740, -127.841, 74.300);
    DescPose p3Desc = new DescPose(146.861, 578.426, 205.598, 175.997, -36.178, -93.437);
    JointPos p3Joint = new JointPos(-112.851, -60.191, 86.566, -80.676, -97.463, 74.300);
    DescPose p4Desc = new DescPose(136.284, 509.876, 225.613, 178.987, 1.372, -100.696);
    JointPos p4Joint = new JointPos(-116.397, -76.281, 113.845, -128.611, -88.654, 74.299);
    DescPose p5Desc = new DescPose(138.395, 505.972, 298.016, 179.134, 2.147, -101.110);
    JointPos p5Joint = new JointPos(-116.814, -82.333, 109.162, -118.662, -88.585, 74.302);
    DescPose p6Desc = new DescPose(105.553, 454.325, 232.017, -179.426, 0.444, -99.952);
    JointPos p6Joint = new JointPos(-115.649, -84.367, 122.447, -128.663, -90.432, 74.303);
    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 100, 0, 0, 0);
    robot.GetForwardKin(p1Joint,ref p1Desc);
    robot.GetForwardKin(p2Joint, ref p2Desc);
    robot.GetForwardKin(p3Joint, ref p3Desc);
    robot.GetForwardKin(p4Joint, ref p4Desc);
    robot.GetForwardKin(p5Joint, ref p5Desc);
    robot.GetForwardKin(p6Joint, ref p6Desc);
    robot.MoveJ(p1Joint, p1Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(1);
    robot.MoveJ(p2Joint, p2Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(2);
    robot.MoveJ(p3Joint, p3Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(3);
```

(continues on next page)

(continued from previous page)

```
    robot.MoveJ(p4Joint, p4Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(4);
    DescPose coordRtn = new DescPose(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
    int rtn = robot.ComputeTcp4(ref coordRtn);
    Console.WriteLine($"4 Point ComputeTool      {rtn} coord is {coordRtn.tran.x} ,{coordRtn.tran.y} ,{coordRtn.tran.z} ,{coordRtn.rpy.rx} ,{coordRtn.rpy.ry} ,{coordRtn.rpy.rz} ");
    robot.SetToolCoord(1, coordRtn, 0, 0, 1, 0);
    robot.GetForwardKin(p1Joint, ref p1Desc);
    robot.GetForwardKin(p2Joint, ref p2Desc);
    robot.GetForwardKin(p3Joint, ref p3Desc);
    robot.SetFocusCalibPoint(1, p1Desc);
    robot.SetFocusCalibPoint(2, p2Desc);
    robot.SetFocusCalibPoint(3, p3Desc);
    DescTran resultPos = new DescTran(0.0, 0.0, 0.0);
    double accuracy = 0.0;
    rtn = robot.ComputeFocusCalib(3, ref resultPos, ref accuracy);
    Console.WriteLine($"ComputeFocusCalib coord is  {rtn},{ resultPos.x} ,{ resultPos.y}, { resultPos.z}, accuracy is {accuracy} ");
    rtn = robot.SetFocusPosition(resultPos);
    robot.GetForwardKin(p5Joint, ref p5Desc);
    robot.GetForwardKin(p6Joint, ref p6Desc);
    robot.MoveL(p5Joint, p5Desc, 1, 0, 10, 100, 100, -1, 0, exaxisPos, 0, 1, offdese);
    robot.MoveL(p6Joint, p6Desc, 1, 0, 10, 100, 100, -1, 0, exaxisPos, 0, 1, offdese);
    robot.FocusStart(50, 19, 710, 90, 0);
    robot.MoveL(p5Joint, p5Desc, 1, 0, 10, 100, 100, -1, 0, exaxisPos, 0, 1, offdese);
    robot.MoveL(p6Joint, p6Desc, 1, 0, 10, 100, 100, -1, 0, exaxisPos, 0, 1, offdese);
    robot.FocusEnd();
}
```

2.2.6.52 Enable joint torque sensor sensitivity calibration function

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Enable joint torque sensor sensitivity calibration function
* @param [in] status 0-Disable1-Enable
* @return Error code
*/
public int JointSensitivityEnable(int status);
```

2.2.6.53 Sensitivity data acquisition of joint torque sensors

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Sensitivity data acquisition of joint torque sensors
* @return Error code
*/
public int JointSensitivityCollect();
```

2.2.6.54 Get the sensitivity calibration results of the joint torque sensor

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Get joint torque sensor sensitivity calibration results
* @param [out] calibResult j1~j6 joint sensitivity [0-1]
* @param [out] linearity j1~j6 joint linearity [0-1]
* @return Error code
*/
public int JointSensitivityCalibration(double calibResult[6], double linearity[6]);
```

2.2.6.55 Get Joint Torque Sensor Hysteresis Error

```
/**
* @brief Get joint torque sensor hysteresis error
* @param [out] hysteresisError j1~j6 joint hysteresis error
* @return Error code
*/
public int JointHysteresisError(ref double[] hysteresisError);
```

2.2.6.56 Get Joint Torque Sensor Repeatability

```
/**
* @brief Get joint torque sensor repeatability
* @param [out] repeatability j1~j6 joint torque sensor repeatability
* @return Error code
*/
public int JointRepeatability(ref double[] repeatability);
```

2.2.6.57 Set Joint Force Sensor Parameters

```
/**
* @brief Set joint force sensor parameters
* @param [in] M J1-J6 mass coefficient [0.001 ~ 10]
* @param [in] B J1-J6 damping coefficient [0.001 ~ 10]
* @param [in] K J1-J6 stiffness coefficient [0.001 ~ 10]
* @param [in] threshold Force control threshold, Nm
* @param [in] sensitivity Sensitivity, Nm/V, [0 ~ 10]
* @param [in] setZeroFlag Function enable flag; 0-Off; 1-On; 2-Record zero point at position 1; 3-Record zero point at position 2
* @return Error code
*/
public int SetAdmittanceParams(double[] M, double[] B, double[] K, double[] threshold, double[] sensitivity, int setZeroFlag);
```

2.2.6.58 Joint torque sensor sensitivity automatic calibration Code Example

New in version C#SDK-V1.1.9: Web-3.8.7

```
public int TestSensitivityCalib()
{
    int rtn;
    rtn = robot.JointSensitivityEnable(0);
    rtn = robot.JointSensitivityEnable(1);
    Console.WriteLine($"JointSensitivityEnable rtn is {rtn}");

    JointPos curJPos = new JointPos(0, 0, 0, 0, 0, 0);
    robot.GetActualJointPosDegree(0, ref curJPos);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    JointPos[] jointPoses = new JointPos[]
    {
        new JointPos(curJPos.jPos[0], 0, 0, -90, 0.02, curJPos.jPos[5]),
        new JointPos(curJPos.jPos[0], -30, 0, -90, 0.02, curJPos.jPos[5]),
        new JointPos(curJPos.jPos[0], -60, 0, -90, 0.02, curJPos.jPos[5]),
        new JointPos(curJPos.jPos[0], -90, 0, -90, 0.02, curJPos.jPos[5]),
        new JointPos(curJPos.jPos[0], -120, 0, -90, 0.02, curJPos.jPos[5]),
        new JointPos(curJPos.jPos[0], -150, 0, -90, 0.02, curJPos.jPos[5]),
        new JointPos(curJPos.jPos[0], -180, 0, -90, 0.02, curJPos.jPos[5])
    };
    for (int i = 0; i < jointPoses.Length; i++)
    {
        DescPose descPos = new DescPose(0, 0, 0, 0, 0, 0);
        robot.GetForwardKin(jointPoses[i], ref descPos);
        robot.MoveJ(jointPoses[i], descPos, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);

        Thread.Sleep(i == 0 ? 200 : 100);
        rtn = robot.JointSensitivityCollect();
        Console.WriteLine($"JointSensitivityCollect {i + 1} rtn is {rtn}");
        Thread.Sleep(100);
```

(continues on next page)

(continued from previous page)

```
    }

    for (int i = jointPoses.Length - 2; i >= 0; i--)
    {
        DescPose descPos = new DescPose();
        robot.GetForwardKin(jointPoses[i], ref descPos);
        robot.MoveJ(jointPoses[i], descPos, 0, 0, 100, 100, 100, epos, -1, 0, offset_
↪pos);
        Thread.Sleep(100);
        rtn = robot.JointSensitivityCollect();
        Console.WriteLine($"JointSensitivityCollect {jointPoses.Length + (jointPoses.
↪Length - 1 - i)} rtn is {rtn}");
        Thread.Sleep(100);
    }
    double[] calibResult = new double[6];
    double[] linearity = new double[6];
    rtn = robot.JointSensitivityCalibration(ref calibResult, ref linearity);
    Console.WriteLine($"JointSensitivityCalibration rtn is {rtn}");
    rtn = robot.JointSensitivityEnable(0);
    Console.WriteLine($"JointSensitivityEnable rtn is {rtn}");
    Console.WriteLine($"jointSensor Calib result is {calibResult[0]:F6}
↪{calibResult[1]:F6} {calibResult[2]:F6} {calibResult[3]:F6} {calibResult[4]:F6}
↪{calibResult[5]:F6}");
    Console.WriteLine($"jointSensor linearity is {linearity[0]:F6} {linearity[1]:F6}
↪{linearity[2]:F6} {linearity[3]:F6} {linearity[4]:F6} {linearity[5]:F6}");
    double[] hysteresisError = new double[6];
    rtn = robot.JointHysteresisError(ref hysteresisError);
    Console.WriteLine($"JointHysteresisError result is {hysteresisError[0]:F6}
↪{hysteresisError[1]:F6} {hysteresisError[2]:F6} {hysteresisError[3]:F6}
↪{hysteresisError[4]:F6} {hysteresisError[5]:F6}");
    double[] repeatability = new double[6];
    rtn = robot.JointRepeatability(ref repeatability);
    Console.WriteLine($"JointRepeatability result is {repeatability[0]:F6}
↪{repeatability[1]:F6} {repeatability[2]:F6} {repeatability[3]:F6} {repeatability[4]:F6}
↪ {repeatability[5]:F6}");
    double[] M = new double[6] { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    double[] B = new double[6] { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    double[] K = new double[6] { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
    double[] threshold = new double[6] { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    int setZeroFlag = 1;
    rtn = robot.SetAdmittanceParams(M, B, K, threshold, calibResult, setZeroFlag);
    Console.WriteLine($"SetAdmittanceParams rtn is {rtn}");
    robot.CloseRPC();
    return 0;
}
```

2.2.6.59 Get the number of 8 slave port error frames of the robot

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Get the number of 8 slave port error frames of the robot
* @param [out] inRecvErr Input receiving error frames
* @param [out] inCRCErr Input CRC error frames
* @param [out] inTransmitErr Input transmit error frames
* @param [out] inLinkErr Input link error frames
* @param [out] outRecvErr Output receiving error frames
* @param [out] outCRCErr Output CRC error frames
* @param [out] outTransmitErr Output transmit error frames
* @param [out] outLinkErr Output link error frames
* @return Error code
*/
public int GetSlavePortErrCounter(ref int[] inRecvErr,ref int[] inCRCErr,ref int[]␣
↪inTransmitErr,ref int[] inLinkErr,ref int[] outRecvErr,ref int[] outCRCErr,ref int[]␣
↪outTransmitErr,ref int[] outLinkErr);
```

2.2.6.60 Clear the slave port error num

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Clear the slave port error num
* @param [in] slaveID slave id 0~7
* @return Error code
*/
public int SlavePortErrCounterClear(int slaveID);
```

2.2.6.61 Gets the slave port error frame code example

New in version C#SDK-V1.1.9: Web-3.8.7

```
public void TestSlavePortErr()
{
    int[] inRecvErr = new int[8];
    int[] inCRCErr = new int[8];
    int[] inTransmitErr = new int[8];
    int[] inLinkErr = new int[8];
    int[] outRecvErr = new int[8];
    int[] outCRCErr = new int[8];
    int[] outTransmitErr = new int[8];
    int[] outLinkErr = new int[8];

    robot.GetSlavePortErrCounter(ref inRecvErr, ref inCRCErr, ref inTransmitErr, ref␣
↪inLinkErr,
        ref outRecvErr, ref outCRCErr, ref outTransmitErr, ref outLinkErr);

    for (int i = 0; i < 8; i++)
```

(continues on next page)

(continued from previous page)

```
    {
        if (inRecvErr[i] != 0)
        {
            Console.WriteLine($"inRecvErr {i} is {inRecvErr[i]}");
        }

        if (inCRCErr[i] != 0)
        {
            Console.WriteLine($"inCRCErr {i} is {inCRCErr[i]}");
        }

        if (inTransmitErr[i] != 0)
        {
            Console.WriteLine($"inTransmitErr {i} is {inTransmitErr[i]}");
        }

        if (inLinkErr[i] != 0)
        {
            Console.WriteLine($"inLinkErr {i} is {inLinkErr[i]}");
        }

        if (outRecvErr[i] != 0)
        {
            Console.WriteLine($"outRecvErr {i} is {outRecvErr[i]}");
        }

        if (outCRCErr[i] != 0)
        {
            Console.WriteLine($"outCRCErr {i} is {outCRCErr[i]}");
        }

        if (outTransmitErr[i] != 0)
        {
            Console.WriteLine($"outTransmitErr {i} is {outTransmitErr[i]}");
        }

        if (outLinkErr[i] != 0)
        {
            Console.WriteLine($"outLinkErr {i} is {outLinkErr[i]}");
        }
    }
    Console.WriteLine("others has no err!");

    for (int i = 0; i < 8; i++)
    {
        robot.SlavePortErrCounterClear(i);
    }

    robot.CloseRPC();
}
```

2.2.6.62 Set the feedforward coefficients of the velocities of each axis

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Set the feedforward coefficients of the velocities of each axis
* @param [in] radio feedforward coefficients of the velocities of each axis
* @return Error code
*/
public int SetVelFeedForwardRatio(double radio[6]);
```

2.2.6.63 Get the feedforward coefficients of the velocities of each axis

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Get the feedforward coefficients of the velocities of each axis
* @param [out] radio feedforward coefficients of the velocities of each axis
* @return Error code
*/
public int GetVelFeedForwardRatio(ref double radio[6]);
```

2.2.6.64 Robot velocity feedforward coefficient code example

New in version C#SDK-V1.1.9: Web-3.8.7

```
public void TestVelFeedForwardRatio()
{

    double[] setRadio = new double[6] { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    robot.SetVelFeedForwardRatio(setRadio);
    double[] getRadio = new double[6] { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
    robot.GetVelFeedForwardRatio(ref getRadio);
    Console.WriteLine($" {getRadio[0]:F6} {getRadio[1]:F6} {getRadio[2]:F6}
↪{getRadio[3]:F6} {getRadio[4]:F6} {getRadio[5]:F6}");
}
```

2.2.6.65 Photoelectric Sensor TCP Calibration - Compute Tool RPY

```
/**
* @brief Photoelectric Sensor TCP Calibration - Compute Tool RPY
* @param [in] Btool Robot Cartesian position
* @param [in] Etool Current tool coordinate values
* @param [in] sensor Current sensor coordinate values (not yet available)
* @param [in] radius Circular motion radius in mm (not yet available)
* @param [in] dz Movement distance along the negative Z-axis of the base coordinate␣
↪system; when dz = 10000, the function directly returns tool RPY
* @param [out] TCPRPY Tool RPY values
* @return Error code
*/
```

(continues on next page)

(continued from previous page)

```
public int TCPComputeRPY(DescPose Btool, DescPose Etool, DescPose sensor, double radius,␣
↪double dz, out Rpy TCPRPY);
```

2.2.6.66 Photoelectric Sensor TCP Calibration - Compute Tool XYZ

```
/**
* @brief Photoelectric Sensor TCP Calibration - Compute Tool XYZ
* @param [in] select 0-Compute tool TCP; 1-Compute sensor origin; 2-Compute sensor␣
↪orientation; 3-Directly return tool TCP; 4-Record current workpiece coordinate system␣
↪and tool coordinate system
* @param [in] originDirection 0-X direction; 1-Y direction; 2-Z direction
* @param [in] pos1 Robot Cartesian position 1
* @param [in] pos2 Robot Cartesian position 2
* @param [in] pos3 Robot Cartesian position 3
* @param [in] pos4 Robot Cartesian position 4
* @param [out] TCP Tool XYZ values
* @return Error code
*/
public int TCPComputeXYZ(int select, double originDirection, DescTran pos1, DescTran␣
↪pos2,DescTran pos3, DescTran pos4, out DescTran TCP);
```

2.2.6.67 Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position

```
/**
* @brief Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position
* @return Error code
*/
public int TCPRecordFlangePosStart();
```

2.2.6.68 Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position

```
/**
* @brief Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position
* @return Error code
*/
public int TCPRecordFlangePosEnd();
```

2.2.6.69 Photoelectric Sensor TCP Calibration - Get Tool Center Point Position

```
/**
* @brief Photoelectric Sensor TCP Calibration - Get Tool Center Point Position
* @param [out] TCP Tool center point position (x, y, z)
* @return Error code
*/
public int TCPGetRecordFlangePos(out DescTran TCP);
```

2.2.6.70 Photoelectric Sensor TCP Calibration

```
/**
* @brief Photoelectric Sensor TCP Calibration
* @param [in] luaPath Automatic calibration Lua program path: For QX version robots - "/fruser/FR_CalibrateTheToolTcp.lua"; For LA version robots - "/usr/local/etc/controller/lua/FR_CalibrateTheToolTcp.lua"
* @param [in] offsetX Teaching point offset (x, y, z) in mm
* @param [out] TCP Calibrated tool coordinate system (x, y, z, rx, ry, rz)
* @return Error code
*/
public int PhotoelectricSensorTCPCalibration(string luaPath, DescTran offset, out DescPose TCP);
```

2.2.6.71 Photoelectric Sensor TCP Calibration Code Example

```
public void TestPhotoelectricSensorTCPCalib()
{
    ROBOT_STATE_PKG pkg =new ROBOT_STATE_PKG();
    DescTran offset = new DescTran( 10.0, 10.0, 3.0 );
    DescPose TCP = new DescPose();
    int rtn = robot.PhotoelectricSensorTCPCalibration("/fruser/FR_CalibrateTheToolTcp.lua", offset, out TCP);
    Console.WriteLine($"PhotoelectricSensorTCPCalibration : {rtn}");
    Console.WriteLine($"Tool TCP Coordinate: X={TCP.tran.x:F3}, Y={TCP.tran.y:F3}, Z={TCP.tran.z:F3}");
    Console.WriteLine($"Tool RPY Orientation: RX={TCP.rpy.rx:F3}, RY={TCP.rpy.ry:F3}, RZ={TCP.rpy.rz:F3}");
}
```

2.2.7 Robot Safety Settings

2.2.7.1 Set collision level

```
/**
* @brief Set collision level
* @param  [in]  mode  0-level, 1-percentage
* @param  [in]  level Collision threshold, level corresponds to range [], percentage corresponds to range [0~1]
* @param  [in]  config 0-do not update configuration file, 1-update configuration file
* @return  Error code
*/
int SetAnticollision(int mode, double[] level, int config);
```

2.2.7.2 Set collision post-strategy

```
/**
* @brief  Set collision post-strategy
* @param  [in] strategy  0 - Pause on error; 1 - Continue running; 2 - Stop on error; 3 - Torque mode; 4 - Oscillation response mode; 5 - Collision rebound mode
* @param  [in] safeTime  safe stop time [1000 - 2000]ms
* @param  [in] safeDistance  Safe stop distance [1-150] mm
* @param  [in] safeVel  TCP safe stop speed [50-250] mm/s
* @param  [in] safetyMargin  j1-j6 safety factor [1-10]
* @return Error code
*/
int SetCollisionStrategy(int strategy, int safeTime, int safeDistance, int safeVel,int[] safetyMargin);
```

2.2.7.3 Custom collision detection threshold function start

```
/**
* @brief  Custom collision detection threshold function starts, sets the collision detection threshold for the joint end and TCP end
* @param  [in] flag 1-only joint detection enabled; 2-only TCP detection enabled; 3-joint and TCP detection enabled simultaneously
* @param  [in] jointDetectionThreshould Joint collision detection threshold j1-j6
* @param  [in] tcpDetectionThreshould TCP collision detection threshold, xyzabc
* @param  [in] block 0-non-blocking; 1-blocking
* @return  Error code
*/
int CustomCollisionDetectionStart(int flag, double[] jointDetectionThreshould, double[] tcpDetectionThreshould, int block);
```

2.2.7.4 Custom collision detection threshold function disabled

```
/**
* @brief Custom collision detection threshold function disabled
* @return Error code
*/
int CustomCollisionDetectionEnd();
```

2.2.7.5 Robot collision level settings code example

```
private void button24_Click(object sender, EventArgs e)
{
    int mode = 0;
    int config = 1;
    double[] level1 = { 1.0f, 2.0f, 3.0f, 4.0f, 5.0f, 6.0f };
    double[] level2 = { 50.0f, 20.0f, 30.0f, 40.0f, 50.0f, 60.0f };

    int rtn = robot.SetAnticollision(mode, level1, config);
```

(continues on next page)

(continued from previous page)

```
    Console.WriteLine($"SetAnticollision mode 0 rtn is {rtn}");
    mode = 1;
    rtn = robot.SetAnticollision(mode, level2, config);
    Console.WriteLine($"SetAnticollision mode 1 rtn is {rtn}");

    JointPos p1Joint = new JointPos(-11.904f, -99.669f, 117.473f, -108.616f, -91.726f, 74.256f);
    JointPos p2Joint = new JointPos(-45.615f, -106.172f, 124.296f, -107.151f, -91.282f, 74.255f);

    DescPose p1Desc = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f, 3.833f);
    DescPose p2Desc = new DescPose(-321.222f, 185.189f, 335.520f, -179.030f, -1.284f, -29.869f);

    ExaxisPos exaxisPos = new ExaxisPos(0.0f, 0.0f, 0.0f, 0.0f);
    DescPose offdese = new DescPose(0.0f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f);
    robot.MoveL( p2Joint,  p2Desc, 0, 0, 100, 100, 100, 2,  exaxisPos, 0, 0,  offdese);
    robot.ResetAllError();
    int[] safety = { 5, 5, 5, 5, 5, 5 };
    rtn = robot.SetCollisionStrategy(3, 1000, 150, 250, safety);
    Console.WriteLine($"SetCollisionStrategy rtn is {rtn}");

    double[] jointDetectionThreshould = { 0.1, 0.1, 0.1, 0.1, 0.1, 0.1 };
    double[] tcpDetectionThreshould = { 60, 60, 60, 60, 60, 60 };
    rtn = robot.CustomCollisionDetectionStart(3, jointDetectionThreshould, tcpDetectionThreshould, 0);
    Console.WriteLine($"CustomCollisionDetectionStart rtn is {rtn}");

    robot.MoveL( p1Joint,  p1Desc, 0, 0, 100, 100, 100, -1,  exaxisPos, 0, 0,  offdese);
    robot.MoveL( p2Joint,  p2Desc, 0, 0, 100, 100, 100, -1,  exaxisPos, 0, 0,  offdese);
    rtn = robot.CustomCollisionDetectionEnd();
    Console.WriteLine($"CustomCollisionDetectionEnd rtn is {rtn}");
}
```

2.2.7.6 Set positive limit

```
/**
* @brief  Set positive limit
* @param  [in] limit Six joint positions, units in deg
* @return  Error code
*/
int SetLimitPositive(double[] limit);
```

2.2.7.7 Set negative limit

```
/**
* @brief  Set negative limit
* @param  [in] limit Six joint positions, units in deg
* @return  Error code
*/
int SetLimitNegative(double[] limit);
```

2.2.7.8 Get joint soft limit angles

```
/**
* @brief  Get joint soft limit angles
* @param  [in] flag 0-block, 1-non-block
* @param  [out] negative  Negative limit angle, in degrees
* @param  [out] positive  Positive limit angle, in degrees
* @return  Error code
*/
int GetJointSoftLimitDeg(byte flag, ref double[] negative, ref double[] positive);
```

2.2.7.9 Robot limit setting code example

```
private void btnRobotSafetySet_Click(object sender, EventArgs e)
{
    double[] plimit = { 170.0f, 80.0f, 150.0f, 80.0f, 170.0f, 160.0f };
    robot.SetLimitPositive(plimit);
    double[] nlimit = { -170.0f, -260.0f, -150.0f, -260.0f, -170.0f, -160.0f };
    robot.SetLimitNegative(nlimit);

    double[] neg_deg = new double[6] {0,0,0,0,0,0 };
    double[] pos_deg = new double[6] { 0, 0, 0, 0, 0, 0 };
    robot.GetJointSoftLimitDeg(0, ref neg_deg,ref pos_deg);
    Console.WriteLine($"neg limit deg:{neg_deg[0]},{neg_deg[1]},{neg_deg[2]},{neg_deg[3]}
↪,{neg_deg[4]},{neg_deg[5]}");
    Console.WriteLine($"pos limit deg:{pos_deg[0]},{pos_deg[1]},{pos_deg[2]},{pos_deg[3]}
↪,{pos_deg[4]},{pos_deg[5]}");
}
```

2.2.7.10 Set robot collision detection method

```
/**
* @brief Set robot collision detection method
* @param  [in] method Collision detection method: 0-current mode; 1-dual encoder; 2-
↪current and dual encoder enabled simultaneously
* @param [in] thresholdMode Collision level threshold mode; 0 - fixed collision level␣
↪threshold mode; 1 - custom collision detection threshold
* @return  Error code
*/
int SetCollisionDetectionMethod(int method, int thresholdMode=0);
```

2.2.7.11 Set collision detection start/stop in static mode

```
/**
* @brief Set collision detection start/stop in static mode
* @param  [in] status 0-disabled; 1-enabled
* @return  Error code
*/
int SetStaticCollisionOnOff(int status);
```

2.2.7.12 Code example for setting the robot collision detection method

```
private void button26_Click(object sender, EventArgs e)
{
    int rtn = robot.SetCollisionDetectionMethod(0, 0);

    rtn = robot.SetStaticCollisionOnOff(1);
    Console.WriteLine($"SetStaticCollisionOnOff On rtn is {rtn}");
    Thread.Sleep(5000);
    rtn = robot.SetStaticCollisionOnOff(0);
    Console.WriteLine($"SetStaticCollisionOnOff Off rtn is {rtn}");
}
```

2.2.7.13 Joint Torque Power Detection

```
/**
* @brief Joint Torque Power Detection
* @param  [in] status 0-off; 1-on
* @param  [in] power Set maximum power (W)
* @return Error code
*/
int SetPowerLimit(int status, double power);
```

2.2.7.14 Joint torque power detection code example

```
private void button26_Click(object sender, EventArgs e)
{
    robot.DragTeachSwitch(1);
    robot.SetPowerLimit(1, 200);
    double[] torques = { 0, 0, 0, 0, 0, 0 };
    robot.GetJointTorques(1, torques);

    int count = 100;
    robot.ServoJTStart();
    int error = 0;
    while (count > 0)
    {
        error = robot.ServoJT(torques, 0.001f);
        count--;
```

(continues on next page)

(continued from previous page)

```
        Thread.Sleep(1);
    }
    error = robot.ServoJTEnd();
    robot.DragTeachSwitch(0);
}
```

2.2.8 Robot Status Check

2.2.8.1 Get current joint position (angle).

```
/**
* @brief Get current joint position (angle).
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] jPos Six joint positions in deg.
* @return Error code.
*/
int GetActualJointPosDegree(byte flag, ref JointPos jPos);
```

2.2.8.2 Get current joint position in degrees of arc.

```
/**
* @brief Get current joint position in radians.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] jPos six joint positions in rad
* @return Error code
*/
int GetActualJointPosRadian(byte flag, ref JointPos jPos);
```

2.2.8.3 Get the joint feedback velocity

```
/**
* @brief Get joint feedback speed -deg/s
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] speed Six joint speeds.
* @return Error code.
*/
int GetActualJointSpeedsDegree(byte flag, ref double[] speed);
```

2.2.8.4 Get joint feedback acceleration

```
/**
* @brief Get the joint feedback acceleration -deg/s^2
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] acc Six joint accelerations.
* @return Error code.
*/
int GetActualJointAccDegree(byte flag, ref double[] acc);
```

2.2.8.5 Get TCP command velocity - joint velocity

```
/**
* @brief Get TCP command speed-combining speed.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] tcp_speed linear_speed
* @param [out] ori_speed Attitude Speed
* @return error code
*/
int GetTargetTCPCompositeSpeed(byte flag, ref double tcp_speed, ref double ori_speed);
```

2.2.8.6 Get TCP feedback speed-composite speed

```
/**
* @brief Get TCP feedback speed-combination speed.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] tcp_speed linear_speed
* @param [out] ori_speed Attitude Speed
* @return error code
*/
int GetActualTCPCompositeSpeed(byte flag, ref double tcp_speed, ref double ori_speed);
```

2.2.8.7 Get TCP command speed-composite speed

```
/**
* @brief Get TCP command speed-split.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] speed [x,y,z,rx,ry,rz] speed
* @return error code
*/
int GetTargetTCPSpeed(byte flag, ref double[] speed);
```

2.2.8.8 Get TCP feedback speed-split speed

```
/**
* @brief Get TCP feedback speed-cents.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] speed [x,y,z,rx,ry,rz] speed
* @return error code
*/
int GetActualTCPSpeed(byte flag, ref double[] speed);
```

2.2.8.9 Get current tool position

```
/**
* @brief Get current tool position
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] desc_pos Tool position.
* @return Error code.
*/
int GetActualTCPPose(byte flag, ref DescPose desc_pos);
```

2.2.8.10 Get the current tool coordinate system number

```
/**
* @brief Get the current tool coordinate system number.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] id tool coordinate system number
* @return Error code.
*/
int GetActualTCPNum(byte flag, ref int id).
```

2.2.8.11 Get the current tool coordinate system number

```
/**
* @brief Get the current workpiece coordinate system number.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] id The workpiece coordinate system number.
* @return Error code.
*/
int GetActualWObjNum(byte flag, ref int id);
```

2.2.8.12 Get the current end flange position

```
/**
* @brief Get the current end flange position.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] desc_pos Flange position.
* @return Error code.
*/
int GetActualToolFlangePose(byte flag, ref DescPose desc_pos);
```

2.2.8.13 Get the current joint torque

```
/**
* @brief Get current joint torque.
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] torques Joint torques.
* @return Error code.
*/
int GetJointTorques(byte flag, float[] torques).
```

2.2.8.14 Get system time

```
/**
* @brief Get the system time.
* @param [out] t_ms unit ms
* @return Error code.
*/
int GetSystemClock(ref double t_ms).
```

2.2.8.15 Queries if the robot movement is complete

```
/**
* @brief Query if robot motion is complete.
* @param [out] state 0-not completed, 1-completed
* @return Error code.
*/
int GetRobotMotionDone(ref byte state).
```

2.2.8.16 Queries the length of the robot motion queue cache

```
/**
* @brief Query the robot motion queue cache length.
* @param [out] len Cache length.
* @return Error code.
*/
int GetMotionQueueLength(ref int len).
```

2.2.8.17 Get the robot emergency stop state

```
/**
* @brief Get the state of the robot's emergency stop.
* @param [out] state Emergency stop state, 0-non-emergency stop, 1-emergency stop.
* @return Error code.
*/
int GetRobotEmergencyStopState(ref byte state).
```

2.2.8.18 Get the state of the communication between the SDK and the robot

```
/**
* @brief Get the communication state of the SDK and the robot.
* @param [out] state communication state, 0-normal communication, 1-abnormal␣
↪communication.
*/
int GetSDKComState(ref int state).
```

2.2.8.19 Get safety stop signal

```
/**
* @brief Get the safe stop signal.
* @param [out] si0_state Safe stop signal SI0, 0-invalid, 1-valid
* @param [out] si1_state safe stop signal SI1, 0-invalid, 1-valid
*/
int GetSafetyStopState(ref byte si0_state, ref byte si1_state);
```

2.2.8.20 Get robot joint actuator temperature (°C)

```
/**
* @brief Get the robot's articulated actuator temperature (°C).
* @return Error code
*/
int GetJointDriverTemperature(double[] temperature).
```

2.2.8.21 Get robot joint driver torque(Nm).

```
/**
* @brief Get robot joint actuator torque (Nm).
* @return Error code
*/
int GetJointDriverTorque(double torque[]).
```

2.2.8.22 Get robot real-time status structure

```
/**
* @brief Get the robot realtime state structure.
* @param [out] pkg Robot real-time status structure.
* @return Error code
*/
int GetRobotRealTimeState(ref ROBOT_STATE_PKG pkg);
```

2.2.8.23 Sample Robot Status Query Code

```
private void button29_Click(object sender, EventArgs e)
{
    ROBOT_STATE_PKG pkg = new ROBOT_STATE_PKG();
    double yangle = 0, zangle = 0;
    robot.GetRobotInstallAngle(ref yangle, ref zangle);
    Console.WriteLine($"yangle:{yangle},zangle:{zangle}");

    JointPos j_deg = new JointPos(0,0,0,0,0,0);
    robot.GetActualJointPosDegree(0, ref j_deg);
    Console.WriteLine($"joint pos deg:{j_deg.jPos[0]},{j_deg.jPos[1]},{j_deg.jPos[2]},{j_
↪deg.jPos[3]},{j_deg.jPos[4]},{j_deg.jPos[5]}");

    double[] jointSpeed = new double[6];
    robot.GetActualJointSpeedsDegree(0, ref jointSpeed);
    Console.WriteLine($"joint speeds deg:{jointSpeed[0]},{jointSpeed[1]},{jointSpeed[2]},
↪{jointSpeed[3]},{jointSpeed[4]},{jointSpeed[5]}");

    double[] jointAcc = new double[6];
    robot.GetActualJointAccDegree(0, ref jointAcc);
    Console.WriteLine($"joint acc deg:{jointAcc[0]},{jointAcc[1]},{jointAcc[2]},
↪{jointAcc[3]},{jointAcc[4]},{jointAcc[5]}");

    double tcp_speed = 0, ori_speed = 0;
    robot.GetTargetTCPCompositeSpeed(0, ref tcp_speed, ref ori_speed);
    Console.WriteLine($"GetTargetTCPCompositeSpeed tcp {tcp_speed}; ori {ori_speed}");

    robot.GetActualTCPCompositeSpeed(0, ref tcp_speed, ref ori_speed);
    Console.WriteLine($"GetActualTCPCompositeSpeed tcp {tcp_speed}; ori {ori_speed}");

    double[] targetSpeed = new double[6];
    robot.GetTargetTCPSpeed(0,ref targetSpeed);
    Console.WriteLine($"GetTargetTCPSpeed {targetSpeed[0]},{targetSpeed[1]},
↪{targetSpeed[2]},{targetSpeed[3]},{targetSpeed[4]},{targetSpeed[5]}");

    double[] actualSpeed = new double[6];
    robot.GetActualTCPSpeed(0, ref actualSpeed);
    Console.WriteLine($"GetTargetTCPSpeed {actualSpeed[0]},{actualSpeed[1]},
↪{actualSpeed[2]},{actualSpeed[3]},{actualSpeed[4]},{actualSpeed[5]}");

    DescPose tcp = new DescPose(0, 0, 0, 0, 0, 0);
```

(continues on next page)

(continued from previous page)

```
    robot.GetActualTCPPose(0, ref tcp);
    Console.WriteLine($"tcp pose:{tcp.tran.x},{tcp.tran.y},{tcp.tran.z},{tcp.rpy.rx},
↪{tcp.rpy.ry},{tcp.rpy.rz}");

    DescPose flange = new DescPose(0, 0, 0, 0, 0, 0);
    robot.GetActualToolFlangePose(0, ref flange);
    Console.WriteLine($"flange pose:{flange.tran.x},{flange.tran.y},{flange.tran.z},
↪{flange.rpy.rx},{flange.rpy.ry},{flange.rpy.rz}");

    int id = 0;
    robot.GetActualTCPNum(0, ref id);
    Console.WriteLine($"tcp num:{id}");

    robot.GetActualWObjNum(0, ref id);
    Console.WriteLine($"wobj num:{id}");

    double[] jtorque = new double[6];
    robot.GetJointTorques(0, jtorque);
    Console.WriteLine($"torques:{jtorque[0]},{jtorque[1]},{jtorque[2]},{jtorque[3]},
↪{jtorque[4]},{jtorque[5]}");

    double t_ms = 0;
    robot.GetSystemClock(ref t_ms);
    Console.WriteLine($"system clock:{t_ms}");

    int config = 0;
    robot.GetRobotCurJointsConfig(ref config);
    Console.WriteLine($"joint config:{config}");

    byte motionDone = 0;
    robot.GetRobotMotionDone(ref motionDone);
    Console.WriteLine($"GetRobotMotionDone :{motionDone}");

    int len = 0;
    robot.GetMotionQueueLength(ref len);
    Console.WriteLine($"GetMotionQueueLength :{len}");

    byte emergState = 0;
    robot.GetRobotEmergencyStopState(ref emergState);
    Console.WriteLine($"GetRobotEmergencyStopState :{emergState}");

    int comstate = 0;
    robot.GetSDKComState(ref comstate);
    Console.WriteLine($"GetSDKComState :{comstate}");

    byte si0_state = 0, si1_state = 0;
    robot.GetSafetyStopState(ref si0_state, ref si1_state);
    Console.WriteLine($"GetSafetyStopState :{si0_state} {si1_state}");

    double[] temp = new double[6];
    robot.GetJointDriverTemperature(temp);
    Console.WriteLine($"Temperature:{temp[0]},{temp[1]},{temp[2]},{temp[3]},{temp[4]},
```

(continues on next page)

(continued from previous page)

```
↪{temp[5]}");

    double[] torque = new double[6];
    robot.GetJointDriverTorque(torque);
    Console.WriteLine($"torque:{torque[0]},{torque[1]},{torque[2]},{torque[3]},
↪{torque[4]},{torque[5]}");

    robot.GetRobotRealTimeState(ref pkg);
}
```

2.2.8.24 Inverse kinematics solution

```
/**
* @brief Inverse kinematics solution
* @param [in] type 0-absolute position (base coordinate system), 1-incremental position␣
↪(base coordinate system), 2-incremental position (instrumental coordinate system)
* @param [in] desc_pos Cartesian Position
* @param [in] config joint_space_configuration, [-1] - solve with reference to the␣
↪current joint position, [0~7] - solve according to a specific joint space configuration
* @param [out] joint_pos Joint position.
* @return Error code
*/
int GetInverseKin(int type, DescPose desc_pos, int config, ref JointPos joint_pos);
```

2.2.8.25 Inverse kinematics solution (reference position)

```
/**
* @brief Inverse kinematics solving, referencing a specified joint position to determine␣
↪if there is a solution.
* @param [in] type 0-absolute position (base coordinate system), 1-incremental position␣
↪(base coordinate system), 2-incremental position (instrumental coordinate system)
* @param [in] desc_pos Cartesian Position
* @param [in] joint_pos_ref reference joint position
* @param [out] result 0-no solution, 1-with solution
* @return Error code
*/
int GetInverseKinRef(int posMode, DescPose desc_pos, JointPos joint_pos_ref, ref␣
↪JointPos joint_pos);
```

2.2.8.26 Inverse Kinematics Solution, Cartesian Space Includes Extended Axis Position

```
/**
* @brief Inverse kinematics solution, Cartesian space includes extended axis position
* @param [in] type 0-Absolute pose (base coordinate system), 1-Incremental pose (base coordinate system), 2-Incremental pose (tool coordinate system)
* @param [in] desc_pos Cartesian pose
* @param [in] exaxis Extended axis position
* @param [in] tool Tool number
* @param [in] workPiece Workpiece number
* @param [out] joint_pos Joint position
* @return Error code
*/
public int GetInverseKinExaxis(int type, DescPose desc_pos, ExaxisPos exaxis, int tool, int workPiece, ref JointPos joint_pos);
```

2.2.8.27 Example Code for Inverse Kinematics Solution Including Extended Axis Position

```
public void TestInverseKinExaxis()
{
    ROBOT_STATE_PKG pkg = new ROBOT_STATE_PKG();


    DescPose desc = new DescPose(99.957f, -0.002f, 29.994f, -176.569f, -6.757f, -167.462f);
    ExaxisPos exaxis = new ExaxisPos(100.0f, 0.0f, 0.0f, 0.0f);
    JointPos jointPos = new JointPos(0,0,0,0,0,0);
    DescPose offsetPos = new DescPose(0.0f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f);
    int rtn;
    robot.GetRobotRealTimeState(ref pkg);
    int toolnum = pkg.tool;
    int workPcsNum = pkg.user;

    robot.GetInverseKinExaxis(0, desc, exaxis, toolnum, workPcsNum, ref jointPos);
    Console.WriteLine($"GetInverseKinExaxis joint is {jointPos.jPos[0]}, {jointPos.jPos[1]}, {jointPos.jPos[2]}, {jointPos.jPos[3]}, {jointPos.jPos[4]}, {jointPos.jPos[5]}");

    robot.ExtAxisMove(exaxis, 100, -1);

    int blendMode = 0;
    int velAccMode = 0;
    float oacc = 100.0f;
    byte flag = 0;
    robot.MoveJ(jointPos, desc, toolnum, workPcsNum, (float)100.0, (float)100.0, (float)100.0, exaxis, -1, 0, offsetPos);
}
```

2.2.8.28 Example Code for Inverse Kinematics Solution Including Extended Axis Position

```
/**
* @brief Inverse kinematics solution, determine if there is a solution for the specified reference joint position.
* @param [in] posMode 0 absolute position, 1 relative position-base coordinate system, 2 relative position-tool coordinate system.
* @param [in] desc_pos Cartesian position.
* @param [in] joint_pos_ref Reference joint position.
* @param [out] hasResult 0-no solution, 1-has solution
* @return Error code
*/
int GetInverseKinHasSolution(int posMode, DescPose desc_pos, JointPos joint_pos_ref, ref bool hasResult);
```

2.2.8.29 Positive kinematics solution

```
/**
* @brief Positive kinematics solution.
* @param [in] joint_pos joint position
* @param [out] desc_pos Cartesian position
* @return Error code.
*/
int GetForwardKin(JointPos joint_pos, ref DescPose desc_pos);
```

2.2.8.30 Robot Forward and Reverse Kinematics Calculation Code Example

```
private void button30_Click(object sender, EventArgs e)
{
    JointPos j1 = new JointPos(-11.904f, -99.669f, 117.473f, -108.616f, -91.726f, 74.256f);
    DescPose desc_pos1 = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f, 3.833f);
    JointPos inverseRtn = new JointPos(0, 0, 0, 0, 0, 0, 0, 0);
    robot.GetInverseKin(0, desc_pos1, -1, ref inverseRtn);
    Console.WriteLine($"dcs1 GetInverseKin rtn is {inverseRtn.jPos[0]} {inverseRtn.jPos[1]} {inverseRtn.jPos[2]} {inverseRtn.jPos[3]} { inverseRtn.jPos[4]} {inverseRtn.jPos[5]}");
    robot.GetInverseKinRef(0, desc_pos1, j1, ref inverseRtn);
    Console.WriteLine($"dcs1 GetInverseKinRef rtn is {inverseRtn.jPos[0]} {inverseRtn.jPos[1]} {inverseRtn.jPos[2]} {inverseRtn.jPos[3]} { inverseRtn.jPos[4]} {inverseRtn.jPos[5]}");
    bool hasResut = false;
    robot.GetInverseKinHasSolution(0, desc_pos1, j1, ref hasResut);
    Console.WriteLine($"dcs1 GetInverseKinRef result {hasResut}");
    DescPose forwordResult = new DescPose(0, 0, 0, 0, 0, 0, 0);
    robot.GetForwardKin(j1, ref forwordResult);
    Console.WriteLine($"jpos1 forwordResult rtn is {forwordResult.tran.x} {forwordResult.tran.y} {forwordResult.tran.z} {forwordResult.rpy.rx } {forwordResult.
```

(continues on next page)

(continued from previous page)

```
↪rpy.ry} {forwordResult.rpy.rz}");
}
```

2.2.8.31 Querying Robot Teaching Management Point Data

```
/**
* @brief Query robot instruction management point data.
* @param [in] name Point name.
* @param [out] data point data double[20]{x,y,z,rx,ry,rz,j1,j2,j3,j4,j5,j6,tool, wobj,
↪speed,acc,e1,e2,e3,e4}
* @return Error code
*/
int GetRobotTeachingPoint(string name, ref double[] data);
```

2.2.8.32 Get the robot DH parameter compensation value

```
/**
* @brief Get the robot DH parameter compensation value.
* @param [out] dhCompensation Robot DH parameter compensation value (mm) [cmpstD1,
↪cmpstA2,cmpstA3,cmpstD4,cmpstD5,cmpstD6]
* @return Error code
*/
int GetDHCompensation(ref double[] dhCompensation).
```

2.2.8.33 Get control box SN code

```
/**
* @brief Get control box SN code.
* @param [out] SNCode Control Box SN Code
* @return Error code.
*/
int GetRobotSN(ref string SNCode);
```

2.2.8.34 Query the robot teaching management point data code example

```
private void button31_Click(object sender, EventArgs e)
{
    string name = "A0";
    double[] data = new double[20];
    int rtn = robot.GetRobotTeachingPoint(name, ref data);
    Console.WriteLine(" {0} name is: {1} \n", rtn, name);
    for (int i = 0; i < 20; i++)
    {
        Console.WriteLine("data is: {0} \n", data[i]);
    }

```

(continues on next page)

(continued from previous page)

```
    int que_len = 0;
    rtn = robot.GetMotionQueueLength(ref que_len);
    Console.WriteLine("GetMotionQueueLength rtn is: {0}, queue length is: {1} \n", rtn,␣
↪que_len);

    double[] dh = { 0, 0, 0, 0, 0, 0 };
    int retval = 0;
    retval = robot.GetDHCompensation(ref dh);
    Console.WriteLine($"retval is  {retval}");
    Console.WriteLine($"dh is {dh[0]}, {dh[1]}, {dh[2]}, {dh[3]}, {dh[4]}, {dh[5]}");
    string SN = "";
    robot.GetRobotSN(ref SN);
    Console.WriteLine($"robot SN is  {SN}");
}
```

2.2.8.35 Get Tool Coordinate System by ID

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Get Tool Coordinate System by ID
* @param [in] id Tool coordinate system ID
* @param [out] coord Coordinate system value
* @return Error code
*/
int GetToolCoordWithID(int id,ref DescPose coord)
```

2.2.8.36 Get Work Object Coordinate System by ID

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Get Work Object Coordinate System by ID
* @param [in]  id Work object coordinate system ID
* @param [out] coord Coordinate system value
* @return Error code
*/
public int GetWObjCoordWithID(int id, ref DescPose coord)
```

2.2.8.37 Get External Tool Coordinate System by ID

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Get External Tool Coordinate System by ID
* @param [in]  id External tool coordinate system ID
* @param [out] coord Coordinate system value
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
public int GetExToolCoordWithID(int id, ref DescPose coord)
```

2.2.8.38 Get Extended Axis Coordinate System by ID

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Get Extended Axis Coordinate System by ID
* @param [in]  id Extended axis coordinate system ID
* @param [out] coord Coordinate system value
* @return Error code
*/
public int GetExAxisCoordWithID(int id, ref DescPose coord)
```

2.2.8.39 Get Current Tool Coordinate System

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
 * @brief Get Current Tool Coordinate System
 * @param [out] coord Coordinate system value
 * @return Error code
 */
public int GetCurToolCoord(ref DescPose coord)
```

2.2.8.40 Get Current Work Object Coordinate System

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
 * @brief Get Current Work Object Coordinate System
 * @param [out] coord Coordinate system value
 * @return Error code
 */
public int GetCurWObjCoord(ref DescPose coord)
```

2.2.8.41 Get Current External Tool Coordinate System

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
 * @brief Get Current External Tool Coordinate System
 * @param  [out] coord Coordinate system value
 * @return Error code
 */
public int GetCurExToolCoord(ref DescPose coord)
```

2.2.8.42 Get Current Extended Axis Coordinate System

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
 * @brief Get Current Extended Axis Coordinate System
 * @param [out] coord Coordinate system value
 * @return Error code
 */
public int GetCurExAxisCoord(ref DescPose coord)
```

2.2.8.43 Get Current Extended Axis Coordinate System

New in version C#SDK-V1.1.8: Web-3.8.6

```
public void TestCoordMain()
{
    DescPose t_coord = new DescPose(0, 0, 0, 0, 0, 0);
    t_coord.tran.x = 1.0;
    t_coord.tran.y = 2.0;
    t_coord.tran.z = 300.0;
    t_coord.rpy.rx = 4.0;
    t_coord.rpy.ry = 5.0;
    t_coord.rpy.rz = 6.0;
    int id = 1;
    DescPose toolCoord = new DescPose();
    robot.GetToolCoordWithID(id, ref toolCoord);
    Console.WriteLine($"GetToolCoordWithID {id}, {toolCoord.tran.x} {toolCoord.tran.y} {toolCoord.tran.z} {toolCoord.rpy.rx} {toolCoord.rpy.ry} {toolCoord.rpy.rz}");
    DescPose wobjCoord = new DescPose();
    robot.GetWObjCoordWithID(id, ref wobjCoord);
    Console.WriteLine($"GetWObjCoordWithID {id}, {wobjCoord.tran.x} {wobjCoord.tran.y} {wobjCoord.tran.z} {wobjCoord.rpy.rx} {wobjCoord.rpy.ry} {wobjCoord.rpy.rz}");
    DescPose extoolCoord = new DescPose();
    robot.GetExToolCoordWithID(id, ref extoolCoord);
    Console.WriteLine($"GetExToolCoordWithID {id}, {extoolCoord.tran.x} {extoolCoord.tran.y} {extoolCoord.tran.z} {extoolCoord.rpy.rx} {extoolCoord.rpy.ry} {extoolCoord.rpy.rz}");
    DescPose exAxisCoord = new DescPose();
    robot.GetExAxisCoordWithID(id, ref exAxisCoord);
    Console.WriteLine($"GetExAxisCoordWithID {id}, {exAxisCoord.tran.x} {exAxisCoord.tran.y} {exAxisCoord.tran.z} {exAxisCoord.rpy.rx} {exAxisCoord.rpy.ry} {exAxisCoord.rpy.rz}");
    double weight = 0.0;
    DescTran cog = new DescTran();
    robot.GetTargetPayloadWithID(id, ref weight, ref cog);
    Console.WriteLine($"GetTargetPayloadWithID {id}, {weight} {cog.x} {cog.y} {cog.z}");
    robot.GetCurToolCoord(ref toolCoord);
    Console.WriteLine($"GetCurToolCoord {toolCoord.tran.x} {toolCoord.tran.y} {toolCoord.tran.z} {toolCoord.rpy.rx} {toolCoord.rpy.ry} {toolCoord.rpy.rz}");

    robot.GetCurWObjCoord(ref wobjCoord);
```

(continues on next page)

(continued from previous page)

```
    Console.WriteLine($"GetCurWObjCoord {wobjCoord.tran.x} {wobjCoord.tran.y} {wobjCoord.tran.z} {wobjCoord.rpy.rx} {wobjCoord.rpy.ry} {wobjCoord.rpy.rz}");
    robot.GetCurExToolCoord(ref extoolCoord);
    Console.WriteLine($"GetExToolCoordWithID {extoolCoord.tran.x} {extoolCoord.tran.y} {extoolCoord.tran.z} {extoolCoord.rpy.rx} {extoolCoord.rpy.ry} {extoolCoord.rpy.rz}");
    robot.GetCurExAxisCoord(ref exAxisCoord);
    Console.WriteLine($"GetCurExAxisCoord {exAxisCoord.tran.x} {exAxisCoord.tran.y} {exAxisCoord.tran.z} {exAxisCoord.rpy.rx} {exAxisCoord.rpy.ry} {exAxisCoord.rpy.rz}");
    double weightT = 0.0f;
    DescTran cogT = new DescTran();
    robot.GetTargetPayload(0, ref weightT);
    robot.GetTargetPayloadCog(0, ref cogT);
    Console.WriteLine($"GetTargetPayload {weightT} {cogT.x} {cogT.y} {cogT.z}");
    DescPose coordSet = new DescPose(0, 10, 2, 3, 4, 5);
    robot.SetToolCoord(2, coordSet, 0, 0, 1, 0);
    DescPose Coordset0 = new DescPose(0, 0, 0, 0, 0, 0);
    DescPose Coordset = new DescPose(1, 2, 3, 4, 5, 6);
    DescPose etcp = new DescPose(10, 20, 30, 40, 50, 60);
    DescPose etctool = new DescPose(0.1, 0.2, 0.3, 0.4, 0.5, 0.6);
    robot.SetToolCoord(id, Coordset, 0, 0, 1, 0);
    Thread.Sleep(100);
    robot.SetWObjCoord(id, Coordset, 0);
    Thread.Sleep(100);
    robot.ExtAxisActiveECoordSys(id, 1, Coordset, 0);
    Thread.Sleep(100);
    robot.SetExToolCoord(id, etcp, etctool);
    Thread.Sleep(100);
    robot.SetLoadWeight(id, (float)1.5);
    //Thread.Sleep(500);
    robot.SetLoadCoord(id, cog);
    Thread.Sleep(100);
}
```

2.2.9 Robot trajectory reproduction

2.2.9.1 Set the parameters for TPD trajectory recording

```
/ * *
* @brief Set the TPD trajectory recording parameters
* @param [in] type records data type, 1- joint position
* @param [in] name Trajectory file name
* @param [in] period_ms data sampling period, fixed value 2ms, 4ms or 8ms
* @param [in] di_choose DI selection,bit0 to bit7 correspond to control box DI0 to DI7, bit8 to bit9 correspond to end DI0 to DI1, 0- no selection, 1- selection
* @param [in] do_choose DO selection,bit0 to bit7 correspond to control box DO0 to DO7, bit8 to bit9 correspond to end DO0 to DO1, 0- do not select, 1- select
* @return error code
* /
int SetTPDParam(int type, string name, int period_ms, UInt16 di_choose, UInt16 do_choose);
```

2.2.9.2 Start TPD trajectory recording

```
/ * *
* @brief Starts TPD trajectory recording
* @param [in] type records data type, 1- joint position
* @param [in] name Trajectory file name
* @param [in] period_ms data sampling period, fixed value 2ms, 4ms or 8ms
* @param [in] di_choose DI selection,bit0 to bit7 correspond to control box DI0 to DI7,␣
↪bit8 to bit9 correspond to end DI0 to DI1, 0- no selection, 1- selection
* @param [in] do_choose DO selection,bit0 to bit7 correspond to control box DO0 to DO7,␣
↪bit8 to bit9 correspond to end DO0 to DO1, 0- do not select, 1- select
* @return error code
* /
int SetTPDStart(int type, string name, int period_ms, UInt16 di_choose, UInt16 do_
↪choose);
```

2.2.9.3 Stop TPD trajectory recording

```
/ * *
* @brief Stop TPD trajectory recording
* @return error code
* /
int SetWebTPDStop();
```

2.2.9.4 Delete the TPD trajectory record

```
/ * *
* @brief Delete TPD trajectory records
* @param [in] name Trajectory file name
* @return error code
* /
int SetTPDDelete(string name);
```

2.2.9.5 TPD trajectory preloading

```
/ * *
* @brief Trajectory Preloading
* @param [in] name Trajectory file name
* @return error code
* /
int LoadTPD(string name);
```

2.2.9.6 Obtain the starting pose of the TPD trajectory

```
/ * *
* @brief Obtain the starting pose of the trajectory
* @param [in] name Trajectory file name
* @param [out] desc_pose trajectory starting pose
* @return error code
* /
int GetTPDStartPose(string name, ref DescPose desc_pose);
```

2.2.9.7 TPD trajectory reproduction

```
/ * *
* @brief Trajectory Reproduction
* @param [in] name Trajectory file name
* @param [in] blend 0- not smooth, 1- smooth
* @param [in] ovl speed scaling percentage, range [0 to 100]
* @return error code
* /
int MoveTPD(string name, byte blend, float ovl);
```

2.2.9.8 A sample code for robot TPD trajectory recording

```
private void btnTPDMove_Click(object sender, EventArgs e)
{
    int type = 1;
    string name = "tpd2025";
    int period_ms = 4;
    ushort di_choose = 0;
    ushort do_choose = 0;

    robot.SetTPDParam(type, name, period_ms, di_choose, do_choose);

    robot.Mode(1);
    Thread.Sleep(1000);
    robot.DragTeachSwitch(1);
    robot.SetTPDStart(type, name, period_ms, di_choose, do_choose);
    Thread.Sleep(10000);
    robot.SetWebTPDStop();
    robot.DragTeachSwitch(0);

    float ovl = 100.0f;
    byte blend = 0;

    DescPose start_pose = new DescPose();

    int rtn = robot.LoadTPD(name);
    Console.WriteLine("LoadTPD rtn is: {0}\n", rtn);

    robot.GetTPDStartPose(name, ref start_pose);
```

(continues on next page)

(continued from previous page)

```
    Console.WriteLine("start pose, xyz is: {0} {1} {2}. rpy is: {3} {4} {5} \n",
        start_pose.tran.x, start_pose.tran.y, start_pose.tran.z,
        start_pose.rpy.rx, start_pose.rpy.ry, start_pose.rpy.rz);
    robot.MoveCart(start_pose, 0, 0, 100, 100, ovl, -1, -1);
    Thread.Sleep(1000);

    rtn = robot.MoveTPD(name, blend, ovl);
    Console.WriteLine("MoveTPD rtn is: {0}\n", rtn);
    Thread.Sleep(5000);

    robot.SetTPDDelete(name);
}
```

2.2.9.9 Preprocessing of external trajectory files

```
/ * *
* @brief External Trajectory File Preprocessing
* @param [in] name Trajectory file name
* @param [in] ovl speed scaling percentage, range [0 to 100]
* @param [in] opt 1- Control point, default is 1
* @return error code
* /
int LoadTrajectoryJ(string name, float ovl, int opt);
```

2.2.9.10 External trajectory file trajectory reproduction

```
/ * *
* @brief External Trajectory file trajectory reproduction
* @return error code
* /
int MoveTrajectoryJ();
```

2.2.9.11 Obtain the starting position of the trajectory file's trajectory

```
/ * *
* @brief Get the starting position of the trajectory file trajectory
* @param [in] name Trajectory file name
* @param [out] desc_pose trajectory starting pose
* @return error code
* /
int GetTrajectoryStartPose(string name, ref DescPose desc_pose);
```

2.2.9.12 Obtain the trajectory point number in the trajectory file

```
/ * *
* @brief Get the trajectory point number
* @param [out] pnum trajectory point number
* @return error code
* /
int GetTrajectoryPointNum(ref int pnum);
```

2.2.9.13 Set the running speed of the trajectory file trajectory

```
/ * *
* @brief Set the running speed of the trajectory file trajectory
* @param [in] ovl speed percentage
* @return error code
* /
int SetTrajectoryJSpeed(double ovl);
```

2.2.9.14 Set the force and torque during the trajectory file's trajectory operation

```
/ * *
* @brief Set the force and torque during the trajectory operation of the trajectory file
* @param [in] ft forces and torques in three directions, units N and Nm
* @return error code
* /
int SetTrajectoryJForceTorque(ForceTorque ft);
```

2.2.9.15 Set the force along the x direction during the trajectory's operation

```
/ * *
* @brief Set the force along the x direction during the trajectory operation
* @param [in] fx the force along the x direction, unit N
* @return error code
* /
int SetTrajectoryJForceFx(double fx);
```

2.2.9.16 Set the force along the y direction during the trajectory's operation

```
/ * *
* @brief Set the force along the y direction during the trajectory operation
* @param [in] fy the force along the y direction, unit N
* @return error code
* /
int SetTrajectoryJForceFy(double fy);
```

2.2.9.17 Set the force along the z direction during the trajectory's operation

```
/ * *
* @brief Set the force along the z direction during the trajectory operation
* @param [in] fz force along the z direction, unit N
* @return error code
* /
int SetTrajectoryJForceFz(double fz);
```

2.2.9.18 Set the torque around the X-axis during the trajectory's operation

```
/ * *
* @brief Set the torque around the X-axis during the trajectory operation
* @param [in] tx Torque around the X-axis, in Nm
* @return error code
* /
int SetTrajectoryJTorqueTx(double tx);
```

2.2.9.19 Set the torque around the Y-axis during the trajectory's operation

```
/ * *
* @brief Set the torque around the Y-axis during the trajectory operation
* @param [in] ty Torque around the Y-axis, in Nm
* @return error code
* /
int SetTrajectoryJTorqueTy(double ty);
```

2.2.9.20 Set the torque around the Z-axis during the trajectory's operation

```
/ * *
* @brief Set the torque around the Z-axis during the trajectory operation
* @param [in] tz the torque around the Z-axis, in Nm
* @return error code
* /
int SetTrajectoryJTorqueTz(double tz);
```

2.2.9.21 Upload the trajectory J file

```
/ * *
* @brief Upload the trajectory J file
* @param [in] filePath upload the full path name of the trajectory file C://test/testJ.
↪txt
* @return error code
* /
int TrajectoryJUpLoad(string filePath);
```

2.2.9.22 Delete the trajectory J file

```
/ * *
* @brief Delete the trajectory J file
* @param [in] fileName file name testJ.txt
* @return error code
* /
int TrajectoryJDelete(string fileName);
```

2.2.9.23 Sample code for reproducing robot trajectory J files

```
private void button33_Click(object sender, EventArgs e)
{
    int rtn = robot.TrajectoryJUpLoad("D://zUP/spray_traj1.txt");
    Console.WriteLine("Upload TrajectoryJ A {0}\n", rtn);

    string traj_file_name = "/fruser/traj/spray_traj1.txt";
    rtn = robot.LoadTrajectoryJ(traj_file_name, 100, 1);
    Console.WriteLine("LoadTrajectoryJ {0}, rtn is: {1}\n", traj_file_name, rtn);

    DescPose traj_start_pose = new DescPose();
    rtn = robot.GetTrajectoryStartPose(traj_file_name, ref traj_start_pose);
    Console.WriteLine("GetTrajectoryStartPose is: {0}\n", rtn);
    Console.WriteLine("desc_pos:{0},{1},{2},{3},{4},{5}\n",
        traj_start_pose.tran.x, traj_start_pose.tran.y, traj_start_pose.tran.z,
        traj_start_pose.rpy.rx, traj_start_pose.rpy.ry, traj_start_pose.rpy.rz);

    Thread.Sleep(1000);

    robot.SetSpeed(50);
    robot.MoveCart(traj_start_pose, 0, 0, 100, 100, 100, -1, -1);

    int traj_num = 0;
    rtn = robot.GetTrajectoryPointNum(ref traj_num);
    Console.WriteLine("GetTrajectoryStartPose rtn is: {0}, traj num is: {1}\n", rtn,␣
↪traj_num);

    rtn = robot.SetTrajectoryJSpeed(50.0f);
    Console.WriteLine("SetTrajectoryJSpeed is: {0}\n", rtn);

    ForceTorque traj_force = new ForceTorque();
    traj_force.fx = 10;
    rtn = robot.SetTrajectoryJForceTorque(traj_force);
    Console.WriteLine("SetTrajectoryJForceTorque rtn is: {0}\n", rtn);

    rtn = robot.SetTrajectoryJForceFx(10.0f);
    Console.WriteLine("SetTrajectoryJForceFx rtn is: {0}\n", rtn);

    rtn = robot.SetTrajectoryJForceFy(0.0f);
    Console.WriteLine("SetTrajectoryJForceFy rtn is: {0}\n", rtn);

```

(continues on next page)

(continued from previous page)

```
    rtn = robot.SetTrajectoryJForceFz(0.0f);
    Console.WriteLine("SetTrajectoryJForceFz rtn is: {0}\n", rtn);

    rtn = robot.SetTrajectoryJTorqueTx(10.0f);
    Console.WriteLine("SetTrajectoryJTorqueTx rtn is: {0}\n", rtn);

    rtn = robot.SetTrajectoryJTorqueTy(10.0f);
    Console.WriteLine("SetTrajectoryJTorqueTy rtn is: {0}\n", rtn);

    rtn = robot.SetTrajectoryJTorqueTz(10.0f);
    Console.WriteLine("SetTrajectoryJTorqueTz rtn is: {0}\n", rtn);

    rtn = robot.MoveTrajectoryJ();
    Console.WriteLine("MoveTrajectoryJ rtn is: {0}\n", rtn);
}
```

2.2.9.24 Trajectory preprocessing (trajectory forward-looking)

```
/ * *
* @brief Trajectory Preprocessing (Trajectory Preview)
* @param [in] name Trajectory file name
* @param [in] mode Sampling mode, 0- no sampling; 1- Equal data interval sampling; 2-␣
↪Equal error limiting sampling
* @param [in] errorLim error limit, effective with linear fitting
* @param [in] type smoothing method, 0-Bezier smoothing
* @param [in] precision Smoothing accuracy, effective when using Bezier smoothing
* @param [in] The maximum speed set by vamx, mm/s
* @param [in] The maximum acceleration set by amax, mm/s2
* @param [in] The maximum acceleration set by jmax, mm/s3
* @return error code
* /
int LoadTrajectoryLA(string name, int mode, double errorLim, int type, double precision,␣
↪double vamx, double amax,  double jmax);
```

2.2.9.25 Trajectory reproduction (Trajectory Foresight)

```
/ * *
* @brief Trajectory Reproduction (Trajectory Preview)
* @return error code
* /
int MoveTrajectoryLA();
```

2.2.9.26 Trajectory reproduction (trajectory forward-looking) code example

```
private void button8_Click(object sender, EventArgs e)
{
    int rtn = 0;

    string nameA = "/fruser/traj/A.txt";
    string nameB = "/fruser/traj/B.txt";

    rtn = robot.LoadTrajectoryLA(nameB, 0, 0, 0, 1, 100.0, 100.0, 1000.0);    //
    Console.WriteLine($"LoadTrajectoryLA rtn is {rtn}");

    DescPose startPos = new DescPose(0, 0, 0, 0, 0, 0);
    robot.GetTrajectoryStartPose(nameA, ref startPos);

    //
    robot.MoveCart(startPos, 1, 0, (float)100.0, (float)100.0, (float)100.0, -1, -1);

    rtn = robot.MoveTrajectoryLA();
    Console.WriteLine($"MoveTrajectoryLA rtn is {rtn}");
}
```

2.2.10 The use of robot WebAPP programs

2.2.10.1 Set the default job program to be automatically loaded upon startup

```
/ * *
* @brief set the default job program to be automatically loaded upon startup
* @param [in] flag 0- Default programs are not automatically loaded upon startup, 1-␣
↪Default programs are automatically loaded upon startup
* @param [in] program_name job program name and path, such as "/fruser/movej.lua", where
↪"/fruser/" is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the fixed␣
↪path for LA.
* @return error code
* /
int LoadDefaultProgConfig(byte flag, string program_name);
```

2.2.10.2 Load the specified job program

```
/ * *
* @brief Load the specified job program
* @param [in] program_name job program name and path, such as "/fruser/movej.lua", where
↪"/fruser/" is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the fixed␣
↪path for LA.
* @return error code
* /
int ProgramLoad(string program_name);
```

2.2.10.3 Get the name of the loaded job program

```
/ * *
* @brief Get the name of the loaded job program
* @param [out] program_name job program name and path, such as "/fruser/movej.lua",␣
↪where "/fruser/" is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the␣
↪fixed path for LA.
* @return error code
* /
int GetLoadedProgram(ref string program_name);
```

2.2.10.4 Obtain the execution line number of the current robot operation program

```
/ * *
* @brief get the line number of the current robot job program being executed
* @param [out] line number
* @return error code
* /
int GetCurrentLine(ref int line);
```

2.2.10.5 Run the currently loaded job program

```
/ * *
* @brief run the currently loaded job program
* @return error code
* /
int ProgramRun();
```

2.2.10.6 Pause the currently running job program

```
/ * *
* @brief Suspend the currently running job program
* @return error code
* /
int ProgramPause();
```

2.2.10.7 Resume the currently suspended job procedure

```
/ * *
* @brief Resume the currently suspended operation procedure
* @return error code
* /
int ProgramResume();
```

2.2.10.8 Terminate the currently running job program

```
/ * *
* @brief Terminate the currently running job program
* @return error code
* /
int ProgramStop();
```

2.2.10.9 Obtain the execution status of the robot's operation program

```
/ * *
* @brief Get the execution status of the robot operation program
* @param [out] state 1- The program stops or no program runs, 2- the program is running,␣
↪3- the program is paused
* @return error code
* /
int GetProgramState(ref byte state);
```

2.2.10.10 Example of robot LUA program operation code

```
private void btnWebApp_Click(object sender, EventArgs e)
{
    string program_name = "/fruser/Text1.lua";
    string loaded_name = "";
    byte state=0;
    int line=0;

    robot.Mode(0);
    robot.LoadDefaultProgConfig(0, program_name);
    robot.ProgramLoad(program_name);
    robot.ProgramRun();
    Thread.Sleep(1000);
    robot.ProgramPause();
    robot.GetProgramState(ref state);
    Console.WriteLine("program state:{0}\n", state);
    robot.GetCurrentLine(ref line);
    Console.WriteLine("current line:{0}\n", line);
    robot.GetLoadedProgram(ref loaded_name);
    Console.WriteLine("program name:{0}\n", loaded_name);
    Thread.Sleep(1000);
    robot.ProgramResume();
    Thread.Sleep(1000);
    robot.ProgramStop();
    Thread.Sleep(1000);
}
```

2.2.10.11 Download the Lua file

New in version C#SDK-v1.0.5.

```
/ * *
* @brief Download Lua files
* @param [in] fileName The job program to be downloaded is "test.lua" or "test.tar.gz"
* @param [in] savePath save the local path of the job program "D://Down/"
* @return error code
* /
int LuaDownLoad(string fileName, string savePath);
```

2.2.10.12 Upload the Lua file

New in version C#SDK-v1.0.5.

```
/ * *
* @brief Upload the Lua file
* @param [in] filePath local job path name "... /test.lua" or "... /test.tar.gz"
* @param [out] errStr error message
* @return error code
* /
int LuaUpload(string filePath, ref string errStr);
```

2.2.10.13 Delete the Lua file

New in version C#SDK-v1.0.5.

```
/ * *
* @brief Delete the Lua file
* @param [in] fileName Name of the job program to be deleted "test.lua"
* @return error code
* /
int LuaDelete(string fileName);
```

2.2.10.14 Get the names of all current lua files

New in version C#SDK-v1.0.5.

```
/ * *
* @brief Get the names of all current lua files
* @param [out] luaNames list of job program names
* @return error code
* /
int GetLuaList(ref List<string> luaNames) ;
```

2.2.10.15 Code example for uploading and downloading robot LUA files

New in version C#SDK-v1.0.5.

```
private void btnUploadLua_Click(object sender, EventArgs e)
{
    int rtn;
    List<string> luaNames = new List<string>();
    rtn = robot.GetLuaList(ref luaNames);
    Console.WriteLine("res is: {0}", rtn);
    Console.WriteLine("size is: {0}", luaNames.Count);
    foreach (var name in luaNames)
    {
    Console.WriteLine(name);
    }
    rtn = robot.LuaDownLoad("TT.lua", "D://zDOWN/");
    Console.WriteLine("LuaDownLoad rtn is {0}", rtn);
    string errStr = "";
    Thread.Sleep(2000);

    rtn = robot.LuaUpload("D://zUP/airlab.lua", ref errStr);
    Console.WriteLine("LuaUpload rtn is {0}", errStr);
    Thread.Sleep(2000);
    rtn = robot.LuaDelete("TT.lua");
    Console.WriteLine("LuaDelete rtn is {0}", rtn);
}
```

2.2.11 Robot Peripherals

2.2.11.1 Configure Gripper

```
/**
* @brief  Configure gripper
* @param  [in] company  Gripper manufacturer, to be determined
* @param  [in] device  Device number, not currently used, default is 0
* @param  [in] softvesion  Software version number, not currently used, default is 0
* @param  [in] bus Device bus position, currently unused, default is 0
* @return  Error code
*/
int SetGripperConfig(int company, int device, int softvesion, int bus);
```

2.2.11.2 Get gripper configuration

```
/**
* @brief  Get gripper configuration
* @param  [in] company  Gripper manufacturer, to be determined
* @param  [in] device  Device number, currently unused, default is 0
* @param  [in] softvesion  Software version number, currently unused, default is 0
* @param  [in] bus Device bus position, currently unused, default is 0
* @return  Error code
```

(continues on next page)

(continued from previous page)

```
*/
int GetGripperConfig(int *company, int *device, int *softvesion, int *bus);
```

2.2.11.3 Activate Gripper

```
/**
* @brief  Activate gripper
* @param  [in] index  Gripper number
* @param  [in] act  0-reset, 1-activate
* @return  Error code
*/
int ActGripper(int index, byte act);
```

2.2.11.4 Control gripper

```
/**
* @brief  Control gripper
* @param  [in] index  Gripper ID
* @param  [in] pos  Position percentage, range [0~100]
* @param  [in] vel  Speed percentage, range [0~100]
* @param  [in] force  Torque percentage, range [0~100]
* @param  [in] max_time  Maximum wait time, range [0~30000], unit ms
* @param  [in] block  0-blocking, 1-non-blocking
* @param  [in] type Gripper type, 0-parallel gripper; 1-rotating gripper
* @param  [in] rotNum Number of rotation cycles
* @param  [in] rotVel Rotation speed percentage [0-100]
* @param  [in] rotTorque Rotation torque percentage [0-100]
* @return Error code
*/
int MoveGripper(int index, int pos, int vel, int force, int max_time, byte block, int␣
↪type, double rotNum, int rotVel, int rotTorque);
```

2.2.11.5 Get gripper motion status

```
/**
* @brief  Get gripper motion status
* @param  [out] fault  0-no error, 1-error
* @param  [out] staus  0-motion not completed, 1-motion completed
* @return  Error code
*/
int GetGripperMotionDone(ref int fault, ref int status);
```

2.2.11.6 Get gripper activation status

```
/**
* @brief  Get gripper activation status
* @param  [out] fault  0-no error, 1-error
* @param  [out] status  bit0~bit15 corresponds to gripper numbers 0~15, bit=0 is not␣
↪activated, bit=1 is activated
* @return  Error code
*/
int GetGripperActivateStatus(ref int fault, ref int status);
```

2.2.11.7 Get Gripper Position

```
/**
* @brief  Get gripper position
* @param  [out] fault  0-no error, 1-error
* @param  [out] position  Position percentage, range 0~100%
* @return  Error code
*/
int GetGripperCurPosition(ref int fault, ref int position);
```

2.2.11.8 Get gripper speed

```
/**
* @brief  Get gripper speed
* @param  [out] fault  0-no error, 1-error
* @param  [out] speed  Speed percentage, range 0~100%
* @return  Error code
*/
int GetGripperCurSpeed(ref int fault, ref int speed);
```

2.2.11.9 Get gripper current

```
/**
* @brief  Get gripper current
* @param  [out] fault  0-no error, 1-error
* @param  [out] current  Current percentage, range 0~100%
* @return Error code
*/
int GetGripperCurrent(ref int fault, ref int current);
```

2.2.11.10 Get gripper voltage

```
/**
* @brief Get gripper voltage
* @param [out] fault 0-no error, 1-error
* @param  [out] voltage  Voltage, unit 0.1V
* @return  Error code
*/
int GetGripperVoltage(ref int fault, ref int voltage);
```

2.2.11.11 Get gripper temperature

```
/**
* @brief  Get gripper temperature
* @param  [out] fault  0-no error, 1-error
* @param  [out] temp  Temperature, unit °C
* @return  Error code
*/
int GetGripperTemp(ref int fault, ref int temp);
```

2.2.11.12 Calculate pre-gripping point - vision

```
/**
* @brief Calculate pre-gripping point - vision
* @param [in] desc_pos Gripping point Cartesian pose
* @param [in] zlength Z-axis offset
* @param [in] zangle Rotation offset around the z-axis
* @param [out] pre_pos Pre-pick point
* @return Error code
*/
int ComputePrePick(DescPose desc_pos, double zlength, double zangle, ref DescPose pre_
↪pos);
```

2.2.11.13 Calculate retreat point - vision

```
/**
* @brief Calculate retreat point - visual
* @param [in] desc_pos Retreat point Cartesian pose
* @param [in] zlength Z-axis offset
* @param [in] zangle Rotation offset around the Z-axis
* @param [out] post_pos Retreat point
* @return Error code
*/
int ComputePostPick(DescPose desc_pos, double zlength, double zangle, ref DescPose post_
↪pos);
```

2.2.11.14 Robot Gripper Operation Code Example

```
private void button36_Click(object sender, EventArgs e)
{
    int company = 4;
    int device = 0;
    int softversion = 0;
    int bus = 2;
    int index = 2;
    byte act = 0;
    int max_time = 30000;
    byte block = 0;
    int status=0;
    int fault=0;
    int active_status = 0;
    int current_pos = 0;
    int current = 0;
    int voltage = 0;
    int temp = 0;
    int speed = 0;

    robot.SetGripperConfig(company, device, softversion, bus);
    Thread.Sleep(1000);
    robot.GetGripperConfig(ref company, ref device, ref softversion, ref bus);
    Console.WriteLine("gripper config:{0},{1},{2},{3}\n", company, device, softversion,␣
↪bus);

    robot.ActGripper(index, act);
    Thread.Sleep(1000);
    act = 1;
    robot.ActGripper(index, act);
    Thread.Sleep(1000);

    robot.MoveGripper(index, 90, 50, 50, max_time, block, 0, 0, 0, 0);
    Thread.Sleep(1000);
    robot.MoveGripper(index, 30, 50, 0, max_time, block, 0, 0, 0, 0);

    robot.GetGripperMotionDone(ref fault, ref status);
    Console.WriteLine("motion status:{0},{1}\n", fault, status);

    robot.GetGripperActivateStatus(ref fault, ref active_status);
    Console.WriteLine("gripper active fault is: {0}, status is: {1}\n", fault, active_
↪status);

    robot.GetGripperCurPosition(ref fault, ref current_pos);
    Console.WriteLine("fault is:{0}, current position is: {1}\n", fault, current_pos);

    robot.GetGripperCurCurrent(ref fault, ref current);
    Console.WriteLine("fault is:{0}, current current is: {1}\n", fault, current);

    robot.GetGripperVoltage(ref fault, ref voltage);
    Console.WriteLine("fault is:{0}, current voltage is: {1} \n", fault, voltage);

```

(continues on next page)

(continued from previous page)

```
    robot.GetGripperTemp(ref fault, ref temp);
    Console.WriteLine("fault is:{0}, current temperature is: {1}\n", fault, temp);

    robot.GetGripperCurSpeed(ref fault, ref speed);
    Console.WriteLine("fault is:{0}, current speed is: {1}\n", fault, speed);

    int retval = 0;
    DescPose prepick_pose = new DescPose();
    DescPose postpick_pose = new DescPose();

    DescPose p1Desc = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f, 3.
↪833f);
    DescPose p2Desc = new DescPose(-321.222f, 185.189f, 335.520f, -179.030f, -1.284f, -
↪29.869f);

    retval = robot.ComputePrePick(p1Desc, 10, 0, ref prepick_pose);
    Console.WriteLine("ComputePrePick retval is: {0}\n", retval);
    Console.WriteLine("xyz is: {0}, {1}, {2}; rpy is: {3}, {4}, {5}\n",
        prepick_pose.tran.x, prepick_pose.tran.y, prepick_pose.tran.z,
        prepick_pose.rpy.rx, prepick_pose.rpy.ry, prepick_pose.rpy.rz);

    retval = robot.ComputePostPick( p2Desc, -10, 0, ref postpick_pose);
    Console.WriteLine("ComputePostPick retval is: {0}\n", retval);
    Console.WriteLine("xyz is: {0}, {1}, {2}; rpy is: {3}, {4}, {5}\n",
        postpick_pose.tran.x, postpick_pose.tran.y, postpick_pose.tran.z,
        postpick_pose.rpy.rx, postpick_pose.rpy.ry, postpick_pose.rpy.rz);

}
```

2.2.11.15 Get the number of rotations of the rotating gripper

```
/**
* @brief  Get the number of rotations of the rotating gripper
* @param  [out] fault  0-no error, 1-error
* @param  [out] num  Number of rotations
* @return  Error code
*/
int GetGripperRotNum(ref UInt16 fault, ref double num);
```

2.2.11.16 Get the rotation speed percentage of the rotating gripper

```
/**
* @brief  Get the rotation speed percentage of the rotating gripper
* @param  [out] fault  0-no error, 1-error
* @param  [out] speed  Rotation speed percentage
* @return  Error code
*/
int GetGripperRotSpeed(ref UInt16 fault, ref int speed);
```

2.2.11.17 Get the rotation torque percentage of the rotating gripper

```
/**
* @brief  Get the rotational torque percentage of the rotating gripper
* @param  [out] fault  0-no error, 1-error
* @param  [out] torque  Rotational torque percentage
* @return  Error code
*/
int GetGripperRotTorque(ref UInt16 fault, ref int torque);
```

2.2.11.18 Example of retrieving the rotational gripper status code

```
int MoveRotGripper(int pos, double rotPos)
{
    robot.ResetAllError();
    robot.ActGripper(1, 1);
    Thread.Sleep(1000);
    int rtn = robot.MoveGripper(1, pos, 50, 50, 5000, 1, 1, rotPos, 50, 100);
    Console.WriteLine($"move gripper rtn is {rtn}" );
    UInt16 fault = 0;
    double rotNum = 0.0;
    int rotSpeed = 0;
    int rotTorque = 0;
    robot.GetGripperRotNum(ref fault, ref rotNum);
    robot.GetGripperRotSpeed(ref fault, ref rotSpeed);
    robot.GetGripperRotTorque(ref fault, ref rotTorque);
    Console.WriteLine($"gripper rot num :{ rotNum}, gripper rotSpeed :{rotSpeed},␣
↪gripper rotTorque : { rotTorque}");
    return 0;
}
```

2.2.11.19 Drive belt start/stop

```
/**
* @brief Conveyor belt start/stop
* @param [in] status Status, 1-start, 0-stop
* @return Error code
*/
int ConveyorStartEnd(byte status);
```

2.2.11.20 Record IO detection points

```
/**
* @brief Record IO detection point
* @return Error code
*/
int ConveyorPointIORecord();
```

2.2.11.21 Record point A

```
/**
* @brief Record Point A
* @return Error code
*/
int ConveyorPointARecord();
```

2.2.11.22 Record reference point

```
/**
* @brief Record reference point
* @return Error code
*/
int ConveyorRefPointRecord();
```

2.2.11.23 Record Point B

```
/**
* @brief Record Point B
* @return Error code
*/
int ConveyorPointBRecord();
```

2.2.11.24 Conveyor belt workpiece IO detection

```
/**
* @brief Conveyor workpiece IO detection
* @param [in] max_t Maximum detection time, unit ms
* @return Error code
*/
int ConveyorIODetect(int max_t);
```

2.2.11.25 Get Object Current Position

```
/**
* @brief Get object current position
* @param [in] mode 1-track grab, 2-track movement, 3-TPD tracking
* @return Error code
*/
int ConveyorGetTrackData(int mode);
```

2.2.11.26 Start conveyor tracking

```
/**
* @brief Start conveyor tracking
* @param [in] status Status, 1-start, 0-stop
* @return Error code
*/
int ConveyorTrackStart(byte status);
```

2.2.11.27 Conveyor tracking stop

```
/**
* @brief Conveyor tracking stop
* @return Error code
*/
int ConveyorTrackEnd();
```

2.2.11.28 Drive Belt Parameter Configuration

```
/**
* @brief Drive belt parameter configuration
* @param [in] para[0] Encoder channel 1~2
* @param [in] para[1] Number of pulses per encoder revolution
* @param [in] para[2] Conveyor belt travel distance per encoder revolution
* @param [in] para[3] Workpiece coordinate system number Select the workpiece coordinate␣
↪system number for tracking motion functionality; set to 0 for tracking grasping and␣
↪TPD tracking
* @param [in] para[4] Whether to configure vision 0: No configuration 1: Configuration
* @param [in] para[5] Speed ratio for conveyor belt tracking and grasping options (1-
↪100). Other options default to 1.
* @param [in] followType Tracking motion type: 0-Tracking motion; 1 - Inspection movement
* @param [in] startDis Inspection tracking requires setting, tracking start distance, -
↪1: automatically calculated (inspection starts automatically when the workpiece␣
↪reaches below the robot), unit mm, default value 0
* @param [in] endDis Inspection tracking requires setting, tracking end distance, unit␣
↪mm, default value 100
* @return Error code
*/
int ConveyorSetParam(int encChannel, int resolution, double lead, int wpAxis, int vision,
↪ double speedRadio, int followType, int startDis=0, int endDis=100);
```

2.2.11.29 Set conveyor belt pickup point compensation

```
/**
* @brief Set conveyor belt catch point compensation
* @param [in] cmp Compensation position double[3]{x, y, z}
* @return Error code
*/
int ConveyorCatchPointComp(double[] cmp);
```

2.2.11.30 Conveyor belt tracking linear motion

```
/**
* @brief Conveyor belt tracking linear motion
* @param [in] name Motion point name
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] wobj Workpiece coordinate number, range [0~14]
* @param [in] vel Velocity percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], currently unavailable
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0] - move to position (blocked), [0~1000.0] - smooth radius␣
↪(unblocked), unit mm
* @return Error code
*/
int ConveyorTrackMoveL(string name, int tool, int wobj, float vel, float acc, float ovl,␣
↪float blendR);
```

2.2.11.31 Conveyor communication input detection

```
/**
* @brief Conveyor communication input detection
* @param [in] timeout Wait timeout in ms
* @return Error code
*/
int ConveyorComDetect(int timeout);
```

2.2.11.32 Conveyor Communication Input Detection Trigger

```
/**
* @brief Conveyor Communication Input Detection Trigger
* @return Error code
*/
int ConveyorComDetectTrigger();
```

2.2.11.33 Conveyor Communication Input Detection Trigger Example Program

```
private void button3_Click(object sender, EventArgs e)
{

    // Disable the button to prevent repeated clicks
    button3.Enabled = false;

    // Execute time-consuming operations in a background thread
    Thread conveyorThread = new Thread(ConveyorTest);
    conveyorThread.IsBackground = true;
    conveyorThread.Start();
}

private void button4_Click(object sender, EventArgs e)
{
    // Get user input
    string input = texBox.Text;
    Console.WriteLine($"please input a number to trigger:{input}");

    int rtn = robot.ConveyorComDetectTrigger();
    Console.WriteLine($"ConveyorComDetectTrigger return value: {rtn}");

}

private void ConveyorTest()
{
    // Use Invoke to update controls on the UI thread
    this.Invoke((MethodInvoker)delegate {
        Console.WriteLine( "Starting conveyor test...");
    });

    int retval = 0;
    int index = 1;
    int max_time = 30000;
    bool block = false;
    retval = 0;

    /* Conveyor belt grabbing process */
    DescPose startdescPose = new DescPose(139.176f, 4.717f, 9.088f, -179.999f, -0.004f, -
↪179.990f);
    JointPos startjointPos = new JointPos(-34.129f, -88.062f, 97.839f, -99.780f, -90.
↪003f, -34.140f);

    DescPose homePose = new DescPose(139.177f, 4.717f, 69.084f, -180.000f, -0.004f, -179.
↪989f);
    JointPos homejointPos = new JointPos(-34.129f, -88.618f, 84.039f, -85.423f, -90.003f,
↪ -34.140f);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    // Move to a safe position
```

(continues on next page)

(continued from previous page)

```
    retval = robot.MoveL(homejointPos, homePose, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 1, 1);
    Console.WriteLine($"MoveL to safe position return value: {retval}");

    // Conveyor detection
    retval = robot.ConveyComDetect(1000 * 10);
    Console.WriteLine($"ConveyorComDetect return value: {retval}");

    // Get tracking data
    retval = robot.ConveyorGetTrackData(2);
    Console.WriteLine($"ConveyorGetTrackData return value: {retval}");

    // Start tracking
    retval = robot.ConveyorTrackStart(2);
    Console.WriteLine($"ConveyorTrackStart return value: {retval}");

    // Move to the starting position
    robot.MoveL(startjointPos, startdescPose, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 1, 1);
    robot.MoveL(startjointPos, startdescPose, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 1, 1);

    // End tracking
    retval = robot.ConveyorTrackEnd();
    Console.WriteLine($"ConveyorTrackEnd return value: {retval}");

    // Return to safe position
    robot.MoveL(homejointPos, homePose, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 1, 1);

    this.Invoke((MethodInvoker)delegate {
        Console.WriteLine( "Conveyor belt test completed!");
        button3.Enabled = true;
    });
}
```

2.2.11.34 Robot Conveyor Belt Operation Example Program

```
private void btnConvert_Click(object sender, EventArgs e)
{
    Robot robot = new Robot();
    robot.RPC( "192.168.58.2");
    DescPose pos1 = new DescPose(0, 0, 0, 0, 0, 0);
    DescPose pos2 = new DescPose(0, 0, 0, 0, 0, 0);

    pos1.tran.x = -351.175;
    pos1.tran.y = 3.389;
    pos1.tran.z = 431.172;
    pos1.rpy.rx = -179.111;
    pos1.rpy.ry = -0.241;
```

(continues on next page)

(continued from previous page)

```
    pos1.rpy.rz = 90.388;

    pos2.tran.x = -333.654;
    pos2.tran.y = -229.003;
    pos2.tran.z = 404.335;
    pos2.rpy.rx = -179.139;
    pos2.rpy.ry = -0.779;
    pos2.rpy.rz = 91.269;
    int rtn = -1;

    double[] cmp = new double[3] { 0, 9.99, 0};
    rtn = robot.ConveyorCatchPointComp(cmp);
    if(rtn != 0)
    {
        return;
    }
    Console.WriteLine($"ConveyorCatchPointComp: rtn  {rtn}");

    rtn = robot.MoveCart(pos1, 0, 0, 100.0f, 180.0f, 100.0f, -1.0f, -1);
    Console.WriteLine($"MoveCart: rtn  {rtn}");

    rtn = robot.ConveyorIODetect(10000);
    Console.WriteLine($"ConveyorIODetect: rtn  {rtn}");

    robot.ConveyorGetTrackData(1);
    rtn = robot. ConveyorTrackStart(1);
    Console.WriteLine($"ConveyorTrackStart: rtn  {rtn}");

    rtn = robot.ConveyorTrackMoveL( "cvrCatchPoint", 0, 0, 100.0f, 0.0f, 100.0f, -1.0f,␣
↪0, 0);
    Console.WriteLine($"ConveyorTrackMoveL: rtn  {rtn}");

    rtn = robot.MoveGripper(1, 59, 43, 21, 30000, 0);
    Console.WriteLine($"MoveGripper: rtn  {rtn}");

    rtn = robot.ConveyorTrackMoveL( "cvrRaisePoint", 0, 0, 100.0f, 0.0f, 100.0f, -1.0f,␣
↪0, 0);
    Console.WriteLine($"ConveyorTrackMoveL: rtn  {rtn}");

    rtn = robot.ConveyorTrackEnd();
    Console.WriteLine($"ConveyorTrackEnd: rtn  {rtn}");

    rtn = robot.MoveCart(pos2, 0, 0, 100.0f, 180.0f, 100.0f, -1.0f, -1);
    Console.WriteLine($"MoveCart: rtn  {rtn}");

    rtn = robot.MoveGripper(1, 100, 43, 21, 30000, 0);
    Console.WriteLine($"MoveGripper: rtn  {rtn}");
}
```

2.2.11.35 End Sensor Configuration

```
/**
* @brief  End Sensor Configuration
* @param  [in] idCompany Manufacturer, 18-JUNKONG; 25-HUIDE
* @param  [in] idDevice Type, 0-JUNKONG/RYR6T.V1.0
* @param  [in] idSoftware Software version, 0-J1.0/HuiDe1.0 (not yet available)
* @param  [in] idBus Mounting location, 1-End 1 port; 2-End 2 port... 8-Endpoint 8 port (not yet available)
* @return Error code
*/
int AxleSensorConfig(int idCompany, int idDevice, int idSoftware, int idBus);
```

2.2.11.36 Get endpoint sensor configuration

```
/**
* @brief  Get terminal sensor configuration
* @param  [out] idCompany Manufacturer, 18-JUNKONG; 25-HUIDE
* @param  [out] idDevice Type, 0-JUNKONG/RYR6T.V1.0
* @return  Error code
*/
int AxleSensorConfigGet(ref int idCompany, ref int idDevice);
```

2.2.11.37 End-of-line sensor activation

```
/**
* @brief  End-of-line sensor activation
* @param  [in] actFlag 0-reset; 1-activate
* @return  Error code
*/
int AxleSensorActivate(int actFlag);
```

2.2.11.38 End Sensor Register Write

```
/**
* @brief  End Sensor Register Write
* @param  [in] devAddr  Device Address Number 0-255
* @param  [in] regHAddr High 8 bits of the register address
* @param  [in] regLAddr Low 8 bits of the register address
* @param  [in] regNum  Number of registers 0-255
* @param  [in] data1 Value to write to the register 1
* @param  [in] data2 Value to write to the register 2
* @param  [in] isNoBlock 0-blocking; 1-non-blocking
* @return  Error code
*/
 int AxleSensorRegWrite(int devAddr, int regHAddr, int regLAddr, int regNum, int data1, int data2, int isNoBlock);
```

2.2.11.39 End Sensor Code Example

```
private void button2_Click_1(object sender, EventArgs e)
{
    robot.AxleSensorConfig(18, 0, 0, 1);
    int company = -1;
    int type = -1;
    robot.AxleSensorConfigGet(ref company, ref type);
    Console.WriteLine( "company is " + company +  ", type is " + type);

    int rtn = robot.AxleSensorActivate(1);
    Console.WriteLine( "AxleSensorActivate rtn is " + rtn);

    Thread.Sleep(1000);

    rtn = robot.AxleSensorRegWrite(1, 4, 6, 1, 0, 0, 0);
    Console.WriteLine( "AxleSensorRegWrite rtn is " + rtn);
}
```

2.2.11.40 Obtain robot peripheral protocol

New in version C#SDK-v1.0.6.

```
/**
* @brief Get robot peripheral protocol
* @param [out] protocol Robot peripheral protocol number 4096-Extended Axle Control Card;
 ↪ 4097-ModbusSlave; 4098-ModbusMaster
* @return Error code
*/
int GetExDevProtocol(ref int protocol);
```

2.2.11.41 Set robot peripheral protocol

New in version C#SDK-v1.0.6.

```
/**
* @brief Set robot peripheral protocol
* @param [in] protocol Robot peripheral protocol number 4096-Extended Axis Control Card;␣
 ↪4097-Modbus Slave; 4098-Modbus Master
* @return Error code
*/
int SetExDevProtocol(int protocol);
```

2.2.11.42 Example program for setting robot peripheral protocol

```
private void btnSetProto_Click(object sender, EventArgs e)
{
  int protocol = 4096;
  int rtn = robot.SetExDevProtocol(protocol);

  Console.WriteLine( "SetExDevProtocol rtn " + rtn);
  rtn = robot.GetExDevProtocol(ref protocol);
  Console.WriteLine( "GetExDevProtocol rtn " + rtn +  " protocol is: " + protocol);
}
```

2.2.11.43 Get end-point communication parameters

```
/**
* @brief Get terminal communication parameters
* @param param Terminal communication parameters
* @return  Error code
*/
int GetAxleCommunicationParam(ref AxleComParam getParam);
```

2.2.11.44 Set terminal communication parameters

```
/**
* @brief Set terminal communication parameters
* @param param  Terminal communication parameters
* @return  Error code
*/
int SetAxleCommunicationParam(AxleComParam param);
```

2.2.11.45 Set terminal file transfer type

```
/**
* @brief Set the terminal file transfer type
* @param type 1-MCU upgrade file; 2-LUA file
* @return  Error code
*/
int SetAxleFileType(int type);
```

2.2.11.46 Set enable terminal LUA execution

```
/**
* @brief Enable LUA execution at the end
* @param enable 0-Disabled; 1-Enabled
* @return  Error code
*/
int SetAxleLuaEnable(int enable);
```

2.2.11.47 End LUA file exception error recovery

```
/**
* @brief End-of-file LUA file exception error recovery
* @param status 0-do not recover; 1-recover
* @return Error code
*/
int SetRecoverAxleLuaErr(int status);
```

2.2.11.48 Get the enable status of the terminal LUA execution

```
/**
* @brief Get the enable status of the terminal LUA execution
* @param [out] status 0-disabled; 1-enabled
* @return  Error code
*/
int GetAxleLuaEnableStatus(ref int status);
```

2.2.11.49 Set the enable type of the terminal LUA terminal device

```
/**
* @brief Set the enable type of the terminal LUA terminal device
* @param [in] forceSensorEnable Force sensor enable status, 0-disabled; 1-enabled
* @param [in] gripperEnable Gripper enable status, 0-disabled; 1-enabled
* @param [in] IOEnable IO device enable status, 0-disabled; 1-enabled
* @return  Error code
*/
int SetAxleLuaEnableDeviceType(int forceSensorEnable, int gripperEnable, int IOEnable);
```

2.2.11.50 Get the enable type of the end-of-line LUA device

```
/**
* @brief Get the enable type of the end LUA device
* @param [out] forceSensorEnable Force sensor enable status, 0-disabled; 1-enabled
* @param [out] gripperEnable Gripper enable status, 0-disabled; 1-enabled
* @param [out] IOEnable IO device enable status, 0-disabled; 1-enabled
* @return  Error code
*/
```

(continues on next page)

(continued from previous page)

```
int GetAxleLuaEnableDeviceType(ref int forceSensorEnable, ref int gripperEnable, ref int␣
↪IOEnable);
```

2.2.11.51 Get the currently configured end device

```
/**
* @brief Get the currently configured end device
* @param [out] forceSensorEnable Force sensor enable device number 0-disabled; 1-enabled
* @param [out] gripperEnable Gripper enable device number, 0-disabled;1-enabled
* @param [out] IODeviceEnable IO device enable ID, 0-disabled; 1-enabled
* @return  Error code
*/
int GetAxleLuaEnableDevice(ref int[] forceSensorEnable, ref int[] gripperEnable, ref␣
↪int[] IODeviceEnable);
```

2.2.11.52 Set enable gripper action control functionality

```
/**
* @brief Set enable claw action control function
* @param [in] id claw device number
* @param [in] func func[0]-claw enable; func[1]-claw initialization; 2-position setting;
↪3-Set speed; 4-Set torque; 6-Read gripper status; 7-Read initialization status; 8-Read␣
↪fault code; 9-Read position; 10-Read speed;11-read torque
* @return  Error code
*/
int SetAxleLuaGripperFunc(int id, int[] func);
```

2.2.11.53 Get enable claw action control function

```
/**
* @brief Get enable claw action control function
* @param [in] id Gripper device ID
* @param [out] func func[0]-Gripper enable; func[1]-Gripper initialization; 2-Position␣
↪setting; 3-Speed setting; 4-Torque setting; 6-Read gripper status; 7-Read␣
↪initialization status; 8-Read fault code; 9-Read position; 10-Read speed; 11-Read␣
↪torque
* @return  Error code
*/
int GetAxleLuaGripperFunc(int id, ref int[] func);
```

2.2.11.54 Writing robot Ethercat slave file

```
/**
* @brief Writing robot Ethercat slave file
* @param [in] type Slave file type, 1-upgrade slave file; 2-upgrade slave configuration file
* @param [in] slaveID Slave ID
* @param [in] fileName Upload file name
* @return  Error code
*/
int SlaveFileWrite(int type, int slaveID, string fileName);
```

2.2.11.55 Upload terminal Lua open protocol file

```
/**
* @brief Upload end Lua open protocol file
* @param filePath Local lua file path name  ".../AXLE_LUA_End_DaHuan.lua"
* @return Error code
*/
int AxleLuaUpload(string filePath);
```

2.2.11.56 Robot Ethercat slave enters boot mode

```
/**
* @brief Robot Ethercat slave enters boot mode
* @return  Error code
*/
int SetSysServoBootMode();
```

2.2.11.57 Robot End-of-Arm LUA File Operation Code Example

```
 private void button41_Click(object sender, EventArgs e)
{
    ROBOT_STATE_PKG pkg = new ROBOT_STATE_PKG();
    robot.AxleLuaUpload("D://zUP/AXLE_LUA_End_JunDuo_Xinjingcheng.lua");

    AxleComParam param = new AxleComParam(7, 8, 1, 0, 5, 3, 1);
    robot.SetAxleCommunicationParam(param);

    AxleComParam getParam = new AxleComParam();
    robot.GetAxleCommunicationParam(ref getParam);
    Console.WriteLine("GetAxleCommunicationParam param is {0} {1} {2} {3} {4} {5} {6}",
        getParam.baudRate, getParam.dataBit, getParam.stopBit, getParam.verify,
        getParam.timeout, getParam.timeoutTimes, getParam.period);

    robot.SetAxleLuaEnable(1);
    int luaEnableStatus = 0;
    robot.GetAxleLuaEnableStatus(ref luaEnableStatus);
```

(continues on next page)

(continued from previous page)

```
    robot.SetAxleLuaEnableDeviceType(0, 1, 0);

    int forceEnable = 0;
    int gripperEnable = 0;
    int ioEnable = 0;
    robot.GetAxleLuaEnableDeviceType(ref forceEnable, ref gripperEnable, ref ioEnable);
    Console.WriteLine("GetAxleLuaEnableDeviceType param is {0} {1} {2}", forceEnable, gripperEnable, ioEnable);

    int[] func = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
    robot.SetAxleLuaGripperFunc(1, func);
    int[] getFunc = new int[16];
    robot.GetAxleLuaGripperFunc(1, ref getFunc);
    int[] getforceEnable = new int[16];
    int[] getgripperEnable = new int[16];
    int[] getioEnable = new int[16];
    robot.GetAxleLuaEnableDevice(ref getforceEnable, ref getgripperEnable, ref getioEnable);
    Console.WriteLine("\ngetforceEnable status : ");
    foreach (int i in getforceEnable)
    {
        Console.Write(i + ",");
    }
    Console.WriteLine("\ngetgripperEnable status : ");
    foreach (int i in getgripperEnable)
    {
        Console.Write(i + ",");
    }
    Console.WriteLine("\ngetioEnable status : ");
    foreach (int i in getioEnable)
    {
        Console.Write(i + ",");
    }
    Console.WriteLine();
    robot.ActGripper(1, 0);
    Thread.Sleep(2000);
    robot.ActGripper(1, 1);
    Thread.Sleep(2000);
    robot.MoveGripper(1, 90, 10, 100, 50000, 0, 0, 0, 0, 0);
    int pos = 0;
    while (true)
    {
        robot.GetRobotRealTimeState(ref pkg);
        Console.WriteLine("gripper pos is " + pkg.gripper_position);
        Thread.Sleep(100);
    }
}
```

2.2.11.58 Get SmartTool button status

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Get SmartTool button status
* @param [out] state SmartTool handle button status; (bit0: 0 - communication normal; 1 - communication disconnected; bit1 - cancel operation; bit2 - clear program;
bit3 - A key; bit4 - B key; bit5 - C key; bit6 - D key; bit7 - E key; bit8 - IO key; bit9 - manual/automatic; bit10 - start)
* @return Error code
*/
int GetSmarttoolBtnState(ref int state);
```

2.2.11.59 Code example

New in version C#SDK-V1.1.3: Web-3.8.2

```
private void button11_Click(object sender, EventArgs e)
{

    ROBOT_STATE_PKG pkg = new ROBOT_STATE_PKG();
    int state = 0;
    while (true)
    {
        int rtn = robot.GetSmarttoolBtnState(ref state);
        string binaryString = Convert.ToString(state, 2).PadLeft(32, '0');
        Console.WriteLine($"GetSmarttoolBtnState rtn (binary): {binaryString}");
        Thread.Sleep(100);
    }

}
```

2.2.11.60 Upload Open Protocol Lua File

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Upload Open Protocol Lua File
* @param  filePath Local open protocol lua file path name
* @return Error code
*/
public int OpenLuaUpload(String filePath)
```

2.2.11.61 Get Slave Board Parameters

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Get Slave Board Parameters
* @param  type  0-Ethercat, 1-CClink, 3-Ethercat, 4-EIP
* @param  version  Protocol version
* @param  connState  0-Disconnected 1-Connected
* @return  Error code
*/
public int GetFieldBusConfig(int[] type, int[] version, int[] connState)
```

2.2.11.62 Write Slave DO

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Write Slave DO
* @param   DOIndex  DO number
* @param   wirteNum  Number to write
* @param   status Value to write, max 8
* @return  Error code
*/
public int FieldBusSlaveWriteDO(int DOIndex, int wirteNum, int[] status)
```

2.2.11.63 Write Slave AO

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Write Slave AO
* @param  AOIndex  AO number
* @param  wirteNum  Number to write
* @param  status Value to write, max 8
* @return  Error code
*/
public int FieldBusSlaveWriteAO(int AOIndex, int wirteNum, int[] status)
```

2.2.11.64 Read Slave DI

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Read Slave DI
* @param  DOIndex  DI number
* @param  readNum  Number to read
* @param  status Read value, max 8
* @return  Error code
*/
public int FieldBusSlaveReadDI(int DOIndex, int readNum, int[] status)
```

2.2.11.65 Read Slave AI

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief  Read Slave AI
* @param  AIIndex  AI number
* @param  readNum  Number to read
* @param  status Read value, max 8
* @return  Error code
*/
public int FieldBusSlaveReadAI(int AIIndex, int readNum, double[] status)
```

2.2.11.66 Wait for Extended DI Input

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Wait for Extended DI Input
* @param  DIIndex DI number
* @param  status 0-Low level; 1-High level
* @param  waitMs Max waiting time (ms)
* @return Error code
*/
public int FieldBusSlaveWaitDI(int DIIndex, int status, int waitMs)
```

2.2.11.67 Wait for Extended AI Input

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Wait for Extended AI Input
* @param  AIIndex AI number
* @param  waitType 0-Greater than; 1-Less than
* @param  value AI value
* @param  waitMs Max waiting time (ms)
* @return Error code
*/
public int FieldBusSlaveWaitAI(int AIIndex, int waitType, double value, int waitMs)
```

2.2.11.68 Slave Mode Related Interface Command Code Example

```
private void button101_Click(object sender, EventArgs e)
{
    int rtn = 0;

    int type = 0, version = 0, connState = 0;
    int[] ctrl = new int[8];
    int[] ctrlAO = new int[8];
    int[] DI = new int[8];
```

(continues on next page)

(continued from previous page)

```
    double[] AI = new double[8];
    if (rtn != 0)
    {
        return;
    }
    // Upload and load open protocol file
    robot.OpenLuaUpload("E://zup/CtrlDev_field.lua");
    Thread.Sleep(2000);
    robot.SetCtrlOpenLUAName(3, "CtrlDev_field.lua");
    robot.UnloadCtrlOpenLUA(3);
    robot.LoadCtrlOpenLUA(3);
    Thread.Sleep(8000);

    // Get protocol type, software version, and connection status with PLC
    robot.GetFieldBusConfig(ref type, ref version, ref connState);
    Console.WriteLine($"type is {type}, version is {version}, connState is {connState}");

    // Write DO0 = 1, DO1 = 0, DO2 = 1
    ctrl[0] = 1;
    ctrl[1] = 0;
    ctrl[2] = 1;
    robot.FieldBusSlaveWriteDO(0, 3, ctrl);

    // Write AO2 = 0x1000
    ctrlAO[0] = 0x1000;
    robot.FieldBusSlaveWriteAO(2, 1, ctrlAO);

    for (int i = 0; i < 100; i++)
    {
        robot.FieldBusSlaveReadDI(0, 4, ref DI);
        Console.WriteLine($"DI0 is {DI[0]}, DI1 is {DI[1]}, DI2 is {DI[2]}, DI3 is
↪{DI[3]}");
        robot.FieldBusSlaveReadAI(0, 3, ref AI);
        Console.WriteLine($"AI0 is {AI[0]}, AI1 is {AI[1]}, AI2 is {AI[2]}");
        Thread.Sleep(10);
    }
    int ret = robot.FieldBusSlaveWaitDI(0, 1, 100);
    Console.WriteLine($"FieldBusSlaveWaitDI result is {ret}");

    ret = robot.FieldBusSlaveWaitAI(0, 0, 400.00f, 100);
    Console.WriteLine($"FieldBusSlaveWaitAI result is {ret}");
}
```

2.2.11.69 Control Array Sucker

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Control Array Sucker
* @param  slaveID Slave ID
* @param  len Length
* @param  ctrlValue Control value 1-Suction at max vacuum; 2-Suction at set vacuum; 3-Stop suction
* @return Error code
*/
public int SetSuckerCtrl(int slaveID, int len, int[] ctrlValue)
```

2.2.11.70 Get Array Sucker Status

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Get Array Sucker Status
* @param  slaveID Slave ID
* @param  state Adsorption state 0-Release object; 1-Workpiece detected and adsorbed successfully; 2-No object adsorbed; 3-Object detached
* @param  pressValue Current vacuum degree Unit kpa
* @param  error Sucker current error code
* @return Error code
*/
public int GetSuckerState(int slaveID, int[] state, int[] pressValue, int[] error)
```

2.2.11.71 Wait for Sucker Status

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Wait for Sucker Status
* @param  slaveID Slave ID
* @param  state Adsorption state 0-Release object; 1-Workpiece detected and adsorbed successfully; 2-No object adsorbed; 3-Object detached
* @param  ms Max waiting time
* @return Error code
*/
public int WaitSuckerState(int slaveID, int state, int ms)
```

2.2.11.72 Array Sucker Control Command Code Example

```
private void TestSucker(Robot robot)
{

    int[] ctrl = new int[20];
    int state=0;
    int pressValue=0;
    int error=0;
    int rtn;


    // Upload and load open protocol file
    robot.OpenLuaUpload(@"C:\SDK\CtrlDev_sucker.lua");
    Thread.Sleep(2000);
    robot.UnloadCtrlOpenLUA(1);
    robot.LoadCtrlOpenLUA(1);
    Thread.Sleep(1000);

    // Control sucker in broadcast mode with maximum adsorption capacity
    ctrl[0] = 1;
    robot.SetSuckerCtrl(0, 1, ctrl);

    // Monitor states of sucker 1 and sucker 12 in a loop
    for (int i = 0; i < 100; i++)
    {
        robot.GetSuckerState(1, ref state, ref pressValue, ref error);
        Console.WriteLine($"sucker1 state is {state}, pressValue is {pressValue}, error␣
↪num is {error}");
        robot.GetSuckerState(12, ref state, ref pressValue, ref error);
        Console.WriteLine($"sucker12 state is {state}, pressValue is {pressValue}, error␣
↪num is {error}");
        Thread.Sleep(100);
    }
    // Wait for sucker 1 to reach adsorbed state, timeout 100ms
    int ret = robot.WaitSuckerState(1, 1, 100);
    Console.WriteLine($"WaitSuckerState result is {ret}");

    // Unicast mode to turn off sucker 1 and 12
    ctrl[0] = 3;
    robot.SetSuckerCtrl(1, 1, ctrl);
    robot.SetSuckerCtrl(12, 1, ctrl);

    robot.CloseRPC();
}
```

2.2.11.73 Laser peripheral on/off function

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser peripheral on/off function
* @param [in] OnOff 0-off 1-on
* @param [in] weldId Weld seam ID, default is 0
* @return Error code
*/
public int LaserTrackingLaserOnOff(int OnOff, int weldId)
```

2.2.11.74 Laser tracking start/stop function

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser tracking start/stop function
* @param [in] OnOff 0-stop 1-start
* @param [in] coordId Laser peripheral tool coordinate system number
* @return Error code
*/
public int LaserTrackingTrackOnOff(int OnOff, int coordId)
```

2.2.11.75 Laser positioning - fixed direction

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser positioning - fixed direction
* @param [in] direction 0-x+ 1-x- 2-y+ 3-y- 4-z+ 5-z-
* @param [in] vel Speed in %
* @param [in] distance Maximum positioning distance in mm
* @param [in] timeout Positioning timeout in ms
* @param [in] posSensorNum Laser calibrated tool coordinate number
* @return Error code
*/
public int LaserTrackingSearchStart_xyz(int direction, int vel, int distance, int␣
↪timeout, int posSensorNum)
```

2.2.11.76 Laser positioning - arbitrary direction

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser positioning - arbitrary direction
* @param [in] directionPoint XYZ coordinates of the positioning input point
* @param [in] vel Speed in %
* @param [in] distance Maximum positioning distance in mm
* @param [in] timeout Positioning timeout in ms
```

(continues on next page)

(continued from previous page)

```
* @param [in] posSensorNum Laser calibrated tool coordinate number
* @return Error code
*/
public int LaserTrackingSearchStart_point(DescTran directionPoint, int vel, int distance,
↪ int timeout, int posSensorNum)
```

2.2.11.77 Laser positioning stop

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser positioning stop
* @return Error code
*/
public int LaserTrackingSearchStop()
```

2.2.11.78 Laser IP configuration

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser IP configuration
* @param [in] ip IP address of the laser peripheral
* @param [in] port Port number of the laser peripheral
* @return Error code
*/
public int LaserTrackingSensorConfig(string ip, int port)
```

2.2.11.79 Laser peripheral sampling period configuration

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser peripheral sampling period configuration
* @param [in] period Laser peripheral sampling period in ms
* @return Error code
*/
public int LaserTrackingSensorSamplePeriod(int period)
```

2.2.11.80 Laser peripheral driver loading

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser peripheral driver loading
* @param [in] type Laser peripheral driver protocol type 101-Ruiniu 102-Chuangxiang 103-
↪Quanshi 104-Tongzhou 105-Aotai
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
public int LoadPosSensorDriver(int type)
```

2.2.11.81 Laser Peripheral Driver Unloading

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser peripheral driver unloading
* @return Error code
*/
public int UnLoadPosSensorDriver()
```

2.2.11.82 Laser Weld Seam Trajectory Recording

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser weld seam trajectory recording
* @param [in] status 0-stop recording 1-real-time tracking 2-start recording
* @param [in] delayTime Delay time in ms
* @return Error code
*/
public int LaserSensorRecord1(int status, int delayTime)
```

2.2.11.83 Laser Weld Seam Trajectory Replay

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser weld seam trajectory replay
* @param [in] delayTime Delay time in ms
* @param [in] speed Speed in %
* @return Error code
*/
public int LaserSensorReplay(int delayTime, double speed)
```

2.2.11.84 Laser Tracking Replay

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Laser tracking replay
* @return Error code
*/
public int MoveLTR()
```

2.2.11.85 Laser Weld Seam Trajectory Recording and Replay

```
/**
* @brief Laser Seam Trajectory Recording and Replay
* @param [in] delayMode Mode 0-Delay Time 1-Delay Distance
* @param [in] delayTime Delay time in milliseconds (ms)
* @param [in] delayDisExAxisNum Extended Axis Number
* @param [in] delayDis Delay distance in millimeters (mm)
* @param [in] sensitivePara Compensation Sensitivity Coefficient
* @param [in] trackMode Fixed-point Tracking Type. 0-Extended Axis Asynchronous Motion;␣
↪1-Robot
* @param [in] triggerMode Fixed-point Tracking Trigger Method. 0-Tracking Duration; 1-IO
* @param [in] runTime Robot Fixed-point Tracking Duration in seconds (s)
* @param [in] speed Speed in percentage (%)
* @return Error Code
*/
public int LaserSensorRecordandReplay(int delayMode, int delayTime, int␣
↪delayDisExAxisNum,double delayDis, double sensitivePara, int trackMode, int␣
↪triggerMode,double runTime, double speed)
```

2.2.11.86 Move to Laser Record Start Point

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Move to laser record start point
* @param [in] moveType 0-PTP 1-LIN
* @param [in] ovl Speed in %
* @return Error code
*/
public int MoveToLaserRecordStart(int moveType, double ovl)
```

2.2.11.87 Move to Laser Record End Point

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Move to laser record end point
* @param [in] moveType 0-PTP 1-LIN
* @param [in] ovl Speed in %
* @return Error code
*/
public int MoveToLaserRecordEnd(int moveType, double ovl)
```

2.2.11.88 Move to Laser Sensor Positioning Point

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Move to laser sensor positioning point
* @param [in] moveFlag Motion type: 0-PTP; 1-LIN
* @param [in] ovl Speed scaling factor, 0-100
* @param [in] dataFlag Weld seam cache data selection: 0-execute planning data; 1-execute recorded data
* @param [in] plateType Plate type: 0-corrugated plate; 1-corrugated cardboard; 2-fence plate; 3-oil drum; 4-corrugated shell steel
* @param [in] trackOffectType Laser sensor offset type: 0-no offset; 1-base coordinate system offset; 2-tool coordinate system offset; 3-laser sensor raw data offset
* @param [in] offset Offset value
* @return Error code
*/
public int MoveToLaserSeamPos(int moveFlag, double ovl, int dataFlag, int plateType, int trackOffectType, DescPose offset)
```

2.2.11.89 Get laser sensor positioning point coordinate information

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Get laser sensor positioning point coordinate information
* @param [in] trackOffectType Laser sensor offset type: 0-no offset; 1-base coordinate system offset; 2-tool coordinate system offset; 3-laser sensor raw data offset
* @param [in] offset Offset value
* @param [out] jPos Joint position [°]
* @param [out] descPos Cartesian position [mm]
* @param [out] tool Tool coordinate system
* @param [out] user Workpiece coordinate system
* @param [out] exaxis Extended axis position [mm]
* @return Error code
*/
public int GetLaserSeamPos(int trackOffectType, DescPose offset, ref JointPos jPos, ref DescPose descPos, ref int tool, ref int user, ref ExaxisPos exaxis)
```

2.2.11.90 Laser Peripheral Sensor Parameter Configuration and Debugging Code Example

New in version C#SDK-V1.1.8: Web-3.8.6

```
void testLaserConfig()
{
    int[] ctrl = new int[20];
    int state;
    int pressValue;
    int error;
    robot.LaserTrackingSensorConfig("192.168.58.20", 5020);
    robot.LaserTrackingSensorSamplePeriod(20);
```

(continues on next page)

(continued from previous page)

```
    robot.LoadPosSensorDriver(101);
    robot.LaserTrackingLaserOnOff(0, 0);
    System.Threading.Thread.Sleep(3000);
    robot.LaserTrackingLaserOnOff(1, 0);
}
```

2.2.11.91 Laser Trajectory Scanning and Trajectory Replay Code Example

New in version C#SDK-V1.1.8: Web-3.8.6

```
void testLaserRecordAndReplay()
{
    int[] ctrl = new int[20];
    int state;
    int pressValue;
    int error;
    robot.OpenLuaUpload("D://zUP/CtrlDev_laser_ruiniu-0117.lua");
    System.Threading.Thread.Sleep(2000);
    robot.SetCtrlOpenLUAName(0, "CtrlDev_laser_ruiniu-0117.lua");
    robot.UnloadCtrlOpenLUA(0);
    robot.LoadCtrlOpenLUA(0);
    System.Threading.Thread.Sleep(8000);
    for (int i=0;i<10;++i)
    {
        JointPos startjointPos = new JointPos(56.205, -117.951, 141.872, -118.149, -94.
↪217, -122.176);
        DescPose startdescPose = new DescPose(-97.552, -282.855, 26.675, 174.182, -1.338,
↪ -91.707);
        ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
        DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

        robot.MoveL(startjointPos, startdescPose, 1, 0, 100, 100, 100, -1, exaxisPos, 0,␣
↪0, offdese, 0);
        robot.LaserSensorRecord1(2, 10);

        JointPos endjointPos = new JointPos(68.809, -87.100, 121.120, -127.233, -95.038,␣
↪-109.555);
        DescPose enddescPose = new DescPose(-103.555, -464.234, 13.076, 174.179, -1.344,␣
↪-91.709);
        robot.MoveL(endjointPos, enddescPose, 1, 0, 50, 100, 100, -1, exaxisPos, 0, 0,␣
↪offdese, 0);

        robot.LaserSensorRecord1(0, 10);
        robot.MoveToLaserRecordStart(1, 30);
        robot.LaserSensorReplay(10, 100);
        robot.MoveLTR();
        robot.LaserSensorRecord1(0, 10);
        Console.WriteLine($"Number of completions : {i+1} ");
    }

}
```

2.2.11.92 Laser Positioning and Real-time Tracking Code Example

New in version C#SDK-V1.1.8: Web-3.8.6

```
public static void testLasertrack()
{
    int[] ctrl = new int[20];
    int state;
    int pressValue;
    int error;
    robot.OpenLuaUpload("D://zUP/CtrlDev_laser_ruiniu-0117.lua");
    System.Threading.Thread.Sleep(2000);
    robot.SetCtrlOpenLUAName(0, "CtrlDev_laser_ruiniu-0117.lua");
    robot.UnloadCtrlOpenLUA(0);
    robot.LoadCtrlOpenLUA(0);
    System.Threading.Thread.Sleep(8000);
    for (int i = 0; i < 10; ++i)
    {
        JointPos startjointPos = new JointPos(56.205, -117.951, 141.872, -118.149, -94.
↪217, -122.176);
        DescPose startdescPose = new DescPose(-97.552, -282.855, 26.675, 174.182, -1.338,
↪ -91.707);
        ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
        DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
        DescTran directionPoint = new DescTran();

        robot.MoveL(startjointPos, startdescPose, 1, 0, 100, 100, 100, -1, exaxisPos, 0,␣
↪0, offdese, 0);
        robot.LaserTrackingSearchStart_xyz(3, 100, 300, 1000, 3);
        robot.LaserTrackingSearchStop();
        robot.MoveToLaserSeamPos(1, 30, 0, 0, 0, offdese);

        robot.LaserTrackingTrackOnOff(1, 3);

        JointPos endjointPos = new JointPos(68.809, -87.100, 121.120, -127.233, -95.038,␣
↪-109.555);
        DescPose enddescPose = new DescPose(-103.555, -464.234, 13.076, 174.179, -1.344,␣
↪-91.709);
        robot.MoveL(endjointPos, enddescPose, 1, 0, 20, 100, 100, -1, exaxisPos, 0, 0,␣
↪offdese, 0);
        robot.LaserTrackingTrackOnOff(0, 3);
        Console.WriteLine($"Number of completions : {i + 1} ");
    }
}
```

2.2.11.93 Extended Axis and Robot Synchronized Laser Tracking Code Example

New in version C#SDK-V1.1.8: Web-3.8.6

```
public void TestLaserTrackAndExitAxis()
{
    ExaxisPos startexaxisPos = new ExaxisPos(0, 0, 0, 0);
    ExaxisPos seamexaxisPos = new ExaxisPos(-10, 0, 0, 0);
    ExaxisPos endexaxisPos = new ExaxisPos(-30, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
    JointPos startjointPos = new JointPos(58.337, -119.628, 146.037, -116.358, -92.224, -117.654);
    DescPose startdescPose = new DescPose(-53.375, -255.363, 0.919, 178.054, 1.077, -94.026);
    for (int i=0;i<10;++i)
    {
        robot.ExtAxisSyncMoveJ(startjointPos, startdescPose, 1, 0, 100, 100, 100, startexaxisPos, -1, 0, offdese);
        Console.WriteLine("11111");
        int ret = robot.LaserTrackingSearchStart_xyz(3, 100, 300, 1000, 2);
        robot.LaserTrackingSearchStop();
        Console.WriteLine("2222");
        int tool = 0;
        int user = 0;
        JointPos seamjointPos = new JointPos();
        DescPose seamdescPose = new DescPose();
        robot.GetLaserSeamPos(0, offdese, ref seamjointPos, ref seamdescPose, ref tool, ref user, ref startexaxisPos);
        Console.WriteLine($"{seamjointPos.jPos[0]}, {seamjointPos.jPos[1]}, {seamjointPos.jPos[2]}, " +
                          $"{seamjointPos.jPos[3]}, {seamjointPos.jPos[4]}, {seamjointPos.jPos[5]}, " +
                          $"{seamdescPose.tran.x}, {seamdescPose.tran.y}, {seamdescPose.tran.z}, " +
                          $"{seamdescPose.rpy.rx}, {seamdescPose.rpy.ry}, {seamdescPose.rpy.rz}");
        if (ret == 0)
        {
            robot.ExtAxisSyncMoveJ(seamjointPos, seamdescPose, 1, 0, 100, 100, 100, seamexaxisPos, -1, 0, offdese);
            Console.WriteLine("3333");
            robot.LaserTrackingTrackOnOff(1, 2);
            JointPos endjointPos = new JointPos(70.580, -90.918, 126.593, -125.154, -92.162, -105.403);
            DescPose enddescPose = new DescPose(-53.375, -419.020, 0.920, 178.054, 1.076, -94.026);
            robot.ExtAxisSyncMoveL(endjointPos, enddescPose, 1, 0, 20, 100, 100, -1, endexaxisPos, 0, offdese);
            robot.LaserTrackingTrackOnOff(0, 2);
        }
        Console.WriteLine($"Number of completions : {i + 1} ");
    }
}
```

2.2.12 Machine Manpower Control

2.2.12.1 Force Sensor Configuration

```
/**
 * @brief Configuring force sensors
 * @param [in] company Force Sensor Manufacturer, 17 - Quintessence Technologies
 * @param [in] device device number, not used, default is 0
 * @param [in] softvesion software version number, not used, the default is 0
 * @param [in] bus device hangs on the end of the bus position, do not use, the default is 0
 * @return Error code
 */
int FT_SetConfig(int company, int device, int softvesion, int bus);
```

2.2.12.2 Get the force transducer configuration

```
/**
 * @brief Get force sensor configuration
 * @param [out] deviceID force sensor number
 * @param [out] company Force Sensor Manufacturer,, Force Sensor Manufacturer, 17-Kunwei Technology, 19-Aerospace 11th Academy, 20-ATI Sensors, 21-Zhongke MiDot, 22-Weihang Minxin
 * @param [out] device device number, Kunwei (0-KWR75B), Aisino Eleven (0-MCS6A-200-4), ATI (0-AXIA80 -M8), Zhongke MiDot (0-MST2010), Weihang Minxin (0-WHC6L-YB-10A)
 * @param [out] softvesion software version number, not used, the default is 0
 * @return Error code
 */
int FT_GetConfig(ref int deviceID, ref int company, ref int device, ref int softvesion);
```

2.2.12.3 Force sensor activation

```
/**
 * @brief Force sensor activation
 * @param [in] act 0-reset, 1-activate
 * @return Error code.
 */
int FT_Activate(byte act).
```

2.2.12.4 Force Transducer Zeroing

```
/**
* @brief Force sensor zeroing
* @param [in] act 0-remove zero, 1-zero correction
* @return Error code
*/
int FT_SetZero(byte act).
```

2.2.12.5 Set the force transducer reference coordinate system

```
/**
* @brief Set the force sensor reference coordinate system.
* @param [in] ref 0-tool coordinate system, 1-base coordinate system
* @return Error code.
*/
int FT_SetRCS(byte type).
```

2.2.12.6 Set the force transducer lower load weight

```
/**
* @brief Set the load weight under the force sensor.
* @param [in] weight load weight kg
* @return Error code.
*/
int SetForceSensorPayLoad(double weight);
```

2.2.12.7 Set the force sensor payload center of mass

```
/**
* @brief Setting the center of mass of a load under a force sensor.
* @param [in] x load center of mass x mm
* @param [in] y load center of mass y mm
* @param [in] z load center of mass z mm
* @return Error code
*/
int SetForceSensorPayLoadCog(double x, double y, double z);
```

2.2.12.8 Get the force sensor pay load weight

```
/**
* @brief Get the load weight under the force sensor.
* @param [in] weight load weight kg
* @return Error code.
*/
int GetForceSensorPayLoad(ref double weight).
```

2.2.12.9 Get force sensor payload center of mass

```
/**
* @brief Get the center of mass of the load under the force transducer.
* @param [out] x load center of mass x mm
* @param [out] y load center of mass y mm
* @param [out] z load center of mass z mm
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
int GetForceSensorPayLoadCog(ref double x, ref double y, ref double z);
```

2.2.12.10 Automatic zeroing of the force sensor.

```
/**
* @brief Force sensor auto-zero
* @param [out] weight Sensor mass kg
* @param [out] pos sensor center of mass mm
* @return Error code
*/
int ForceSensorAutoComputeLoad(ref double weight, ref DescTran pos);
```

2.2.12.11 Get force/torque data in reference coordinate system.

```
/**
* @brief Get force/torque data in reference coordinate system.
* @param [out] ft force/torque, fx,fy,fz,tx,ty,tz
* @return Error code
*/
int FT_GetForceTorqueRCS(byte flag, ref ForceTorque ft);
```

2.2.12.12 Get force sensor raw force/torque data

```
/**
* @brief Get force sensor raw force/torque data.
* @param [out] ft force/torque, fx,fy,fz,tx,ty,tz
* @return Error code.
*/
int FT_GetForceTorqueOrigin(byte flag, ref ForceTorque ft);
```

2.2.12.13 Force Transducer Configuration and Auto-Zero Code Example

```
 private void button54_Click(object sender, EventArgs e)
{
    int company = 24;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    int index = 1;

    robot.FT_SetConfig(company, device, softversion, bus);
    Thread.Sleep(1000);
    robot.FT_GetConfig(ref company, ref device, ref softversion, ref bus);
    Console.WriteLine($"FT config:{company},{device},{softversion},{bus}");
    Thread.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
    robot.FT_Activate(0);
    Thread.Sleep(1000);
    robot.FT_Activate(1);
    Thread.Sleep(1000);

    Thread.Sleep(1000);
    robot.FT_SetZero(0);
    Thread.Sleep(1000);

    ForceTorque ft = new ForceTorque(0, 0, 0, 0, 0, 0);
    robot.FT_GetForceTorqueOrigin(0, ref ft);
    Console.WriteLine($"ft origin:{ft.fx},{ft.fy},{ft.fz},{ft.tx},{ft.ty},{ft.tz}");
    robot.FT_SetZero(1);
    Thread.Sleep(1000);

    DescPose ftCoord = new DescPose(0, 0, 0, 0, 0, 0);
    robot.FT_SetRCS(0, ftCoord);

    robot.SetForceSensorPayLoad(0.824);
    robot.SetForceSensorPayLoadCog(0.778, 2.554, 48.765);
    double weight = 0;
    double x = 0, y = 0, z = 0;
    robot.GetForceSensorPayLoad(ref weight);
    robot.GetForceSensorPayLoadCog(ref x, ref y, ref z);
    Console.WriteLine($"the FT load is {weight}, {x} {y} {z}");

    robot.SetForceSensorPayLoad(0);
    robot.SetForceSensorPayLoadCog(0, 0, 0);

    double computeWeight = 0;
    DescTran tran = new DescTran(0, 0, 0);
    robot.ForceSensorAutoComputeLoad(ref weight, ref tran);
    Console.WriteLine($"the result is weight {weight} pos is {tran.x} {tran.y} {tran.z}
↪");

}
```

2.2.12.14 Load Weight Recognition Record

```
/**
* @brief Load weight recognition record.
* @param [in] id Sensor coordinate system number in the range [1~14].
* @return Error code.
*/
int FT_PdIdenRecord(int id).
```

2.2.12.15 Load weight recognition calculation

```
/**
* @brief Load Weight Recognition Calculation
* @param [out] weight Weight of load in kg.
* @return Error code
*/
int FT_PdIdenCompute(ref double weight).
```

2.2.12.16 Load center of mass identification record

```
/**
* @brief Load center-of-mass identification record.
* @param [in] id Sensor coordinate system number, range [1~14].
* @param [in] index point number, range [1~3].
* @return Error code
*/
int FT_PdCogIdenRecord(int id, int index).
```

2.2.12.17 Load center of mass identification calculation

```
/**
* @brief Load center of mass identification calculation.
* @param [out] cog load center of mass in mm.
* @return Error code.
*/
int FT_PdCogIdenCompute(ref DescTran cog);
```

2.2.12.18 Force Transducer Load Recognition Code Example

```
 private void btnFTPdCog_Click(object sender, EventArgs e)
{
    int company = 24, device = 0, softversion = 0, bus = 1;

    robot.FT_SetConfig(company, device, softversion, bus);
    Thread.Sleep(1000);
    robot.FT_GetConfig(ref company, ref device, ref softversion, ref bus);
    Console.WriteLine($"FT config: {company}, {device}, {softversion}, {bus}");
    Thread.Sleep(1000);

    robot.FT_Activate(0);
    Thread.Sleep(1000);
    robot.FT_Activate(1);
    Thread.Sleep(1000);

    Thread.Sleep(1000);
    robot.FT_SetZero(0);
    Thread.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```

    ForceTorque ft = new ForceTorque(0,0,0,0,0,0);
    robot.FT_GetForceTorqueOrigin(0, ref ft);
    Console.WriteLine($"ft origin: {ft.fx}, {ft.fy}, {ft.fz}, {ft.tx}, {ft.ty}, {ft.tz}
↪");
    robot.FT_SetZero(1);
    Thread.Sleep(1000);

    DescPose tcoord = new DescPose(0, 0, 35.0, 0, 0, 0);
    robot.SetToolCoord(10, tcoord, 1, 0, 0, 0);

    robot.FT_PdIdenRecord(10);
    Thread.Sleep(1000);

    double weight = 0.0f;
    robot.FT_PdIdenCompute(ref weight);
    Console.WriteLine($"payload weight: {weight}");

    DescPose desc_p1 = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_p2 = new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);
    DescPose desc_p3 = new DescPose(-327.622, 402.230, 320.402, -178.067, 2.127, -46.
↪207);

    robot.MoveCart( desc_p1, 0, 0, 100.0f, 100.0f, 100.0f, -1.0f, -1);
    Thread.Sleep(1000);
    robot.FT_PdCogIdenRecord(10, 1);
    robot.MoveCart( desc_p2, 0, 0, 100.0f, 100.0f, 100.0f, -1.0f, -1);
    Thread.Sleep(1000);
    robot.FT_PdCogIdenRecord(10, 2);
    robot.MoveCart( desc_p3, 0, 0, 100.0f, 100.0f, 100.0f, -1.0f, -1);
    Thread.Sleep(1000);
    robot.FT_PdCogIdenRecord(10, 3);

    DescTran cog = new DescTran(0,0,0);
    robot.FT_PdCogIdenCompute(ref cog);
    Console.WriteLine($"cog: {cog.x}, {cog.y}, {cog.z}");
}
```

2.2.12.19 Collision Guard

```
/**
* @brief collision guarding
* @param [in] flag 0- turn off collision guarding, 1- turn on collision guarding
* @param [in] sensor_id Force sensor number.
* @param [in] select Whether to detect collision in six degrees of freedom, 0-no␣
↪detection, 1-detection.
* @param [in] ft collision force/torque, fx,fy,fz,tx,ty,tz
* @param [in] max_threshold max_threshold
* @param [in] min_threshold min_threshold
```

(continues on next page)

(continued from previous page)

```
* @note Force/torque detection range: (ft-min_threshold, ft+max_threshold)
* @return Error code
*/
int FT_Guard(int flag, int sensor_id, int[] select, ForceTorque ft, double[] max_
↪threshold, double[] min_threshold);
```

2.2.12.20 Collision Guard Code Example

```
 private void btnFTGuard_Click(object sender, EventArgs e)
{
    int company = 24, device = 0, softversion = 0, bus = 1;

    robot.FT_SetConfig(company, device, softversion, bus);
    Thread.Sleep(1000);
    robot.FT_GetConfig(ref company, ref device, ref softversion, ref bus);
    Console.WriteLine($"FT config: {company}, {device}, {softversion}, {bus}");
    Thread.Sleep(1000);

    robot.FT_Activate(0);
    Thread.Sleep(1000);
    robot.FT_Activate(1);
    Thread.Sleep(1000);

    Thread.Sleep(1000);
    robot.FT_SetZero(0);
    Thread.Sleep(1000);

    byte sensor_id = 1;
    int[] select = { 1, 1, 1, 1, 1, 1 };
    double[] max_threshold = { 10.0f, 10.0f, 10.0f, 10.0f, 10.0f, 10.0f };
    double[] min_threshold = { 5.0f, 5.0f, 5.0f, 5.0f, 5.0f, 5.0f };

    ForceTorque ft = new ForceTorque();
    DescPose desc_p1 = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_p2 = new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);
    DescPose desc_p3 = new DescPose(-327.622, 402.230, 320.402, -178.067, 2.127, -46.
↪207);

    robot.FT_Guard(1, sensor_id, select,  ft, max_threshold, min_threshold);
    robot.MoveCart( desc_p1, 0, 0, 100.0f, 100.0f, 100.0f, -1.0f, -1);
    robot.MoveCart( desc_p2, 0, 0, 100.0f, 100.0f, 100.0f, -1.0f, -1);
    robot.MoveCart( desc_p3, 0, 0, 100.0f, 100.0f, 100.0f, -1.0f, -1);

    robot.FT_Guard(0, sensor_id, select, ft, max_threshold, min_threshold);
}
```

2.2.12.21 Constant force control

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief  Constant Force Control
* @param  [in] flag 0-Disable constant force control, 1-Enable constant force control
* @param  [in] sensor_id Force sensor ID
* @param  [in] select  Select whether to detect collision for the six degrees of freedom, 0-Do not detect, 1-Detect
* @param  [in] ft  Collision force/torque, fx,fy,fz,tx,ty,tz
* @param  [in] ft_pid Force PID parameters, torque PID parameters
* @param  [in] adj_sign Adaptive start/stop control, 0-Disable, 1-Enable
* @param  [in] ILC_sign ILC start/stop control, 0-Stop, 1-Training, 2-Operation
* @param  [in] max_dis Maximum adjustment distance, unit mm
* @param  [in] max_ang Maximum adjustment angle, unit deg
* @param  [in] M rx, ry mass parameters [0.1-10], default 2
* @param  [in] B rx, ry damping parameters [0.1-50], default 8
* @param  [in] threshold rx, ry activation thresholds [0-10], default 0.2
* @param  [in] adjustCoeff rx, ry torque adjustment coefficients [0-1], default 1
* @param  [in] polishRadio Polishing radius, unit mm
* @param  [in] filter_Sign Filter enable flag 0-Off; 1-On, default off
* @param  [in] posAdapt_sign Pose compliance enable flag 0-Off; 1-On, default off
* @param  [in] isNoBlock Blocking flag, 0-Blocking; 1-Non-blocking
* @return  Error code
*/
public int FT_Control(byte flag, int sensor_id, byte[] select, ForceTorque ft, float[] ft_pid,byte adj_sign, byte ILC_sign, float max_dis, float max_ang,double[] M, double[] B, double[] threshold, double[] adjustCoeff,double polishRadio, int filter_Sign, int posAdapt_sign, int isNoBlock)
```

2.2.12.22 Constant force control with damping code example

New in version C#SDK-V1.1.9: Web-3.8.7

```
public void TestFTControlWithAdjustCoeff()
{
    int rtn;
    int sensor_id = 10;
    byte[] select = new byte[6] { 0, 0, 1, 0, 0, 0 };
    float[] ft_pid = new float[6] { 0.0008f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f };
    byte adj_sign = 0;
    byte ILC_sign = 0;
    float max_dis = 1000.0f;
    float max_ang = 20.0f;
    ForceTorque ft = new ForceTorque();
    ft.fz = -10.0f;
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);
    JointPos j1 = new JointPos(80.765f, -98.795f, 106.548f, -97.734f, -89.999f, 94.842f);
    JointPos j2 = new JointPos(43.067f, -84.429f, 92.620f, -98.175f, -90.011f, 57.144f);
    DescPose desc_p1 = new DescPose(5.009f, -547.463f, 262.053f, -179.999f, -0.019f, 75.923f);
```

(continues on next page)

(continued from previous page)

```
    DescPose desc_p2 = new DescPose(-347.966f, -547.463f, 262.048f, -180.000f, -0.019f, 75.923f);
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    double[] M = new double[2] { 2.0, 2.0 };
    double[] B = new double[2] { 15.0, 15.0 };
    double[] threshold = new double[2] { 1.0, 1.0 };
    double[] adjustCoeff = new double[2] { 1.0, 0.8 };
    double polishRadio = 0.0;
    int filter_Sign = 0;
    int posAdapt_sign = 1;
    int isNoBlock = 0;
    while (true)
    {
        rtn = robot.FT_Control(1, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis, max_ang, M, B, threshold, adjustCoeff, 0, 0, 1, 0);
        Console.WriteLine($"FT_Control start rtn is {rtn}");
        robot.MoveL(j1, desc_p1, 1, 0, 100, 100, 100, -1, 0, epos, 0, 0, offset_pos, 0, 0, 10);
        robot.MoveL(j2, desc_p2, 1, 0, 100, 100, 100, -1, 0, epos, 0, 0, offset_pos, 0, 0, 10);
        rtn = robot.FT_Control(0, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis, max_ang, M, B, threshold, adjustCoeff, 0, 0, 1, 0);
        Console.WriteLine($"FT_Control end rtn is {rtn}");
    }
}
```

2.2.12.23 Rotational Insertion

```
/**
* @brief Rotational Insertion
* @param [in] rcs Reference coordinate system, 0 - Tool coordinate system, 1 - Base coordinate system
* @param [in] angVelRot Rotational angular velocity, unit deg/s
* @param [in] ft Force/Torque threshold, fx,fy,fz,tx,ty,tz, range [0~100]
* @param [in] max_angle Maximum rotation angle, unit deg
* @param [in] orn Force/Torque direction, 1 - Along Z-axis direction, 2 - Around Z-axis direction
* @param [in] max_angAcc Maximum rotational acceleration, unit deg/s^2, currently unused, default is 0
* @param [in] rotorn Rotation direction, 1 - Clockwise, 2 - Counterclockwise
* @param [in] strategy Processing strategy for undetected force/torque, 0 - Error; 1 - Warning, continue motion
* @return Error code
*/
public int FT_RotInsertion(int rcs, double angVelRot, double ft, double max_angle, int orn, double max_angAcc, int rotorn, int strategy)
```

2.2.12.24 Robot Force Sensor Rotational Insertion Code Example

```
public void TestMove()
{
    int rtn;
    JointPos j1 = new JointPos(-11.904f, -99.669f, 117.473f, -108.616f, -91.726f, 74.256f);
    JointPos j2 = new JointPos(-45.615f, -106.172f, 124.296f, -107.151f, -91.282f, 74.255f);
    JointPos j3 = new JointPos(-29.777f, -84.536f, 109.275f, -114.075f, -86.655f, 74.257f);
    JointPos j4 = new JointPos(-31.154f, -95.317f, 94.276f, -88.079f, -89.740f, 74.256f);
    DescPose desc_pos1 = new DescPose(-419.524f, -13.000f, 351.569f, -178.118f, 0.314f,3.833f);
    DescPose desc_pos2 = new DescPose(-321.222f, 185.189f, 335.520f, -179.030f, -1.284f,-29.869f);
    DescPose desc_pos3 = new DescPose(-487.434f, 154.362f, 308.576f, 176.600f, 0.268f, -14.061f);
    DescPose desc_pos4 = new DescPose(-443.165f, 147.881f, 480.951f, 179.511f, -0.775f, -15.409f);
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 100.0f;
    float ovl = 100.0f;
    float oacc = 100.0f;
    float blendT = 0.0f;
    float blendR = 0.0f;
    byte flag = 0;
    byte search = 0;
    int blendMode = 0;
    int velAccMode = 0;
    robot.SetSpeed(20);
    rtn = robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag,offset_pos);
    Console.WriteLine($"movej errcode:{rtn}");
    rtn = robot.MoveL(j2, desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, epos,search, flag, offset_pos, oacc, velAccMode,0,10);
    Console.WriteLine($"movel errcode:{rtn}");
    rtn = robot.MoveC(j3, desc_pos3, tool, user, vel, acc, epos, flag, offset_pos,j4,desc_pos4, tool, user, vel, acc, epos, flag, offset_pos, ovl, blendR, oacc,velAccMode);
    Console.WriteLine($"movec errcode:{rtn}");
    rtn = robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag,offset_pos);
    Console.WriteLine($"movej errcode:{rtn}");
    rtn = robot.Circle(j3, desc_pos3, tool, user, vel, acc, epos,j1, desc_pos1, tool,user, vel, acc, epos,ovl, flag, offset_pos, oacc, -1, velAccMode);
    Console.WriteLine($"circle errcode:{rtn}");
    rtn = robot.MoveCart(desc_pos4, tool, user, vel, acc, ovl, blendT, -1);
    Console.WriteLine($"MoveCart errcode:{rtn}");
```

(continues on next page)

(continued from previous page)

```
    rtn = robot.MoveJ(j1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos);
    Console.WriteLine($"movej errcode:{rtn}");
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, epos,␣
↪search, flag, offset_pos, -1, velAccMode);
    Console.WriteLine($"movel errcode:{rtn}");
    rtn = robot.MoveC(desc_pos3, tool, user, vel, acc, epos, flag, offset_pos,desc_pos4,␣
↪tool, user, vel, acc, epos, flag, offset_pos,ovl, blendR, -1, velAccMode);
    Console.WriteLine($"movec errcode:{rtn}");
    rtn = robot.MoveJ(j2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos);
    Console.WriteLine($"movej errcode:{rtn}");
    rtn = robot.Circle(desc_pos3, tool, user, vel, acc, epos, desc_pos1, tool, user, vel,
↪ acc, epos,ovl, flag, offset_pos, oacc, blendR, -1, velAccMode);
    Console.WriteLine($"circle errcode:{rtn}");
}
```

2.2.12.25 Flex control on

```
/**
* @brief Smooth control on
* @param [in] p Position adjustment coefficient or softening coefficient.
* @param [in] force Soft opening force threshold in N
* @return Error Code
*/
int FT_ComplianceStart(float p, float force);
```

2.2.12.26 Flex control off

```
/**
* @brief Soft control off
* @return Error code
*/
int FT_ComplianceStop();
```

2.2.12.27 Sample Flex Control Code

```
 private void btnComplience_Click(object sender, EventArgs e)
{
    int company = 24, device = 0, softversion = 0, bus = 1;
    robot.FT_SetConfig(company, device, softversion, bus);
    Thread.Sleep(1000);
    robot.FT_GetConfig(ref company, ref device, ref softversion, ref bus);
    Console.WriteLine($"FT config: {company}, {device}, {softversion}, {bus}");
    Thread.Sleep(1000);

    robot.FT_Activate(0);
    Thread.Sleep(1000);
    robot.FT_Activate(1);
```

(continues on next page)

(continued from previous page)

```
    Thread.Sleep(1000);

    robot.FT_SetZero(0);
    Thread.Sleep(1000);

    byte flag = 1;
    int sensor_id = 1;
    int[] select = { 1, 1, 1, 0, 0, 0 };
    double[] ft_pid = { 0.0005f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f };
    byte adj_sign = 0, ILC_sign = 0;
    float max_dis = 100.0f, max_ang = 0.0f;

    ForceTorque ft = new ForceTorque { fx = -10.0, fy = -10.0, fz = -10.0 };
    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);

    JointPos j1 = new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2 = new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose desc_p1 = new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_p2 = new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);

    robot.FT_Control(flag, (byte)sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_
↪dis, max_ang);
    float p = 0.00005f;
    float force = 30.0f;
    int rtn = robot.FT_ComplianceStart(p, force);
    Console.WriteLine($"FT_ComplianceStart rtn is {rtn}");

    int count = 5;
    while (count-- > 0)
    {
    robot.MoveL(j1, desc_p1, 0, 0, 100.0f, 180.0f, 100.0f, -1.0f, epos, 0, 1, offset_
↪pos);
    robot.MoveL(j2, desc_p2, 0, 0, 100.0f, 180.0f, 100.0f, -1.0f, epos, 0, 0, offset_
↪pos);
    }

    robot.FT_ComplianceStop();
    Console.WriteLine($"FT_ComplianceStop rtn is {rtn}");

    flag = 0;
    robot.FT_Control(flag, (byte)sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_
↪dis, max_ang);
}
```

2.2.12.28 Load recognition initialization

New in version C#SDK-v1.0.4.

```
/**
* @brief Load recognition initialization.
* @return Error code
*/
int LoadIdentifyDynFilterInit();
```

2.2.12.29 Load identification variable initialization

New in version C#SDK-v1.0.4.

```
/**
* @brief Initialization of load recognition variables.
* @return Error code
*/
int LoadIdentifyDynVarInit();
```

2.2.12.30 Load Identification Main Program

New in version C#SDK-v1.0.4.

```
/**
* @brief Load recognition main program.
* @param [in] joint_torque Joint torque.
* @param [in] joint_pos Joint position.
* @param [in] t Sampling period.
* @return Error code
*/
int LoadIdentifyMain(double[] joint_torque, double[] joint_pos, double t);
```

2.2.12.31 Get the load identification result

New in version C#SDK-v1.0.4.

```
/**
* @brief Get load recognition results
* @param [in] gain Gravity term coefficient double[6], centrifugal term coefficient␣
↪double[6].
* @param [out] weight load weight
* @param [out] cog load center of mass
* @return error code
*/
int LoadIdentifyGetResult(double[] gain, ref double weight, ref DescTran cog);
```

2.2.12.32 Robot Load Identification Code Example

```
private void button74_Click(object sender, EventArgs e)
{
    int rtn;
    int retval = 0;

    retval = robot.LoadIdentifyDynFilterInit();
    Console.WriteLine("LoadIdentifyDynFilterInit retval is: " + retval);

    retval = robot.LoadIdentifyDynVarInit();
    Console.WriteLine("LoadIdentifyDynVarInit retval is: " + retval);

    JointPos posJ = new JointPos(0,0,0,0,0,0);
    DescPose posDec = new DescPose(0, 0, 0, 0, 0, 0);
    double[] joint_toq = new double[6] { 0.0f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f };
    robot.GetActualJointPosDegree(0, ref posJ);
    posJ.jPos[1] = posJ.jPos[1] + 10;
    robot.GetJointTorques(0, joint_toq);
    joint_toq[1] = joint_toq[1] + 2;

    double[] tmpTorque = new double[6] { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
    for (int i = 0; i < 6; i++)
    {
        tmpTorque[i] = joint_toq[i];
    }

    retval = robot.LoadIdentifyMain(tmpTorque, posJ.jPos, 1);
    Console.WriteLine("LoadIdentifyMain retval is: " + retval);

    double[] gain = new double[12] { 0, 0.05, 0, 0, 0, 0, 0, 0.02, 0, 0, 0, 0 };
    double weight = 0;
    DescTran load_pos = new DescTran(0, 0, 0);
    retval = robot.LoadIdentifyGetResult(gain, ref weight, ref load_pos);
    Console.WriteLine("LoadIdentifyGetResult retval is: {0}; weight is {1} cog is {2} {3}
↪ {4}", retval, weight, load_pos.x, load_pos.y, load_pos.z);
}
```

2.2.12.33 Force sensor assisted drag

New in version C#SDK-V1.1.4: Web-3.8.3

```
/**
* @brief Force sensor assisted drag
* @param [in] status control status, 0-off; 1-on
* @param [in] asaptiveFlag Adaptive on flag, 0-off; 1-on
* @param [in] interfereDragFlag Interference area drag flag, 0-off; 1-on
* @param [in] ingularityConstraintsFlag singularity strategy, 0-avoidance; 1-crossing
* @param [in] forceCollisionFlag Robot collision detection flag during auxiliary drag; 0-
↪off; 1-on
* @param [in] M inertia coefficient
* @param [in] B Damping coefficient
```

(continues on next page)

(continued from previous page)

```
* @param [in] K stiffness coefficient
* @param [in] F Drag six-dimensional force threshold
* @param [in] Fmax Maximum drag force limit Nm
* @param [in] Vmax Maximum joint speed limit °/s
* @return Error code
*/
int EndForceDragControl(int status, int asaptiveFlag, int interfereDragFlag,int␣
↪ingularityConstraintsFlag,int forceCollisionFlag, double[] M , double[] B, double[] K,␣
↪double[] F, double Fmax, double Vmax);
```

2.2.12.34 Get the state of the force sensor drag switch

```
/**
* @brief Get force sensor drag switch state.
* @param [out] dragState force sensor assist drag control state, 0-off; 1-on
* @param [out] sixDimensionalDragState sixDimensionalForceAssistedDragState, 0-off; 1-on
* @return ErrorCode
*/
int GetForceAndTorqueDragState(ref int dragState, ref int sixDimensionalDragState);
```

2.2.12.35 The force sensor is automatically turned on after the error is cleared

```
/**
* @brief The force sensor is automatically turned on after an error is cleared.
* @param [in] status Control status, 0-off, 1-on.
* @return Error code
*/
int SetForceSensorDragAutoFlag(int status);
```

2.2.12.36 Force Sensor Assisted Drag Code Example

```
 private void button61_Click(object sender, EventArgs e)
{
    robot.SetForceSensorDragAutoFlag(1);
    double[] M = { 15.0, 15.0, 15.0, 0.5, 0.5, 0.1 };
    double[] B = { 150.0, 150.0, 150.0, 5.0, 5.0, 1.0 };
    double[] K = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
    double[] F = { 10.0, 10.0, 10.0, 1.0, 1.0, 1.0 };

    robot.EndForceDragControl(1, 0, 0, 0, M, B, K, F, 50, 100);
    robot.WaitMs(5000);

    int dragState = 0;
    int sixDimensionalDragState = 0;
    robot.GetForceAndTorqueDragState(ref dragState, ref sixDimensionalDragState);
    Console.WriteLine($"the drag state is {dragState} {sixDimensionalDragState}");

```

(continues on next page)

(continued from previous page)

```
    robot.EndForceDragControl(0, 0, 0, 0, M, B, K, F, 50, 100);
}
```

2.2.12.37 Setting up the six-dimensional force and joint impedance hybrid drag switch and parameters

```
/**
* @brief Setting up a six-dimensional force and joint impedance hybrid drag switch and␣
↪parameters.
* @param [in] status Control status, 0-off, 1-on.
* @param [in] impedanceFlag impedance on flag, 0-off; 1-on
* @param [in] lamdeDain drag gain
* @param [in] KGain Stiffness Gain
* @param [in] BGain damping gain
* @param [in] dragMaxTcpVel Drag End Max Linear Velocity Limit
* @param [in] dragMaxTcpOriVel Maximum angular velocity limit at drag end
* @return Error code
*/
int ForceAndJointImpedanceStartStop(int status, impedanceFlag, double[] lamdeDain,␣
↪double[] KGain, double[] BGain, double dragMaxTcpVel, double dragMaxTcpOriVel);
```

2.2.12.38 Force Sensor Assisted Drag Code Example

```
 private void button62_Click(object sender, EventArgs e)
{
    robot.DragTeachSwitch(1);
    double[] lambdaGain = { 3.0, 2.0, 2.0, 2.0, 2.0, 3.0 };
    double[] kGain = { 0, 0, 0, 0, 0, 0 };
    double[] bGain = { 150, 150, 150, 5.0, 5.0, 1.0 };
    int rtn = robot.ForceAndJointImpedanceStartStop(1, 0, lambdaGain, kGain, bGain, 1000,
↪ 180);
    Console.WriteLine($"ForceAndJointImpedanceStartStop rtn is {rtn}");
    Thread.Sleep(5000);
    robot.DragTeachSwitch(0);
    rtn = robot.ForceAndJointImpedanceStartStop(0, 0, lambdaGain, kGain, bGain, 1000,␣
↪180);
    Console.WriteLine($"ForceAndJointImpedanceStartStop rtn is {rtn}");
}
```

2.2.12.39 Setting up the Wire Seek Expansion IO Port

```
/**
* @brief Setting up a wire-seeking extended IO port.
* @param searchDoneDINum Successful DO port (0-127) for wire seek.
* @param searchStartDONum The start/stop control DO port (0-127).
* @return Error Code
*/
int SetWireSearchExtDIONum(int searchDoneDINum, int searchStartDONum);
```

2.2.12.40 Impedance Control Start/Stop

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Impedance Control Start/Stop
* @param [in] status 0: disable; 1-enable
* @param [in] workSpace 0-joint space; 1-Cartesian space
* @param [in] forceThreshold Trigger force threshold (N)
* @param [in] m Mass parameter
* @param [in] b Damping parameter
* @param [in] k Stiffness parameter
* @param [in] maxV Max linear velocity (mm/s)
* @param [in] maxVA Max linear acceleration (mm/s2)
* @param [in] maxW Max angular velocity (°/s)
* @param [in] maxWA Max angular acceleration (°/s2)
* @return Error code
*/
public int ImpedanceControlStartStop(int status, int workSpace, double[] forceThreshold,␣
↪double[] m, double[] b, double[] k, double maxV, double maxVA, double maxW, double␣
↪maxWA)
```

2.2.12.41 Robot Impedance Control Start/Stop Code Example

New in version C#SDK-V1.1.8: Web-3.8.6

```
public void TestImpedanceControl()
{
    int[] ctrl = new int[20];
    int state;
    int pressValue;
    int error;
    int rtn;
    JointPos j1 = new JointPos(102.622, -135.990, 120.769, -73.950, -90.848, 35.507);
    JointPos j2 = new JointPos(93.674, -80.062, 82.947, -92.199, -90.967, 26.559);
    DescPose desc_pos1 = new DescPose(136.552, -149.799, 449.532, 179.817, -1.172, 157.
↪123);
    DescPose desc_pos2 = new DescPose(136.540, -561.048, 449.542, 179.819, -1.172, 157.
↪122);

    DescPose offset_pos = new DescPose(0, 0, 0, 0, 0, 0);
```

(continues on next page)

(continued from previous page)

```
    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    float vel = 100.0f;
    float acc = 200.0f;
    float ovl = 100.0f;
    float blendT = -1.0f;
    float blendR = -1.0f;

    byte flag = 0;

    byte search = 0;
    robot.SetSpeed(20);
    int company = 17;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    robot.FT_SetConfig(company, device, softversion, bus);
    Thread.Sleep(1000);
    robot.FT_GetConfig(ref company, ref device, ref softversion, ref bus);
    Console.WriteLine($"FT config:{company},{device},{softversion},{bus}");
    Thread.Sleep(1000);

    robot.FT_Activate(0);
    Thread.Sleep(1000);
    robot.FT_Activate(1);
    Thread.Sleep(1000);
    Thread.Sleep(1000);
    robot.FT_SetZero(0);
    Thread.Sleep(1000);
    robot.FT_SetZero(1);
    Thread.Sleep(1000);

    double[] forceThreshold = new double[] { 30, 30, 30, 5, 5, 5 };
    double[] m = new double[] { 0.1, 0.1, 0.1, 0.02, 0.02, 0.02 };
    double[] b = new double[] { 1, 1, 1, 0.08, 0.08, 0.08 };
    double[] k = new double[] { 0, 0, 0, 0, 0, 0 };
    rtn = robot.ImpedanceControlStartStop(1, 1, forceThreshold, m, b, k, 1000, 500, 100,␣
↪100);
    Console.WriteLine($"ImpedanceControlStartStop errcode:{rtn}");
    rtn = robot.MoveL(desc_pos1, tool, user, vel, acc, ovl, blendR, 0, epos, search,␣
↪flag, offset_pos, -1, 0);
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, 0, epos, search,␣
↪flag, offset_pos, -1, 0);
    rtn = robot.MoveL(desc_pos1, tool, user, vel, acc, ovl, blendR, 0, epos, search,␣
↪flag, offset_pos, -1, 0);
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, 0, epos, search,␣
↪flag, offset_pos, -1, 0);
    rtn = robot.MoveL(desc_pos1, tool, user, vel, acc, ovl, blendR, 0, epos, search,␣
↪flag, offset_pos, -1, 0);
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, 0, epos, search,␣
```

(continues on next page)

(continued from previous page)

```
↪flag, offset_pos, -1, 0);
    Console.WriteLine($"movel errcode:{rtn}");
    robot.ImpedanceControlStartStop(0, 1, forceThreshold, m, b, k, 1000, 500, 100, 100);
}
```

2.2.12.42 Enable torque compensation function and compensation coefficient

```
/**
* @brief Enable torque compensation function and compensation coefficient
* @param [in] status Switch, 0-Disable; 1-Enable
* @param [in] torqueCoeff J1-J6 torque compensation coefficient [0-1]
* @return Error code
*/
public int SerCoderCompenParams(int status, double[] torqueCoeff)
```

2.2.13 Extended axis

2.2.13.1 Set 485 extended axis parameters

New in version C#SDK-v1.0.6.

```
/**
* @brief Set 485 extended axis parameters
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] servoCompany Servo driver manufacturer, 1-Dynatech
* @param [in] servoModel Servo driver model, 1-FD100-750C
* @param [in] servoSoftVersion Servo driver software version, 1-V1.0
* @param [in] servoResolution Encoder resolution
* @param [in] axisMechTransRatio Mechanical transmission ratio
* @return Error code
*/
int AuxServoSetParam(int servoId, int servoCompany, int servoModel, int servoSoftVersion,
↪ int servoResolution, double axisMechTransRatio);
```

2.2.13.2 Get 485 extended axis configuration parameters

New in version C#SDK-v1.0.6.

```
/**
* @brief Get 485 extended axis configuration parameters
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [out] servoCompany Servo driver manufacturer, 1-Dynatec
* @param [out] servoModel Servo drive model, 1-FD100-750C
* @param [out] servoSoftVersion Servo drive software version, 1-V1.0
* @param [out] servoResolution Encoder resolution
* @param [out] axisMechTransRatio Mechanical transmission ratio
* @return Error code
*/
```

(continues on next page)

(continued from previous page)

```
int AuxServoGetParam(int servoId, ref int servoCompany, ref int servoModel, ref int␣
↪servoSoftVersion, ref int servoResolution, ref double axisMechTransRatio);
```

2.2.13.3 Set 485 extended axis enable/disable

New in version C#SDK-v1.0.6.

```
* @brief Set 485 extension axis enable/disable
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] status Enable status, 0-disable, 1-enable
* @return Error code
*/
int AuxServoEnable(int servoId, int status);
```

2.2.13.4 Set 485 extension axis control mode

New in version C#SDK-v1.0.6.

```
/**
* @brief Set 485 extension axis control mode
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] mode Control mode, 0-position mode, 1-speed mode
* @return Error code
*/
int AuxServoSetControlMode(int servoId, int mode);
```

2.2.13.5 Set the target position of the 485 extended axis (position mode)

New in version C#SDK-v1.0.6.

```
/**
* @brief Set the target position of the 485 extended axis (position mode)
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] pos Target position, mm or °
* @param [in] speed Target speed, mm/s or °/s
* @return Error code
*/
int AuxServoSetTargetPos(int servoId, double pos, double speed);
```

2.2.13.6 Set the target speed of the 485 extended axis (speed mode)

New in version C#SDK-v1.0.6.

```
/**
* @brief Set the target speed of the 485 extended axis (speed mode)
* @param [in] servoId Servo driver ID, range [1-15], corresponding to the slave ID
* @param [in] speed Target speed, mm/s or °/s
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
int AuxServoSetTargetSpeed(int servoId, double speed);
```

2.2.13.7 Set the target torque of the 485 extended axis (torque mode) – Not available yet

New in version C#SDK-v1.0.6.

```
/**
* @brief Set the target torque of the 485 extended axis (torque mode) -- not yet available
* @param [in] servoId Servo driver ID, range [1-15], corresponding to the slave ID
* @param [in] torque Target torque, Nm
* @return Error code
*/
int AuxServoSetTargetTorque(int servoId, double torque);
```

2.2.13.8 Set 485 extended axis zero

New in version C#SDK-v1.0.6.

```
/**
* @brief Set 485 extended axis zero
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @param [in] mode Home mode, 1-current position home; 2-negative limit home; 3-positive limit home
* @param [in] searchVel Home speed, mm/s or °/s
* @param [in] latchVel Latching speed, mm/s or °/s
* @return Error code
*/
int AuxServoHoming(int servoId, int mode, double searchVel, double latchVel);
```

2.2.13.9 Clear 485 extended axis error information

New in version C#SDK-v1.0.6.

```
/**
* @brief Clear 485 extended axis error information
* @param [in] servoId Servo driver ID, range [1-15], corresponding to slave ID
* @return Error code
*/
int AuxServoClearError(int servoId);
```

2.2.13.10 Get 485 extended axis servo status

New in version C#SDK-v1.0.6.

```
/**
* @brief Get the status of the 485 extended axis servo
* @param [in] servoId Servo driver ID, range [1-15], corresponding to the slave ID
* @param [out] servoErrCode Servo driver error code
* @param [out] servoState Servo driver status  bit0: 0-Disabled; 1-Enabled;  bit1: 0-Not␣
↪moving; 1-Moving;  bit2 0-Positive limit not triggered; 1-Positive limit triggered;␣
↪bit3 0-Negative limit not triggered; 1-Negative limit triggered; bit4 0-Positioning␣
↪not completed;1 - Positioning complete;  bit5: 0 - Not zeroed; 1 - Zeroing complete
* @param [out] servoPos Servo current position mm or °
* @param [out] servoSpeed Servo current speed mm/s or °/s
* @param [out] servoTorque Servo current torque Nm
* @return Error code
*/
int AuxServoGetStatus(int servoId, ref int servoErrCode, ref int servoState, ref double␣
↪servoPos, ref double servoSpeed, ref double servoTorque);
```

2.2.13.11 Set the 485 extended axis data axis number in the status feedback

New in version C#SDK-v1.0.6.

```
/**
* @brief Set the data axis number of the 485 extended axis in the status feedback
* @param [in] servoId Servo driver ID, range [1-15], corresponding to the slave ID
* @return Error code
*/
int AuxServosetStatusID(int servoId);
```

2.2.13.12 Set the acceleration and deceleration of the 485 extended axis

```
/**
* @brief Set the acceleration and deceleration of the 485 extended axis
* @param [in] acc 485 extended axis acceleration
* @param [in] dec 485 extended axis deceleration
* @return  Error code
*/
int AuxServoSetAcc(double acc, double dec);
```

2.2.13.13 Set 485 extended axis emergency stop acceleration and deceleration

```
/**
* @brief Set 485 extended axis emergency stop acceleration and deceleration
* @param [in] acc 485 extended axis emergency stop acceleration
* @param [in] dec 485 extended axis emergency stop deceleration
* @return  Error code
*/
int AuxServoSetEmergencyStopAcc(double acc, double dec);
```

2.2.13.14 Get 485 extended axis movement acceleration and deceleration

```
/**
* @brief Get 485 extended axis motion acceleration and deceleration
* @param [out] acc 485 extended axis motion acceleration
* @param [out] dec 485 extended axis motion deceleration
* @return  Error code
*/
int AuxServoGetAcc(ref double acc, ref double dec);
```

2.2.13.15 Get 485 extended axis emergency stop acceleration and deceleration

```
/**
* @brief Get 485 extended axis emergency stop acceleration and deceleration
* @param [out] acc 485 extended axis emergency stop acceleration
* @param [out] dec 485 extended axis emergency stop deceleration
* @return  Error code
*/
int AuxServoGetEmergencyStopAcc(ref double acc, ref double dec);
```

2.2.13.16 Extended axis control code example

New in version C#SDK-V1.1.3: Web-3.8.2

```
private void button64_Click(object sender, EventArgs e)
{
    int retval = robot.AuxServoSetParam(1, 1, 1, 1, 131072, 15.45);
    Console.WriteLine($"AuxServoSetParam is: {retval}");

    int servoCompany = 0;
    int servoModel = 0;
    int servoSoftVersion = 0;
    int servoResolution = 0;
    double axisMechTransRatio = 0;
    retval = robot.AuxServoGetParam(1, ref servoCompany, ref servoModel, ref␣
↪servoSoftVersion, ref servoResolution, ref axisMechTransRatio);
    Console.WriteLine($"servoCompany {servoCompany}\n" +
        $"servoModel {servoModel}\n" +
        $"servoSoftVersion {servoSoftVersion}\n" +
```

(continues on next page)

(continued from previous page)

```
        $"servoResolution {servoResolution}\n" +
        $"axisMechTransRatio {axisMechTransRatio}\n");

    retval = robot.AuxServoSetParam(1, 10, 11, 12, 13, 14);
    Console.WriteLine($"AuxServoSetParam is: {retval}");

    retval = robot.AuxServoGetParam(1, ref servoCompany, ref servoModel, ref␣
↪servoSoftVersion, ref servoResolution, ref axisMechTransRatio);
    Console.WriteLine($"servoCompany {servoCompany}\n" +
        $"servoModel {servoModel}\n" +
        $"servoSoftVersion {servoSoftVersion}\n" +
        $"servoResolution {servoResolution}\n" +
        $"axisMechTransRatio {axisMechTransRatio}\n");

    retval = robot.AuxServoSetParam(1, 1, 1, 1, 131072, 36);
    Console.WriteLine($"AuxServoSetParam is: {retval}");
    Thread.Sleep(3000);

    robot.AuxServoSetAcc(3000, 3000);
    robot.AuxServoSetEmergencyStopAcc(5000, 5000);
    Thread.Sleep(1000);
    double emagacc = 0, acc = 0;
    double emagdec = 0, dec = 0;
    robot.AuxServoGetEmergencyStopAcc(ref emagacc, ref emagdec);
    Console.WriteLine($"emergency acc is {emagacc}  dec is {emagdec}");
    robot.AuxServoGetAcc(ref acc, ref dec);
    Console.WriteLine($"acc is {acc}  dec is {dec}");

    robot.AuxServoSetControlMode(1, 0);
    Thread.Sleep(2000);

    retval = robot.AuxServoEnable(1, 0);
    Console.WriteLine($"AuxServoEnable disenable {retval}");
    Thread.Sleep(1000);
    int servoerrcode = 0;
    int servoErrCode = 0;
    int servoState = 0;
    double servoPos = 0;
    double servoSpeed = 0;
    double servoTorque = 0;
    retval = robot.AuxServoGetStatus(1, ref servoErrCode, ref servoState, ref servoPos,␣
↪ref servoSpeed, ref servoTorque);
    Console.WriteLine($"AuxServoGetStatus servoState {servoState}");
    Thread.Sleep(1000);

    retval = robot.AuxServoEnable(1, 1);
    Console.WriteLine($"AuxServoEnable enable {retval}");
    Thread.Sleep(1000);
    retval = robot.AuxServoGetStatus(1, ref servoErrCode, ref servoState, ref servoPos,␣
↪ref servoSpeed, ref servoTorque);
    Console.WriteLine($"AuxServoGetStatus servoState {servoState}");
    Thread.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```

    retval = robot.AuxServoHoming(1, 1, 5, 1);
    Console.WriteLine($"AuxServoHoming {retval}");
    Thread.Sleep(3000);

    retval = robot.AuxServoSetTargetPos(1, 200, 30);
    Console.WriteLine($"AuxServoSetTargetPos {retval}");
    Thread.Sleep(1000);
    retval = robot.AuxServoGetStatus(1, ref servoErrCode, ref servoState, ref servoPos,␣
↪ref servoSpeed, ref servoTorque);
    Console.WriteLine($"AuxServoGetStatus servoSpeed {servoSpeed}");
    Thread.Sleep(8000);

    robot.AuxServoSetControlMode(1, 1);
    Thread.Sleep(2000);

    robot.AuxServoEnable(1, 0);
    Thread.Sleep(1000);
    robot.AuxServoEnable(1, 1);
    Thread.Sleep(1000);
    robot.AuxServoSetTargetSpeed(1, 100, 80);

    Thread.Sleep(5000);
    robot.AuxServoSetTargetSpeed(1, 0, 80);
}
```

2.2.13.17 UDP extended axis communication parameter configuration

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief UDP extended axis communication parameter configuration
* @param [in] ip PLC IP address
* @param [in] port  Port number
* @param [in] period    Communication cycle (ms, default is 2, do not modify this␣
↪parameter)
* @param [in] lossPkgTime   Packet loss detection time (ms)
* @param [in] lossPkgNum    Number of packet losses
* @param [in] disconnectTime    Communication disconnection confirmation time
* @param [in] reconnectEnable   Enable automatic reconnection after communication␣
↪disconnect 0-Disabled 1-Enabled
* @param [in] reconnectPeriod   Reconnection period (ms)
* @param [in] reconnectNum  Reconnection count
* @param [in] selfConnect Whether to automatically establish a connection after power␣
↪failure; 0-Do not establish a connection; 1-Establish a connection
* @return Error code
*/
int ExtDevSetUDPComParam(std::string ip, int port, int period, int lossPkgTime, int␣
↪lossPkgNum, int disconnectTime, int reconnectEnable, int reconnectPeriod, int␣
↪reconnectNum, int selfConnect);
```

2.2.13.18 Get UDP extended axis communication parameter configuration

New in version C#SDK-v1.0.7.

```
/**
* @brief Get UDP extended axis communication parameters
* @param [out] ip PLC IP address
* @param [out] port Port number
* @param [out] period   Communication cycle (ms, default is 2, do not modify this␣
↪parameter)
* @param [out] lossPkgTime  Packet loss detection time (ms)
* @param [out] lossPkgNum   Number of packet losses
* @param [out] disconnectTime   Communication disconnection confirmation time
* @param [out] reconnectEnable  Automatic reconnection after disconnection 0-disabled 1-
↪enabled
* @param [out] reconnectPeriod  Reconnection period (ms)
* @param [out] reconnectNum Number of reconnection attempts
* @return Error code
*/
int ExtDevGetUDPComParam(std::string& ip, int& port, int& period, int& lossPkgTime, int&␣
↪lossPkgNum, int& disconnectTime, int& reconnectEnable, int& reconnectPeriod, int&␣
↪reconnectNum);
```

2.2.13.19 Load UDP communication

New in version C#SDK-v1.0.7.

```
/**
* @brief Load UDP communication
* @return Error code
*/
int ExtDevLoadUDPDriver();
```

2.2.13.20 Unload UDP communication

New in version C#SDK-v1.0.7.

```
/**
* @brief Unload UDP communication
* @return Error code
*/
int ExtDevUnloadUDPDriver();
```

2.2.13.21 Restore connection after abnormal disconnection of UDP extended axis communication

New in version C#SDK-v1.0.7.

```
/**
* @brief Restore connection after abnormal disconnection of UDP extended axis communication
* @return Error code
*/
int ExtDevUDPClientComReset();
```

2.2.13.22 Close communication after abnormal disconnection of UDP extended axis communication

New in version C#SDK-v1.0.7.

```
/**
* @brief Close communication after abnormal disconnection of UDP extended axis communication
* @return Error code
*/
int ExtDevUDPClientComClose();
```

2.2.13.23 UDP extended axis parameter configuration

New in version C#SDK-v1.0.7.

```
/**
* @brief UDP extended axis parameter configuration
* @param [in] axisID Axis number
* @param [in] axisType Extended axis type 0-translation; 1-rotation
* @param [in] axisDirection Extended axis direction 0-forward; 1-backward
* @param [in] axisMax Maximum position of the extended axis mm
* @param [in] axisMin Minimum position of the extended axis mm
* @param [in] axisVel Speed mm/s
* @param [in] axisAcc Acceleration mm/s2
* @param [in] axisLead Lead mm
* @param [in] encResolution Encoder resolution
* @param [in] axisOffect Weld start point extension axis offset
* @param [in] axisCompany Driver manufacturer 1-Hechuan; 2-Huichuan; 3-Panasonic
* @param [in] axisModel Driver model 1-Hechuan-SV-XD3EA040L-E, 2-Hechuan-SV-X2EA150A-A, 1-Huichuan-SV620PT5R4I,1-Panasonic-MADLN15SG, 2-Panasonic-MSDLN25SG, 3-Panasonic-MCDLN35SG
* @param [in] axisEncType Encoder type  0-Incremental; 1-Absolute
* @return Error code
*/
int ExtAxisParamConfig(int axisID, int axisType, int axisDirection, double axisMax, double axisMin, double axisVel, double axisAcc, double axisLead, long encResolution, double axisOffect, int axisCompany, int axisModel, int axisEncType);
```

2.2.13.24 Set the installation position of the extended axis

New in version C#SDK-v1.0.7.

```
/**
* @brief Set the installation position of the extended axis
* @param [in] installType 0-robot installed on external axis, 1-robot installed outside external axis
* @return Error code
*/
int SetRobotPosToAxis(int installType);
```

2.2.13.25 Set extended axis system DH parameter configuration

New in version C#SDK-v1.0.7.

```
/**
* @brief Set the DH parameter configuration for the extended axis system
* @param [in]  axisConfig External axis configuration, 0-single-degree-of-freedom linear slide rail, 1-two-degree-of-freedom L-type indexer, 2-three-degree-of-freedom, 3-four-degree-of-freedom, 4-single-degree-of-freedom indexer
* @param [in]  axisDHd1 External axis DH parameter d1 mm
* @param [in]  axisDHd2 External axis DH parameter d2 mm
* @param [in]  axisDHd3 External axis DH parameter d3 mm
* @param [in]  axisDHd4 External axis DH parameter d4 mm
* @param [in]  axisDHa1 External axis DH parameter 11 mm
* @param [in]  axisDHa2 External axis DH parameter a2 mm
* @param [in]  axisDHa3 External axis DH parameter a3 mm
* @param [in]  axisDHa4 External axis DH parameter a4 mm
* @return Error code
*/
int SetAxisDHParaConfig(int axisConfig, double axisDHd1, double axisDHd2, double axisDHd3, double axisDHd4, double axisDHa1, double axisDHa2, double axisDHa3, double axisDHa4);
```

2.2.13.26 UDP extended axis enable

New in version C#SDK-v1.0.7.

```
/**
* @brief UDP extended axis enable
* @param [in] axisID Axis number [1-4]
* @param [in] status 0-disable; 1-enable
* @return Error code
*/
int ExtAxisServoOn(int axisID, int status);
```

2.2.13.27 UDP extended axis reset

New in version C#SDK-v1.0.7.

```
/**
* @brief UDP extended axis zero return
* @param [in] axisID axis number [1-4]
* @param [in] mode zero return mode 0-return to current position, 1-return to negative limit, 2-return to positive limit
* @param [in] searchVel zero search speed (mm/s)
* @param [in] latchVel Latching speed (mm/s)
* @return Error code
*/
int ExtAxisSetHoming(int axisID, int mode, double searchVel, double latchVel);
```

2.2.13.28 UDP extended axis jog start

New in version C#SDK-v1.0.7.

```
/**
* @brief UDP extended axis jog start
* @param [in] axisID Axis number [1-4]
* @param [in] direction Rotation direction 0-reverse; 1-forward
* @param [in] vel Speed (mm/s)
* @param [in] acc Acceleration (mm/s2)
* @param [in] maxDistance Maximum jog distance
* @return Error code
*/
int ExtAxisStartJog(int axisID, int direction, double vel, double acc, double maxDistance);
```

2.2.13.29 UDP extended axis jog stop

New in version C#SDK-v1.0.7.

```
/**
* @brief UDP extended axis jog stop
* @param [in] axisID axis ID[1-4]
* @return Error code
*/
int ExtAxisStopJog(int axisID);
```

2.2.13.30 UDP extended axis configuration and jog code example

```
private void btnJog_Click(object sender, EventArgs e)
{
    int rtn = robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100,␣
↪5,1);
    Console.WriteLine("ExtDevSetUDPComParam rtn is " + rtn);
    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;
    rtn = robot.ExtDevGetUDPComParam(ref ip, ref port, ref period, ref lossPkgTime, ref␣
↪lossPkgNum, ref disconnectTime, ref reconnectEnable, ref reconnectPeriod, ref␣
↪reconnectNum);
    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();
    Console.WriteLine("ExtDevGetUDPComParam rtn is " + rtn + param);

    robot.ExtDevLoadUDPDriver();

    rtn = robot.ExtAxisServoOn(1, 1);
    Console.WriteLine("ExtAxisServoOn axis id 1 rtn is " + rtn);
    rtn = robot.ExtAxisServoOn(2, 1);
    Console.WriteLine("ExtAxisServoOn axis id 2 rtn is " + rtn);
    Thread.Sleep(2000);

    rtn = robot.ExtAxisSetHoming(1, 0, 10, 2);
    Console.WriteLine("ExtAxisSetHoming 1 rtnn is  " + rtn);
    Thread.Sleep(2000);
    rtn = robot.ExtAxisSetHoming(2, 0, 10, 2);
    Console.WriteLine("ExtAxisSetHoming 2 rtnn is  " + rtn);

    Thread.Sleep(4000);

    rtn = robot.SetRobotPosToAxis(1);
    Console.WriteLine("SetRobotPosToAxis rtn is " + rtn);
    rtn = robot.SetAxisDHParaConfig(10, 20, 0, 0, 0, 0, 0, 0, 0);
    Console.WriteLine("SetAxisDHParaConfig rtn is " + rtn);
    rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905f, 262144, 200,
↪ 1, 0, 0);
    Console.WriteLine("ExtAxisParamConfig axis 1 rtn is " + rtn);
    rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444f, 262144, 200,
↪ 1, 0, 0);
    Console.WriteLine("ExtAxisParamConfig axis 2 rtn is " + rtn);

    Thread.Sleep(3000);
    robot.ExtAxisStartJog(1, 0, 10, 10, 30);
    Thread.Sleep(1000);
    robot.ExtAxisStopJog(1);
    Thread.Sleep(3000);
    robot.ExtAxisServoOn(1, 0);
```

(continues on next page)

(continued from previous page)

```

    Thread.Sleep(3000);
    robot.ExtAxisStartJog(2, 0, 10, 10, 30);
    Thread.Sleep(1000);
    robot.ExtAxisStopJog(2);
    Thread.Sleep(3000);
    robot.ExtAxisServoOn(2, 0);
    Thread.Sleep(3000);
    robot.ExtDevUnloadUDPDriver();
}
```

2.2.13.31 Set the reference point of the extended axis coordinate system - four-point method

New in version C#SDK-v1.0.7.

```
/**
* @brief Set extended axis coordinate system reference point - four-point method
* @param [in]  pointNum Point number [1-4]
* @return Error code
*/
int ExtAxisSetRefPoint(int pointNum);
```

2.2.13.32 Calculate extended axis coordinate system - four-point method

New in version C#SDK-v1.0.7.

```
/**
* @brief Calculate extended axis coordinate system - four-point method
* @param [out]  coord Coordinate system value
* @return Error code
*/
int ExtAxisComputeECoordSys(DescPose& coord);
```

2.2.13.33 Apply extended axis coordinate system

New in version C#SDK-v1.0.7.

```
/**
* @brief Apply extended axis coordinate system
* @param [in]  applyAxisId Extended axis ID bit0-bit3 correspond to extended axis ID 1-4.
↪ For example, if extended axes 1 and 3 are applied, the value is 0b 0000 0101, which␣
↪is 5
* @param [in]  axisCoordNum Extended axis coordinate system ID
* @param [in]  coord Coordinate system value
* @param [in]  calibFlag Calibration flag 0-no, 1-yes
* @return Error code
*/
int ExtAxisActiveECoordSys(int applyAxisId, int axisCoordNum, DescPose coord, int␣
↪calibFlag);
```

2.2.13.34 Set the calibration reference point in the end coordinate system of the positioner

New in version C#SDK-v1.0.7.

```
/**
* @brief Set the calibration reference point in the end position coordinate system of the positioner
* @param [in] pos Position value
* @return Error code
*/
int SetRefPointInExAxisEnd(DescPose pos);
```

2.2.13.35 Set the reference point of the positioner coordinate system

New in version C#SDK-v1.0.7.

```
/**
* @brief Set the reference point in the positioner coordinate system
* @param [in]  pointNum Point number [1-4]
* @return Error code
*/
int PositionorSetRefPoint(int pointNum);
```

2.2.13.36 Positioner coordinate system calculation - four-point method

New in version C#SDK-v1.0.7.

```
/**
* @brief Positioner coordinate system calculation - four-point method
* @param [out] coord Coordinate system value
* @return Error code
*/
int PositionorComputeECoordSys(DescPose& coord);
```

2.2.13.37 Get extended axis coordinate system

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Get extended axis coordinate system
* @param [out] coord Extended axis coordinate system
* @return Error code
*/
int ExtAxisGetCoord(ref DescPose coord);
```

2.2.13.38 Extended axis coordinate system calibration code example

New in version C#SDK-V1.1.3: Web-3.8.2

```
private void button66_Click(object sender, EventArgs e)
{
    int rtn = robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100,5,1);
    Console.WriteLine("ExtDevSetUDPComParam rtn is " + rtn);
    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;
    rtn = robot.ExtDevGetUDPComParam(ref ip, ref port, ref period, ref lossPkgTime, ref lossPkgNum, ref disconnectTime, ref reconnectEnable, ref reconnectPeriod, ref reconnectNum);
    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();
    Console.WriteLine("ExtDevGetUDPComParam rtn is " + rtn + param);

    robot.ExtDevLoadUDPDriver();

    rtn = robot.ExtAxisServoOn(1, 1);
    Console.WriteLine("ExtAxisServoOn axis id 1 rtn is " + rtn);
    rtn = robot.ExtAxisServoOn(2, 1);
    Console.WriteLine("ExtAxisServoOn axis id 2 rtn is " + rtn);
    Thread.Sleep(2000);

    robot.ExtAxisSetHoming(1, 0, 10, 2);
    Thread.Sleep(2000);
    rtn = robot.ExtAxisSetHoming(2, 0, 10, 2);
    Console.WriteLine("ExtAxisSetHoming rtnn is  " + rtn);

    Thread.Sleep(4000);

    rtn = robot.SetRobotPosToAxis(1);
    Console.WriteLine("SetRobotPosToAxis rtn is " + rtn);
    rtn = robot.SetAxisDHParaConfig(1, 128.5f, 206.4f, 0, 0, 0, 0, 0, 0);
    Console.WriteLine("SetAxisDHParaConfig rtn is " + rtn);
    rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905f, 262144, 200, 1, 0, 0);
    Console.WriteLine("ExtAxisParamConfig axis 1 rtn is " + rtn);
    rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444f, 262144, 200, 1, 0, 0);
    Console.WriteLine("ExtAxisParamConfig axis 1 rtn is " + rtn);

    DescPose toolCoord = new DescPose(0, 0, 210, 0, 0, 0);
    robot.SetToolCoord(1, toolCoord, 0, 0, 1, 0);

    JointPos jSafe = new JointPos(115.193f, -96.149f, 92.489f, -87.068f, -89.15f, -83.488f);
```

(continues on next page)

(continued from previous page)

```
    JointPos j1 = new JointPos(117.559f, -92.624f, 100.329f, -96.909f, -94.057f, -83.
↪488f);
    JointPos j2 = new JointPos(112.239f, -90.096f, 99.282f, -95.909f, -89.824f, -83.
↪488f);
    JointPos j3 = new JointPos(110.839f, -83.473f, 93.166f, -89.22f, -90.499f, -83.487f);
    JointPos j4 = new JointPos(107.935f, -83.572f, 95.424f, -92.873f, -87.933f, -83.
↪488f);

    DescPose descSafe = new DescPose();
    DescPose desc1 = new DescPose();
    DescPose desc2 = new DescPose();
    DescPose desc3 = new DescPose();
    DescPose desc4 = new DescPose();
    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    robot.GetForwardKin( jSafe,  ref descSafe);
    robot.MoveJ( jSafe,  descSafe, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    Thread.Sleep(2000);

    robot.GetForwardKin( j1, ref desc1);
    robot.MoveJ( j1,  desc1, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    Thread.Sleep(2000);

    DescPose actualTCPPos = new DescPose();
    robot.GetActualTCPPose(0, ref actualTCPPos);
    robot.SetRefPointInExAxisEnd(actualTCPPos);
    rtn = robot.PositionorSetRefPoint(1);
    Console.WriteLine("PositionorSetRefPoint 1 rtn is " + rtn);
    Thread.Sleep(2000);

    robot.MoveJ( jSafe,  descSafe, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.ExtAxisStartJog(1, 0, 50, 50, 10);
    Thread.Sleep(1000);
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    Thread.Sleep(1000);
    robot.GetForwardKin( j2, ref desc2);
    rtn = robot.MoveJ( j2,  desc2, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    rtn = robot.PositionorSetRefPoint(2);
    Console.WriteLine("PositionorSetRefPoint 2 rtn is " + rtn);
    Thread.Sleep(2000);

    robot.MoveJ( jSafe,  descSafe, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.ExtAxisStartJog(1, 0, 50, 50, 10);
    Thread.Sleep(1000);
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    Thread.Sleep(1000);
    robot.GetForwardKin( j3, ref desc3);
    robot.MoveJ( j3,  desc3, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    rtn = robot.PositionorSetRefPoint(3);
    Console.WriteLine("PositionorSetRefPoint 3 rtn is " + rtn);
    Thread.Sleep(2000);
```

(continues on next page)

(continued from previous page)

```

    robot.MoveJ( jSafe,  descSafe, 1, 0, 100, 100, 100,  exaxisPos, -1, 0,  offdese);
    robot.ExtAxisStartJog(1, 0, 50, 50, 10);
    Thread.Sleep(1000);
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    Thread.Sleep(1000);
    robot.GetForwardKin(j4, ref desc4);
    robot.MoveJ(j4, desc4, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    rtn = robot.PositionorSetRefPoint(4);
    Console.WriteLine("PositionorSetRefPoint 4 rtn is " + rtn);
    Thread.Sleep(2000);

    DescPose axisCoord = new DescPose();
    robot.PositionorComputeECoordSys(ref axisCoord);
    robot.MoveJ(jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    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);
    rtn = robot.ExtAxisActiveECoordSys(3, 1, axisCoord, 1);
    Console.WriteLine("ExtAxisActiveECoordSys rtn is " + rtn);
}
```

2.2.13.39 Extended axis motion via UDP

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Extended axis motion via UDP
* @param [in] pos Target position
* @param [in] ovl Speed percentage
* @param [in] blend Smoothing parameter (mm or ms)
* @return Error code
*/
int ExtAxisMove(ExaxisPos pos, double ovl, double blend=-1);
```

2.2.13.40 UDP extended axis movement code example

```
private void button66_Click(object sender, EventArgs e)
{
    ExaxisPos axisPos;
    axisPos.ePos[0] = 20;
    axisPos.ePos[1] = 0;
    axisPos.ePos[2] = 0;
    axisPos.ePos[3] = 0;
    robot.ExtAxisMove(axisPos, 50);
}
```

2.2.13.41 UDP extended axis and robot joint motion synchronization

New in version C#SDK-v1.0.7.

```
/**
* @brief UDP extension axis and robot joint motion synchronization
* @param [in] joint_pos Target joint position, unit deg
* @param [in] desc_pos Target Cartesian pose
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] user Workpiece coordinate number, range [0~14]
* @param [in] vel Speed percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], not currently available
* @param [in] ovl Velocity scaling factor, range [0~100]
* @param [in] epos Extended axis position, unit mm
* @param [in] blendT [-1.0]-Move to position (blocked), [0~500.0]-Smooth time (non-blocked), unit ms
* @param [in] offset_flag 0-no offset, 1-offset relative to base coordinate system/workpiece coordinate system, 2-offset relative to tool coordinate system
* @param [in] offset_pos Pose offset
* @return Error code
*/
int ExtAxisSyncMoveJ(JointPos joint_pos, DescPose desc_pos, int tool, int user, float vel, float acc, float ovl, ExaxisPos epos, float blendT, byte offset_flag, DescPose offset_pos);
```

2.2.13.42 Code example

```
private void btnSyncMoveJ_Click(object sender, EventArgs e)
{
    Robot robot = new Robot();
    robot.RPC("192.168.58.2");
    //1. Calibrate and apply the robot tool coordinate system. You can use the four-point method or six-point method to calibrate and apply the tool coordinate system. The interfaces involved in calibrating the tool coordinate system are as follows:

    //    int SetToolPoint(int point_num);  //Set tool reference point - six-point method
    //    int ComputeTool(ref DescPose tcp_pose);  //Calculate tool coordinate system
    //    int SetTcp4RefPoint(int point_num);    //Set tool reference point - four-point method

    //    int ComputeTcp4(ref DescPose tcp_pose);   // Calculate tool coordinate system - four-point method
    //    int SetToolCoord(int id, DescPose coord, int type, int install);  // Set application tool coordinate system
    //    int SetToolList(int id, DescPose coord, int type, int install);   // Set application tool coordinate system list

    //2. Set UDP communication parameters and load UDP communication
    robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10);
    robot.ExtDevLoadUDPDriver();
    //3. Set extended axis parameters, including extended axis type, extended axis driver parameters, and extended axis DH parameters
```

(continues on next page)

(continued from previous page)

```
    robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner␣
↪and DH parameters
    robot.SetRobotPosToAxis(1);  //Extended axis installation position
    robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //␣
↪Servo drive parameters. In this example, a single-axis positioner is used, so only one␣
↪drive parameter needs to be set. If you select an extended axis type with multiple␣
↪axes, each axis must have its drive parameters set.
    //4. Enable the selected axis and return to zero
    robot.ExtAxisServoOn(1, 0);

    robot.ExtAxisSetHoming(1, 0, 20, 3);
    //5. Perform extended axis coordinate system calibration and application (Note: The␣
↪calibration interfaces for positioners and linear rails are different. The following␣
↪is the calibration interface for positioners)
    DescPose pos = new DescPose(/* Enter your calibration point coordinates */);

    robot.SetRefPointInExAxisEnd(pos);
    robot.PositionorSetRefPoint(1); /* You need to calibrate the extended axis using␣
↪four points at different positions, so this interface must be called four times to␣
↪complete the calibration */
    DescPose coord = new DescPose( );
    robot.PositionorComputeECoordSys(ref coord); // Calculate the extended axis␣
↪calibration results

    robot.ExtAxisActiveECoordSys(1, 1, coord, 1);  //Apply the calibration results to␣
↪the extended axis coordinate system
    //6. Calibrate the workpiece coordinate system on the extended axis. You need to use␣
↪the following interfaces
    //int SetWObjCoordPoint(int point_num);

    //int ComputeWObjCoord(int method, ref DescPose wobj_pose);
    //int SetWObjCoord(int id, DescPose coord);
    //int SetWObjList(int id, DescPose coord);
    //7. Record the starting point of your synchronized joint motion

    DescPose startdescPose = new DescPose(/*Enter your coordinates*/);
    JointPos startjointPos = new JointPos(/*Enter your coordinates*/);
    ExaxisPos startexaxisPos = new ExaxisPos(/* Enter your extended axis starting point␣
↪coordinates */);
    //8. Record the endpoint coordinates of your synchronized joint motion

    DescPose enddescPose = new DescPose(/*enter your coordinates*/);
    JointPos endjointPos = new JointPos(/*enter your coordinates*/);
    ExaxisPos endexaxisPos = new ExaxisPos(/* Enter your extended axis endpoint␣
↪coordinates */);
    //9. Write the synchronized motion program
    //Move to the starting point, assuming the tool coordinate system and workpiece␣
↪coordinate system are both 1
    robot.ExtAxisMove(startexaxisPos, 20);

    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
    robot.MoveJ(startjointPos, startdescPose, 1, 1, 100, 100, 100, startexaxisPos, 0, 0,␣
```

(continues on next page)

(continued from previous page)

```
↪offdese);

    //Start synchronized movement
    robot.ExtAxisSyncMoveJ(endjointPos, enddescPose, 1, 1, 100, 100, 100, endexaxisPos, -
↪1, 0, offdese);
}
```

2.2.13.43 UDP extended axis and robot linear motion synchronization

New in version C#SDK-v1.0.7.

```
/**
* @brief UDP extended axis and robot linear motion synchronization
* @param [in] joint_pos  Target joint position, unit: deg
* @param [in] desc_pos   Target Cartesian pose
* @param [in] tool  Tool coordinate number, range[0~14]
* @param [in] user  Workpiece coordinate index, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not currently available
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Move to position (blocked), [0~1000.0]-Smoothing radius
(non-blocking), unit mm
* @param [in] epos  Extended axis position, unit mm
* @param [in] offset_flag  0-no offset, 1-offset relative to base coordinate system/
↪workpiece coordinate system, 2-offset relative to tool coordinate system
* @param [in] offset_pos  Position offset
*/
int ExtAxisSyncMoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, float␣
↪vel, float acc, float ovl, float blendR, ExaxisPos epos, int offset_flag, DescPose␣
↪offset_pos);
```

2.2.13.44 Code example

```
private void btnSyncMoveL_Click(object sender, EventArgs e)
{
Robot robot = new Robot();
robot.RPC("192.168.58.2");

//1. Calibrate and apply the robot tool coordinate system. You can use the four-point␣
↪method or six-point method to calibrate and apply the tool coordinate system. The␣
↪interfaces involved in calibrating the tool coordinate system are as follows:
    //    int SetToolPoint(int point_num);  //Set the tool reference point - six-point␣
↪method

//    int ComputeTool(ref DescPose tcp_pose);  // Calculate tool coordinate system
    //    int SetTcp4RefPoint(int point_num);    // Set tool reference point - four-
↪point method

//    int ComputeTcp4(ref DescPose tcp_pose);   // Calculate tool coordinate system -␣
```

(continues on next page)

(continued from previous page)

```
↪four-point method
    //    int SetToolCoord(int id, DescPose coord, int type, int install);  // Set␣
↪application tool coordinate system
//    int SetToolList(int id, DescPose coord, int type, int install);   // Set␣
↪application tool coordinate system list

//2. Set UDP communication parameters and load UDP communication
robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10);
robot.ExtDevLoadUDPDriver();
//3. Set extended axis parameters, including extended axis type, extended axis driver␣
↪parameters, and extended axis DH parameters
robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner and␣
↪DH parameters
robot.SetRobotPosToAxis(1);  // Extended axis installation position
robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //Servo␣
↪drive parameters. In this example, a single-axis positioner is used, so only one drive␣
↪parameter needs to be set. If you select an extended axis type with multiple axes,␣
↪each axis must have its drive parameters set.
//4.Enable the selected axis and set it to home position.
robot.ExtAxisServoOn(1, 0);
robot.ExtAxisSetHoming(1, 0, 20, 3);
//5. Perform extended axis coordinate system calibration and apply
DescPose pos = new DescPose(/* Enter your calibration point coordinates */);
robot.SetRefPointInExAxisEnd(pos);
robot.PositionorSetRefPoint(1); /* You need to calibrate the extended axis using four␣
↪different points, so you must call this interface four times to complete the␣
↪calibration */

DescPose coord = new DescPose();
robot.PositionorComputeECoordSys(ref coord); // Calculate the extended axis calibration␣
↪results
robot.ExtAxisActiveECoordSys(1, 1, coord, 1);  //Apply the calibration results to the␣
↪extended axis coordinate system

DescPose startdescPose = new DescPose(/*Enter your coordinates*/);

JointPos startjointPos = new JointPos(/*Enter your coordinates*/);
    ExaxisPos startexaxisPos = new ExaxisPos(/* Enter your extended axis starting point␣
↪coordinates */);
//8. Record the endpoint coordinates of your synchronized linear motion
DescPose enddescPose = new DescPose(/*Enter your coordinates*/);

JointPos endjointPos = new JointPos(/*Enter your coordinates*/);
ExaxisPos endexaxisPos = new ExaxisPos(/* Enter your extended axis endpoint coordinates␣
↪*/);
//9. Write the synchronized motion program
//Move to the starting point, assuming that the tool coordinate system and workpiece␣
↪coordinate system are both 1
robot.ExtAxisMove(startexaxisPos, 20);
DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
robot.MoveJ(startjointPos, startdescPose, 1, 1, 100, 100, 100, startexaxisPos, 0, 0,␣
↪offdese);
```

(continues on next page)

(continued from previous page)

```

//Start synchronized movement
robot.ExtAxisSyncMoveL(endjointPos, enddescPose, 1, 1, 100, 100, 100, 0, endexaxisPos, 0,
↪ offdese);
}
```

2.2.13.45 UDP extended axis and robot arc motion synchronization

New in version C#SDK-v1.0.7.

```
/**
* @brief UDP extended axis and robot arc motion synchronization
* @param [in] joint_pos_p  Path point joint position, unit deg
* @param [in] desc_pos_p   Path point Cartesian pose
* @param [in] ptool  Tool coordinate number, range [0~14]
* @param [in] puser  Workpiece coordinate number, range [0~14]
* @param [in] pvel  Speed percentage, range[0~100]
* @param [in] pacc  Acceleration percentage, range [0~100], not currently available
* @param [in] epos_p  Intermediate point extension axis position, unit mm
* @param [in] poffset_flag  0-no offset, 1-offset relative to base coordinate system/
↪workpiece coordinate system, 2-offset relative to tool coordinate system
* @param [in] offset_pos_p  Position offset
* @param [in] joint_pos_t  Target point joint position, unit deg
* @param [in] desc_pos_t   Target point Cartesian position
* @param [in] ttool  Tool coordinate number, range [0~14]
* @param [in] tuser  Workpiece coordinate index, range [0~14]
* @param [in] tvel  Velocity percentage, range [0~100]
* @param [in] tacc  Acceleration percentage, range [0~100], not currently available
* @param [in] epos_t  Extended axis position, unit mm
* @param [in] toffset_flag  0-no offset, 1-offset relative to base coordinate system/
↪workpiece coordinate system, 2-offset relative to tool coordinate system
* @param [in] offset_pos_t Pose offset
* @param [in] ovl Speed scaling factor, range [0~100]
* @param [in] blendR [-1.0]-Move to position (blocked), [0~1000.0]-Smooth radius␣
↪(unblocked), unit mm
* @return Error code
*/
int ExtAxisSyncMoveC(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser,␣
↪float pvel, float pacc, ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p,␣
↪JointPos joint_pos_t, DescPose desc_pos_t, int ttool, int tuser, float tvel, float␣
↪tacc, ExaxisPos epos_t, int toffset_flag, DescPose offset_pos_t, float ovl, float␣
↪blendR);
```

2.2.13.46 Code example

```
private void btnSyncMoveC_Click(object sender, EventArgs e)
{
    Robot robot = new Robot();
    robot.RPC("192.168.58.2");
    //1. Calibrate and apply the robot tool coordinate system. You can use the four-point method or six-point method to calibrate and apply the tool coordinate system. The interfaces involved in calibrating the tool coordinate system are as follows:

    //    int SetToolPoint(int point_num);  //Set tool reference point - six-point method
    //    int ComputeTool(ref DescPose tcp_pose);  //Compute tool coordinate system
    //    int SetTcp4RefPoint(int point_num);    //Set tool reference point - four-point method

    //    int ComputeTcp4(ref DescPose tcp_pose);   // Calculate tool coordinate system - four-point method
    //    int SetToolCoord(int id, DescPose coord, int type, int install);  // Set application tool coordinate system
    //    int SetToolList(int id, DescPose coord, int type, int install);   // Set application tool coordinate system list

    //2. Set UDP communication parameters and load UDP communication
    robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10);
    robot.ExtDevLoadUDPDriver();

    //3. Set the extended axis parameters, including extended axis type, extended axis driver parameters, and extended axis DH parameters
    robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner and DH parameters
    robot.SetRobotPosToAxis(1);  // Extended axis installation position
    robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //Servo drive parameters. In this example, it is a single-axis positioner, so only one drive parameter needs to be set. If you select an extended axis type with multiple axes, each axis must have its drive parameters set

    //4. Enable the selected axis and set it to home position
    robot.ExtAxisServoOn(1, 0);
    robot.ExtAxisSetHoming(1, 0, 20, 3);
    //5. Perform extended axis coordinate system calibration and application

    DescPose pos = new DescPose(/* Enter your calibration point coordinates */);
    robot.SetRefPointInExAxisEnd(pos);
    robot.PositionorSetRefPoint(1); /*You need to calibrate the extended axis using four different points, so you need to call this interface four times to complete the calibration */
    DescPose coord = new DescPose();

    robot.PositionorComputeECoordSys(ref coord); // Calculate the extended axis calibration results
    robot.ExtAxisActiveECoordSys(1, 1, coord, 1);  // Apply the calibration results to the extended axis coordinate system
    //6. Calibrate the workpiece coordinate system on the extended axis. You will need
```

(continues on next page)

(continued from previous page)

```
the following interfaces

    //int SetWObjCoordPoint(int point_num);
    //int ComputeWObjCoord(int method, ref DescPose wobj_pose);
    //int SetWObjCoord(int id, DescPose coord);
    //int SetWObjList(int id, DescPose coord);
    //7. Record the starting point of your synchronized arc motion

    DescPose startdescPose = new DescPose(/*Enter your coordinates*/);
    JointPos startjointPos = new JointPos(/*Enter your coordinates*/);
    ExaxisPos startexaxisPos = new ExaxisPos(/* Enter your extended axis starting point coordinates */);
    //8. Record the endpoint coordinates of your synchronized circular arc motion

    DescPose endDescPose = new DescPose(/*Enter your coordinates*/);
    JointPos endJointPos = new JointPos(/*Enter your coordinates*/);

    ExaxisPos endexaxisPos = new ExaxisPos(/* Enter your extended axis endpoint coordinates */);
    //8. Record your synchronized arc motion midpoint coordinates
    DescPose middescPose = new DescPose(/* Enter your coordinates */);
    JointPos midjointPos = new JointPos(/* Enter your coordinates */);

    ExaxisPos midexaxisPos = new ExaxisPos(/* Enter the extended axis coordinates when the robot is at the arc midpoint */);
    //9. Write the synchronized motion program

    //Move to the starting point, assuming the tool coordinate system and workpiece coordinate system are both 1
    robot.ExtAxisMove(startexaxisPos, 20);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    robot.MoveJ(startjointPos, startdescPose, 1, 1, 100, 100, 100, startexaxisPos, 0, 0, offdese);
    //Start synchronized motion
    robot.ExtAxisSyncMoveC(midjointPos, middescPose, 1, 1, 100, 100, midexaxisPos, 0, offdese, endjointPos, enddescPose, 1, 1, 100, 100, endexaxisPos, 0, offdese, 100, 0);
}
```

2.2.13.47 Set extended DO

New in version C#SDK-v1.0.7.

```
/**
* @brief Set extended DO
* @param [in] DONum DO number
* @param [in] bOpen switch true-on; false-off
* @param [in] smooth whether to smooth
* @param [in] block whether to block
* @return Error code
*/
```

(continues on next page)

(continued from previous page)

```
int SetAuxDO(int DONum, bool bOpen, bool smooth, bool block);
```

2.2.13.48 Set extended AO

New in version C#SDK-v1.0.7.

```
/**
* @brief Set extended AO
* @param [in] AONum AO number
* @param [in] value Analog value [0-4095]
* @param [in] block Whether to block
* @return Error code
*/
int SetAuxAO(int AONum, double value, bool block);
```

2.2.13.49 Set extended DI input filter time

New in version C#SDK-v1.0.7.

```
/**
* @brief Set extended DI input filter time
* @param [in] filterTime Filter time (ms)
* @return Error code
*/
int SetAuxDIFilterTime(int filterTime);
```

2.2.13.50 Set extended AI input filter time

New in version C#SDK-v1.0.7.

```
/**
* @brief Set extended AI input filter time
* @param [in] filterTime Filter time (ms)
* @return Error code
*/
int SetAuxAIFilterTime(int filterTime);
```

2.2.13.51 Wait for extended DI input

New in version C#SDK-v1.0.7.

```
/**
* @brief Wait for extended DI input
* @param [in] DINum DI number
* @param [in] bOpen Switch 0-off; 1-on
* @param [in] time Maximum wait time (ms)
* @param [in] errorAlarm Whether to continue movement
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
int WaitAuxDI(int DINum, bool bOpen, int time, bool errorAlarm);
```

2.2.13.52 Wait for extended AI input

New in version C#SDK-v1.0.7.

```
/**
* @brief Wait for extended AI input
* @param [in] AINum AI number
* @param [in] sign 0-greater than; 1-less than
* @param [in] value AI value
* @param [in] time Maximum wait time (ms)
* @param [in] errorAlarm Continue movement
* @return Error code
*/
int WaitAuxAI(int AINum, int sign, int value, int time, bool errorAlarm);
```

2.2.13.53 Get extended DI value

New in version C#SDK-v1.0.7.

```
/**
* @brief Get extended DI value
* @param [in] DINum DI number
* @param [in] isNoBlock Whether to block
* @param [out] isOpen 0-off; 1-on
* @return Error code
*/
int GetAuxDI(int DINum, bool isNoBlock, bool& isOpen);
```

2.2.13.54 Get extended AI value

New in version C#SDK-v1.0.7.

```
/**
* @brief Get extended AI value
* @param [in] AINum AI number
* @param [in] isNoBlock Whether to block
* @param [in] value Input value
* @return Error code
*/
int GetAuxAI(int AINum, bool isNoBlock, int& value);
```

2.2.13.55 Extended IO code example

```
private void btnAODO_Click(object sender, EventArgs e)
{
    int rtn;
    for (int i = 0; i < 128; i++)
    {
        robot.SetAuxDO(i, true, false, true);
        Thread.Sleep(100);
    }
    for (int i = 0; i < 128; i++)
    {
        robot.SetAuxDO(i, false, false, true);
        Thread.Sleep(100);
    }

    for (int i = 0; i < 409; i++)
    {
        robot.SetAuxAO(0, i * 10, true);
        robot.SetAuxAO(1, 4095 - i * 10, true);
        robot.SetAuxAO(2, i * 10, true);
        robot.SetAuxAO(3, 4095 - i * 10, true);
        Thread.Sleep(10);
    }

    robot.SetAuxDIFilterTime(10);
    robot.SetAuxAIFilterTime(0, 10);

    for (int i = 0; i < 20; i++)
    {
        bool curValue = false;
        rtn = robot.GetAuxDI(i, false, ref curValue);
        Console.WriteLine("DI" + i + "    " + curValue);
    }
    int curValueAI = -1;
    for (int i = 0; i < 4; i++)
    {
        rtn = robot.GetAuxAI(i, true, ref curValueAI);
    }

    robot.WaitAuxDI(1, false, 1000, false);
    robot.WaitAuxAI(1, 1, 132, 1000, false);
}
```

2.2.13.56 Enable movable device

New in version C#SDK-v1.0.9.

```
/**
* @brief Enable movable device
* @param enable false-disable; true-enable
* @return Error code
*/
int TractorEnable(bool enable);
```

2.2.13.57 Stop movable device movement

New in version C#SDK-v1.0.9.

```
/**
* @brief Stop the movement of the movable device
* @return Error code
*/
int TractorStop();
```

2.2.13.58 Reset the movable device

New in version C#SDK-v1.0.9.

```
/**
* @brief Movable device returns to zero
* @return Error code
*/
int TractorHoming();
```

2.2.13.59 Movable device moves in a straight line

New in version C#SDK-v1.0.9.

```
/**
* @brief Movable device moves in a straight line
* @param distance Linear movement distance (mm)
* @param vel Linear movement speed percentage (0-100)
* @return Error code
*/
int TractorMoveL(double distance, double vel);
```

2.2.13.60 Circular motion of a movable device

New in version C#SDK-v1.0.9.

```
/**
* @brief Circular motion of a movable device
* @param radio Circular motion radius (mm)
* @param angle Arc movement angle (°)
* @param vel Linear movement speed percentage (0-100)
* @return Error code
*/
int TractorMoveC(double radio, double angle, double vel);
```

2.2.13.61 Code example

```
private void button6_Click(object sender, EventArgs e)
{
    int rtn;
    robot.ExtDevSetUDPComParam("192.168.58.2", 2021, 2, 50, 5, 50, 1, 50, 10,1);
    robot.ExtDevLoadUDPDriver();
    rtn = robot.ExtAxisServoOn(1, 1);
    rtn = robot.ExtAxisServoOn(2, 1);
    Thread.Sleep(2000);
    robot.ExtAxisSetHoming(1, 0, 10, 2);
    Thread.Sleep(2000);
    rtn = robot.ExtAxisSetHoming(2, 0, 10, 2);
    Thread.Sleep(4000);
    robot.ExtAxisParamConfig(1, 0, 0, 50000, -50000, 1000, 1000, 6.280f, 16384, 200, 0,␣
↪0, 0);
    robot.ExtAxisParamConfig(2, 0, 0, 50000, -50000, 1000, 1000, 6.280f, 16384, 200, 0,␣
↪0, 0);
    robot.SetAxisDHParaConfig(5, 0, 0, 0, 0, 0, 0, 0, 0);
    robot.TractorEnable(false);
    Thread.Sleep(2000);
    robot.TractorEnable(true);
    Thread.Sleep(2000);
    robot.TractorHoming();
    Thread.Sleep(2000);
    robot.TractorMoveL(100, 2);
    Thread.Sleep(5000);
    robot.TractorStop();
    robot.TractorMoveL(-100, 20);
    Thread.Sleep(5000);
    robot.TractorMoveC(300, 90, 20);
    Thread.Sleep(10000);
    robot.TractorMoveC(300, -90, 20);
    Thread.Sleep(1000);
    robot.TractorStop();
}
```

2.2.13.62 Set the synchronous motion strategy of the extension axis and the robot

New in version C#SDK-V1.1.7: Web-3.8.5

```
/**
* @brief Set the synchronous movement strategy of the extended axis with the robot
* @param strategy Strategy; 0 - Robot as the main; 1 - Extended axis synchronous with the robot
* @return Error code */

int SetExAxisRobotPlan(int strategy)
```

2.2.13.63 Code example for setting up the extended axis to move synchronously with the robot

New in version C#SDK-V1.1.7: Web-3.8.5

```
private void button94_Click(object sender, EventArgs e)
{
    JointPos joint_pos1 = new JointPos(-22.016, -49.217, 124.714, -161.100, -85.108, -0.333);
    JointPos joint_pos2 = new JointPos(-21.083, -46.613, 110.079, -147.796, -80.757, -0.330);
    JointPos joint_pos3 = new JointPos(-25.572, -60.090, 135.397, -163.889, -82.489, -0.345);
    DescPose desc_pos1 = new DescPose(2.637, -0.001, 30.673, 178.786, -4.134, 68.326);
    DescPose desc_pos2 = new DescPose(213.812, -1.440, 47.311, 177.410, 0.166, 68.946);
    DescPose desc_pos3 = new DescPose(444.342, -12.723, 82.470, -177.701, -1.325, 65.151);
    ExaxisPos epos1 = new ExaxisPos(0.001, 0.000, 0.000, 0.000);
    ExaxisPos epos2 = new ExaxisPos(299.977, 0.000, 0.000, 0.000);
    ExaxisPos epos3 = new ExaxisPos(399.969, 0.000, 0.000, 0.000);
    DescPose offset_pos = new DescPose(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
    int rtn = robot.SetExAxisRobotPlan(0);
    Console.WriteLine($"SetExAxisRobotPlan rtn is {rtn}");
    Thread.Sleep(1000);
    rtn = robot.ExtAxisSyncMoveL(joint_pos1, desc_pos1, 1, 0, 100, 100, 100, -1, epos1, 0, offset_pos);
    Console.WriteLine($"ExtAxisSyncMoveL 1 rtn is {rtn}");

    rtn = robot.ExtAxisSyncMoveL(joint_pos2, desc_pos2, 1, 0, 100, 100, 100, -1, epos2, 0, offset_pos);
    Console.WriteLine($"ExtAxisSyncMoveL 2 rtn is {rtn}");
    rtn = robot.ExtAxisSyncMoveL(joint_pos3, desc_pos3, 1, 0, 100, 100, 100, -1, epos3, 0, offset_pos);
    Console.WriteLine($"ExtAxisSyncMoveL 3 rtn is {rtn}");
    Thread.Sleep(8000);
}
```

2.2.14 Robotic Welding

2.2.14.1 Setting the welding process parameters

```
/**
* @brief Sets the welding process curve parameters.
* @param [in] id Welding process number (1-99)
* @param [in] startCurrent startCurrent(A)
* @param [in] startVoltage startVoltage(V)
* @param [in] startTime startTime(ms)
* @param [in] weldCurrent weldCurrent(A)
* @param [in] weldVoltage Welding voltage (V)
* @param [in] endCurrent Welding current(A)
* @param [in] endVoltage weldVoltage(V)
* @param [in] endTime Arc recovery time (ms)
* @return Error code
*/
int WeldingSetProcessParam(int id, double startCurrent, double startVoltage, double␣
↪startTime, double weldCurrent, double weldVoltage, double endCurrent, double␣
↪endVoltage, double endTime);
```

2.2.14.2 Get weld process curve parameters

```
/**
* @brief Get weld process curve parameters.
* @param [in] id Welding process number (1-99)
* @param [out] startCurrent startCurrent (A)
* @param [out] startVoltage startVoltage(V)
* @param [out] startTime startTime(ms)
* @param [out] weldCurrent weldCurrent(A)
* @param [out] weldVoltage Welding voltage (V)
* @param [out] endCurrent Welding current(A)
* @param [out] endVoltage weldVoltage(V)
* @param [out] endTime Arc recovery time (ms)
* @return Error code
*/
int WeldingGetProcessParam(int id, ref double startCurrent, ref double startVoltage, ref␣
↪double startTime, ref double weldCurrent, ref double weldVoltage, ref double␣
↪endCurrent, ref double endVoltage, ref double endTime);
```

2.2.14.3 Setting the weld current in relation to the output analog

New in version C#SDK-v1.0.4.

```
/**
* @brief Set weld current to output analog.
* @param [in] currentMin Welding current-analog output linear relationship left point␣
↪current value (A)
* @param [in] currentMax Welding current - analog output linear relationship right point␣
↪current value (A)
```

(continues on next page)

(continued from previous page)

```
* @param [in] outputVoltageMin Welding current - analog output linear relationship left point analog output voltage value (V)
* @param [in] outputVoltageMax Welding current - analog output linear relationship of the right point analog output voltage value (V)
* @return Error Code
*/
int WeldingSetCurrentRelation(double currentMin, double currentMax, double outputVoltageMin, double outputVoltageMax);
```

2.2.14.4 Set Welding Voltage and Output Analog Relation

New in version C#SDK-v1.0.4.

```
/**
* @brief Set weld voltage to output analog.
* @param [in] weldVoltageMin weldVoltageMin Welding voltage-analog output linear relationship between the left point weld voltage value (A)
* @param [in] weldVoltageMax Welding voltage - analog output linear relationship between the right point welding voltage value (A)
* @param [in] outputVoltageMin Welding voltage-analog output linear relationship left point analog output voltage value (V)
* @param [in] outputVoltageMax Welding voltage - analog output linear relationship of the right point analog output voltage value (V)
* @return Error Code
*/
int WeldingSetVoltageRelation(double weldVoltageMin, double weldVoltageMax, double outputVoltageMin, double outputVoltageMax);
```

2.2.14.5 Get the relationship between the welding current and the output analog

New in version C#SDK-v1.0.4.

```
/**
* @brief Get weld current to output analog.
* @param [out] currentMin Welding current-analog output linear relationship of the left point current value (A).
* @param [out] currentMax Welding current - analog output linear relationship right point current value (A)
* @param [out] outputVoltageMin Welding current - analog output linear relationship left point analog output voltage value (V)
* @param [out] outputVoltageMax Welding current - analog output linear relationship between the right point analog output voltage value (V)
* @return Error Code
*/
int WeldingGetCurrentRelation(ref double currentMin, ref double currentMax, ref double outputVoltageMin, ref double outputVoltageMax);
```

2.2.14.6 Get Welding Voltage Relation to Output Analog

New in version C#SDK-v1.0.4.

```
/**
* @brief Get weld voltage to output analog
* @param [out] weldVoltageMin Welding Voltage - Analog Output Linear Relationship Left Point Welding Voltage Value (A)
* @param [out] weldVoltageMax Welding Voltage - Analog Output Linear Relationship Welding Voltage Value (A) at the Right Point
* @param [out] outputVoltageMin Welding voltage-analog output linear relationship left point analog output voltage value (V)
* @param [out] outputVoltageMax Welding voltage - analog output linear relationship right point analog output voltage value (V)
* @return Error Code
*/
int WeldingGetVoltageRelation(ref double weldVoltageMin, ref double weldVoltageMax, ref double outputVoltageMin, ref double outputVoltageMax);
```

2.2.14.7 Set weld current

New in version C#SDK-v1.0.4.

```
/**
* @brief Set weld current.
* @param [in] ioType Control IO type 0-control box IO; 1-expansion IO
* @param [in] current Welding current value (A)
* @param [in] AOIndex Welding current control box analog output port (0-1)
* @return Error Code
*/
int WeldingSetCurrent(int ioType, double current, int AOIndex);
```

2.2.14.8 Set Welding Voltage

New in version C#SDK-v1.0.4.

```
/**
* @brief Set the solder voltage.
* @param [in] ioType Control IO type 0-Control box IO; 1-Extension IO
* @param [in] voltage Welding voltage value (A)
* @param [in] AOIndex Welding voltage control box analog output port (0-1)
* @return Error Code
*/
int WeldingSetVoltage(int ioType, double voltage, int AOIndex);
```

2.2.14.9 Set the swing parameter

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Set the swing parameter
* @param [in] weaveNum The configuration number of the weave parameter.
* @param [in] weaveType Oscillation type 0-planar triangular oscillation; 1-vertical L-shaped triangular oscillation; 2-clockwise circular oscillation; 3-counterclockwise circular oscillation; 4-planar sinusoidal oscillation; 5-vertical L-shaped sinusoidal oscillation; 6-vertical triangular oscillation; 7-vertical sine wave oscillation
* @param [in] weaveFrequency swing frequency (Hz)
* @param [in] weaveIncStayTime wait mode 0-cycle without wait time; 1-cycle with wait time
* @param [in] weaveRange Weave range (mm)
* @param [in] weaveLeftRange vertical triangle swing left chord length (mm)
* @param [in] weaveRightRange the length of the right chord of the vertical triangle swing (mm)
* @param [in] additionalStayTime The stay time of the vertical triangle (mm).
* @param [in] weaveLeftStayTime weaveLeftStayTime (ms)
* @param [in] weaveRightStayTime weaveRightStayTime (ms)
* @param [in] weaveCircleRadio Circle swing-back ratio (0-100%)
* @param [in] weaveStationary swing position wait, 0 - wait time for position to continue moving; 1 - wait time for position to be stationary
* @param [in] weaveYawAngle swing direction azimuth (rotation around swing Z-axis), in °.
* @return Error code
*/
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=0);
```

2.2.14.10 Sample code for setting welding parameters

New in version C#SDK-V1.1.3: Web-3.8.2

```
private void button7_Click(object sender, EventArgs e)
{
   robot.WeldingSetProcessParam(1, 177, 27, 1000, 178, 28, 176, 26, 1000);
   robot.WeldingSetProcessParam(2, 188, 28, 555, 199, 29, 133, 23, 333);

   double startCurrent = 0;
   double startVoltage = 0;
   double startTime = 0;
   double weldCurrent = 0;
   double weldVoltage = 0;
   double endCurrent = 0;
   double endVoltage = 0;
   double endTime = 0;

   robot.WeldingGetProcessParam(1, ref startCurrent, ref startVoltage, ref startTime, ref weldCurrent, ref weldVoltage, ref endCurrent, ref endVoltage, ref endTime);
```

(continues on next page)

(continued from previous page)

```
    Console.WriteLine("the Num 1 process param is " + startCurrent + " " + startVoltage + " " + startTime + " " + weldCurrent + " " + weldVoltage + " " + endCurrent + " " + endVoltage + " " + endTime);
    robot.WeldingGetProcessParam(2, ref startCurrent, ref startVoltage, ref startTime, ref weldCurrent, ref weldVoltage, ref endCurrent, ref endVoltage, ref endTime);
    Console.WriteLine("the Num 2 process param is " + startCurrent + " " + startVoltage + " " + startTime + " " + weldCurrent + " " + weldVoltage + " " + endCurrent + " " + endVoltage + " " + endTime);

    int rtn = robot.WeldingSetCurrentRelation(0, 400, 0, 10, 0);
    Console.WriteLine("WeldingSetCurrentRelation rtn is: " + rtn);

    rtn = robot.WeldingSetVoltageRelation(0, 40, 0, 10, 1);
    Console.WriteLine("WeldingSetVoltageRelation rtn is: " + rtn);

    double current_min = 0;
    double current_max = 0;
    double vol_min = 0;
    double vol_max = 0;
    double output_vmin = 0;
    double output_vmax = 0;
    int curIndex = 0;
    int volIndex = 0;
    rtn = robot.WeldingGetCurrentRelation(ref current_min, ref current_max, ref output_vmin, ref output_vmax, ref curIndex);
    Console.WriteLine("WeldingGetCurrentRelation rtn is: " + rtn);
    Console.WriteLine("current min " + current_min + " current max " + current_max + " output vol min " + output_vmin + " output vol max " + output_vmax);

    rtn = robot.WeldingGetVoltageRelation(ref vol_min, ref vol_max, ref output_vmin, ref output_vmax, ref volIndex);
    Console.WriteLine("WeldingGetVoltageRelation rtn is: " + rtn);
    Console.WriteLine("vol min " + vol_min + " vol max " + vol_max + " output vol min " + output_vmin + " output vol max " + output_vmax);

    rtn = robot.WeldingSetCurrent(1, 100, 0, 0);
    Console.WriteLine("WeldingSetCurrent rtn is: " + rtn);

    System.Threading.Thread.Sleep(3000);

    rtn = robot.WeldingSetVoltage(1, 10, 0, 0);
    Console.WriteLine("WeldingSetVoltage rtn is: " + rtn);

    rtn = robot.WeaveSetPara(0, 0, 2.000000, 0, 10.000000, 0.000000, 0.000000, 0, 0, 0, 0, 0, 60.000000);
    Console.WriteLine("rtn is: " + rtn);

    robot.WeaveOnlineSetPara(0, 0, 1, 0, 20, 0, 0, 0, 0);

    rtn = robot.WeldingSetCheckArcInterruptionParam(1, 200);
    Console.WriteLine("WeldingSetCheckArcInterruptionParam    " + rtn);
    rtn = robot.WeldingSetReWeldAfterBreakOffParam(1, 5.7, 98.2, 0);
```

(continues on next page)

(continued from previous page)

```
   Console.WriteLine("WeldingSetReWeldAfterBreakOffParam    " + rtn);
   int enable = 0;
   double length = 0;
   double velocity = 0;
   int moveType = 0;
   int checkEnable = 0;
   int arcInterruptTimeLength = 0;
   rtn = robot.WeldingGetCheckArcInterruptionParam(ref checkEnable, ref arcInterruptTimeLength);
   Console.WriteLine("WeldingGetCheckArcInterruptionParam  checkEnable  " + checkEnable + "   arcInterruptTimeLength  " + arcInterruptTimeLength);
   rtn = robot.WeldingGetReWeldAfterBreakOffParam(ref enable, ref length, ref velocity, ref moveType);
   Console.WriteLine("WeldingGetReWeldAfterBreakOffParam  enable = " + enable + ", length = " + length + ", velocity = " + velocity + ", moveType = " + moveType);

   robot.SetWeldMachineCtrlModeExtDoNum(17);
   for (int i = 0; i < 5; i++)
   {
      robot.SetWeldMachineCtrlMode(0);
      Thread.Sleep(1000);
      robot.SetWeldMachineCtrlMode(1);
      Thread.Sleep(1000);
   }

}
```

2.2.14.11 Setting the swing parameters on the fly

New in version C#SDK-v1.0.4.

```
/**
* @brief Instant setup of swing parameters
* @param [in] weaveNum The configuration number of the weave parameter.
* @param [in] weaveType Oscillation type 0-planar triangular oscillation; 1-vertical L-shaped triangular oscillation; 2-clockwise circular oscillation; 3-counterclockwise circular oscillation; 4-planar sinusoidal oscillation; 5-vertical L-shaped sinusoidal oscillation; 6-vertical triangular oscillation; 7-vertical sinusoidal oscillation
* @param [in] weaveFrequency swing frequency (Hz)
* @param [in] weaveIncStayTime wait mode 0-cycle without wait time; 1-cycle with wait time
* @param [in] weaveRange Weave range (mm)
* @param [in] weaveLeftStayTime weaveLeftStayTime (ms)
* @param [in] weaveRightStayTime weaveRightStayTime(ms)
* @param [in] weaveCircleRadio Circle swing-back ratio (0-100%)
* @param [in] weaveStationary swing position wait, 0 - wait time for position to continue moving; 1 - wait time for position to be stationary
* @return error code
*/
int WeaveOnlineSetPara(int weaveNum, int weaveType, double weaveFrequency, int weaveIncStayTime, double weaveRange, int weaveLeftStayTime, int weaveRightStayTime,
```

(continues on next page)

(continued from previous page)

```
↪int weaveCircleRadio, int weaveStationary).
```

2.2.14.12 Set parameters for detecting unexpected interruptions of the robot's welding arc

```
/**
* @brief Setting parameters for detecting unexpected interruptions of a robotic welding␣
↪arc.
* @param [in] checkEnable whether or not to enable detection; 0-don't enable; 1-enable
* @param [in] arcInterruptTimeLength arcInterruptAcknowledgmentTimeLength(ms)
* @return Error code
*/
int WeldingSetCheckArcInterruptionParam(int checkEnable, int arcInterruptTimeLength)
```

2.2.14.13 Get parameters for detecting accidental interruptions of the robot's welding arc

```
/**
* @brief Get parameters for detecting unexpected interruptions of a robotic welding arc.
* @param [out] checkEnable Whether to enable detection; 0-don't enable; 1-enable
* @param [out] arcInterruptTimeLength arcInterruptAcknowledgmentTimeLength(ms)
* @return Error code
*/
int WeldingGetCheckArcInterruptionParam(ref int checkEnable, ref int␣
↪arcInterruptTimeLength)
```

2.2.14.14 Set the robot welding interruption recovery parameter

```
/**
* @brief Setting the robot weld interrupt recovery parameters
* @param[in] enable Whether to enable weld interrupt recovery or not.
* @param[in] length The overlap distance of the weld (mm).
* @param[in] velocity the speed of the robot to return to the restart point (0-100)
* @param[in] moveType Robot movement to restart point type; 0-LIN; 1-PTP
* @return Error code
*/
int WeldingSetReWeldAfterBreakOffParam(int enable, double length, double velocity, int␣
↪moveType)
```

2.2.14.15 Get the robot welding break recovery parameter

```
/**
* @brief Get robot weld interrupt recovery parameters.
* @param [out] enable Whether to enable weld interrupt recovery.
* @param [out] length The overlap distance of the weld (mm).
* @param [out] velocity the speed of the robot to return to the restart point (0-100)
* @param [out] moveType the way the robot moves to the restart point; 0-LIN; 1-PTP
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
int WeldingGetReWeldAfterBreakOffParam(ref int enable, ref double length, ref double␣
↪velocity, ref int moveType)
```

2.2.14.16 Set welder control mode extension DO port

```
/**
* @brief Set welder control mode extended DO port.
* @param DONum Welder control mode DO port (0-127)
* @return Error code
*/
int SetWeldMachineCtrlModeExtDoNum(int DONum).
```

2.2.14.17 Set weld machine control mode

```
/**
* @brief Set welder control mode
* @param mode Welder control mode; 0-unitary; 1-binary
* @return Error code
*/
int SetWeldMachineCtrlMode(int mode).
```

2.2.14.18 Weld start

New in version C#SDK-v1.0.4.

```
/**
* @brief Welding start
* @param [in] ioType io type 0-controller IO; 1-extended IO
* @param [in] arcNum Welder profile number
* @param [in] timeout Arc start timeout
* @return Error code
*/
int ARCStart(int ioType, int arcNum, int timeout);
```

2.2.14.19 Weld End

New in version C#SDK-v1.0.4.

```
/**
* @brief End of welding.
* @param [in] ioType io type 0-controller IO; 1-extended IO
* @param [in] arcNum Welder profile number
* @param [in] timeout Arc-off timeout timeout
* @return Error code
*/
int ARCEnd(int ioType, int arcNum, int timeout).
```

2.2.14.20 Oscillation start

New in version C#SDK-v1.0.4.

```
/**
* @brief The start of the swing.
* @param [in] weaveNum Pendulum weld parameter configuration number
* @return ErrorCode
*/
int WeaveStart(int weaveNum).
```

2.2.14.21 WeaveStart(int weaveNum); int WeaveStart(int weaveNum).

New in version C#SDK-v1.0.4.

```
/**
* @brief End of swing
* @param [in] weaveNum Pendulum weld parameter configuration number
* @return ErrorCode
*/
int WeaveEnd(int weaveNum).
```

2.2.14.22 Forward Wire Feed

New in version C#SDK-v1.0.4.

```
/**
* @brief Forward wire feed
* @param [in] ioType io type 0-controller IO; 1-extended IO
* @param [in] wireFeed wire feed control 0-stop wire feed; 1-wire feed
* @return Error code
*/
int SetForwardWireFeed(int ioType, int wireFeed).
```

2.2.14.23 Reverse wire feed

New in version C#SDK-v1.0.4.

```
/**
* @brief Reverse wire feed
* @param [in] ioType ioType 0-Controller IO; 1-Extended IO
* @param [in] wireFeed wireFeedControl 0-Stop wire feed; 1-Wire feed
* @return Error code
*/
int SetReverseWireFeed(int ioType, int wireFeed).
```

2.2.14.24 Wire Feed

New in version C#SDK-v1.0.4.

```
/**
* @brief air delivery
* @param [in] ioType io type 0-controller IO; 1-extended IO
* @param [in] airControl airControl 0-stop air delivery; 1-feed air
* @return Error code
*/
int SetAspirated(int ioType, int airControl).
```

2.2.14.25 Setting the robot to resume welding after an interruption

```
/**
* @brief Set the robot to resume welding after a weld interruption.
* @return Error code
*/
int WeldingStartReWeldAfterBreakOff()
```

2.2.14.26 Sets the robot to exit welding after a weld break

```
/**
* @brief Set the robot to quit welding after a weld interruption.
* @return Error code
*/
int WeldingAbortWeldAfterBreakOff()
```

2.2.14.27 Code example

```
private void button7_Click(object sender, EventArgs e)
{
   robot.WeldingSetCurrent(1, 230, 0, 0);
   robot.WeldingSetVoltage(1, 24, 0, 1);

   DescPose p1Desc = new DescPose(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
   JointPos p1Joint = new JointPos(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);

   DescPose p2Desc = new DescPose(-333.302, -435.580, 449.866, -174.997, 2.017, 109.815);
   JointPos p2Joint = new JointPos(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);

   ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
   DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

   robot.MoveJ(p1Joint, p1Desc, 13, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
   robot.ARCStart(1, 0, 10000);
   robot.WeaveStart(0);
   robot.MoveL (p2Joint, p2Desc, 13, 0, 20, 100, 100, -1, 0, exaxisPos, 0, 0, offdese);
   robot.ARCEnd(1, 0, 10000);
```

(continues on next page)

(continued from previous page)

```
    robot.WeaveEnd(0);
}
```

2.2.14.28 Start of segment welding

New in version C#SDK-v1.0.4.

```
/**
* @brief Start of segment welding
* @param [in] startDesePos start point Cartesian position
* @param [in] endDesePos end point Cartesian position
* @param [in] startJPos start point joint position
* @param [in] endJPos end point joint position
* @param [in] weldLength weldLength (mm)
* @param [in] noWeldLength length (mm)
* @param [in] weldIOType Weld IO type (0-control box IO; 1-expansion IO)
* @param [in] arcNum Welder profile number
* @param [in] weldTimeout start/recovery arc timeout time
* @param [in] isWeave whether or not to wiggle
* @param [in] weaveNum weave parameter configuration number
* @param [in] tool tool number
* @param [in] user Workpiece number
* @param [in] vel velocity percentage, range [0~100] * @param [in] acc.
* @param [in] acc Acceleration percentage, range [0~100], not open yet.
* @param [in] ovl velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-motion in place (blocking), [0~1000.0]-smoothing radius (non-
↪blocking) in mm
* @param [in] epos Extended axis position in mm
* @param [in] search 0-no wire seek, 1-wire seek
* @param [in] offset_flag 0-no offset, 1-offset in base/work coordinate system, 2-offset␣
↪in tool coordinate system
* @param [in] offset_pos Position offset
* @return Error code
*/
int SegmentWeldStart(DescPose startDesePos, DescPose endDesePos, JointPos startJPos,␣
↪JointPos endJPos, double weldLength, double noWeldLength, int weldIOType, int arcNum,␣
↪int weldTimeout,bool isWeave, int weaveNum, int tool, int user, float vel, float acc,␣
↪float ovl, float blendR, ExaxisPos epos, byte search, byte offset_flag, DescPose␣
↪offset_pos);
```

2.2.14.29 Robot segment welding code example

New in version C#SDK-v1.0.4.

```
private void btnWeldStart_Click(object sender, EventArgs e)
{
    robot.WeldingSetCurrent(1, 230, 0, 0);
    robot.WeldingSetVoltage(1, 24, 0, 1);

    DescPose p1Desc = new DescPose(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
```

(continues on next page)

(continued from previous page)

```
    JointPos p1Joint = new JointPos(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);

    DescPose p2Desc = new DescPose(-333.302, -435.580, 449.866, -174.997, 2.017, 109.815);
    JointPos p2Joint = new JointPos(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    int rtn = robot.SegmentWeldStart( p1Desc,  p2Desc,  p1Joint,  p2Joint, 20, 20, 0, 0,␣
↪5000, false, 0, 0, 0, 100, 100, 100, -1,  exaxisPos, 0, 0,  offdese);
    Console.WriteLine("SegmentWeldStart rtn is {0}", rtn);
}
```

2.2.14.30 Simulate the start of the swing

```
/**
* @brief Simulate swing start
* @param [in] weaveNum swing parameter number
* @return ErrorCode
*/
int WeaveStartSim(int weaveNum).
```

2.2.14.31 End of swing simulation

```
/**
* @brief Simulate end of swing
* @param [in] weaveNum swing parameter number
* @return ErrorCode
*/
int WeaveEndSim(int weaveNum).
```

2.2.14.32 Start trajectory detection warning (no motion)

```
/**
* @brief Start trajectory detection warning (no motion).
* @param [in] weaveNum swing parameter number
* @return Error code
*/
int WeaveInspectStart(int weaveNum).
```

2.2.14.33 End trajectory detection warning (no motion)

```
/**
* @brief End trajectory detection warning (no motion).
* @param [in] weaveNum swing parameter number
* @return Error code.
*/
int WeaveInspectEnd(int weaveNum).
```

2.2.14.34 Weave fading start

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Swinging gradient start
* @param [in] weaveChangeFlag 1-change swing parameter; 2-change swing parameter + weld␣
↪speed
* @param [in] weaveNum weaveNumber
* @param [in] velStart Welding start speed, (cm/min)
* @param [in] velEnd Weld end speed, (cm/min)
* @return Error code
*/
int WeaveChangeStart(int weaveChangeFlag, int weaveNum, double velStart, double velEnd);
```

2.2.14.35 Weave fade end

```
/**
* @brief End of swing gradient
* @return Error code
*/
int WeaveChangeEnd()
```

2.2.14.36 Sample Welding Code for Robot Weave Change

New in version C#SDK-V1.1.3: Web-3.8.2

```
private void btnweld_Click(object sender, EventArgs e)
{
   DescPose p1Desc = new DescPose(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
   JointPos p1Joint = new JointPos(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);

   DescPose p2Desc = new DescPose(-333.302, -435.580, 449.866, -174.997, 2.017, 109.815);
   JointPos p2Joint = new JointPos(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);

   ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
   DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

   robot.MoveJ(p1Joint, p1Desc, 13, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
   robot.WeaveStartSim(0);
```

(continues on next page)

(continued from previous page)

```
    robot.MoveL(p2Joint, p2Desc, 13, 0, 20, 100, 100, -1, 0, exaxisPos, 0, 0, offdese);
    robot.WeaveEndSim(0);
    robot.MoveJ(p1Joint, p1Desc, 13, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
    robot.WeaveInspectStart(0);
    robot.MoveL(p2Joint, p2Desc, 13, 0, 20, 100, 100, -1, 0, exaxisPos, 0, 0, offdese);
    robot.WeaveInspectEnd(0);

    robot.WeldingSetVoltage(1, 19, 0, 0);
    robot.WeldingSetCurrent(1, 190, 0, 0);
    robot.MoveL( p1Joint,  p1Desc, 1, 1, 100, 100, 50, -1,  exaxisPos, 0, 0,  offdese);
    robot.ARCStart(1, 0, 10000);
    robot.ArcWeldTraceControl(1, 0, 1, 0.06, 5, 5, 60, 1, 0.06, 5, 5, 80, 0, 0, 4, 1, 10,␣
↪0, 0);
    robot.WeaveStart(0);
    robot.WeaveChangeStart(1, 0, 50, 30);
    robot.MoveL( p2Joint,  p2Desc, 1, 1, 100, 100, 1, -1,  exaxisPos, 0, 0,  offdese);
    robot.WeaveChangeEnd();
    robot.WeaveEnd(0);
    robot.ArcWeldTraceControl(0, 0, 1, 0.06, 5, 5, 60, 1, 0.06, 5, 5, 80, 0, 0, 4, 1, 10,␣
↪0, 0);
    robot.ARCEnd(1, 0, 10000);
}
```

2.2.14.37 Extended IO-Configuration of Welder Gas Detection Signal

```
/**
* @brief Extended IO-Configuration Welder Gas Detection Signal
* @param [in] DONum Gas detection signal extended DO number
* @return error-code
*/
int SetAirControlExtDoNum(int DONum).
```

2.2.14.38 Extended IO-Configuration of the welder arc start signal

```
/**
* @brief Extended IO-Configuration Welder Arc Start Signal
* @param [in] DONum Welder arc start signal extended DO number
* @return Error code.
*/
int SetArcStartExtDoNum(int DONum).
```

2.2.14.39 Extended IO-Configuring the Welder Reverse Wire Feed Signal

```
/**
* @brief Extended IO-Configuration of the welder reverse wire feed signal.
* @param [in] DONum Reverse Wire Feed Signal Extension DO Number
* @return Error code.
*/
int SetWireReverseFeedExtDoNum(int DONum).
```

2.2.14.40 Extended IO-Configuration Welder Forward Wire Feed Signal

```
/**
* @brief Extended IO-Configuration of the welder's forward wire feed signal.
* @param [in] DONum Forward Wire Feed Signal Extension DO Number
* @return Error code.
*/
int SetWireForwardFeedExtDoNum(int DONum);
```

2.2.14.41 Extended IO-Configuration Welder Arc Start Success Signal

```
/**
* @brief Extended IO-Configuration Welder Arc Start Success Signal
* @param [in] DINum Extended DI number for arc success signaling.
* @return Error code.
*/
int SetArcDoneExtDiNum(int DINum).
```

2.2.14.42 Extended IO-Configuration Welder Ready Signal

```
/**
* @brief Extended IO-Configuration Welder Ready Signal
* @param [in] DINum Welder ready signal extended DI number
* @return error-code
*/
int SetWeldReadyExtDiNum(int DINum).
```

2.2.14.43 Extended IO-Configure weld interrupt recovery signal

```
/**
* @brief Extended IO-Configuration Weld Interrupt Recovery Signal
* @param [in] reWeldDINum Resume Weld Signal after Weld Interrupt Extended DI Number
* @param [in] abortWeldDINum Exit Weld Signal Extended DI Number after a Weld Interrupt
* @return Error Code
*/
int SetExtDIWeldBreakOffRecover(int reWeldDINum, int abortWeldDINum);
```

2.2.14.44 Set Extended IO Weld Signal Code Example

```
private void button51_Click(object sender, EventArgs e)
{
   robot.SetArcStartExtDoNum(10);
   robot.SetAirControlExtDoNum(20);
   robot.SetWireForwardFeedExtDoNum(30);
   robot.SetWireReverseFeedExtDoNum(40);

   robot.SetWeldReadyExtDiNum(50);
   robot.SetArcDoneExtDiNum(60);
   robot.SetExtDIWeldBreakOffRecover(70, 80);
   robot.SetWireSearchExtDIONum(0, 1);
}
```

2.2.14.45 Arc tracking control

```
/**
 * @brief Arc tracking control
 * @param [in] flag switch, 0-off; 1-on
 * @param [in] dalayTime hysteresis time, in ms
 * @param [in] isLeftRight left-right deviation compensation
 * @param [in] klr left/right adjustment factor (sensitivity)
 * @param [in] tStartLr left/right start time cyc
 * @param [in] stepMaxLr Maximum compensation in mm per pass * @param [in] stepMaxLr␣
↪Maximum compensation in mm per pass
 * @param [in] sumMaxLr total max compensation mm
 * @param [in] isUpLow Compensation for up and down deviation
 * @param [in] kud up/down adjustment factor (sensitivity)
 * @param [in] tStartUd up/down compensation time cyc
 * @param [in] stepMaxUd Maximum compensation in mm for each step.
 * @param [in] sumMaxUd up and down total max. compensation mm
 * @param [in] axisSelect upper and lower coordinate system selection, 0-swing; 1-tool;␣
↪2-base
 * @param [in] referenceType upper and lower reference current setting mode, 0-feedback;␣
↪1-constant
 * @param [in] referSampleStartUd upper and lower reference current sample start count␣
↪(feedback), cyc
 * @param [in] referSampleCountUd upper and lower reference current sampling cycle count␣
↪(feedback), cyc
 * @param [in] referenceCurrent upper and lower reference current mA
 * @param [in] offsetType offset tracking type, 0-no offset; 1-sample; 2-percentage /
↪version 3.7.9
 * @param [in] offsetParameter offset parameter; sampling (offset sampling start time,␣
↪default one cycle); percentage (offset percentage (-100 ~ 100)) /version 3.7.9
 * @return Error code
 */
int ArcWeldTraceControl(int flag, double delaytime, int isLeftRight, double klr, double␣
↪tStartLr, double stepMaxLr, double sumMaxLr, int isUpLow, double kud, double tStartUd,␣
↪double stepMaxUd, double sumMaxUd, int axisSelect, int referenceType, double␣
↪referSampleStartUd, double referSampleCountUd, double referenceCurrent, int offsetType,
↪ int offsetParameter).
```

2.2.14.46 Arc tracking AI passband selection

```
/**
* @brief Arc-tracking AI passband selection.
* @param [in] channel Arc-tracking AI passband selection, [0-3].
* @return Error code.
*/
int ArcWeldTraceExtAIChannelConfig(int channel).
```

2.2.14.47 Arc Trace + Multi-Layer Multi-Channel Compensation on

```
/**
* @brief Arc tracing + multilayer compensation on.
* @return Error code
*/
int ArcWeldTraceReplayStart();
```

2.2.14.48 ArcWeldTrace + MultiLayerMultiChannelCompensation OFF

```
/**
* @brief Arc tracing + multilayer multichannel compensation off .
* @return Error code
*/
int ArcWeldTraceReplayEnd();
```

2.2.14.49 Offset Coordinate Change - Multi-Layer Multi-Pass Welding

```
 /**
 * @brief Offset Coordinate Change - Multi-Layer Multi-Pass Weld
 * @param [in] point0 Cartesian orientation of datum.
 * @param [in] pointX Cartesian position of datum X in offset direction.
 * @param [in] pointZ Cartesian position of the datum Z to the offset direction point.
 * @param [in] dx x-direction offset (mm)
 * @param [in] z offset (mm)
 * @param [in] offset around y-axis (°)
 * @param [out] Calculation result offset (mm)
 * @return Error Code
 */
int MultilayerOffsetTrsfToBase(DescTran point0, DescTran pointX, DescTran pointZ, double␣
↪dx, double dy, double db, ref DescPose offset);
```

2.2.14.50 Multi-layer multi-pass welding arc tracking code example

```
private void button52_Click(object sender, EventArgs e)
{
   JointPos mulitilineorigin1_joint = new JointPos(-24.090, -63.501, 84.288, -111.940, -93.426, 57.669);
   DescPose mulitilineorigin1_desc = new DescPose(-677.559, 190.951, -1.205, 1.144, -41.482, -82.577);

   DescTran mulitilineX1_desc = new DescTran();
   mulitilineX1_desc.x = -677.556;
   mulitilineX1_desc.y = 211.949;
   mulitilineX1_desc.z = -1.206;

   DescTran mulitilineZ1_desc = new DescTran();
   mulitilineZ1_desc.x = -677.564;
   mulitilineZ1_desc.y = 190.956;
   mulitilineZ1_desc.z = 19.817;

   JointPos mulitilinesafe_joint = new JointPos(-25.734, -63.778, 81.502, -108.975, -93.392, 56.021);
   DescPose mulitilinesafe_desc = new DescPose(-677.561, 211.950, 19.812, 1.144, -41.482, -82.577);
   JointPos mulitilineorigin2_joint = new JointPos(-29.743, -75.623, 101.241, -116.354, -94.928, 55.735);
   DescPose mulitilineorigin2_desc = new DescPose(-563.961, 215.359, -0.681, 2.845, -40.476, -87.443);

   DescTran mulitilineX2_desc = new DescTran();
   mulitilineX2_desc.x = -563.965;
   mulitilineX2_desc.y = 220.355;
   mulitilineX2_desc.z = -0.680;

   DescTran mulitilineZ2_desc = new DescTran();
   mulitilineZ2_desc.x = -563.968;
   mulitilineZ2_desc.y = 215.362;
   mulitilineZ2_desc.z = 4.331;

   ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);
   DescPose offset = new DescPose(0, 0, 0, 0, 0, 0);

   Thread.Sleep(10);
   int error = robot.MoveJ( mulitilinesafe_joint,  mulitilinesafe_desc, 13, 0, 10, 100, 100,  epos, -1, 0,  offset);
   Console.WriteLine("MoveJ return: {0}", error);

   error = robot.MoveL( mulitilineorigin1_joint,  mulitilineorigin1_desc, 13, 0, 10, 100, 100, -1,  epos, 0, 0,  offset, 0, 100);
   Console.WriteLine("MoveL return: {0}", error);

   error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100, epos, -1, 0, offset);
   Console.WriteLine("MoveJ return: {0}", error);
```

(continues on next page)

(continued from previous page)

```
43
44    error = robot.MoveL(mulitilineorigin2_joint, mulitilineorigin2_desc, 13, 0, 10, 100,␣
↪100, -1,  epos, 0, 0,  offset, 0, 100);
45    Console.WriteLine("MoveL return: {0}", error);
46
47    error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
↪epos, -1, 0, offset);
48    Console.WriteLine("MoveJ return: {0}", error);
49
50    error = robot.MoveL( mulitilineorigin1_joint,  mulitilineorigin1_desc, 13, 0, 10, 100,
↪ 100, -1,  epos, 0, 0,  offset, 0, 100);
51    Console.WriteLine("MoveL return: {0}", error);
52
53    error = robot.ARCStart(1, 0, 3000);
54    Console.WriteLine("ARCStart return: {0}", error);
55
56    error = robot.WeaveStart(0);
57    Console.WriteLine("WeaveStart return: {0}", error);
58
59    error = robot.ArcWeldTraceControl(1, 0, 1, 0.06, 5, 5, 50, 1, 0.06, 5, 5, 55, 0, 0, 4,
↪ 1, 10);
60    Console.WriteLine("ArcWeldTraceControl return: {0}", error);
61
62    error = robot.MoveL( mulitilineorigin2_joint,  mulitilineorigin2_desc, 13, 0, 1, 100,␣
↪100, -1,  epos, 0, 0,  offset, 0, 100);
63    Console.WriteLine("MoveL return: {0}", error);
64
65    error = robot.ArcWeldTraceControl(0, 0, 1, 0.06, 5, 5, 50, 1, 0.06, 5, 5, 55, 0, 0, 4,
↪ 1, 10);
66    Console.WriteLine("ArcWeldTraceControl return: {0}", error);
67
68    error = robot.WeaveEnd(0);
69    Console.WriteLine("WeaveEnd return: {0}", error);
70
71    error = robot.ARCEnd(1, 0, 10000);
72    Console.WriteLine("ARCEnd return: {0}", error);
73
74    error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
↪epos, -1, 0, offset);
75    Console.WriteLine("MoveJ return: {0}", error);
76
77    error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin1_desc.tran, mulitilineX1_
↪desc, mulitilineZ1_desc, 10.0, 0.0, 0.0, ref offset);
78    Console.WriteLine("MultilayerOffsetTrsfToBase return: {0}  offect is {1} {2} {3}",␣
↪error, offset.tran.x, offset.tran.y, offset.tran.z);
79
80    error = robot.MoveL( mulitilineorigin1_joint,  mulitilineorigin1_desc, 13, 0, 10, 100,
↪ 100, -1,  epos, 0, 1,  offset, 0, 100);
81    Console.WriteLine("MoveL return: {0}", error);
82
83    error = robot.ARCStart(1, 0, 3000);
84    Console.WriteLine("ARCStart return: {0}", error);
```

(continues on next page)

(continued from previous page)

```
85
86    error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin2_desc.tran, mulitilineX2_
↪desc, mulitilineZ2_desc, 10, 0, 0, ref offset);
87    Console.WriteLine("MultilayerOffsetTrsfToBase return: {0}  offect is {1} {2} {3}",␣
↪error, offset.tran.x, offset.tran.y, offset.tran.z);
88
89    error = robot.ArcWeldTraceReplayStart();
90    Console.WriteLine("ArcWeldTraceReplayStart return: {0}", error);
91
92    error = robot.MoveL( mulitilineorigin2_joint,  mulitilineorigin2_desc, 13, 0, 2, 100,␣
↪100, -1,  epos, 0, 1,  offset, 0, 100);
93    Console.WriteLine("MoveL return: {0}", error);
94
95    error = robot.ArcWeldTraceReplayEnd();
96    Console.WriteLine("ArcWeldTraceReplayEnd return: {0}", error);
97
98    error = robot.ARCEnd(1, 0, 10000);
99    Console.WriteLine("ARCEnd return: {0}", error);
100
101   error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
↪epos, -1, 0, offset);
102   Console.WriteLine("MoveJ return: {0}", error);
103
104   error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin1_desc.tran, mulitilineX1_
↪desc, mulitilineZ1_desc, 0, 10, 0, ref offset);
105   Console.WriteLine("MultilayerOffsetTrsfToBase return: {0}  offect is {1} {2} {3}",␣
↪error, offset.tran.x, offset.tran.y, offset.tran.z);
106
107   error = robot.MoveL( mulitilineorigin1_joint,  mulitilineorigin1_desc, 13, 0, 10, 100,
↪ 100, -1,  epos, 0, 1,  offset, 0, 100);
108   Console.WriteLine("MoveL return: {0}", error);
109
110   error = robot.ARCStart(1, 0, 3000);
111   Console.WriteLine("ARCStart return: {0}", error);
112
113   error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin2_desc.tran, mulitilineX2_
↪desc, mulitilineZ2_desc, 0, 10, 0, ref offset);
114   Console.WriteLine("MultilayerOffsetTrsfToBase return: {0}  offect is {1} {2} {3}",␣
↪error, offset.tran.x, offset.tran.y, offset.tran.z);
115
116   error = robot.ArcWeldTraceReplayStart();
117   Console.WriteLine("MoveJ return: {0}", error);
118
119   error = robot.MoveL(mulitilineorigin2_joint, mulitilineorigin2_desc, 13, 1, 2, 100,␣
↪100, -1, epos, 1, 1, offset, 1, 100);
120   Console.WriteLine("MoveL return: {0}", error);
121
122   error = robot.ArcWeldTraceReplayEnd();
123   Console.WriteLine("ArcWeldTraceReplayEnd return: {0}", error);
124
125   error = robot.ARCEnd(1, 0, 3000);
126   Console.WriteLine("ARCEnd return: {0}", error);
```

(continues on next page)

(continued from previous page)

```
127
128   error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
↪epos, -1, 0, offset);
129   Console.WriteLine("MoveJ return: {0}", error);
130 }
```

2.2.14.51 Arc Tracking Welder Current Feedback AI Channel Selection

New in version C#SDK-V1.1.3: Web-3.8.2

```
1 /**
2  * @brief Arc tracking welder current feedback AI channel selection
3  * @param [in] channel channel; 0-extension AI0; 1-extension AI1; 2-extension AI2; 3-
↪extension AI3; 4-control box AI0; 5-control box AI1
4  * @return error code
5  */
6 int ArcWeldTraceAIChannelCurrent(int channel).
```

2.2.14.52 Arc tracking welder voltage feedback AI channel selection

New in version C#SDK-V1.1.3: Web-3.8.2

```
1 /**
2  * @brief Arc Tracking Welder Voltage Feedback AI Channel Selection
3  * @param [in] channel channel; 0-extension AI0; 1-extension AI1; 2-extension AI2; 3-
↪extension AI3; 4-control box AI0; 5-control box AI1
4  * @return error code
5  */
6 int ArcWeldTraceAIChannelVoltage(int channel).
```

2.2.14.53 Arc tracking welder current feedback conversion parameters

New in version C#SDK-V1.1.3: Web-3.8.2

```
1 /**
2  * @brief Arc tracking welder current feedback conversion parameters
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] currentLow AI channel lower limit corresponds to the welder current value,
↪ default value 0V, range [0-200V].
6  * @param [in] currentHigh AI channel upper limit corresponds to the welder current␣
↪value, default value 100V, range [0-200V].
7  * @return Error code
8  */
9 int ArcWeldTraceCurrentPara(float AILow, float AIHigh, float currentLow, float␣
↪currentHigh);
```

2.2.14.54 Arc Trace Welder Voltage Feedback Conversion Parameters

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
 * @brief Arc Tracker Welder Voltage Feedback Conversion Parameters
 * @param [in] AILow AI channel lower limit, default value 0V, range [0-10V].
 * @param [in] AIHigh AI channel upper limit, default value 10V, range [0-10V].
 * @param [in] voltageLow AI channel lower limit corresponds to the welder voltage value, default value 0V, range [0-200V].
 * @param [in] voltageHigh AI channel upper limit corresponds to the welder voltage value, default value 100V, range [0-200V] * @param [in] voltageHigh AI channel upper limit corresponds to the welder voltage value, default value 100V, range [0-200V
 * @return Error code
 */
int ArcWeldTraceVoltagePara(float AILow, float AIHigh, float voltageLow, float voltageHigh);
```

2.2.14.55 Arc Trace Code Example

New in version C#SDK-V1.1.3: Web-3.8.2

```
private void btnweld_Click(object sender, EventArgs e)
{
   DescPose safetydescPose = new DescPose(-504.043, 275.181, 40.908, -28.002, -42.025, -14.044);
   JointPos safetyjointPos = new JointPos(-39.078, -76.732, 87.227, -99.47, -94.301, 18.714);
   DescPose startdescPose = new DescPose(-473.86, 257.879, -20.849, -37.317, -42.021, 2.543);
   JointPos startjointPos = new JointPos(-43.487, -76.526, 95.568, -104.445, -89.356, 3.72);



   DescPose enddescPose = new DescPose(-499.844, 141.225, 7.72, -34.856, -40.17, 13.13);
   JointPos endjointPos = new JointPos(-31.305, -82.998, 99.401, -104.426, -89.35, 3.696);

   ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
   DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
   robot.MoveJ(safetyjointPos, safetydescPose, 1, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);

   robot.WeldingSetCurrentRelation(0, 495, 1, 10, 0);
   robot.WeldingSetVoltageRelation(10, 45, 1, 10, 1);
   robot.WeldingSetVoltage(0, 25, 1, 0);// ----
   robot.WeldingSetCurrent(0, 260, 0, 0);// ----

   int rtn = robot.ArcWeldTraceAIChannelCurrent(4);
   Console.WriteLine("ArcWeldTraceAIChannelCurrent rtn is " + rtn);
   rtn = robot.ArcWeldTraceAIChannelVoltage(5);
```

(continues on next page)

(continued from previous page)

```
    Console.WriteLine("ArcWeldTraceAIChannelVoltage rtn is " + rtn);
    rtn = robot.ArcWeldTraceCurrentPara((float)0, (float)5, (float)0, (float)500);
    Console.WriteLine("ArcWeldTraceCurrentPara rtn is " + rtn);
    rtn = robot.ArcWeldTraceVoltagePara((float)1.018, (float)10, (float)0, (float)50);
    Console.WriteLine("ArcWeldTraceVoltagePara rtn is " + rtn);

    robot.MoveJ(startjointPos, startdescPose, 1, 0, 20, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1,␣
↪10, 0, 0);
    robot.ARCStart(0, 0, 10000);
    robot.WeaveStart(0);
    // robot.MoveL(endjointPos, enddescPose, 1, 0, 100, 100, 2, -1, exaxisPos, 0, 0,␣
↪offdese);
    robot.ARCEnd(0, 0, 10000);
    robot.WeaveEnd(0);
    robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1,␣
↪10, 0, 0);
    robot.MoveJ(safetyjointPos, safetydescPose, 1, 0, 20, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
}
```

2.2.14.56 Start of wire position finding

```
/**
 * @brief Welding wire seek start.
 * @param [in] refPos 1-datum point 0-contact point
 * @param [in] searchVel SearchVelocity %.
 * @param [in] searchDis Search distance mm
 * @param [in] autoBackFlag autoBackFlag, 0 - not auto; - auto
 * @param [in] autoBackVel autoBackVelocity % * @param [in] searchDis searchDistance mm
 * @param [in] autoBackDis autoBack distance mm
 * @param [in] offectFlag 1-with offset seek; 0-teach point seek
 * @return error code
 */
int WireSearchStart(int refPos, double searchVel, int searchDis, int autoBackFlag,␣
↪double autoBackVel, int autoBackDis, int offectFlag);
```

2.2.14.57 End of wire search

```
/**
* @brief End of wire seek.
* @param [in] refPos 1-datum point 2-contact point
* @param [in] searchVel searchVelocity %
* @param [in] searchDis Search distance mm
* @param [in] autoBackFlag autoBackFlag, 0 - not auto; - auto
* @param [in] autoBackVel autoBackVelocity % * @param [in] searchDis searchDistance mm
* @param [in] autoBackDis autoBack distance mm
* @param [in] offectFlag 1-with offset seek; 2-teach point seek
```

(continues on next page)

(continued from previous page)

```
* @return error code
*/
int WireSearchEnd(int refPos, double searchVel, int searchDis, int autoBackFlag, double␣
↪autoBackVel, int autoBackDis, int offectFlag);
```

2.2.14.58 Calculate the wire seek offset

```
/**
* @brief Calculate wire-seeking offsets.
* @param [in] seamType weldType
* @param [in] method Calculate the method.
* @param [in] varNameRef datums 1-6, "#" means no point variable
* @param [in] varNameRes contact points 1-6, "#" means no point variable
* @param [out] offectFlag 0-offset directly superimposed on the command point; 1-offset␣
↪requires a coordinate transformation of the command point
* @param [out] offect Offset pose [x, y, z, a, b, c]
* @return Error code
*/
int GetWireSearchOffset(int seamType, int method, string[] varNameRef, string[]␣
↪varNameRes, ref int offsetFlag, ref DescPose offset);;
```

2.2.14.59 Wait for the wire search to complete

```
/**
* @brief Waiting for wire seek to complete.
* @return Error code.
*/
int WireSearchWait(string name).
```

2.2.14.60 Write wire search contact to database.

```
/**
* @brief Welding wire seek contacts written to database.
* @param [in] varName Contact name "RES0" ~ "RES99".
* @param [in] pos contact data [x, y, x, a, b, c]
* @return Error Code
*/
int SetPointToDatabase(string varName, DescPose pos);
```

2.2.14.61 Robot Welding Wire Position Finding Code Example

```
private void button53_Click(object sender, EventArgs e)
{
   DescPose toolCoord=new DescPose(0, 0, 200, 0, 0, 0);
   robot.SetToolCoord(1, toolCoord, 0, 0, 1, 0);
   DescPose wobjCoord=new DescPose(0, 0, 0, 0, 0, 0);
   robot.SetWObjCoord(1, wobjCoord, 0);

   int rtn0, rtn1, rtn2 = 0;
   ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
   DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

   DescPose descStart = new DescPose(216.543, 445.175, 93.465, 179.683, 1.757, -112.641);
   JointPos jointStart = new JointPos(-128.345, -86.660, 114.679, -119.625, -89.219, 74.
↪303);

   DescPose descEnd = new DescPose(111.143, 523.384, 87.659, 179.703, 1.835, -97.750);
   JointPos jointEnd = new JointPos(-113.454, -81.060, 109.328, -119.954, -89.218, 74.
↪302);

   robot.MoveL(jointStart, descStart, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
   robot.MoveL(jointEnd, descEnd, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);

   DescPose descREF0A = new DescPose(142.135, 367.604, 86.523, 179.728, 1.922, -111.089);
   JointPos jointREF0A = new JointPos(-126.794, -100.834, 128.922, -119.864, -89.218, 74.
↪302);

   DescPose descREF0B = new DescPose(254.633, 463.125, 72.604, 179.845, 2.341, -114.704);
   JointPos jointREF0B = new JointPos(-130.413, -81.093, 112.044, -123.163, -89.217, 74.
↪303);

   DescPose descREF1A = new DescPose(92.556, 485.259, 47.476, -179.932, 3.130, -97.512);
   JointPos jointREF1A = new JointPos(-113.231, -83.815, 119.877, -129.092, -89.217, 74.
↪303);

   DescPose descREF1B = new DescPose(203.103, 583.836, 63.909, 179.991, 2.854, -103.372);
   JointPos jointREF1B = new JointPos(-119.088, -69.676, 98.692, -121.761, -89.219, 74.
↪303);

   rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
   robot.MoveL(jointREF0A, descREF0A, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
↪  //
   robot.MoveL(jointREF0B, descREF0B, 1, 1, 100, 100, 100, -1, exaxisPos, 1, 0, offdese);
↪  //
   rtn1 = robot.WireSearchWait("REF0");
   rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

   rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
   robot.MoveL(jointREF1A, descREF1A, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
↪  //
   robot.MoveL(jointREF1B, descREF1B, 1, 1, 100, 100, 100, -1, exaxisPos, 1, 0, offdese);
↪  //
```

(continues on next page)

(continued from previous page)

```
    rtn1 = robot.WireSearchWait("REF1");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF0A, descREF0A, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
↪   //
    robot.MoveL(jointREF0B, descREF0B, 1, 1, 100, 100, 100, -1, exaxisPos, 1, 0, offdese);
↪   //
    rtn1 = robot.WireSearchWait("RES0");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF1A, descREF1A, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
↪   //
    robot.MoveL(jointREF1B, descREF1B, 1, 1, 100, 100, 100, -1, exaxisPos, 1, 0, offdese);
↪   //
    rtn1 = robot.WireSearchWait("RES1");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    string[] varNameRef = { "REF0", "REF1", "#", "#", "#", "#" };
    string[] varNameRes = { "RES0", "RES1", "#", "#", "#", "#" };
    int offectFlag = 0;
    DescPose offectPos = new DescPose(0, 0, 0, 0, 0, 0);
    rtn0 = robot.GetWireSearchOffset(0, 0, varNameRef, varNameRes, ref offectFlag, ref␣
↪offectPos);
    robot.PointsOffsetEnable(0, offectPos);
    robot.MoveL(jointStart, descStart, 1, 1, 100, 100, 100, -1, exaxisPos, 0, 0, offdese);
    robot.MoveL(jointEnd, descEnd, 1, 1, 100, 100, 100, -1, exaxisPos, 1, 0, offdese);
    robot.PointsOffsetDisable();
}
```

2.2.14.62 Setting the end of the welding voltage gradient

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Sets the weld voltage gradient to start.
* @param [in] IOType Control type; 0-control box IO; 1-digital communication protocol␣
↪(UDP); 2-digital communication protocol (ModbusTCP)
* @param [in] voltageStart Start welding voltage (V)
* @param [in] voltageEnd End weld voltage (V)
* @param [in] AOIndex control box AO port number (0-1)
* @param [in] blend Whether smooth or not 0-not smooth; 1-smooth
* @return Error code
*/
int WeldingSetVoltageGradualChangeStart(int IOType, double voltageStart, double␣
↪voltageEnd, int AOIndex, int blend);
```

2.2.14.63 Set weld voltage gradual change end

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Setting the end of the welding voltage gradient.
* @return Error Code
*/
int WeldingSetVoltageGradualChangeEnd();
```

2.2.14.64 Sets the weld current gradual change end

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Setting the weld current gradient to start.
* @param [in] IOType Control type; 0-control box IO; 1-digital communication protocol␣
↪(UDP); 2-digital communication protocol (ModbusTCP)
* @param [in] voltageStart Start welding current (A)
* @param [in] voltageEnd End weld current (A)
* @param [in] AOIndex Control box AO port number (0-1)
* @param [in] blend Whether smooth or not 0-not smooth; 1-smooth
* @return Error code
*/
int WeldingSetCurrentGradualChangeStart(int IOType, double currentStart, double␣
↪currentEnd, int AOIndex, int blend);
```

2.2.14.65 Set weld current gradual change end

New in version C#SDK-V1.1.3: Web-3.8.2

```
/**
* @brief Setting the end of the welding current gradient.
* @return Error code
*/
int WeldingSetCurrentGradualChangeEnd();
```

2.2.14.66 Robot Welding Current Voltage Gradual Change Code Example

New in version C#SDK-V1.1.3: Web-3.8.2

```
 private void btnweld_Click(object sender, EventArgs e)
 {
    DescPose startdescPose = new DescPose(-319.303, -240.689, 116.379, -175.879, -0.337,
↪ 148.239);
    JointPos startjointPos = new JointPos(20.474, -103.554, 126.774, -116.682, -87.746,␣
↪-37.709);

    DescPose enddescPose = new DescPose(-454.166, -327.159, 62.217, 177.199, -2.276,␣
↪154.955);
```

(continues on next page)

(continued from previous page)

```
    JointPos endjointPos = new JointPos(27.176, -74.423, 104.557, -119.315, -93.514, -37.698);

    DescPose safedescPose = new DescPose(-375.533, -543.319, 19.798, 177.486, -2.489, 175.825);
    JointPos safejointPos = new JointPos(48.074, -59.714, 89.955, -119.777, -93.508, -37.683);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    robot.WeldingSetCurrentRelation(0, 495, 1, 10, 0);
    robot.WeldingSetVoltageRelation(10, 45, 1, 10, 1);

    robot.WeldingSetVoltage(0, 25, 1, 0);//
    robot.WeldingSetCurrent(0, 260, 0, 0);//

    robot.MoveJ(safejointPos, safedescPose, 1, 0, 5, 100, 100, exaxisPos, -1, 0, offdese);

    int rtn = robot.WeldingSetCurrentGradualChangeStart(0, 260, 220, 0, 0);
    Console.WriteLine($"WeldingSetCurrentGradualChangeStart rtn is {rtn}");
    rtn = robot.WeldingSetVoltageGradualChangeStart(0, 25, 22, 1, 0);
    Console.WriteLine($"WeldingSetVoltageGradualChangeStart rtn is {rtn}");

    rtn = robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
    Console.WriteLine($"ArcWeldTraceControl rtn is {rtn}");

    robot.MoveJ(startjointPos, startdescPose, 1, 0, 5, 100, 100, exaxisPos, -1, 0, offdese);

    robot.ARCStart(0, 0, 10000);
    robot.WeaveStart(0);
    rtn = robot.WeaveChangeStart(2, 1, 24, 36);
    Console.WriteLine($"WeaveChangeStart rtn is {rtn}");
    //robot.MoveL(endjointPos, enddescPose, 1, 0, 100, 100, 2, -1, exaxisPos, 0, 0, offdese);
    robot.ARCEnd(0, 0, 10000);
    robot.WeaveChangeEnd();
    robot.WeaveEnd(0);
    robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0);
    robot.WeldingSetCurrentGradualChangeEnd();
    robot.WeldingSetVoltageGradualChangeEnd();
  }
```

2.2.14.67 Set custom weaving parameters

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Set custom weaving parameters
* @param [in] id Custom weaving number: 0-2
* @param [in] pointNum Number of weaving points: 0-10
* @param [in] point Moving endpoint data x,y,z
* @param [in] stayTime Weaving dwell time in ms
* @param [in] frequency Weaving frequency in Hz
* @param [in] incStayType Wait mode: 0-cycle does not include wait time; 1-cycle␣
↪includes wait time
* @param [in] stationary Weaving position wait: 0-continue motion during wait time; 1-
↪position static during wait time
* @return  Error code
*/
public int CustomWeaveSetPara(int id, int pointNum, DescTran[] points, double[]␣
↪stayTimes, double frequency, int incStayType, int stationary)
```

2.2.14.68 Get custom weaving parameters

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
* @brief Get custom weaving parameters
* @param [in] id Custom weaving number: 0-2
* @param [out] pointNum Number of weaving points: 0-10
* @param [out] point Moving endpoint data x,y,z
* @param [out] stayTime Weaving dwell time in ms
* @param [out] frequency Weaving frequency in Hz
* @param [out] incStayType Wait mode: 0-cycle does not include wait time; 1-cycle␣
↪includes wait time
* @param [out] stationary Weaving position wait: 0-continue motion during wait time; 1-
↪position static during wait time
* @return  Error code
*/
public int CustomWeaveGetPara(int id, ref int pointNum, ref DescTran[] points, ref␣
↪double[] stayTimes, ref double frequency, ref int incStayType, ref int stationary)
```

2.2.14.69 Custom Weaving Parameters Code Example

New in version C#SDK-V1.1.8: Web-3.8.6

```
public void TestCoordMain5()
{
    DescTran[] points = new DescTran[10];
    for (int i = 0; i < 10; i++)
    {
        points[i] = new DescTran();
    }
```

(continues on next page)

(continued from previous page)

```
    points[0].x = -3;
    points[0].y = -3;
    points[0].z = 0;
    points[1].x = -6;
    points[1].y = 0;
    points[1].z = 0;
    points[2].x = -3;
    points[2].y = 3;
    points[2].z = 0;
    points[3].x = 0;
    points[3].y = 0;
    points[3].z = 0;
    double[] stayTimes = new double[10] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
    int rtn = robot.CustomWeaveSetPara(2, 4, points, stayTimes, 1.000, 0, 0);
    Console.WriteLine($"CustomWeaveSetPara rtn is {rtn}");
    System.Threading.Thread.Sleep(1000);
    int pointNum = 0;
    double frequency = 0;
    int incStayType = 0;
    int stationary = 0;
    rtn = robot.CustomWeaveGetPara(2, ref pointNum, ref points, ref stayTimes, ref frequency, ref incStayType, ref stationary);
    Console.WriteLine($"pointNum is {pointNum}");
    for (int i = 0; i < pointNum; i++)
    {
        Console.WriteLine($"point {i}, point x y z {points[i].x:F6} {points[i].y:F6} {points[i].z:F6}");
    }
    Console.WriteLine($"fre is {frequency:F6}, stay is {incStayType} {stationary}");
    robot.WeaveSetPara(0, 9, 1.000000, 1, 5.000000, 6.000000, 5.000000, 50, 100, 100, 0, 1, 0.000000, 0.000000);
    DescPose desc_p1 = new DescPose(-288.650, 367.807, 288.404, 0.000, -0.001, 0.001);
    DescPose desc_p2 = new DescPose(-431.714, 367.815, 288.415, 0.001, 0.001, 0.000);
    DescPose desc_p3 = new DescPose(-348.666, 427.798, 288.404, -0.000, -0.000, 0.001);
    JointPos j1 = new JointPos(140.656,  -84.560,  -91.707, -93.734,  90.000,50.655 );
    JointPos j2 = new JointPos ( 149.873, -98.298, -77.599,  -94.103,  90.000,  59.873 );
    JointPos j3 = new JointPos (139.773,  -96.173, -80.014,  -93.814,  90.000,  49.772 );
    ExaxisPos epos = new ExaxisPos(0,0,0,0);
    DescPose offset_pos = new DescPose(0,0,0,0,0,0);
    robot.MoveJ(j1, desc_p1, 3, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.WeaveStart(0);
    robot.Circle(j3, desc_p3, 3, 0, 100, 100, epos, j2, desc_p2, 3, 0, 100, 100, epos, 10, -1, offset_pos, 100, -1, 0);
    robot.WeaveEnd(0);
    robot.MoveJ(j1, desc_p1, 3, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.WeaveStart(0);
    robot.MoveC(j3, desc_p3, 3, 0, 100, 100, epos, 0, offset_pos, j2, desc_p2, 3, 0, 100, 100, epos, 0, offset_pos, 10, -1, 0);
    robot.WeaveEnd(0);
    robot.MoveJ(j1, desc_p1, 3, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.WeaveStart(0);
    robot.MoveL(j2, desc_p2, 3, 0, 100, 100, 10, -1, epos, 0, 0, offset_pos, 0, 0, 10);
```

(continues on next page)

(continued from previous page)

```
    robot.WeaveEnd(0);
}
```

2.2.15 Other interfaces

2.2.15.1 Get SSH public key

```
/**
* @brief Get SSH public key
* @param [out] keygen Public key
* @return Error code
*/
int GetSSHKeygen(ref string keygen);
```

2.2.15.2 Send SCP command

New in version C#SDK-V1.1.4: Web-3.8.3

```
/**
* @brief Send SCP command
* @param [in] mode 0-Upload (host computer -> controller), 1-Download (controller ->␣
↪host computer)
* @param [in] sshname Host computer username
* @param [in] sship Host computer IP address
* @param [in] usr_file_url Host computer file path
* @param [in] robot_file_url Robot controller file path
* @return Error code
*/
int SetSSHScpCmd(int mode, string sshname, string sship, string usr_file_url, string␣
↪robot_file_url);
```

2.2.15.3 Calculate the MD5 value of a file in a specified path

```
/**
* @brief Calculate the MD5 value of a file in the specified path
* @param [in] file_path File path including file name, default Traj folder path is "/
↪fruser/traj/", such as "/fruser/traj/trajHelix_aima_1.txt"
* @param [out] md5 The MD5 value of the file
* @return Error code
*/
int ComputeFileMD5(string file_path, ref string md5);
```

2.2.15.4 Robot SSH and MD5 command code example

New in version C#SDK-V1.1.4: Web-3.8.3

```
private void button46_Click(object sender, EventArgs e)
{
    string file_path = "/fruser/airlab.lua";
    string md5 = "";
    byte emerg_state = 0;
    byte si0_state = 0;
    byte si1_state = 0;
    int sdk_com_state = 0;

    string ssh_keygen = "";
    int retval = robot.GetSSHKeygen(ref ssh_keygen);
    Console.WriteLine("GetSSHKeygen retval is: {0}", retval);
    Console.WriteLine("ssh key is: {0}", ssh_keygen);

    string ssh_name = "fr";
    string ssh_ip = "192.168.58.45";
    string ssh_route = "/home/fr";
    string ssh_robot_url = "/root/robot/dhpara.config";
    retval = robot.SetSSHScpCmd(1, ssh_name, ssh_ip, ssh_route, ssh_robot_url);
    Console.WriteLine("SetSSHScpCmd retval is: {0}", retval);
    Console.WriteLine("robot url is: {0}", ssh_robot_url);

    robot.ComputeFileMD5(file_path, ref md5);
    Console.WriteLine("md5 is: {0}", md5);
}
```

2.2.15.5 Set robot 20004 port feedback cycle

```
/**
* @brief Set the robot's 20004 port feedback period
* @param [in] period Robot's 20004 port feedback period (ms)
* @return Error code
*/
int SetRobotRealtimeStateSamplePeriod(int period);
```

2.2.15.6 Get the robot's 20004 port feedback period

```
/**
* @brief Get the feedback period for robot port 20004
* @param [out] period Feedback period for robot port 20004 (ms)
* @return Error code
*/
int GetRobotRealtimeStateSamplePeriod((ref int period);
```

2.2.15.7 Robot 20004 port status feedback period configuration code example

```
private void button47_Click(object sender, EventArgs e)
{
    robot.SetRobotRealtimeStateSamplePeriod(10);
    int getPeriod = 0;
    robot.GetRobotRealtimeStateSamplePeriod(ref getPeriod);
    Console.WriteLine("period is {0}", getPeriod);
    Thread.Sleep(1000);
}
```

2.2.15.8 Robot Software Upgrade

```
/**
* @brief Robot software upgrade
* @param [in] filePath Full path of the software upgrade package
* @param [in] block Whether to block until the upgrade is complete true: block; false: non-blocking
* @return Error code
*/
int SoftwareUpgrade(string filePath, bool block);
```

2.2.15.9 Get robot software upgrade status

```
/**
* @brief  Get robot software upgrade status
* @param [out] state Robot software package upgrade status  0-idle or uploading upgrade package; 1~100: upgrade completion percentage; -1: upgrade software failed; -2: verification failed; -3: Version verification failed; -4: Unzipping failed; -5: User configuration upgrade failed; -6: Peripheral configuration upgrade failed; -7: Extended axis configuration upgrade failed; -8: Robot configuration upgrade failed; -9: DH parameter configuration upgrade failed
* @return Error code
*/
int GetSoftwareUpgradeState(ref int state);
```

2.2.15.10 Robot software upgrade code example

```
private void button48_Click(object sender, EventArgs e)
{
    robot.SoftwareUpgrade("D://zUP/QNX382/software.tar.gz", false);
    while (true)
    {
        int curState = -1;
        robot.GetSoftwareUpgradeState(ref curState);
        Console.WriteLine("upgrade state is {0}", curState);
        Thread.Sleep(300);
    }
}
```

2.2.15.11 Download point table

```
/**
* @brief Download the point table from the robot controller to the local computer
* @param [in] pointTableName The name of the point table in the controller: pointTable1.db
* @param [in] saveFilePath The path where the point table is downloaded to the computer: C://test/
* @return Error code
*/
int PointTableDownLoad(string pointTableName, string saveFilePath);
```

2.2.15.12 Upload point table

```
/**
* @brief Upload the point table from the local computer to the robot controller
* @param [in] pointTableFilePath The absolute path of the point table on the local computer C://test/pointTable1.db
* @return Error code
*/
int PointTableUpLoad(string pointTableFilePath);
```

2.2.15.13 Point Table Update Lua Program

```
/**
* @brief Use the given point table to update the points in the Lua program
* @param [in] pointTableName The name of the point table in the controller: "pointTable1.db". When the point table is empty (i.e., ""), it indicates that the Lua program will be updated to the initial program without applying the point table.
* @param [in] luaFileName The name of the Lua file to be updated: "test.lua"
* @param [out] errorStr Error message for updating the point table in Lua
* @return Error code
*/
int PointTableUpdateLua(string pointTableName, string luaFileName, ref string errorStr);
```

2.2.15.14 Switch point tables and apply

```
/**
* @brief Switch point table and apply
* @param [in] pointTableName Name of the point table to switch to   "pointTable1.db"
* @param [out] errorStr Error message for switching point tables
* @return Error code
*/
int PointTableSwitch(string pointTableName, ref string errorStr);
```

2.2.15.15 Robot point table operation code example

```
private void btnUpload_Click(object sender, EventArgs e)
{
    string save_path = "D://zDOWN/";
    string point_table_name = "test_point_A.db";
    int rtn = robot.PointTableDownLoad(point_table_name, save_path);
    Console.WriteLine("download : {0} fail: {1}", point_table_name, rtn);

    string upload_path = "D://zUP/test_point_A.db";
    rtn = robot.PointTableUpLoad(upload_path);
    Console.WriteLine("retval is: {0}", rtn);

    string point_tablename = "test_point_A.db";
    string lua_name = "Text1.lua";

    string errorStr = "";
    rtn = robot.PointTableUpdateLua(point_tablename, lua_name, ref errorStr);
    Console.WriteLine("retval is: {0}", rtn);
}
```

2.2.15.16 Controller log download

```
/**
* @brief  Controller log download
* @param [in] savePath Save file path "D://zDown/"
* @return Error code
*/
int RbLogDownload(string savePath);
```

2.2.15.17 All Data Source Download

```
/**
* @brief All Data Source Download
* @param [in] savePath Save file path "D://zDown/"
* @return Error code
*/
int AllDataSourceDownload(string savePath);
```

2.2.15.18 Data backup package download

```
/**
* @brief Data backup package download
* @param [in] savePath Save file path "D://zDown/"
* @return Error code
*/
int DataPackageDownload(string savePath);
```

2.2.15.19 Download controller data code example

```
private void button50_Click(object sender, EventArgs e)
{
    int rtn = robot.RbLogDownload("D://zDOWN/");
    Console.WriteLine("RbLogDownload rtn is {0}", rtn);

    rtn = robot.AllDataSourceDownload("D://zDOWN/");
    Console.WriteLine("AllDataSourceDownload rtn is {0}", rtn);

    rtn = robot.DataPackageDownload("D://zDOWN/");
    Console.WriteLine("DataPackageDownload rtn is {0}", rtn);
}
```

2.2.15.20 Robot Operating System Upgrade (LA Control Box)

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
 * @brief Robot Operating System Upgrade (LA Control Box)
 * @param [in] filePath Full path of the operating system upgrade package
 * @return Error code
 */
public int KernelUpgrade(string filePath)
```

2.2.15.21 Get Robot Operating System Upgrade Result (LA Control Box)

New in version C#SDK-V1.1.8: Web-3.8.6

```
/**
 * @brief Get Robot Operating System Upgrade Result (LA Control Box)
 * @param [out] result Upgrade result: 0: Success; -1: Failure
 * @return Error code
 */
public int GetKernelUpgradeResult(ref int[] result)
```

2.2.15.22 Set encoder upgrade

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Set encoder upgrade
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
int SetEncoderUpgrade(string path);
```

2.2.15.23 Set joint firmware upgrade

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Set joint firmware upgrade
* @param [in] type Upgrade file type; 1 - Upgrade firmware; 2 - Upgrade slave␣
↪configuration file
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
int SetJointFirmwareUpgrade(int type, string path);
```

2.2.15.24 Set firmware upgrade for control box

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Set firmware upgrade for control box
* @param [in] type Type of upgrade file; 1 - Upgrade firmware; 2 - Upgrade slave station␣
↪configuration file
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
int SetCtrlFirmwareUpgrade(int type, string path);
```

2.2.15.25 Set end firmware upgrade

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Set end firmware upgrade
* @param [in] type Upgrade file type; 1 - Upgrade firmware; 2 - Upgrade slave␣
↪configuration file
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
int SetEndFirmwareUpgrade(int type, string path);
```

2.2.15.26 Upgrade of the joint full parameter configuration file

New in version C#SDK-V1.1.5: Web-3.8.4

```
/**
* @brief Upgrade of the joint full parameter configuration file
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
int JointAllParamUpgrade(string path);
```

2.2.15.27 Example of upgrading code for robot from firmware

New in version C#SDK-V1.1.5: Web-3.8.4

```
private void button83_Click(object sender, EventArgs e)
{
    robot.RobotEnable(0);
    Thread.Sleep(200);
    int rtn = robot.JointAllParamUpgrade("D://zUP/upgrade/jointallparameters.db");
    Console.WriteLine($"robot JointAllParamUpgrade rtn is{rtn}");
    rtn = robot.SetCtrlFirmwareUpgrade(2, "D://zUP/upgrade/FAIR_Cobot_Cbd_Asix_V2.0.bin
↪");
    Console.WriteLine($"robot SetCtrlFirmwareUpgrade rtn is{rtn}");
    rtn = robot.SetEndFirmwareUpgrade(2, "D://zUP/upgrade/FAIR_Cobot_Axle_Asix_V2.4.bin
↪");
    Console.WriteLine($"robot SetEndFirmwareUpgrade rtn is {rtn}");
    robot.SetSysServoBootMode();
    rtn = robot.SetCtrlFirmwareUpgrade(1, "D://zUP/upgrade/FR_CTRL_PRIMCU_FV201212_MAIN_
↪U4_T01_20250428(MT).bin");
    Console.WriteLine($"robot SetCtrlFirmwareUpgrade rtn is{rtn}");
    rtn = robot.SetEndFirmwareUpgrade(1, "D://zUP/upgrade/FR_END_FV201009_MAIN_U1_T01_
↪20250428.bin");
    Console.WriteLine($"robot SetEndFirmwareUpgrade rtn is {rtn}");
    rtn = robot.SetJointFirmwareUpgrade(1, "D://zUP/upgrade/FR_SERVO_FV504214_MAIN_U7_
↪T07_20250519.bin");
    Console.WriteLine($"robot SetJointFirmwareUpgrade rtn is{rtn}");
}
```

2.2.15.28 Robot MCU log generation

New in version C#SDK-V1.1.9: Web-3.8.7

```
/**
* @brief Robot MCU log generation
* @return Error code
*/
public int RobotMCULogCollect();
```

2.2.16 Appendix

2.2.16.1 Source Code Download

Find the “Downloads” section in the FAIRINO documentation (https://fairino-doc-en.readthedocs.io/latest/), click the “C# SDK” button, and then click “FAIRINO C# SDK” on the right-hand page. Wait for the browser to complete the download.

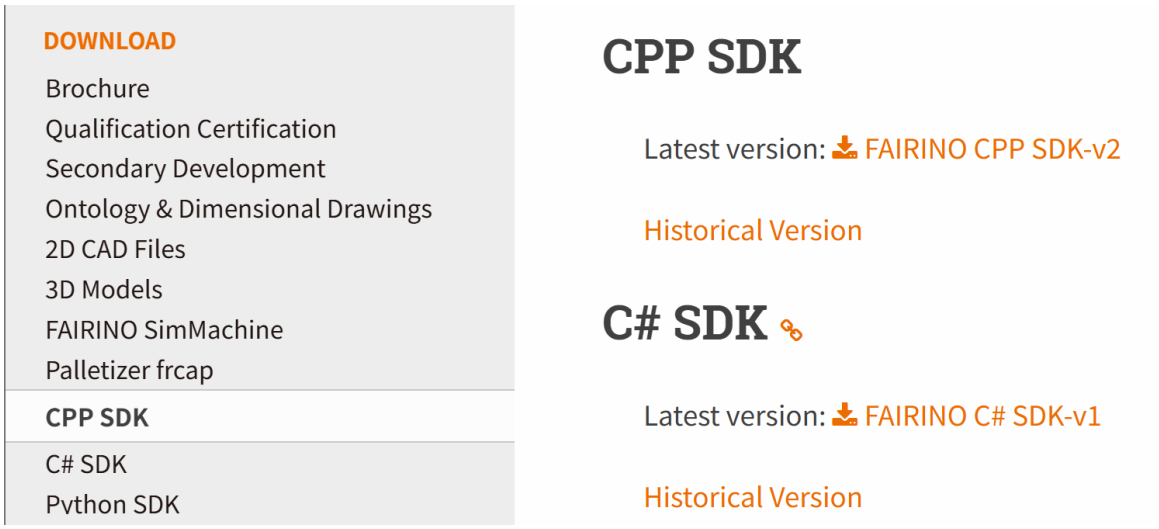


Figure 15.1-1 C# SDK Source Code Download

Download and extract the C# SDK. The project directory is shown below. The "examples" folder contains test samples, the "src" folder contains the C# SDK, and "Fairino.sln" is the project solution file. "Dlls" contains library files.



Figure 15.1-2 C# SDK File Structure Example

Locate the solution file named "fairino.sln" and double-click to open it. The file structure is shown below.

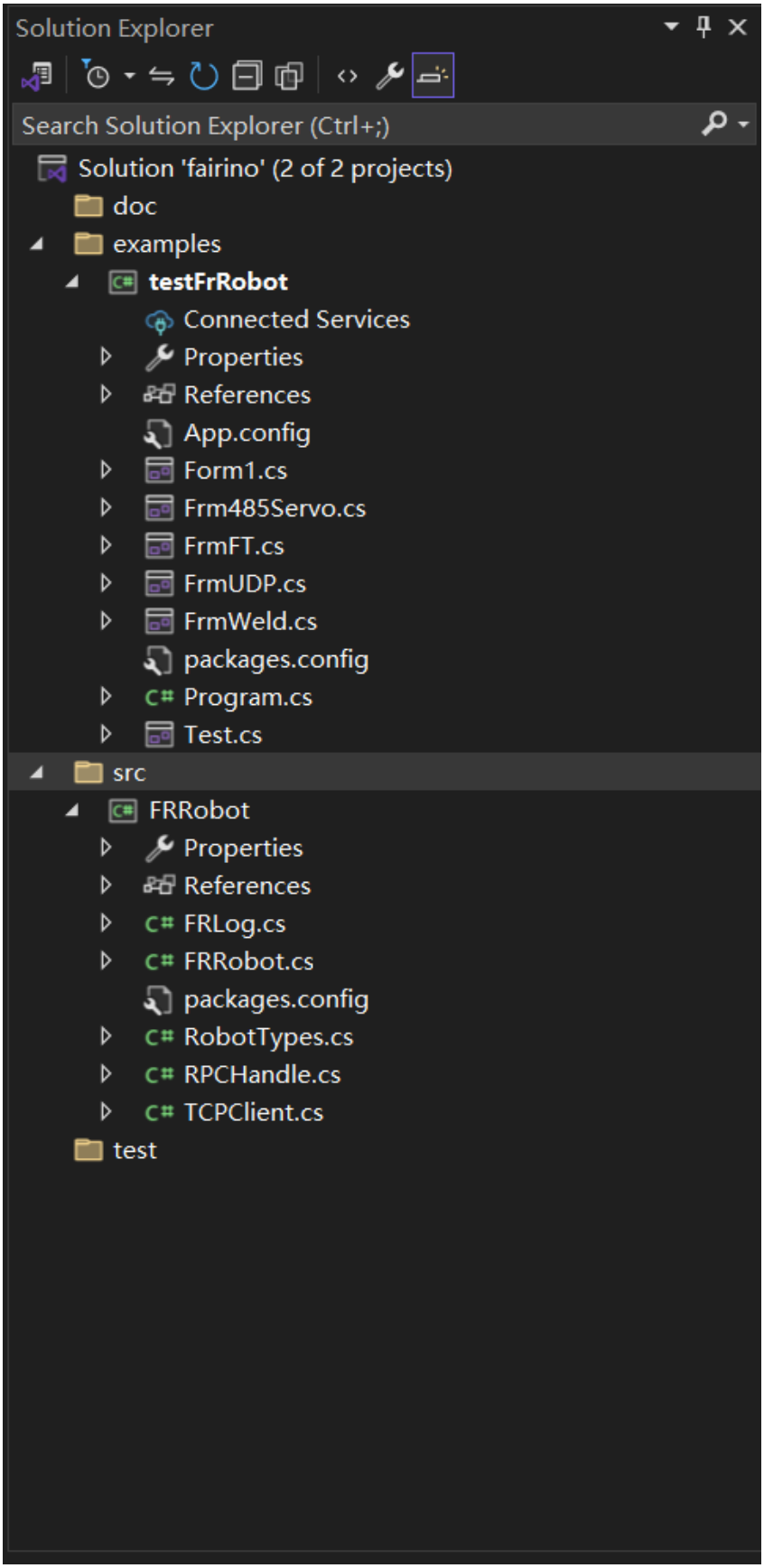

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'fairino' (2 of 2 projects)
doc
examples
testFrRobot
Connected Services
Properties
References
App.config
Form1.cs
Frm485Servo.cs
FrmFT.cs
FrmUDP.cs
FrmWeld.cs
packages.config
Program.cs
Test.cs
src
FRRobot
Properties
References
FRLog.cs
FRRobot.cs
packages.config
RobotTypes.cs
RPCHandle.cs
TCPClient.cs
test

Figure 15.1-3 Project File Structure Example in Visual Studio 2022

2.2.16.2 Windows Platform Source Code Compilation

2.2.16.2.1 C# SDK Compilation

Right-click the FRRobot project, select Properties, and choose the .NET framework version.

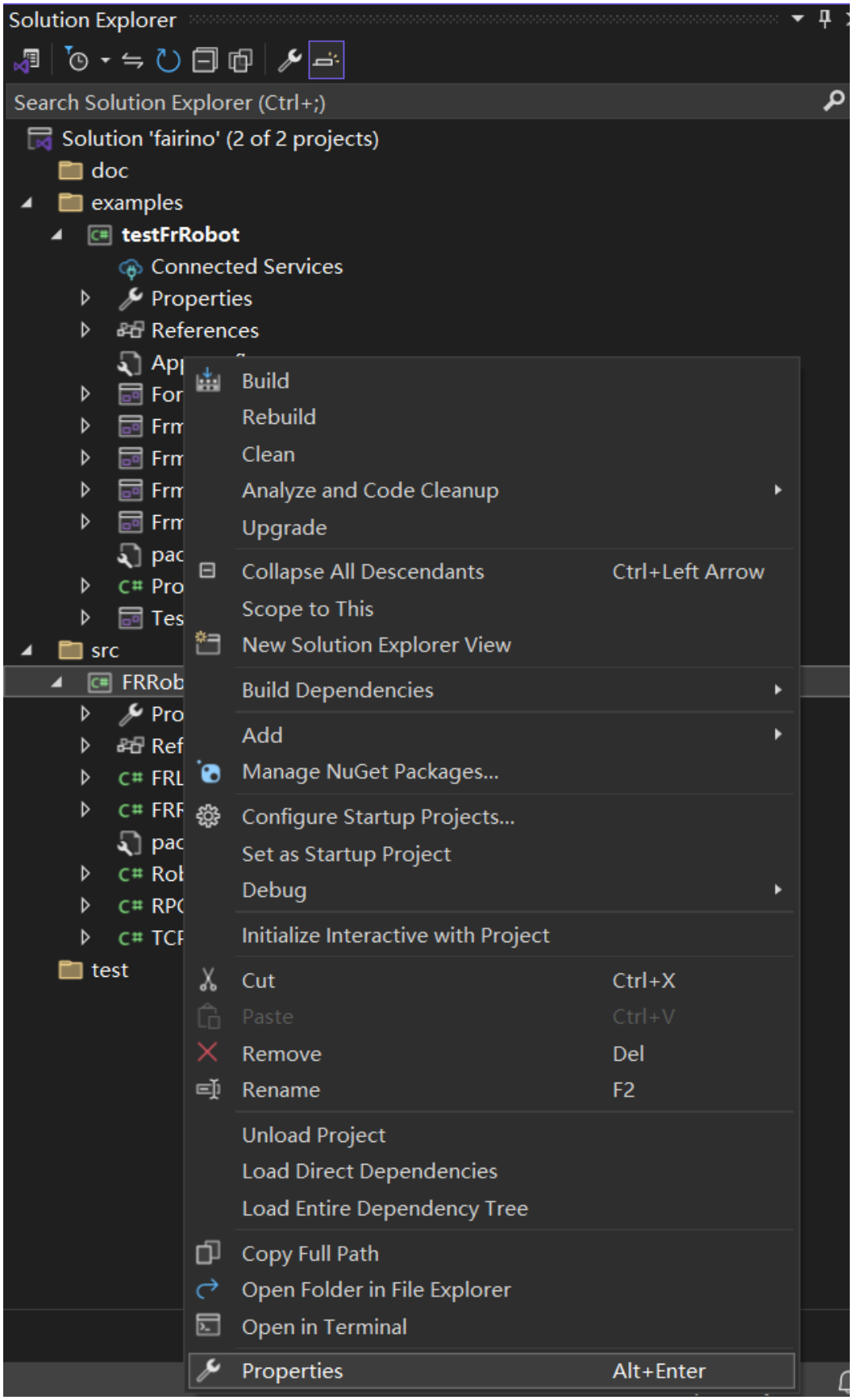

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'fairino' (2 of 2 projects)
doc
examples
testFrRobot
Connected Services
Properties
References
src
test
Build
Rebuild
Clean
Analyze and Code Cleanup
Upgrade
Collapse All Descendants Ctrl+Left Arrow
Scope to This
New Solution Explorer View
Build Dependencies
Add
Manage NuGet Packages...
Configure Startup Projects...
Set as Startup Project
Debug
Initialize Interactive with Project
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename F2
Unload Project
Load Direct Dependencies
Load Entire Dependency Tree
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Properties Alt+Enter

Figure 15.2-1 Setting Properties

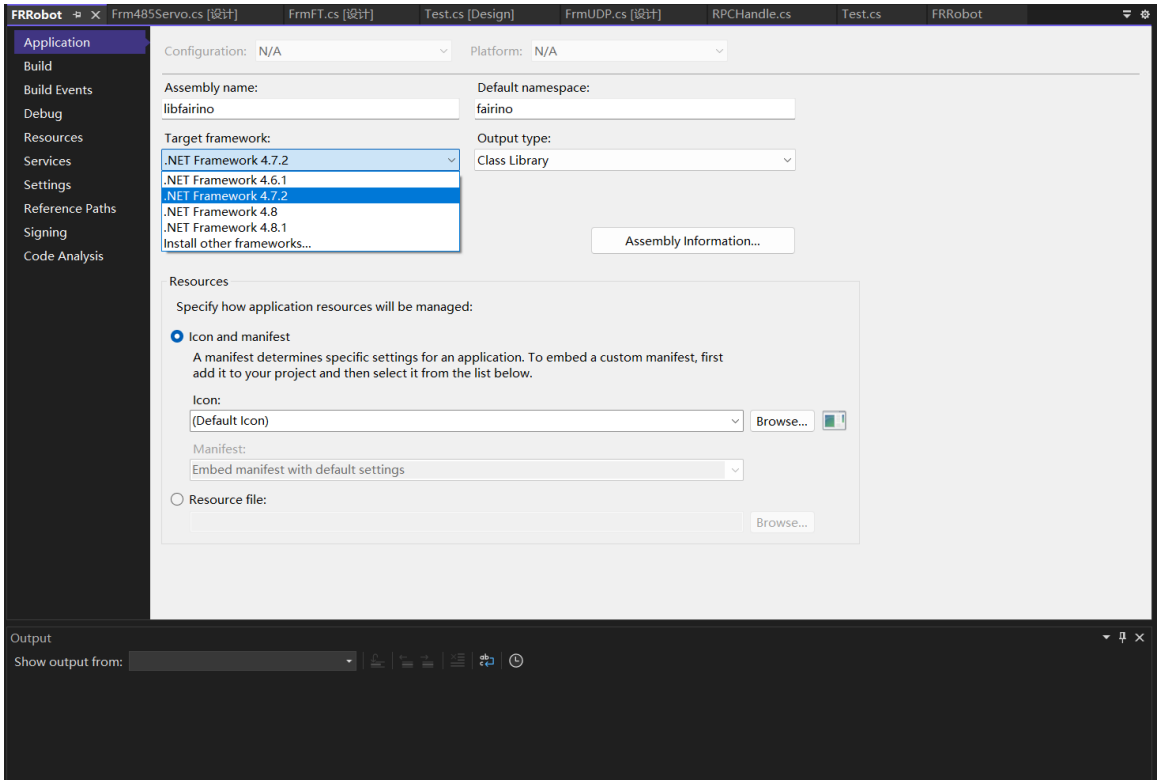


Figure 15.2-2 Selecting .NET Framework

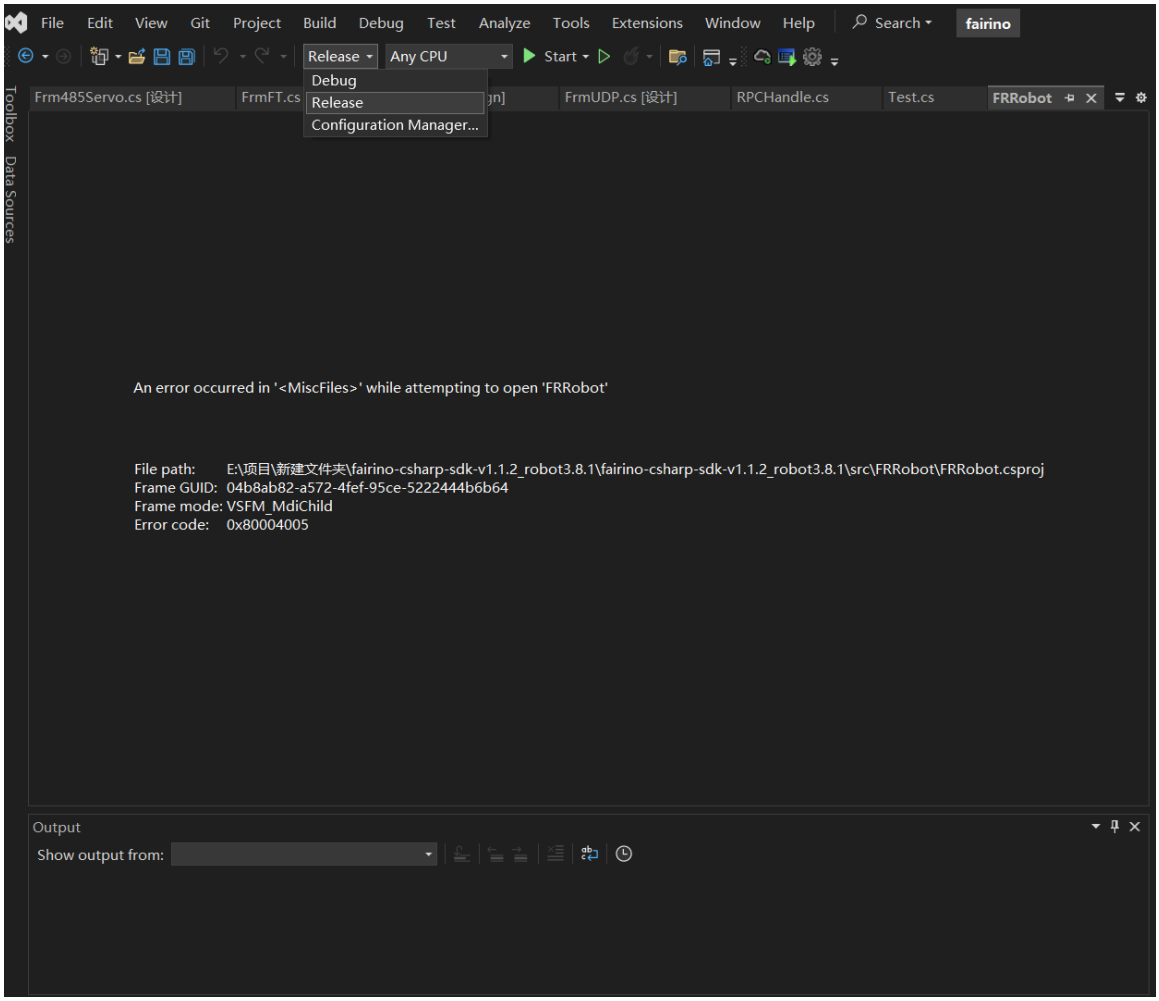


Figure 15.2-3 Building FRRobot Project in Release Mode

Switch Visual Studio 2022 to Release mode and rebuild the FRRobot project. The DLL dynamic link library will be generated in the binRelease folder.

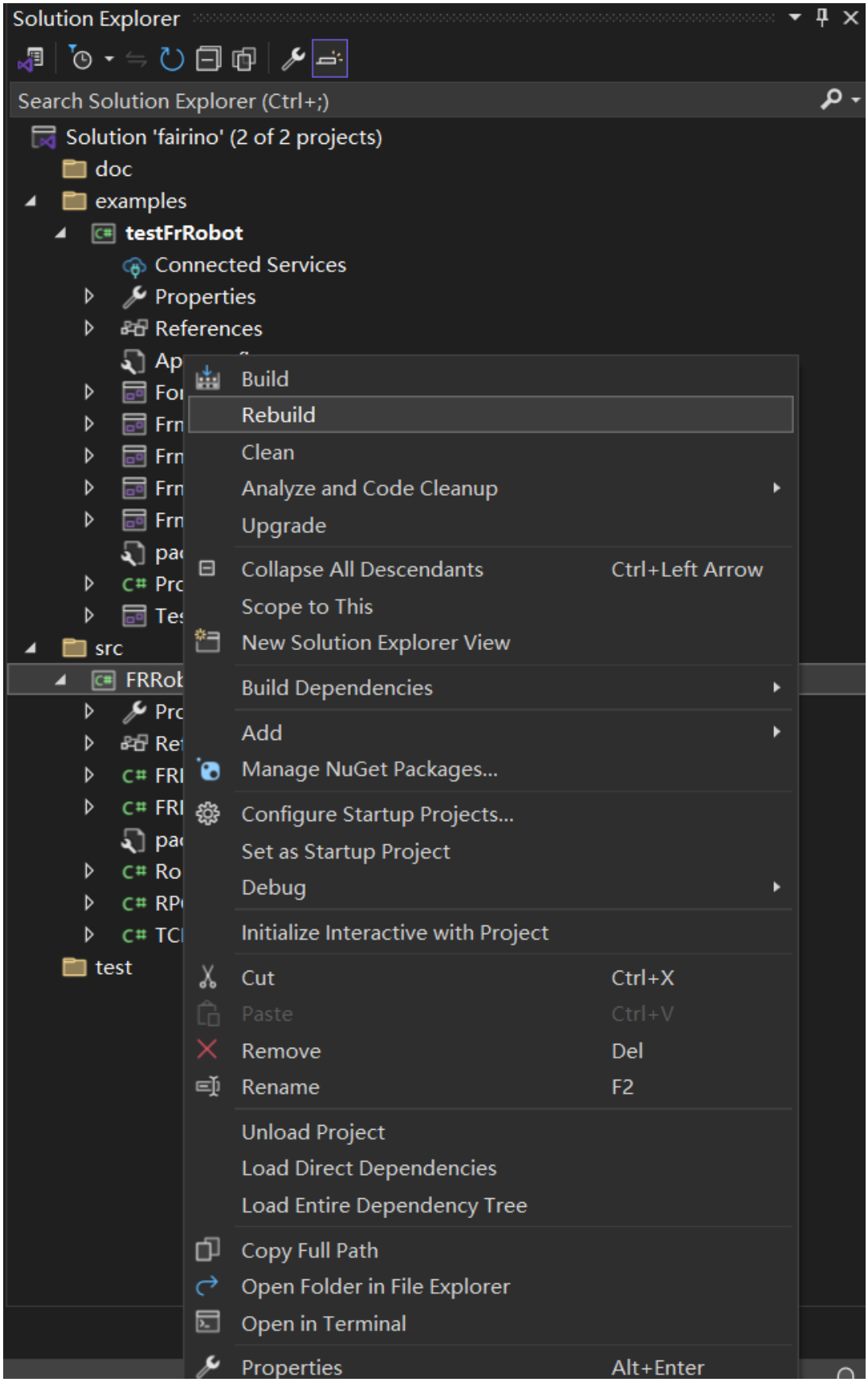

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'fairino' (2 of 2 projects)
doc
examples
testFrRobot
Connected Services
Properties
References
src
test
Build
Rebuild
Clean
Analyze and Code Cleanup
Upgrade
Collapse All Descendants Ctrl+Left Arrow
Scope to This
New Solution Explorer View
Build Dependencies
Add
Manage NuGet Packages...
Configure Startup Projects...
Set as Startup Project
Debug
Initialize Interactive with Project
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename F2
Unload Project
Load Direct Dependencies
Load Entire Dependency Tree
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Properties Alt+Enter

Figure 15.2-4 Setting Release Mode

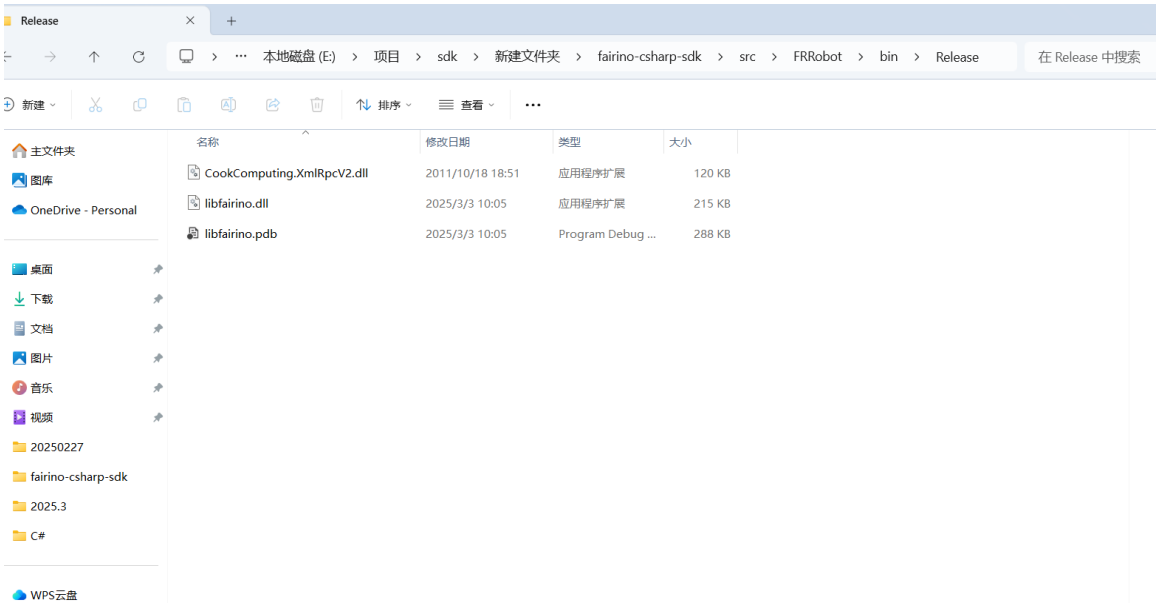


Figure 15.2-5 Rebuilding FRRobot Project in Release Mode

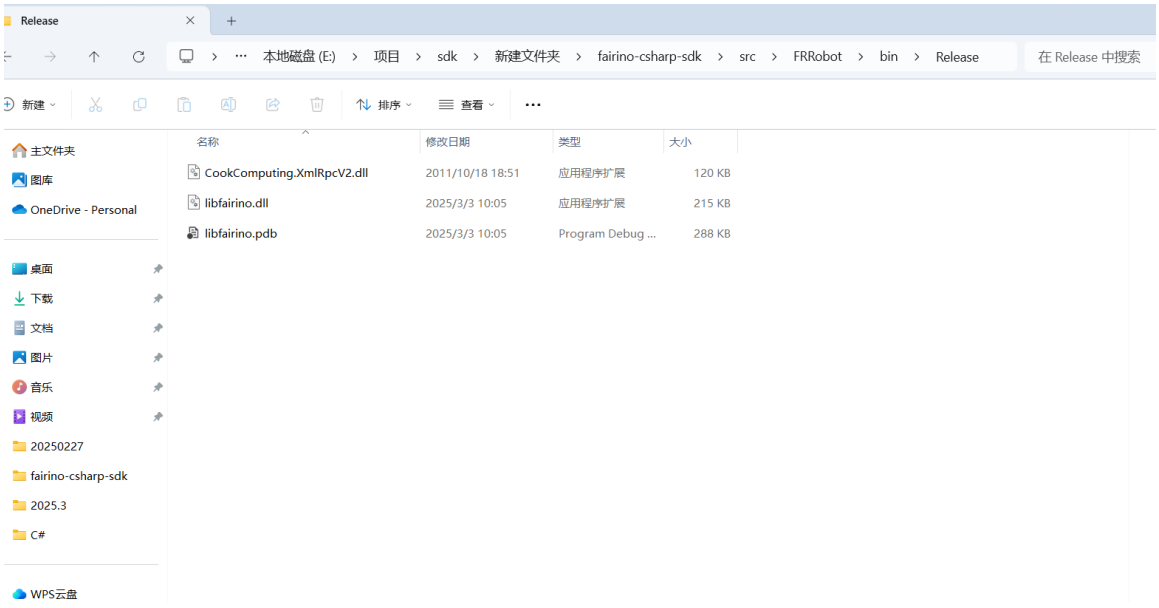


Figure 15.2-6 Generating DLL Dynamic Link Library

2.2.16.2.2 C# SDK Usage

Right-click the testFrRobot project and set it as the startup project.

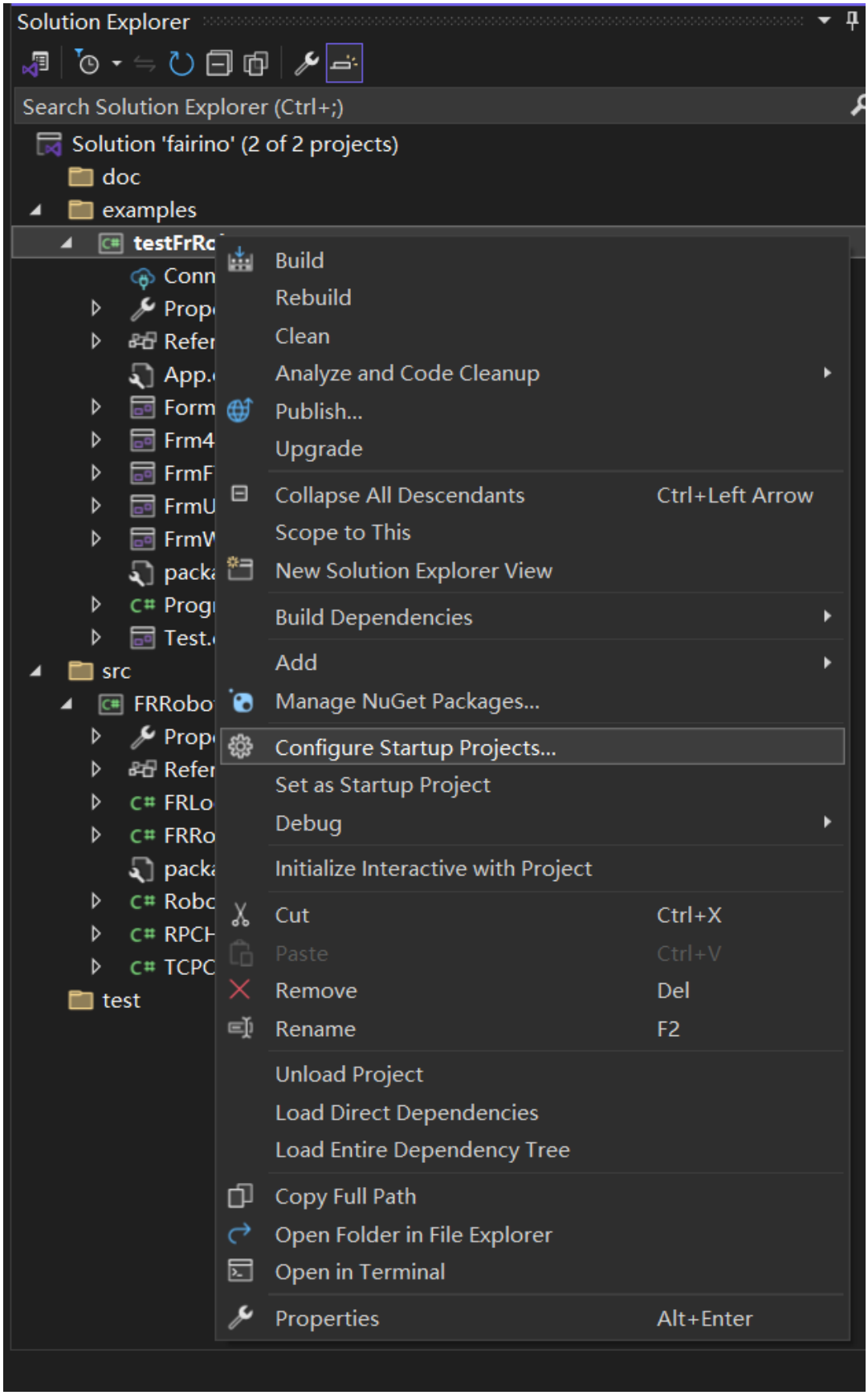

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'fairino' (2 of 2 projects)
doc
examples
src
test
Build
Rebuild
Clean
Analyze and Code Cleanup
Publish...
Upgrade
Collapse All Descendants Ctrl+Left Arrow
Scope to This
New Solution Explorer View
Build Dependencies
Add
Manage NuGet Packages...
Configure Startup Projects...
Set as Startup Project
Debug
Initialize Interactive with Project
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename F2
Unload Project
Load Direct Dependencies
Load Entire Dependency Tree
Copy Full Path
Open Folder in File Explorer
Open in Terminal
Properties Alt+Enter

Figure 15.2-7 Setting as Startup Project

The C# SDK test interface is shown below.



Figure 15.2-8 C# SDK Test Interface

2.2.16.3 Notes

2.2.16.3.1 Potential Issues

2.2.16.3.1.1 Handling No Effect After Code Updates

If the project still executes old code after rewriting and restarting, consider the following steps:

Rebuild the project: Follow the instructions in step 3.2 to rebuild or update the project configuration and files.

2.2.16.3.1.2 Error Codes

A return value of 0 indicates normal operation. If the return value is not 0, refer to the error code table.

2.3 Java

This document is the Java version of the secondary development interface documentation.

Important: Description of robot parameter units: robot position in millimeters (mm) and attitude in degrees (°).

Important:

1) All code examples in the documentation default to the robot being powered on and enabled unless otherwise specified;
2) All code examples in the documentation default to no interference in the robot's workspace;

3) Please use the data from the live robot for actual use testing.

Note: The current documentation applies to SDK-v3.8.5 and is backward compatible with v1.x/v2.x versions.

2.3.1 Version Update Description

Version	Date	Update Description
V3.9.3	2026-02-11	1.ServoCart() interface added extended axis parameters 2.SetOutputResetCtlBoxDO() interface added parameter for whether to reload DO state before reset after pause resume 3.SetOutputResetCtlBoxAO() interface added parameter for whether to reload DO state before reset after pause resume 4.SetOutputResetAxleDO() interface added parameter for whether to reload DO state before reset after pause resume 5.SetOutputResetAxleAO() interface added parameter for whether to reload DO state before reset after pause resume 6.SetOutputResetExtDO() interface added parameter for whether to reload DO state before reset after pause resume 7.SetOutputResetExtAO() interface added parameter for whether to reload DO state before reset after pause resume 8.SetOutputResetSmartToolDO() interface added parameter for whether to reload DO state before reset after pause resume 9.Added GetInverseKinExaxis() inverse kinematics solution interface including extended axis position
V3.9.2	2026-01-26	1. Added a processing strategy parameter for undetected force/torque to the FT_RotInsertion() interface 2. Added robot fixed-point tracking related parameters to the LaserSensorRecordandReplay() interface 3. Added the MoveStationary() interface 4. Added the TCPComputeRPY() interface 5. Added the TCPComputeXYZ() interface 6. Added the TCPRecordFlangePosStart() interface 7. Added the TCPRecordFlangePosEnd() interface 8. Added the TCPGetRecordFlangePos() interface 9. Added the PhotoelectricSensorTCPCalibration() interface
V3.9.1	2025-12-25	1. Added oacc speed scaling factor parameter / physical acceleration parameter to the MoveL() interface; 2. Added oacc speed scaling factor parameter / physical acceleration parameter to the MoveC() interface; 3. Optimized parameter descriptions for physical speed and physical acceleration in the Circle() interface; 4. Added FT_Control() overloaded function with rx, ry activation thresholds and torque adjustment coefficient parameters; 5. Added SerCoderCompenParams() interface;
V3.9.0	2025-11-26	1. JointSensitivityCalibration() interface adds j1~j6 joint linearity return 2. Added JointHysteresisError() interface 3. Added JointRepeatability() interface 4. Added SetAdmittanceParams() interface 5. Added MoveToIntersectLineStart() interface

2.3.2 Data Structure Description

2.3.2.1 Joint Position Data Type

```
/**
* @brief Joint position data type
*/
public class JointPos
{
  double J1;
  double J2;
  double J3;
  double J4;
  double J5;
  double J6;

  public JointPos(double j1, double j2, double j3, double j4, double j5, double j6)
  {
    J1 = j1;
    J2 = j2;
    J3 = j3;
    J4 = j4;
    J5 = j5;
    J6 = j6;
  }

  public JointPos()
  {

  }
}
```

2.3.2.2 Cartesian Space Position Data Type

```
/**
* @brief Cartesian space position data type
*/
public class DescTran
{
  public double x = 0.0;    /* X-axis coordinate, unit: mm */
  public double y = 0.0;    /* Y-axis coordinate, unit: mm */
  public double z = 0.0;    /* Z-axis coordinate, unit: mm */
  public DescTran(double posX, double posY, double posZ)
  {
    x = posX;
    y = posY;
    z = posZ;
  }

  public DescTran()
  {
```

(continues on next page)

(continued from previous page)

```
  }

}
```

2.3.2.3 Euler Angle Attitude Data Type

```
/**
* @brief Euler angle attitude data type
*/
public class Rpy
{
  public double rx = 0.0;   /* Rotation angle around fixed X-axis, unit: deg */
  public double ry = 0.0;   /* Rotation angle around fixed Y-axis, unit: deg */
  public double rz = 0.0;   /* Rotation angle around fixed Z-axis, unit: deg */
  public Rpy(double rotateX, double rotateY, double rotateZ)
  {
    rx = rotateX;
    ry = rotateY;
    rz = rotateZ;
  }
}
```

2.3.2.4 Cartesian Space Pose Data Type

```
/**
*@brief Cartesian space pose type
*/
public class DescPose
{
  public DescTran tran = new DescTran(0.0, 0.0, 0.0);      /* Cartesian space position */
  public Rpy rpy = new Rpy(0.0, 0.0, 0.0);                          /* Cartesian space␣
↪attitude */

  public DescPose()
  {

  }

  public DescPose(DescTran descTran, Rpy rotateRpy)
  {
    tran = descTran;
    rpy = rotateRpy;
  }

  public DescPose(double tranX, double tranY, double tranZ, double rX, double ry, double␣
↪rz)
  {
    tran.x = tranX;
```

(continues on next page)

(continued from previous page)

```
    tran.y = tranY;
    tran.z = tranZ;
    rpy.rx = rX;
    rpy.ry = ry;
    rpy.rz = rz;
  }

  public String toString()
  {
    return String.valueOf(tran.x) + "," + String.valueOf(tran.y) + "," + String.
↪valueOf(tran.z) + "," + String.valueOf(rpy.rx) + "," + String.valueOf(rpy.ry) + "," +␣
↪String.valueOf(rpy.rz);
  }
}
```

2.3.2.5 Extended Axis Position Data Type

```
/**
* @brief Extended axis position data type
*/
public class ExaxisPos
{
  public double axis1 = 0.0;
  public double axis2 = 0.0;
  public double axis3 = 0.0;
  public double axis4 = 0.0;

  public ExaxisPos()
  {

  }
  public ExaxisPos(double[] exaxisPos)
  {
    axis1 = exaxisPos[0];
    axis2 = exaxisPos[1];
    axis3 = exaxisPos[2];
    axis4 = exaxisPos[3];
  }

  public ExaxisPos(double pos1, double pos2, double pos3, double pos4)
  {
    axis1 = pos1;
    axis2 = pos2;
    axis3 = pos3;
    axis4 = pos4;
  }
}
```

2.3.2.6 Force Torque Sensor Data Type

```
/**
* @brief Force component and torque component of force sensor
*/
public class ForceTorque
{
  public double fx;  /* Force component along X-axis, unit: N */
  public double fy;  /* Force component along Y-axis, unit: N */
  public double fz;  /* Force component along Z-axis, unit: N */
  public double tx;  /* Torque component around X-axis, unit: Nm */
  public double ty;  /* Torque component around Y-axis, unit: Nm */
  public double tz;  /* Torque component around Z-axis, unit: Nm */
  public ForceTorque(double fX, double fY, double fZ, double tX, double tY, double tZ)
  {
    fx = fX;
    fy = fY;
    fz = fZ;
    tx = tX;
    ty = tY;
    tz = tZ;
  }
}
```

2.3.2.7 Spiral Parameter Data Type

```
/**
* @brief Spiral parameter data type
*/
public class SpiralParam
{
    public int circle_num;           /* Number of spiral turns  */
    public double circle_angle;         /* Spiral pitch angle  */
    public double rad_init;             /* Initial spiral radius, in mm  */
    public double rad_add;              /* Radius increment  */
    public double rotaxis_add;          /* Rotation axis increment  */
    public int rot_direction;  /* Rotation direction: 0-clockwise, 1-counterclockwise  */
    public int velAccMode;     /* Velocity acceleration parameter mode: 0-constant␣
↪angular velocity; 1- Constant linear velocity */
    public SpiralParam(int circleNum, double circleAngle, double radInit, double radAdd,␣
↪double rotaxisAdd, int rotDirection, int vel_AccMode)
    {
        circle_num = circleNum;
        circle_angle = circleAngle;
        rad_init = radInit;
        rad_add = radAdd;
        rotaxis_add = rotaxisAdd;
        rot_direction = rotDirection;
        velAccMode=vel_AccMode;
    }
}
```

2.3.2.8 Extended Axis Status Type

```
/**
* @brief Extended axis status type
*/
public class EXT_AXIS_STATUS
{
 public double pos = 0;         // Extended axis position
 public double vel = 0;         // Extended axis velocity
 public int errorCode = 0;      // Extended axis error code
 public int ready = 0;         // Servo ready
 public int inPos = 0;         // Servo in position
 public int alarm = 0;         // Servo alarm
 public int flerr = 0;         // Following error
 public int nlimit = 0;        // Negative limit reached
 public int pLimit = 0;        // Positive limit reached
 public int mdbsOffLine = 0;   // Driver 485 bus offline
 public int mdbsTimeout = 0;   // Communication timeout between control card and control␣
↪box via 485
 public int homingStatus = 0; // Extended axis homing status
}
```

2.3.2.9 Sensor Type

```
/**
* @brief Sensor type
*/
public class DeviceConfig
{
  int company = 0;           // Manufacturer
  int device = 0;            // Type/device number
  int softwareVersion = 0;   // Software version
  int bus = 0;               // Mounting location

  public DeviceConfig()
  {

  }

  public DeviceConfig(int company, int device, int softwareVersion, int bus)
  {
    this.company = company;
    this.device = device;
    this.softwareVersion = softwareVersion;
    this.bus = bus;
  }
}
```

2.3.2.10 485 Extended Axis Configuration

```
/**
* @brief 485 extended axis configuration
*/
public class Axis485Param
{
  int servoCompany;           // Servo driver manufacturer, 1-DynaTech
  int servoModel;             // Servo driver model, 1-FD100-750C
  int servoSoftVersion;       // Servo driver software version, 1-V1.0
  int servoResolution;        // Encoder resolution
  double axisMechTransRatio;  // Mechanical transmission ratio

  public Axis485Param(int company, int model, int softVersion, int resolution, double␣
↪mechTransRatio)
  {
    servoCompany = company;
    servoModel = model;
    servoSoftVersion = softVersion;
    servoResolution = resolution;
    axisMechTransRatio = mechTransRatio;
  }

  public Axis485Param()
  {

  }
}
```

2.3.2.11 Servo Controller Status

```
/**
* @brief Servo controller status
*/
public class ROBOT_AUX_STATE
{
  public int servoId = 0;           // Servo driver ID
  public int servoErrCode = 0;       // Servo driver error code
  public int servoState = 0;         // Servo driver status
  public double servoPos = 0;        // Current servo position
  public float servoVel = 0;         // Current servo velocity
  public float servoTorque = 0;      // Current servo torque
}
```

2.3.2.12 Welding Breakoff Status

```java
/**
* @brief Welding breakoff status
*/
public class WELDING_BREAKOFF_STATE
{
  public int breakOffState = 0;  // Welding breakoff status
  public int weldArcState = 0;   // Welding arc breakoff status
}
```

2.3.2.13 UDP Extended Axis Communication Parameters

```java
/**
* @brief UDP extended axis communication parameters
*/
public class UDP_EXT_AXIS_PARAM
{
  public String ip = "192.168.58.88"; // IP address
  public int port = 2021;             // Port number
  public int period = 2;              // Communication period (ms, default is 2, do not␣
↪modify this parameter)
  public int lossPkgTime = 50;        // Packet loss detection time (ms)
  public int lossPkgNum = 2;          // Number of packet losses
  public int disconnectTime = 100;    // Communication disconnection confirmation duration
  public int reconnectEnable = 0;     // Communication disconnection auto-reconnect␣
↪enable 0-disable 1-enable
  public int reconnectPeriod = 100;  // Reconnection interval (ms)
  public int reconnectNum = 3;        // Number of reconnection attempts
  public int selfConnect = 0;        // Whether to automatically establish connection␣
↪after power restart; 0-do not establish connection; 1-establish connection
}
```

2.3.2.14 Robot State Feedback Structure Type

```java
/**
* @brief Robot state feedback structure type
*/
public class ROBOT_STATE_PKG
{
  public short frame_head = 0;            // Frame header 0x5A5A
  public byte frame_cnt = 0;              // Frame count
  public short data_len = 0;              // Data length
  public int program_state = 0;          // Program running state, 1-stop; 2-run; 3-pause
  public int robot_state = 0;            // Robot motion state, 1-stop; 2-run; 3-pause;␣
↪4-drag
  public int main_code = 0;               // Main error code
  public int sub_code = 0;                // Sub error code
  public int robot_mode = 0;             // Robot mode, 0-auto mode; 1-manual mode
  public double[] jt_cur_pos = new double[6];                 // Joint current position
```

(continues on next page)

(continued from previous page)

```
  public double[] tl_cur_pos = new double[6];                    // Tool current pose
  public double[] flange_cur_pos = new double[6];                // End flange current pose
  public double[] actual_qd = new double[6];                     // Robot current joint velocity
  public double[] actual_qdd = new double[6];                    // Robot current joint acceleration
  public double[] target_TCP_CmpSpeed = new double[2];           // Robot TCP composite command velocity
  public double[] target_TCP_Speed = new double[6];              // Robot TCP command velocity
  public double[] actual_TCP_CmpSpeed = new double[2];           // Robot TCP composite actual velocity
  public double[] actual_TCP_Speed = new double[6];              // Robot TCP actual velocity
  public double[] jt_cur_tor = new double[6];                    // Current torque
  public int tool = 0;                          // Tool number
  public int user = 0;                          // Workpiece number
  public int cl_dgt_output_h = 0;               // Digital output 15-8
  public int cl_dgt_output_l = 0;               // Digital output 7-0
  public int tl_dgt_output_l = 0;               // Tool digital output 7-0 (only bit0-bit1 valid)
  public int cl_dgt_input_h = 0;                // Digital input 15-8
  public int cl_dgt_input_l = 0;                // Digital input 7-0
  public int tl_dgt_input_l = 0;                // Tool digital input 7-0 (only bit0-bit1 valid)
  public short[] cl_analog_input = new short[2];          // Control box analog input
  public short tl_anglog_input = 0;                       // Tool analog input
  public double[] ft_sensor_raw_data = new double[6];     // Force/torque sensor raw data
  public double[] ft_sensor_data = new double[6];         // Reference coordinate system force/torque sensor data
  public int ft_sensor_active = 0;              // Force/torque sensor activation state, 0-reset, 1-active
  public int EmergencyStop = 0;                 // Emergency stop flag
  public int motion_done = 0;                    // In-position signal
  public int gripper_motiondone = 0;            // Gripper motion completion signal
  public int mc_queue_len = 0;                   // Motion queue length
  public int collisionState = 0;                // Collision detection, 1-collision; 0-no collision
  public int trajectory_pnum = 0;                // Trajectory point number
  public int safety_stop0_state = 0;  /* Safety stop signal SI0 */
  public int safety_stop1_state = 0;  /* Safety stop signal SI1 */
  public int gripper_fault_id = 0;    /* Error gripper number */
  public short gripper_fault = 0;      /* Gripper fault */
  public short gripper_active = 0;     /* Gripper activation state */
  public int gripper_position = 0;    /* Gripper position */
  public int gripper_speed = 0;       /* Gripper speed */
  public int gripper_current = 0;     /* Gripper current */
  public int gripper_tmp = 0;          /* Gripper temperature */
  public int gripper_voltage = 0;      /* Gripper voltage */
  public ROBOT_AUX_STATE auxState = new ROBOT_AUX_STATE(); /* 485 extended axis status */
  public EXT_AXIS_STATUS extAxisStatus0 = new EXT_AXIS_STATUS();
  public EXT_AXIS_STATUS extAxisStatus1 = new EXT_AXIS_STATUS();
```

(continues on next page)

(continued from previous page)

```
    public EXT_AXIS_STATUS extAxisStatus2 = new EXT_AXIS_STATUS();
    public EXT_AXIS_STATUS extAxisStatus3 = new EXT_AXIS_STATUS();
    public short[] extDIState = new short[8];        // Extended DI input
    public short[] extDOState = new short[8];        // Extended DO output
    public short[] extAIState = new short[4];        // Extended AI input
    public short[] extAOState = new short[4];        // Extended AO output
    public int rbtEnableState = 0;       // Robot enable state
    public double[] jointDriverTorque = new double[6];       // Joint driver current torque
    public double[] jointDriverTemperature = new double[6];  // Joint driver current␣
↪temperature
    public ROBOT_TIME robotTime = new ROBOT_TIME();
    public int softwareUpgradeState = 0;   // Robot software upgrade state 0-idle or␣
↪uploading upgrade package; 1~100: upgrade completion percentage; -1: upgrade software␣
↪failed; -2: verification failed; -3: version verification failed; -4: decompression␣
↪failed; -5: user configuration upgrade failed; -6: peripheral configuration upgrade␣
↪failed; -7: extended axis configuration upgrade failed; -8: robot configuration␣
↪upgrade failed; -9: DH parameter configuration upgrade failed
    public int endLuaErrCode;              // End LUA running state

    public int[] cl_analog_output = new int[2];  // Control box analog output
    public int tl_analog_output;              // Tool analog output
    public float gripperRotNum;               // Rotary gripper current rotation number
    public int gripperRotSpeed;                // Rotary gripper current rotation speed␣
↪percentage
    public int gripperRotTorque;                  // Rotary gripper current rotation␣
↪torque percentage

    public WELDING_BREAKOFF_STATE weldingBreakOffstate = new WELDING_BREAKOFF_STATE(); //␣
↪Welding breakoff state

    public double[] jt_tgt_tor = new double[6];    // Joint command torque
    public int smartToolState;         // SmartTool handle button state

    public float wideVoltageCtrlBoxTemp;        // Wide voltage control box temperature
    public int wideVoltageCtrlBoxFanVel;   // Wide voltage control box fan speed (mA)

    public double[] toolCoord=new double[6];           // Tool coordinate system
    public double[] wobjCoord=new double[6];             // Work object coordinate system
    public double[] extoolCoord=new double[6];                   // External tool␣
↪coordinate system
    public double[] exAxisCoord=new double[6];                   // Extended axis␣
↪coordinate system
    public double load;                    // Payload mass
    public double[] loadCog=new double[3];             // Payload center of gravity

    public double[] lastServoTarget = new double[6];      // Last servo target position in␣
↪the queue
    public int servoJCmdNum;                           // Servo command count

    public short check_sum = 0;          /* Checksum */

    public ROBOT_STATE_PKG()
```

(continues on next page)

(continued from previous page)

```
  {

  }
}
```

2.3.3 Robot basics

2.3.3.1 Instantiate robot

```
/**
* @brief Robot interface class constructor
*/
Robot robot = new Robot();
```

2.3.3.2 Establish communication with controller

```
/**
* @brief Establish communication with robot controller
* @param [in] ip Controller ip address, default is 192.168.58.2
* @return Error code
*/
int RPC(String ip);
```

2.3.3.3 Close communication with robot

```
/**
* @brief Close communication with robot
* @return Error code
*/
int CloseRPC();
```

2.3.3.4 Query sdk version

```
/**
* @brief Query sdk version
* @return Version number
*/
String GetSDKVersion();
```

2.3.3.5 Get controller ip

```
/**
* @brief Get controller ip
* @param [out] ip Controller ip
* @return Error code
*/
int GetControllerIP(String[] ip);
```

2.3.3.6 Control robot to enter or exit drag teaching mode

```
/**
* @brief Control robot to enter or exit drag teaching mode
* @param [in] state 0-exit drag teaching mode, 1-enter drag teaching mode
* @return Error code
*/
int DragTeachSwitch(int state);
```

2.3.3.7 Query whether robot is in drag teaching mode

```
/**
* @brief Query whether robot is in drag teaching mode
* @param [in] state 0-not in drag teaching mode, 1-in drag teaching mode
* @return Error code
*/
int IsInDragTeach(List<Number> state);
```

2.3.3.8 Control robot enable or disable

```
/**
* @brief Control robot enable or disable, robot is enabled by default after power on
* @param [in] state 0-disable, 1-enable
* @return Error code
*/
int RobotEnable(int state);
```

2.3.3.9 Control robot manual/auto mode switch

```
/**
* @brief Control robot manual/auto mode switch
* @param [in] mode 0-auto mode, 1-manual mode
* @return Error code
*/
int Mode(int mode);
```

2.3.3.10 Shut down robot os

New in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief Shut down robot os
* @return Error code
*/
int ShutDownRobotOS();
```

2.3.3.11 Set robot communication reconnect parameters

```
/**
* @brief Set robot communication reconnect parameters
* @param [in] enable Whether to enable, true:enable, false:disable
* @param [in] times Reconnect times
* @param [in] period Reconnect interval
* @return Error code
*/
int SetReconnectParam(boolean enable, int times, int period);
```

2.3.3.12 Initialize log parameters

```
/**
* @brief Initialize log parameters
* @param [in] logType Output mode, DIRECT-direct output; BUFFER-buffer output; ASYNC-
↪async output
* @param [in] logLevel Log filter level, ERROR-error; WARNING-warning; INFO-information;␣
↪DEBUG-debug
* @param [in] filePath File save path, e.g. "D://Log/"
* @param [in] saveFileNum Save file count, files exceeding both save file count and save␣
↪file days will be deleted
* @param [in] saveDays Save file days, files exceeding both save file count and save␣
↪file days will be deleted
* @return Error code
*/
int LoggerInit(FrLogType logType, FrLogLevel logLevel, String filePath, int saveFileNum,␣
↪int saveDays)
```

2.3.3.13 Set log filter level

```
/**
* @brief Set log filter level
* @param [in] logLevel Log filter level, ERROR-error; WARNING-warning; INFO-information;␣
↪DEBUG-debug
* @return Error code
*/
int SetLoggerLevel(FrLogLevel logLevel)
```

2.3.3.14 Robot basic control code example

```java
public static void main(String[] args)
{
    Robot robot = new Robot();
    robot.SetReconnectParam(true,20,500);
    robot.LoggerInit(FrLogType.DIRECT, FrLogLevel.INFO, "D://log", 10, 10);
    int rtn = robot.RPC("192.168.58.2");
    if(rtn == 0)
    {
        System.out.println("Rpc connection success");
    }
    else
    {
        System.out.println("Rpc connection fail");
        return ;
    }
    String[] ip={""};
    String version = "";
    version=robot.GetSDKVersion();
    System.out.println("SDK version : " + version);
    int rtn = robot.GetControllerIP(ip);
    System.out.println("Controller ip : " +  ip[0] + "  " + rtn);
    robot.Mode(1);//1-manual mode  0-auto mode
    robot.Sleep(1000);
    robot.DragTeachSwitch(1);//Enter drag mode
    robot.Sleep(1000);
    ROBOT_STATE_PKG pkg = robot.GetRobotRealTimeState();
    System.out.println("Drag state : " + pkg.robot_state);
    robot.Sleep(1000);
    robot.DragTeachSwitch(0);//Exit drag mode
    robot.Sleep(1000);
    pkg = robot.GetRobotRealTimeState();
    System.out.println("Drag state : " + pkg.robot_state);

    if (pkg.robot_state ==4){
       System.out.println("Drag mode");
    }else {
       System.out.println("Non-drag mode");
    }
}
```

2.3.3.15 Get robot software version

```java
/**
* @brief Get robot software version
* @param [out] robotModel Robot model
* @param [out] webVersion Web version
* @param [out] controllerVersion Controller version
* @return Error code
*/
int GetSoftwareVersion(String robotModel, String webVersion, String controllerVersion);
```

2.3.3.16 Get robot hardware version

```
/**
* @brief Get robot hardware version
* @param [out] ctrlBoxBoardVersion Control box board hardware version
* @param [out] driver1Version Driver 1 hardware version
* @param [out] driver2Version Driver 2 hardware version
* @param [out] driver3Version Driver 3 hardware version
* @param [out] driver4Version Driver 4 hardware version
* @param [out] driver5Version Driver 5 hardware version
* @param [out] driver6Version Driver 6 hardware version
* @param [out] endBoardVersion End board hardware version
* @return Error code
*/
int GetHardwareVersion(String ctrlBoxBoardVersion, String driver1Version, String␣
↪driver2Version, String driver3Version,
                                        String driver4Version, String driver5Version,␣
↪String driver6Version, String endBoardVersion);
```

2.3.3.17 Get robot firmware version

```
/**
* @brief Get robot firmware version
* @param [out] ctrlBoxBoardVersion Control box board firmware version
* @param [out] driver1Version Driver 1 firmware version
* @param [out] driver2Version Driver 2 firmware version
* @param [out] driver3Version Driver 3 firmware version
* @param [out] driver4Version Driver 4 firmware version
* @param [out] driver5Version Driver 5 firmware version
* @param [out] driver6Version Driver 6 firmware version
* @param [out] endBoardVersion End board firmware version
* @return Error code
*/
int GetFirmwareVersion(String ctrlBoxBoardVersion, String driver1Version, String␣
↪driver2Version, String driver3Version,
                                        String driver4Version, String driver5Version,␣
↪String driver6Version, String endBoardVersion);
```

2.3.3.18 Get robot software/firmware version code example

```
public static void main(String[] args)
{
    Robot robot = new Robot();
    robot.SetReconnectParam(true,20,500);//Set reconnect times and interval
    robot.LoggerInit(FrLogType.DIRECT, FrLogLevel.INFO, "D://log", 10, 10);
    int rtn = robot.RPC("192.168.58.2");
    if(rtn == 0)
    {
        System.out.println("Rpc connection success");
    }
```

(continues on next page)

(continued from previous page)

```
    else
    {
        System.out.println("Rpc connection fail");
        return ;
    }
    String ctrlBoxBoardVersion = "";
    String driver1Version = "";
    String driver2Version = "";
    String driver3Version = "";
    String driver4Version = "";
    String driver5Version = "";
    String driver6Version = "";
    String endBoardVersion = "";
    robot.GetHardwareVersion(ctrlBoxBoardVersion ,driver1Version,  driver2Version, ␣
↪driver3Version,
             driver4Version,  driver5Version,  driver6Version,  endBoardVersion);

    robot.GetFirmwareVersion(ctrlBoxBoardVersion, driver1Version, driver2Version,␣
↪driver3Version,
            driver4Version, driver5Version, driver6Version, endBoardVersion);
}
```

2.3.4 Robot movement

2.3.4.1 Jog movement

```
/**
* @brief Jog movement
* @param [in] refType 0-joint jog, 2-base coordinate jog, 4-tool coordinate jog, 8-
↪workpiece coordinate jog
* @param [in] nb 1-joint 1 (or x-axis), 2-joint 2 (or y-axis), 3-joint 3 (or z-axis), 4-
↪joint 4 (or rotation about x-axis), 5-joint 5 (or rotation about y-axis), 6-joint 6␣
↪(or rotation about z-axis)
* @param [in] dir 0-negative direction, 1-positive direction
* @param [in] vel Speed percentage, [0~100]
* @param [in] acc Acceleration percentage, [0~100]
* @param [in] max_dis Maximum angle per jog movement in [°] or distance in [mm]
* @return Error code
*/
int StartJOG(int refType, int nb, int dir, double vel, double acc, double max_dis);
```

2.3.4.2 Jog deceleration stop

```
/**
* @brief Jog deceleration stop
* @param [in] stopType 1-joint jog stop, 3-base coordinate jog stop, 5-tool coordinate jog stop, 9-workpiece coordinate jog stop
* @return Error code
*/
int StopJOG(int stopType);
```

2.3.4.3 Jog immediate stop

```
/**
* @brief Jog immediate stop
* @return Error code
*/
int ImmStopJOG();
```

2.3.4.4 Robot jog control code example

```
public static  int TestJOG(Robot robot)
{
    for (int i = 0; i < 6; i++)
    {
        robot.StartJOG(0, i + 1, 0, 20.0, 20.0, 30.0);
        robot.Sleep(1000);
        robot.ImmStopJOG();
        robot.Sleep(1000);
    }

    for (int i = 0; i < 6; i++)
    {
        robot.StartJOG(2, i + 1, 0, 20.0, 20.0, 30.0);
        robot.Sleep(1000);
        robot.ImmStopJOG();
        robot.Sleep(1000);
    }

    for (int i = 0; i < 6; i++)
    {
        robot.StartJOG(4, i + 1, 0, 20.0, 20.0, 30.0);
        robot.Sleep(1000);
        robot.StopJOG(5);
        robot.Sleep(1000);
    }

    for (int i = 0; i < 6; i++)
    {
        robot.StartJOG(8, i + 1, 0, 20.0, 20.0, 30.0);
        robot.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
        robot.StopJOG(9);
        robot.Sleep(1000);
    }
    return 0;
}
```

2.3.4.5 Joint space movement

```
/**
* @brief  Joint space movement
* @param  [in] joint_pos  Target joint position in deg
* @param  [in] desc_pos  Target cartesian pose
* @param  [in] tool  Tool coordinate number, range [0~14]
* @param  [in] user  Workpiece coordinate number, range [0~14]
* @param  [in] vel  Speed percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not currently available
* @param  [in] ovl  Speed scaling factor, range [0~100]
* @param  [in] epos  Extended axis position in mm
* @param  [in] blendT [-1.0]-move to position (blocking), [0~500.0]-smoothing time (non-blocking) in ms
* @param  [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate, 2-offset in tool coordinate
* @param  [in] offset_pos  Pose offset
* @return  Error code
*/
int MoveJ(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel, double acc, double ovl, ExaxisPos epos, double blendT, int offset_flag, DescPose offset_pos);
```

2.3.4.6 Joint space motion (automatic forward kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Joint space motion (automatic forward kinematics calculation)
* @param [in] joint_pos  Target joint position, unit deg
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] epos  Extended axis position, unit mm
* @param [in] blendT [-1.0]-move to position (blocking), [0~500.0]-smoothing time (non-blocking), unit ms
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @return Error code
*/
int MoveJ(JointPos joint_pos, int tool, int user, double vel, double acc, double ovl, ExaxisPos epos, double blendT, int offset_flag, DescPose offset_pos)
```

2.3.4.7 Cartesian space linear movement

Changed in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief  Cartesian Space Linear Motion (Overloaded Function 1 with blendMode)
* @param  joint_pos  Target joint positions, unit: deg
* @param  desc_pos   Target Cartesian pose
* @param  tool  Tool coordinate system index, range [1~15]
* @param  user  Workpiece/User coordinate system index, range [1~15]
* @param  vel  Velocity percentage, range [0~100]
* @param  acc  Acceleration percentage, range [0~100], currently unavailable
* @param  ovl  Velocity scaling factor [0~100] / Physical velocity (mm/s)
* @param  blendR  [-1.0]-Block until motion completes (blocking), [0~1000.0]-Blend␣
↪radius (non-blocking), unit: mm
* @param  blendMode  Transition method; 0-Tangent blend; 1-Corner blend
* @param  epos  Extended axis position, unit: mm
* @param  search  0-No wire search, 1-Wire search
* @param  offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-
↪Offset in tool coordinate system
* @param  offset_pos  Pose offset value
* @param  oacc  Acceleration scaling factor [0-100] / Physical acceleration (mm/s^2)
* @param  velAccParamMode  Velocity/acceleration parameter mode; 0-Percentage; 1-
↪Physical velocity (mm/s) and acceleration (mm/s^2)
* @param  overSpeedStrategy  Overspeed handling strategy; 1-Standard; 2-Stop with error␣
↪on overspeed; 3-Adaptive speed reduction, default 0
* @param  speedPercent  Allowable speed reduction threshold percentage [0-100], default␣
↪10%
* @return  Error code
*/
public int MoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel,␣
↪double acc, double ovl, double blendR, int blendMode, ExaxisPos epos, int search, int␣
↪offset_flag, DescPose offset_pos, double oacc,int velAccParamMode, int␣
↪overSpeedStrategy, int speedPercent)
```

2.3.4.8 Cartesian space linear motion (automatic inverse kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Cartesian space linear motion (automatic inverse kinematics calculation)
* @param [in] desc_pos  Target cartesian pose
* @param [in] tool  Tool coordinate number, range [1~15]
* @param [in] user  Workpiece coordinate number, range [1~15]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius␣
↪(non-blocking), unit mm
* @param [in] blendMode Transition mode; 0-tangent transition; 1-corner transition
* @param [in] epos  Extended axis position, unit mm
* @param [in] search  0-no wire search, 1-wire search
```

(continues on next page)

(continued from previous page)

```
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based on current joint position, [0~7]-solve according to specific joint space configuration
* @param [in] overSpeedStrategy  Overspeed handling strategy, 1-standard; 2-stop on error when overspeed; 3-adaptive speed reduction, default is 0
* @param [in] speedPercent  Allowed speed reduction threshold percentage [0-100], default 10%
* @return Error code
*/
int MoveL(DescPose desc_pos, int tool, int user, double vel, double acc, double ovl, double blendR, int blendMode, ExaxisPos epos, int search, int offset_flag, DescPose offset_pos, int config, int overSpeedStrategy, int speedPercent)
```

2.3.4.9 Cartesian Space Linear Motion (Added velAccParamMode parameter for velocity and acceleration modes)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Cartesian Space Linear Motion (Added velAccParamMode parameter for velocity and acceleration modes)
* @param  [in] joint_pos  Target joint position, unit deg
* @param  [in] desc_pos   Target Cartesian pose
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Velocity percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not yet open
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-blocking), unit mm
* @param  [in] epos  Extended axis position, unit mm
* @param  [in] search  0-No wire search, 1-Wire search
* @param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
* @param  [in] overSpeedStrategy  Overspeed handling strategy, 1-Standard; 2-Stop with error on overspeed; 3-Adaptive deceleration, default is 0
* @param  [in] speedPercent  Allowed deceleration threshold percentage [0-100], default 10%
* @return  Error code
*/
public int MoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel, double acc, double ovl, double blendR, ExaxisPos epos, int search, int offset_flag, DescPose offset_pos, int velAccParamMode, int overSpeedStrategy, int speedPercent)
```

2.3.4.10 Cartesian Space Linear Motion (Overload Function 1, Added blendMode)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Cartesian Space Linear Motion (Overload Function 1, Added blendMode)
* @param  [in] joint_pos  Target joint position, unit deg
* @param  [in] desc_pos   Target Cartesian pose
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Velocity percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not yet open
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-blocking), unit mm
* @param  [in] blendMode Transition mode; 0-Tangent transition; 1-Corner transition
* @param  [in] epos  Extended axis position, unit mm
* @param  [in] search  0-No wire search, 1-Wire search
* @param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
* @param  [in] overSpeedStrategy  Overspeed handling strategy, 1-Standard; 2-Stop with error on overspeed; 3-Adaptive deceleration, default is 0
* @param  [in] speedPercent  Allowed deceleration threshold percentage [0-100], default 10%
* @return  Error code
*/
public int MoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel, double acc, double ovl, double blendR, int blendMode, ExaxisPos epos, int search, int offset_flag, DescPose offset_pos, int velAccParamMode, int overSpeedStrategy, int speedPercent)
```

2.3.4.11 Cartesian Space Linear Motion (Overload Function 2, No Joint Position Input Required)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Cartesian Space Linear Motion (Overload Function 2, No Joint Position Input Required)
* @param  [in] desc_pos   Target Cartesian pose
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Velocity percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not yet open
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-blocking), unit mm
* @param  [in] blendMode Transition mode; 0-Tangent transition; 1-Corner transition
* @param  [in] epos  Extended axis position, unit mm
* @param  [in] search  0-No wire search, 1-Wire search
```

(continues on next page)

(continued from previous page)

```
* @param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset
* @param  [in] config Inverse kinematic joint space configuration, [-1]-Reference current joint position for calculation, [0~7]-Solve based on specific joint space configuration
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
* @param  [in] overSpeedStrategy  Overspeed handling strategy, 1-Standard; 2-Stop with error on overspeed; 3-Adaptive deceleration, default is 0
* @param  [in] speedPercent  Allowed deceleration threshold percentage [0-100], default 10%
* @return  Error code
*/
public int MoveL(DescPose desc_pos, int tool, int user, double vel, double acc, double ovl, double blendR, int blendMode, ExaxisPos epos, int search, int offset_flag, DescPose offset_pos, int config, int velAccParamMode, int overSpeedStrategy, int speedPercent)
```

2.3.4.12 Cartesian space circular movement

```
/**
* @brief  Cartesian Space Circular Arc Motion
* @param  joint_pos_p  Path point joint positions, unit: deg
* @param  desc_pos_p  Path point Cartesian pose
* @param  ptool  Tool coordinate system index, range [1~15]
* @param  puser  Workpiece/User coordinate system index, range [1~15]
* @param  pvel  Velocity percentage, range [0~100]
* @param  pacc  Acceleration percentage, range [0~100], currently unavailable
* @param  epos_p  Extended axis position, unit: mm
* @param  poffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  offset_pos_p  Pose offset value
* @param  joint_pos_t  Target point joint positions, unit: deg
* @param  desc_pos_t  Target point Cartesian pose
* @param  ttool  Tool coordinate system index, range [1~15]
* @param  tuser  Workpiece/User coordinate system index, range [1~15]
* @param  tvel  Velocity percentage, range [0~100]
* @param  tacc  Acceleration percentage, range [0~100], currently unavailable
* @param  epos_t  Extended axis position, unit: mm
* @param  toffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  offset_pos_t  Pose offset value
* @param  ovl  Velocity scaling factor [0~100] / Physical velocity (mm/s)
* @param  blendR [-1.0]-Block until motion completes (blocking), [0~1000.0]-Blend radius (non-blocking), unit: mm
* @param  oacc Acceleration scaling factor [0-100] / Physical acceleration (mm/s²)
* @param  velAccParamMode Velocity/acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s²)
* @return  Error code
```

(continues on next page)

(continued from previous page)

```
*/
public int MoveC(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser, double␣
↪pvel, double pacc, ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p, JointPos␣
↪joint_pos_t, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc,␣
↪ExaxisPos epos_t, int toffset_flag, DescPose offset_pos_t, double ovl, double blendR,␣
↪double oacc, int velAccParamMode)
```

2.3.4.13 Cartesian Space Arc Motion (Added velAccParamMode parameter for velocity and acceleration modes)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Cartesian Space Arc Motion (Added velAccParamMode parameter for velocity and␣
↪acceleration modes)
* @param  [in] joint_pos_p  Path point joint position, unit deg
* @param  [in] desc_pos_p   Path point Cartesian pose
* @param  [in] ptool  Tool coordinate number, range [1~15]
* @param  [in] puser  Workpiece coordinate number, range [1~15]
* @param  [in] pvel  Velocity percentage, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_p  Extended axis position, unit mm
* @param  [in] poffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system,␣
↪2-Offset in tool coordinate system
* @param  [in] offset_pos_p  Pose offset
* @param  [in] joint_pos_t  Target point joint position, unit deg
* @param  [in] desc_pos_t   Target point Cartesian pose
* @param  [in] ttool  Tool coordinate number, range [1~15]
* @param  [in] tuser  Workpiece coordinate number, range [1~15]
* @param  [in] tvel  Velocity percentage, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_t  Extended axis position, unit mm
* @param  [in] toffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system,␣
↪2-Offset in tool coordinate system
* @param  [in] offset_pos_t  Pose offset
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-
↪blocking), unit mm
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-
↪Physical velocity (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int MoveC(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser, double␣
↪pvel, double pacc, ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p, JointPos␣
↪joint_pos_t, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc,␣
↪ExaxisPos epos_t, int toffset_flag, DescPose offset_pos_t, double ovl, double blendR,␣
↪int velAccParamMode)
```

2.3.4.14 Cartesian Space Arc Motion (Overload Function 1, No Joint Position Input Required)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Cartesian Space Arc Motion (Overload Function 1, No Joint Position Input␣
↪Required)
* @param  [in] desc_pos_p   Path point Cartesian pose
* @param  [in] ptool  Tool coordinate number, range [1~15]
* @param  [in] puser  Workpiece coordinate number, range [1~15]
* @param  [in] pvel  Velocity percentage, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_p  Extended axis position, unit mm
* @param  [in] poffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system,␣
↪2-Offset in tool coordinate system
* @param  [in] offset_pos_p  Pose offset
* @param  [in] desc_pos_t   Target point Cartesian pose
* @param  [in] ttool  Tool coordinate number, range [1~15]
* @param  [in] tuser  Workpiece coordinate number, range [1~15]
* @param  [in] tvel  Velocity percentage, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_t  Extended axis position, unit mm
* @param  [in] toffset_flag  0-No offset, 1-Offset in base/workpiece coordinate system,␣
↪2-Offset in tool coordinate system
* @param  [in] offset_pos_t  Pose offset
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] blendR [-1.0]-Motion complete (blocking), [0~1000.0]-Blending radius (non-
↪blocking), unit mm
* @param  [in] config Inverse kinematic joint space configuration, [-1]-Reference␣
↪current joint position for calculation, [0~7]-Solve based on specific joint space␣
↪configuration
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-
↪Physical velocity (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int MoveC(DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc,␣
↪ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p, DescPose desc_pos_t, int␣
↪ttool, int tuser, double tvel, double tacc, ExaxisPos epos_t, int toffset_flag,␣
↪DescPose offset_pos_t, double ovl, double blendR, int config, int velAccParamMode)
```

2.3.4.15 Cartesian space full circle movement

Changed in version Java: SDK-v1.0.6-3.8.3

```
/**
* @brief  Cartesian Space Full Circle Motion
* @param  joint_pos_p  Path point 1 joint positions, unit: deg
* @param  desc_pos_p   Path point 1 Cartesian pose
* @param  ptool  Tool coordinate system index, range [1~15]
* @param  puser  Workpiece/User coordinate system index, range [1~15]
* @param  pvel  Velocity percentage, range [0~100]
* @param  pacc  Acceleration percentage, range [0~100], currently unavailable
```

(continues on next page)

(continued from previous page)

```
* @param  epos_p  Extended axis position, unit: mm
* @param  joint_pos_t  Path point 2 joint positions, unit: deg
* @param  desc_pos_t  Path point 2 Cartesian pose
* @param  ttool  Tool coordinate system index, range [1~15]
* @param  tuser  Workpiece/User coordinate system index, range [1~15]
* @param  tvel  Velocity percentage, range [0~100]
* @param  tacc  Acceleration percentage, range [0~100], currently unavailable
* @param  epos_t  Extended axis position, unit: mm
* @param  ovl  Velocity scaling factor [0~100] / Physical velocity (mm/s)
* @param  offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  offset_pos  Pose offset value
* @param  oacc Acceleration scaling factor [0-100] / Physical acceleration (mm/s^2)
* @param  blendR -1: Blocking; 0~1000: Blend radius
* @param  velAccParamMode Velocity/acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int Circle(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc, ExaxisPos epos_p, JointPos joint_pos_t, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc, ExaxisPos epos_t, double ovl, int offset_flag, DescPose offset_pos, double oacc, double blendR, int velAccParamMode)
```

2.3.4.16 Cartesian space full circle motion (automatic inverse kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Cartesian space full circle motion (automatic inverse kinematics calculation)
* @param [in] desc_pos_p  Path point 1 cartesian pose
* @param [in] ptool  Tool coordinate number, range [0~14]
* @param [in] puser  Workpiece coordinate number, range [0~14]
* @param [in] pvel  Velocity percentage, range [0~100]
* @param [in] pacc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos_p  Extended axis position, unit mm
* @param [in] desc_pos_t  Path point 2 cartesian pose
* @param [in] ttool  Tool coordinate number, range [0~14]
* @param [in] tuser  Workpiece coordinate number, range [0~14]
* @param [in] tvel  Velocity percentage, range [0~100]
* @param [in] tacc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos_t  Extended axis position, unit mm
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @param [in] oacc Acceleration percentage
* @param [in] blendR -1: blocking; 0~1000: smoothing radius
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
```

(continues on next page)

(continued from previous page)

```
int Circle(DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc,␣
↪ExaxisPos epos_p, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc,␣
↪ExaxisPos epos_t, double ovl, int offset_flag, DescPose offset_pos, double oacc,␣
↪double blendR,int config)
```

2.3.4.17 Cartesian Space Full Circle Motion (Added velAccParamMode parameter for velocity and acceleration modes)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
*@brief  Cartesian Space Full Circle Motion (Added velAccParamMode parameter for␣
↪velocity and acceleration modes)
*@param  [in] joint_pos_p  Path point 1 joint position, unit deg
*@param  [in] desc_pos_p   Path point 1 Cartesian pose
*@param  [in] ptool  Tool coordinate number, range [1~15]
*@param  [in] puser  Workpiece coordinate number, range [1~15]
*@param  [in] pvel  Velocity percentage, range [0~100]
*@param  [in] pacc  Acceleration percentage, range [0~100], not yet open
*@param  [in] epos_p  Extended axis position, unit mm
*@param  [in] joint_pos_t  Path point 2 joint position, unit deg
*@param  [in] desc_pos_t   Path point 2 Cartesian pose
*@param  [in] ttool  Tool coordinate number, range [1~15]
*@param  [in] tuser  Workpiece coordinate number, range [1~15]
*@param  [in] tvel  Velocity percentage, range [0~100]
*@param  [in] tacc  Acceleration percentage, range [0~100], not yet open
*@param  [in] epos_t  Extended axis position, unit mm
*@param  [in] ovl  Velocity scaling factor, range [0~100]
*@param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-
↪Offset in tool coordinate system
*@param  [in] offset_pos  Pose offset
*@param  [in] oacc Acceleration percentage
*@param  [in] blendR -1: Blocking; 0~1000: Blending radius
*@param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-
↪Physical velocity (mm/s) and acceleration (mm/s^2)
*@return  Error code
*/
public int Circle(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser,␣
↪double pvel, double pacc, ExaxisPos epos_p, JointPos joint_pos_t, DescPose desc_pos_t,␣
↪int ttool, int tuser, double tvel, double tacc, ExaxisPos epos_t, double ovl, int␣
↪offset_flag, DescPose offset_pos, double oacc, double blendR, int velAccParamMode)
```

2.3.4.18 Cartesian Space Full Circle Motion (Overload Function 1, No Joint Position Input Required)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Cartesian Space Full Circle Motion (Overload Function 1, No Joint Position Input Required)
* @param  [in] desc_pos_p   Path point 1 Cartesian pose
* @param  [in] ptool  Tool coordinate number, range [0~14]
* @param  [in] puser  Workpiece coordinate number, range [0~14]
* @param  [in] pvel  Velocity percentage, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_p  Extended axis position, unit mm
* @param  [in] desc_pos_t   Path point 2 Cartesian pose
* @param  [in] ttool  Tool coordinate number, range [0~14]
* @param  [in] tuser  Workpiece coordinate number, range [0~14]
* @param  [in] tvel  Velocity percentage, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], not yet open
* @param  [in] epos_t  Extended axis position, unit mm
* @param  [in] ovl  Velocity scaling factor, range [0~100]
* @param  [in] offset_flag  0-No offset, 1-Offset in base/workpiece coordinate system, 2-Offset in tool coordinate system
* @param  [in] offset_pos  Pose offset
* @param  [in] oacc Acceleration percentage
* @param  [in] blendR -1: Blocking; 0~1000: Blending radius
* @param  [in] config Inverse kinematic joint space configuration, [-1]-Reference current joint position for calculation, [0~7]-Solve based on specific joint space configuration
* @param  [in] velAccParamMode Velocity and acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2)
* @return  Error code
*/
public int Circle(DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc, ExaxisPos epos_p, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc, ExaxisPos epos_t, double ovl, int offset_flag, DescPose offset_pos, double oacc, double blendR, int config, int velAccParamMode)
```

2.3.4.19 Cartesian space point-to-point movement

```
/**
* @brief Cartesian space point-to-point movement
* @param [in] desc_pos  Target cartesian pose or pose increment
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Speed percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not currently available
* @param [in] ovl  Speed scaling factor, range [0~100]
* @param [in] blendT [-1.0]-move to position (blocking), [0~500.0]-smoothing time (non-blocking) in ms
* @param [in] config  Joint space configuration, [-1]-calculate with reference to current joint position, [0~7]-calculate with reference to specific joint space configuration, default -1
```

(continues on next page)

(continued from previous page)

```
* @return Error code
*/
int MoveCart(DescPose desc_pos, int tool, int user, double vel, double acc, double ovl,␣
↪double blendT, int config);
```

2.3.4.20 Basic robot movement command code example

```
public static int TestMove(Robot robot)
{
    int rtn=-1;
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3=new JointPos(-29.777, -84.536, 109.275, -114.075, -86.655, 74.257);
    JointPos j4=new JointPos(-31.154, -95.317, 94.276, -88.079, -89.740, 74.256);
    DescPose desc_pos1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);
    DescPose desc_pos3=new DescPose(-487.434, 154.362, 308.576, 176.600, 0.268, -14.061);
    DescPose desc_pos4=new DescPose(-443.165, 147.881, 480.951, 179.511, -0.775, -15.
↪409);
    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 100.0;
    double oacc = 100.0;
    double blendT = 0.0;
    double blendR = 0.0;
    int flag = 0;
    int search = 0;
    int blendMode = 0;
    int velAccMode = 0;
    robot.SetSpeed(20);
    rtn = robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag,␣
↪offset_pos);
    System.out.printf("movej errcode:%d\n", rtn);
    rtn = robot.MoveL(j2, desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, epos,␣
↪search, flag, offset_pos, oacc, velAccMode,0,10);
    System.out.printf("movel errcode:%d\n", rtn);
    rtn = robot.MoveC(j3, desc_pos3, tool, user, vel, acc, epos, flag, offset_pos, j4,␣
↪desc_pos4, tool, user, vel, acc, epos, flag, offset_pos, ovl, blendR, oacc,␣
↪velAccMode);
    System.out.printf("movec errcode:%d\n", rtn);
    rtn = robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag,␣
↪offset_pos);
    System.out.printf("movej errcode:%d\n", rtn);
    rtn = robot.Circle(j3, desc_pos3, tool, user, vel, acc, epos, j1, desc_pos1, tool,␣
↪user, vel, acc, epos, ovl, flag, offset_pos, oacc, -1, velAccMode);
```

(continues on next page)

(continued from previous page)

```
    System.out.printf("circle errcode:%d\n", rtn);
    rtn = robot.MoveCart(desc_pos4, tool, user, vel, acc, ovl, blendT, -1);
    System.out.printf("MoveCart errcode:%d\n", rtn);
    rtn = robot.MoveJ(j1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos);
    System.out.printf("movej errcode:%d\n", rtn);
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, epos,␣
↪search, flag, offset_pos, -1, velAccMode,0,10);
    System.out.printf("movel errcode:%d\n", rtn);
    rtn = robot.MoveC(desc_pos3, tool, user, vel, acc, epos, flag, offset_pos, desc_pos4,
↪ tool, user, vel, acc, epos, flag, offset_pos, ovl, blendR, -1, velAccMode);
    System.out.printf("movec errcode:%d\n", rtn);
    rtn = robot.MoveJ(j2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos);
    System.out.printf("movej errcode:%d\n", rtn);
    rtn = robot.Circle(desc_pos3, tool, user, vel, acc, epos, desc_pos1, tool, user, vel,
↪ acc, epos, ovl, flag, offset_pos, oacc, blendR, -1, velAccMode);
    System.out.printf("circle errcode:%d\n", rtn);
    return 0;
}
```

2.3.4.21 Cartesian space spiral movement

```
/**
* @brief Cartesian space spiral movement
* @param [in] joint_pos  Target joint position in deg
* @param [in] desc_pos   Target cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Speed percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not currently available
* @param [in] epos  Extended axis position in mm
* @param [in] ovl  Speed scaling factor, range [0~100]
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate, 2-offset␣
↪in tool coordinate
* @param [in] offset_pos  Pose offset
* @return Error code
*/
int NewSpiral(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel,␣
↪double acc, ExaxisPos epos, double ovl, int offset_flag, DescPose offset_pos,␣
↪SpiralParam spiral_param);
```

2.3.4.22 Cartesian space spiral motion (automatic inverse kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Cartesian space spiral motion (automatic inverse kinematics calculation)
* @param [in] desc_pos  Target cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
```

(continues on next page)

(continued from previous page)

```
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos  Extended axis position, unit mm
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @param [in] spiral_param  Spiral parameters
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
int NewSpiral(DescPose desc_pos, int tool, int user, double vel, double acc, ExaxisPos epos, double ovl, int offset_flag, DescPose offset_pos, SpiralParam spiral_param,int config)
```

2.3.4.23 Spiral movement code example

```
public static int TestSpiral(Robot robot)
{
    int rtn=-1;
    JointPos j=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    DescPose desc_pos=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose offset_pos1=new DescPose(50, 0, 0, -30, 0, 0);
    DescPose offset_pos2=new DescPose(50, 0, 0, -5, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);
    SpiralParam sp=new SpiralParam(1,5.0,50.0,10.0,10.0,0);

    int tool = 0;
    int user = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 100.0;
    double blendT = 0.0;
    int flag = 2;

    rtn = robot.MoveJ(j, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos1);
    System.out.println("movej errcode:"+ rtn);

    rtn = robot.NewSpiral(desc_pos, tool, user, vel, acc, epos, ovl, flag, offset_pos2,sp,-1);
    System.out.println("newspiral errcode:"+ rtn);

    return 0;
}
```

2.3.4.24 Servo movement start

```
/**
* @brief Servo movement start, used with ServoJ, ServoCart commands
* @return Error code
*/
int ServoMoveStart();
```

2.3.4.25 Servo movement end

```
/**
* @brief Servo movement end, used with ServoJ, ServoCart commands
* @return Error code
*/
int ServoMoveEnd();
```

2.3.4.26 Joint space servo mode movement

Changed in version Java: SDK-v1.0.6-3.8.3

```
/**
* @brief  Joint space servo mode movement
* @param  [in] joint_pos  Target joint position in deg
* @param  [in] axisPos  External axis position in mm
* @param  [in] acc  Acceleration percentage, range [0~100], not currently available, default 0
* @param  [in] vel  Speed percentage, range [0~100], not currently available, default 0
* @param  [in] cmdT  Command cycle in s, recommended range [0.001~0.0016]
* @param  [in] filterT Filter time in s, not currently available, default 0
* @param  [in] gain  Target position proportional amplifier, not currently available, default 0
* @param  [in] id  servoJ command ID, default 0
* @return  Error code
*/
int ServoJ(JointPos joint_pos, ExaxisPos axisPos, double acc, double vel, double cmdT, double filterT, double gain, int id);
```

2.3.4.27 Joint space servo mode movement example program

```
public static void TestServoJ()
{
    Robot robot = new Robot();
    robot.SetReconnectParam(true,20,500);//Set reconnection times and interval
    robot.LoggerInit(FrLogType.DIRECT, FrLogLevel.INFO, "D://log", 10, 10);
    int rtn = robot.RPC("192.168.58.2");
    if(rtn == 0)
    {
        System.out.println("rpc connection success");
    }
```

(continues on next page)

(continued from previous page)

```
    else
    {
        System.out.println("rpc connection fail");
        return ;
    }
    JointPos j5 = new JointPos();
    ExaxisPos ePos=new ExaxisPos();
    int ret = robot.GetActualJointPosDegree(j5);
    if (ret == 0)
    {
        int count = 200;
        while (count > 0)
        {
            robot.ServoJ(j5, ePos,100, 100, 0.008, 0, 0);
            j5.J1 += 0.2;//Joint 1 position increase
            count -= 1;
            robot.WaitMs((int)(8));
        }
    }
}
```

2.3.4.28 Joint torque control start

```
/**
* @brief  Joint torque control start
* @return  Error code
*/
int ServoJTStart()
```

2.3.4.29 Joint torque control

```
/**
* @brief Joint Torque Control
* @param torque Torque for joints j1~j6, unit: Nm
* @param interval Command cycle, unit: s, range: [0.001~0.008]
* @param checkFlag Detection strategy
*                  0-No restrictions;
*                  1-Power limit;
*                  2-Velocity limit;
*                  3-Both power and velocity limits
* @param jPowerLimit Maximum joint power limit for each joint (W)
* @param jVelLimit Maximum joint velocity for each joint (°/s)
* @return Error code
*/
public int ServoJT(double[] torque, double interval, int checkFlag, double[] jPowerLimit,
↪ double[] jVelLimit)
```

2.3.4.30 Joint torque control end

```
/**
* @brief  Joint torque control end
* @return  Error code
*/
int ServoJTEnd()
```

2.3.4.31 Joint space servo mode movement example program

```
public static int TestServoJT(Robot robot)
{

    robot.DragTeachSwitch(1);
    List<Number> joint_toq=new ArrayList<>();
    joint_toq=robot.GetJointTorques(1);

    int count = 100;
    robot.ServoJTStart(); //   #servoJT start
    int error = 0;
    while (count > 0)
    {
        error = robot.ServoJT(torques, 0.001);
        count = count - 1;
        robot.Sleep(1);
    }
    error = robot.ServoJTEnd();
    robot.DragTeachSwitch(0);

    robot.CloseRPC();
    return 0;
}
```

2.3.4.32 Joint Torque Control Code Example with Overspeed Protection

```
public static void ServoJTWithSafety(Robot robot)
{
    robot.ResetAllError();
    robot.Sleep(500);
    List<Number> torques;
    torques = robot.GetJointTorques(1);
    robot.ServoJTStart(); // Start servoJT
    ROBOT_STATE_PKG pkg = new ROBOT_STATE_PKG();
    robot.DragTeachSwitch(1);
    int checkFlag = 3; // -1,3 - Both power and velocity limits
    // double[] jPowerLimit = {1.0,1.0,1.0,1.0,1.0,1.0}; // 5001
    double[] jPowerLimit = { 10.0, 10.0, 10.0, 10.0, 10.0, 10.0 }; // Power limits for␣
↪each joint (W)
    double[] jVelLimit = { 50, 50, 50, 50, 50, 50 }; // 180.1,-1 - Velocity limits for␣
↪each joint (°/s)
```

(continues on next page)

(continued from previous page)

```
    int count = 800000;
    int error = 0;
    double[] tor = new double[]{(double)torques.get(1), (double)torques.get(2),(double)torques.get(3),
                                (double)torques.get(4), (double)torques.get(5),(double)torques.get(6)};
    while (count > 0)
    {
        tor[2] = tor[2] + 0.01; // Increase torque of axis 1 by 0.01NM each time, moving 100 times
        error = robot.ServoJT(tor, 0.01, checkFlag, jPowerLimit, jVelLimit); // Joint space servo mode motion
        System.out.printf("ServoJT return is %d\n", error);
        count = count - 1;
        robot.Sleep(1);
        pkg = robot.GetRobotRealTimeState();
        System.out.printf("maincode %d, subcode %d\n", pkg.main_code, pkg.sub_code);
    }
    robot.DragTeachSwitch(0);
    error = robot.ServoJTEnd(); // End servo motion
}
```

2.3.4.33 Cartesian Space Servo Mode Motion

```
/**
* @brief Cartesian space servo mode motion
* @param mode 0-Absolute motion (base coordinate system), 1-Incremental motion (base coordinate system), 2-Incremental motion (tool coordinate system)
* @param desc_pose Target Cartesian pose or pose increment
* @param exaxis Extended axis position
* @param pos_gain Pose increment proportionality coefficient, only effective in incremental motion, range [0~1]
* @param acc Acceleration percentage, range [0~100], temporarily not available, default 0
* @param vel Velocity percentage, range [0~100], temporarily not available, default 0
* @param cmdT Command transmission period, unit s, recommended range [0.001~0.016]
* @param filterT Filter time, unit s, temporarily not available, default 0
* @param gain Proportional amplifier for target position, temporarily not available, default 0
* @return Error code
*/
public int ServoCart(int mode, DescPose desc_pose, ExaxisPos exaxis, double[] pos_gain, double acc, double vel, double cmdT, double filterT, double gain)
```

2.3.4.34 Cartesian Space Servo Mode Motion Code Example

```
public static void TestServoCart1(Robot robot)
{
    DescPose desc_pos_dt = new DescPose(83.00800, 50.525000 , 29.246 , 179.629 , -7.138 , -166.975 );
    ExaxisPos exaxis = new ExaxisPos( 100.0, 0.0, 0.0, 0.0 );
    double[] pos_gain = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
    int mode = 0;
    double vel = 0.0;
    double acc = 0.0;
    double cmdT = 0.001;
    double filterT = 0.0;
    double gain = 0.0;
    int flag = 0;
    int count = 5000;
    robot.SetSpeed(20);
    while (count>0)
    {
        int rtn = robot.ServoCart(mode, desc_pos_dt, exaxis, pos_gain, acc, vel, cmdT, filterT, gain);
        System.out.printf("ServoCart rtn is %d\n", rtn);
        count -= 1;
        desc_pos_dt.tran.x += 0.01;
        exaxis.axis1 += 0.01;
    }
    robot.CloseRPC();
}
```

2.3.4.35 Spline movement start

```
/**
* @brief  Spline movement start
* @return  Error code
*/
int SplineStart();
```

2.3.4.36 Joint movement PTP

```
/**
* @brief  Joint space spline movement
* @param  [in] joint_pos  Target joint position in deg
* @param  [in] desc_pos   Target cartesian pose
* @param  [in] tool  Tool coordinate number, range [0~14]
* @param  [in] user  Workpiece coordinate number, range [0~14]
* @param  [in] vel  Speed percentage, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not currently available
* @param  [in] ovl  Speed scaling factor, range [0~100]
* @return  Error code
*/
```

(continues on next page)

(continued from previous page)

```
int SplinePTP(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel,␣
↪double acc, double ovl);
```

2.3.4.37 Joint space spline motion (automatic forward kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Joint space spline motion (automatic forward kinematics calculation)
* @param [in] joint_pos  Target joint position, unit deg
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @return Error code
*/
int SplinePTP(JointPos joint_pos, int tool, int user, double vel, double acc, double ovl)
```

2.3.4.38 Spline movement end

```
/**
* @brief  Spline movement end
* @return  Error code
*/
int SplineEnd();
```

2.3.4.39 Spline movement code example

```
public static int TestSpline(Robot robot)
{
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3=new JointPos(-61.954, -84.409, 108.153, -116.316, -91.283, 74.260);
    JointPos j4=new JointPos(-89.575, -80.276, 102.713, -116.302, -91.284, 74.267);
    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 100.0;
    double blendT = -1.0;
    int flag = 0;

    int err1 = robot.MoveJ(j1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
```

(continues on next page)

(continued from previous page)

```
    System.out.println("movej errcode:"+ err1);
    robot.SplineStart();
    robot.SplinePTP(j1, tool, user, vel, acc, ovl);
    robot.SplinePTP(j2, tool, user, vel, acc, ovl);
    robot.SplinePTP(j3, tool, user, vel, acc, ovl);
    robot.SplinePTP(j4, tool, user, vel, acc, ovl);
    robot.SplineEnd();
    return 0;
}
```

2.3.4.40 New spline movement start

```
/**
* @brief New spline movement start
* @param [in] type   0-circular transition, 1-given points as path points
* @param [in] averageTime  Global average transition time(ms)(10 ~  ), default 2000
* @return Error code
*/
int NewSplineStart(int type, int averageTime);
```

2.3.4.41 New spline command point

```
/**
* @brief Add spline movement command point
* @param [in] joint_pos  Target joint position in deg
* @param [in] desc_pos   Target cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Speed percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not currently available
* @param [in] ovl  Speed scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius␣
↪(non-blocking) in mm
* @param [in] lastFlag Whether it is the last point, 0-no, 1-yes
* @return Error code
*/
int NewSplinePoint(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel,
↪ double acc, double ovl, double blendR, int lastFlag);
```

2.3.4.42 New spline command point (automatic inverse kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief New spline command point (automatic inverse kinematics calculation)
* @param [in] desc_pos  Target cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius␣
↪(non-blocking), unit mm
* @param [in] lastFlag Whether it is the last point, 0-no, 1-yes
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based␣
↪on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
int NewSplinePoint(DescPose desc_pos, int tool, int user, double vel, double acc, double␣
↪ovl, double blendR, int lastFlag,int config)
```

2.3.4.43 New spline movement end

```
/**
* @brief New spline movement start
* @return Error code
*/
int NewSplineEnd();
```

2.3.4.44 New spline movement code example

```
public static int TestNewSpline(Robot robot)
{
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    DescPose desc_pos1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);
    DescPose desc_pos3=new DescPose(-327.622, 402.230, 320.402, -178.067, 2.127, -46.
↪207);
    DescPose desc_pos4=new DescPose(-104.066, 544.321, 327.023, -177.715, 3.371, -73.
↪818);
    DescPose desc_pos5=new DescPose(-33.421, 732.572, 275.103, -177.907, 2.709, -79.482);
    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);


    int tool = 0;
    int user = 0;
    double vel = 100.0;
```

(continues on next page)

(continued from previous page)

```
    double acc = 100.0;
    double ovl = 100.0;
    double blendT = -1.0;
    int flag = 0;


    int err1 = robot.MoveJ(j1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    System.out.println("movej errcode:"+ err1);
    robot.NewSplineStart(1, 2000);
    robot.NewSplinePoint(desc_pos1, tool, user, vel, acc, ovl, -1, 0,-1);
    robot.NewSplinePoint(desc_pos2, tool, user, vel, acc, ovl, -1, 0,-1);
    robot.NewSplinePoint(desc_pos3, tool, user, vel, acc, ovl, -1, 0,-1);
    robot.NewSplinePoint(desc_pos4, tool, user, vel, acc, ovl, -1, 0,-1);
    robot.NewSplinePoint(desc_pos5, tool, user, vel, acc, ovl, -1, 0,-1);
    robot.NewSplineEnd();
    return 0;
}
```

2.3.4.45 Stop movement

```
/**
* @brief Stop movement
* @return  Error code
*/
int StopMotion();
```

2.3.4.46 Pause movement

```
/**
  * @brief Pause movement
  * @return Error code
*/
int PauseMotion();
```

2.3.4.47 Resume movement

```
/**
* @brief Resume movement
* @return Error code
*/
int ResumeMotion();
```

2.3.4.48 Movement pause, resume, stop code example

```
public static int TestPause(Robot robot)
{
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j5=new JointPos(-95.228, -54.621, 73.691, -112.245, -91.280, 74.268);
    DescPose desc_pos1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos5=new DescPose(-33.421, 732.572, 275.103, -177.907, 2.709, -79.482);
    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 100.0;
    double blendT = -1.0;
    int flag = 0;

    robot.SetSpeed(20);
    int rtn=-1;
    rtn = robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag,␣
↪offset_pos);
    rtn = robot.MoveJ(j5, desc_pos5, tool, user, vel, acc, ovl, epos, 1, flag, offset_
↪pos);
    robot.Sleep(1000);
    robot.PauseMotion();

    robot.Sleep(1000);
    robot.ResumeMotion();

    robot.Sleep(1000);
    robot.StopMotion();

    robot.Sleep(1000);

    return 0;
}
```

2.3.4.49 Point global offset start

```
/**
* @brief  Point global offset start
* @param  [in]  flag  0-offset in base/workpiece coordinate, 2-offset in tool coordinate
* @param  [in]  offset_pos  Pose offset
* @return  Error code
*/
int PointsOffsetEnable(int flag, DescPose offset_pos);
```

2.3.4.50 Point global offset end

```
/**
* @brief  Point global offset end
* @return  Error code
*/
int PointsOffsetDisable();
```

2.3.4.51 Point offset code example

```
public static int TestOffset(Robot robot)
{
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);

    DescPose desc_pos1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);

    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    DescPose offset_pos1=new DescPose(0, 0, 50, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 100.0;
    double blendT = -1.0;
    int flag = 0;

    robot.SetSpeed(20);

    robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.Sleep(1000);
    robot.PointsOffsetEnable(0, offset_pos1);
    robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.PointsOffsetDisable();

    return 0;
}
```

2.3.4.52 Controller AO flying start

```
/**
* @brief Controller AO flying start
* @param [in] AONum Controller AO number
* @param [in] maxTCPSpeed Maximum TCP speed value[1-5000mm/s], default 1000
* @param [in] maxAOPercent AO percentage corresponding to maximum TCP speed, default 100%
* @param [in] zeroZoneCmp Dead zone compensation value AO percentage, integer, default 20%, range [0-100]
* @return Error code
*/
int MoveAOStart(int AONum, int maxTCPSpeed, int maxAOPercent, int zeroZoneCmp);
```

2.3.4.53 Controller AO flying stop

```
/**
* @brief Controller AO flying stop
* @return Error code
*/
int MoveAOStop();
```

2.3.4.54 End effector AO flying start

```
/**
* @brief End effector AO flying start
* @param [in] AONum End effector AO number
* @param [in] maxTCPSpeed Maximum TCP speed value[1-5000mm/s], default 1000
* @param [in] maxAOPercent AO percentage corresponding to maximum TCP speed, default 100%
* @param [in] zeroZoneCmp Dead zone compensation value AO percentage, integer, default 20%, range [0-100]
* @return Error code
*/
int MoveToolAOStart(int AONum, int maxTCPSpeed, int maxAOPercent, int zeroZoneCmp);
```

2.3.4.55 End effector AO flying stop

```
/**
* @brief End effector AO flying stop
* @return Error code
*/
int MoveToolAOStop();
```

2.3.4.56 AO flying code example

```
public static int TestMoveAO(Robot robot)
{
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);

    DescPose desc_pos1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);

    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    DescPose offset_pos1=new DescPose(0, 0, 50, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);

    int tool = 0;
    int user = 0;
    double vel = 20.0;
    double acc = 20.0;
    double ovl = 100.0;
    double blendT = -1.0;
    int flag = 0;

    robot.SetSpeed(20);

    robot.MoveAOStart(0, 100, 100, 20);
    robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveAOStop();

    robot.Sleep(1000);

    robot.MoveToolAOStart(0, 100, 100, 20);
    robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_
↪pos);
    robot.MoveToolAOStop();

    return 0;
}
```

2.3.4.57 Start Ptp movement FIR filtering

Changed in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Start Ptp movement FIR filtering
* @param [in] maxAcc Maximum acceleration limit (deg/s2)
* @param [in] maxJek Unified joint jerk limit (deg/s3)
* @return Error code
*/
int PtpFIRPlanningStart(double maxAcc,double maxJek);
```

2.3.4.58 Close Ptp movement FIR filtering

```
/**
* @brief Close Ptp movement FIR filtering
* @return Error code
*/
int PtpFIRPlanningEnd();
```

2.3.4.59 Start LIN, ARC movement FIR filtering

```
/**
* @brief Start LIN, ARC movement FIR filtering
* @param [in] maxAccLin Linear acceleration limit (mm/s2)
* @param [in] maxAccDeg Angular acceleration limit (deg/s2)
* @param [in] maxJerkLin Linear jerk limit (mm/s3)
* @param [in] maxJerkDeg Angular jerk limit (deg/s3)
* @return Error code
*/
int LinArcFIRPlanningStart(double maxAccLin, double maxAccDeg, double maxJerkLin, double␣
↪maxJerkDeg);
```

2.3.4.60 Close LIN, ARC movement FIR filtering

```
/**
* @brief Close LIN, ARC movement FIR filtering
* @return Error code
*/
int LinArcFIRPlanningEnd();
```

2.3.4.61 FIR filtering code example

```
public static int TestFIR(Robot robot)
{
    JointPos startjointPos=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos midjointPos=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos endjointPos=new JointPos(-29.777, -84.536, 109.275, -114.075, -86.655, 74.257);

    DescPose startdescPose=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose middescPose=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose enddescPose=new DescPose(-487.434, 154.362, 308.576, 176.600, 0.268, -14.061);

    ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
    DescPose offdese=new DescPose(0, 0, 0, 0, 0, 0);

    int rtn = robot.PtpFIRPlanningStart(1000, 1000);
    robot.MoveJ(startjointPos, startdescPose, 0, 0, 100, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    robot.MoveJ(endjointPos, enddescPose, 0, 0, 100, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    robot.PtpFIRPlanningEnd();

    robot.LinArcFIRPlanningStart(1000, 1000, 1000, 1000);
    robot.MoveL(startjointPos, startdescPose, 0, 0, 100, 100, 100, -1, 0,exaxisPos, 0, 0,
↪ offdese, 1, 1);
    robot.MoveC(midjointPos, middescPose, 0, 0, 100, 100, exaxisPos, 0, offdese,␣
↪endjointPos, enddescPose, 0, 0, 100, 100, exaxisPos, 0, offdese, 100, -1);
    robot.LinArcFIRPlanningEnd();
    return 0;
}
```

2.3.4.62 Acceleration smoothing enable

New in version Java: SDK-v1.0.4-3.8.1

```
/**
 * @brief Acceleration smoothing enable
 * @param [in] saveFlag Whether to save after power off
 * @return  Error code
 */
public int AccSmoothStart(boolean saveFlag)
```

2.3.4.63 Acceleration smoothing disable

New in version Java: SDK-v1.0.4-3.8.1

```
/**
 * @brief Acceleration smoothing disable
 * @param [in] saveFlag Whether to save after power off
 * @return  Error code
 */
public int AccSmoothEnd(boolean saveFlag)
```

2.3.4.64 Acceleration smoothing code example

```
public static int TestAccSmooth(Robot robot)
{
    JointPos startjointPos=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos endjointPos=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);

    DescPose startdescPose=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose enddescPose=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);

    ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
    DescPose offdese=new DescPose(0,0,0,0,0,0);
    int rtn = robot.AccSmoothStart(false);
    robot.MoveJ(startjointPos, startdescPose, 0, 0, 100, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    robot.MoveJ(endjointPos, enddescPose, 0, 0, 100, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    rtn = robot.AccSmoothEnd(false);

    robot.CloseRPC();
    return 0;
}
```

2.3.4.65 Specified pose speed enable

```
/**
 * @brief Specified pose speed enable
 * @param [in] ratio Pose speed percentage [0-300]
 * @return  Error code
 */
int AngularSpeedStart(int ratio)
```

2.3.4.66 Specified pose speed disable

```
/**
 * @brief Specified pose speed disable
 * @return  Error code
 */
int AngularSpeedEnd();
```

2.3.4.67 Robot specified pose speed code example

```
public static int TestAngularSpeed(Robot robot)
{
    JointPos startjointPos=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos endjointPos=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);

    DescPose startdescPose=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose enddescPose=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);

    ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
    DescPose offdese=new DescPose(0, 0, 0, 0, 0, 0);
    int rtn = robot.AngularSpeedStart(50);
    robot.MoveJ(startjointPos, startdescPose, 0, 0, 100, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    robot.MoveJ(endjointPos, enddescPose, 0, 0, 100, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    rtn = robot.AngularSpeedEnd();

    return 0;
}
```

2.3.4.68 Start singular pose protection

```
/**
* @brief  Start singular pose protection
* @param  [in]  protectMode Singular protection mode, 0: joint mode; 1-cartesian mode
* @param  [in]  minShoulderPos Shoulder singular adjustment range(mm), default 100
* @param  [in]  minElbowPos Elbow singular adjustment range(mm), default 50
* @param  [in]  minWristPos Wrist singular adjustment range(°), default 10
* @return  Error code
*/
int SingularAvoidStart(int protectMode, double minShoulderPos, double minElbowPos,␣
↪double minWristPos);
```

2.3.4.69 Stop singular pose protection

```
/**
* @brief  Stop singular pose protection
* @return  Error code
*/
int SingularAvoidEnd();
```

2.3.4.70 Robot singular pose protection code example

```
public static int TestAngularSpeed(Robot robot)
{
    JointPos startjointPos=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos endjointPos=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);

    DescPose startdescPose=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose enddescPose=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);

    ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
    DescPose offdese=new DescPose(0, 0, 0, 0, 0, 0);
    int rtn = robot.AngularSpeedStart(50);
    robot.MoveJ(startjointPos, startdescPose, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.MoveJ(endjointPos, enddescPose, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    rtn = robot.AngularSpeedEnd();

    return 0;
}
```

2.3.4.71 Clear Motion Command Queue

```
/**
* @brief Clears the motion command queue
* @return Error code
*/
public int MotionQueueClear()
```

2.3.4.72 Move to Intersecting Line Start Point

```
/**
* @brief Move to intersecting line start point
* @param [in] mainPoint Cartesian poses of 6 taught points on the main pipe
* @param [in] mainExaxisPos Extended axis positions for 6 taught points on the main pipe
* @param [in] piecePoint Cartesian poses of 6 taught points on the branch pipe
* @param [in] pieceExaxisPos Extended axis positions for 6 taught points on the branch␣
↪pipe
* @param [in] extAxisFlag Whether to enable extended axis; 0-Disable; 1-Enable
* @param [in] exaxisPos Start point extended axis position
* @param [in] tool Tool coordinate system number
* @param [in] wobj Workpiece coordinate system number
* @param [in] vel Velocity percentage
* @param [in] acc Acceleration percentage
* @param [in] ovl Velocity scaling factor
* @param [in] oacc Acceleration scaling factor
* @param [in] moveType Motion type; 0-PTP; 1-LIN
* @param [in] moveDirection Motion direction; 0-Clockwise; 1-Counterclockwise
* @param [in] offset Offset value
* @return Error code
*/
public int MoveToIntersectLineStart(DescPose[] mainPoint, ExaxisPos[] mainExaxisPos,␣
↪DescPose[] piecePoint, ExaxisPos[] pieceExaxisPos, int extAxisFlag, ExaxisPos␣
↪exaxisPos, int tool, int wobj, double vel, double acc, double ovl, double oacc, int␣
↪moveType, int moveDirection, DescPose offset);
```

2.3.4.73 Intersecting Line Motion

```
/**
* @brief Intersecting line motion
* @param [in] mainPoint Cartesian poses of 6 taught points on the main pipe
* @param [in] mainExaxisPos Extended axis positions for 6 taught points on the main pipe
* @param [in] piecePoint Cartesian poses of 6 taught points on the branch pipe
* @param [in] pieceExaxisPos Extended axis positions for 6 taught points on the branch␣
↪pipe
* @param [in] extAxisFlag Whether to enable extended axis; 0-Disable; 1-Enable
* @param [in] exaxisPos Start point extended axis positions
* @param [in] tool Tool coordinate system number
* @param [in] wobj Workpiece coordinate system number
* @param [in] vel Velocity percentage
* @param [in] acc Acceleration percentage
* @param [in] ovl Velocity scaling factor
* @param [in] oacc Acceleration scaling factor
* @param [in] moveDirection Motion direction; 0-Clockwise; 1-Counterclockwise
* @param [in] offset Offset value
* @return Error code
*/
public int MoveIntersectLine(DescPose[] mainPoint, ExaxisPos[] mainExaxisPos, DescPose[]␣
↪piecePoint, ExaxisPos[] pieceExaxisPos, int extAxisFlag, ExaxisPos[] exaxisPos, int␣
↪tool, int wobj, double vel, double acc, double ovl, double oacc, int moveDirection,␣
↪DescPose offset);
```

2.3.4.74 Robot Intersecting Line Motion Code Example

```
public static void TestIntersectLineMove(Robot robot)
{
    DescPose[] mainPoint = new DescPose[6];
    DescPose[] piecePoint = new DescPose[6];
    ExaxisPos[] mainExaxisPos = new ExaxisPos[6];
    ExaxisPos[] pieceExaxisPos = new ExaxisPos[6];
    int extAxisFlag = 1;
    ExaxisPos[] exaxisPos = new ExaxisPos[4];
    DescPose offset =new DescPose(0.0, 2.0 ,30.0, -2.0, 0.0, 0.0 );
    mainPoint[0] = new DescPose(490.004, -383.194, 402.735, -9.332, -1.528, 69.594);
    mainPoint[1] = new DescPose(444.950, -407.117, 389.011, -5.546, -2.196, 65.279);
    mainPoint[2] = new DescPose(445.168, -463.605, 355.759, -1.544, -10.886, 57.104);
    mainPoint[3] = new DescPose(507.529, -485.385, 343.013, -0.786, -4.834, 61.799);
    mainPoint[4] = new DescPose(554.390, -442.647, 367.701, -4.761, -10.181, 64.925);
    mainPoint[5] = new DescPose(532.552, -394.003, 396.467, -13.732, -13.592, 67.411);
    mainExaxisPos[0] = new ExaxisPos(-29.996, 0.000, 0.000, 0.000 );
    mainExaxisPos[1] = new ExaxisPos(-29.996, 0.000, 0.000, 0.000 );
    mainExaxisPos[2] = new ExaxisPos(-29.996, 0.000, 0.000, 0.000 );
    mainExaxisPos[3] = new ExaxisPos(-29.996, 0.000, 0.000, 0.000 );
    mainExaxisPos[4] = new ExaxisPos(-29.996, 0.000, 0.000, 0.000 );
    mainExaxisPos[5] = new ExaxisPos(-29.996, 0.000, 0.000, 0.000 );
    piecePoint[0] = new DescPose( 505.571, -192.408, 316.759, 38.098, 37.051, 139.447);
    piecePoint[1] =new DescPose(533.837, -201.558, 332.340, 34.644, 42.339, 137.748);
    piecePoint[2] =new DescPose(530.386, -225.085, 373.808, 35.431, 45.111, 137.560);
    piecePoint[3] =new DescPose(485.646, -229.195, 383.778, 33.870, 45.173, 137.064);
    piecePoint[4] =new DescPose(460.551, -212.161, 354.256, 28.856, 45.602, 135.930);
    piecePoint[5] =new DescPose(474.217, -197.124, 324.611, 42.469, 41.133, 148.167);
    pieceExaxisPos[0] = new ExaxisPos( -29.996, -0.000, 0.000, 0.000);
    pieceExaxisPos[1] = new ExaxisPos( -29.996, -0.000, 0.000, 0.000);
    pieceExaxisPos[2] = new ExaxisPos( -29.996, -0.000, 0.000, 0.000);
    pieceExaxisPos[3] = new ExaxisPos( -29.996, -0.000, 0.000, 0.000);
    pieceExaxisPos[4] = new ExaxisPos( -29.996, -0.000, 0.000, 0.000);
    pieceExaxisPos[5] = new ExaxisPos( -29.996, -0.000, 0.000, 0.000);
    exaxisPos[0] = new ExaxisPos(-29.996, -0.000, 0.000, 0.000);
    exaxisPos[1] = new ExaxisPos(-44.994, 90.000, 0.000, 0.000);
    exaxisPos[2] = new ExaxisPos(-59.992, 0.002, 0.000, 0.000);
    exaxisPos[3] = new ExaxisPos(-44.994, -89.997, 0.000, 0.000);
    int tool = 2;
    int wobj = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 12.0;
    double oacc = 12.0;
    int moveType = 1;
    int moveDirection = 1;
    int  rtn = robot.MoveToIntersectLineStart(mainPoint, mainExaxisPos, piecePoint,␣
↪pieceExaxisPos, extAxisFlag, exaxisPos[0], tool, wobj, vel, acc, ovl, oacc, moveType,␣
↪moveDirection, offset);
    System.out.printf("MoveToIntersectLineStart rtn is %d\n", rtn);
    rtn = robot.MoveIntersectLine(mainPoint, mainExaxisPos, piecePoint, pieceExaxisPos,␣
↪extAxisFlag, exaxisPos, tool, wobj, vel, acc, 5.0, 5.0, moveDirection, offset);
```

(continues on next page)

(continued from previous page)

```
    System.out.printf("MoveIntersectLine rtn is %d\n", rtn);
    robot.CloseRPC();
    return ;
}
```

2.3.4.75 Stationary Air Motion

```
/**
* @brief Stationary Air Motion
* @return Error code
*/
public int MoveStationary()
```

2.3.4.76 Stationary Air Motion Code Example

```
public static void test_RecordandReplay(Robot robot)
{
    int rtn = robot.LaserSensorRecordandReplay(0, 10, 1, 0, 0.1, 1, 1, 10, 100);
    System.out.printf("LaserSensorRecordandReplay rtn is %d\n", rtn);
    rtn = robot.MoveStationary();
    System.out.printf("MoveStationary rtn is %d\n", rtn);
    rtn = robot.LaserSensorRecord1(0, 10);
    System.out.printf("LaserSensorRecordandReplay rtn is %d\n", rtn);
    robot.CloseRPC();
    robot.Sleep(9999999);
}
```

2.3.5 Robot IO

2.3.5.1 Set control box digital output

```
/**
* @brief  Set control box digital output
* @param  [in] id  IO number, range [0~15]
* @param  [in] status 0-off, 1-on
* @param  [in] smooth 0-no smoothing, 1-smoothing
* @param  [in] block  0-blocking, 1-non-blocking
* @return  Error code
*/
int SetDO(int id, int status, int smooth, int block);
```

2.3.5.2 Set tool digital output

```
/**
* @brief  Set tool digital output
* @param  [in] id  IO number, range [0~1]
* @param  [in] status 0-off, 1-on
* @param  [in] smooth 0-no smoothing, 1-smoothing
* @param  [in] block  0-blocking, 1-non-blocking
* @return  Error code
*/
int SetToolDO(int id, int status, int smooth, int block);
```

2.3.5.3 Set control box analog output

```
/**
* @brief  Set control box analog output
* @param  [in] id  IO number, range [0~1]
* @param  [in] value Current or voltage value percentage, range [0~100] corresponding to␣
↪current [0~20mA] or voltage [0~10V]
* @param  [in] block  0-blocking, 1-non-blocking
* @return  Error code
*/
int SetAO(int id, double value, int block);
```

2.3.5.4 Set tool analog output

```
/**
* @brief  Set tool analog output
* @param  [in] id  IO number, range [0]
* @param  [in] value Current or voltage value percentage, range [0~100] corresponding to␣
↪current [0~20mA] or voltage [0~10V]
* @param  [in] block  0-blocking, 1-non-blocking
* @return  Error code
*/
int SetToolAO(int id, double value, int block);
```

2.3.5.5 Digital and analog output setting example

```
public static int TestAODO(Robot robot)
{
    int status = 1;
    int smooth = 0;
    int block = 0;

    for (int i = 0; i < 16; i++)
    {
        robot.SetDO(i, status, smooth, block);
        robot.Sleep(300);
```

(continues on next page)

(continued from previous page)

```
    }

    status = 0;

    for (int i = 0; i < 16; i++)
    {
        robot.SetDO(i, status, smooth, block);
        robot.Sleep(300);
    }

    status = 1;

    for (int i = 0; i < 2; i++)
    {
        robot.SetToolDO(i, status, smooth, block);
        robot.Sleep(1000);
    }

    status = 0;

    for (int i = 0; i < 2; i++)
    {
        robot.SetToolDO(i, status, smooth, block);
        robot.Sleep(1000);
    }

    for (int i = 0; i < 100; i++)
    {
        robot.SetAO(0, i, block);
        robot.Sleep(30);
    }

    for (int i = 0; i < 100; i++)
    {
        robot.SetToolAO(0, i, block);
        robot.Sleep(30);
    }

    robot.CloseRPC();
    return 0;
}
```

2.3.5.6 Get control box digital input

```
/**
* @brief  Get control box digital input
* @param  [in] id  IO number, range [0~15]
* @param  [in] block  0-blocking, 1-non-blocking
* @param  [out] level  0-low level, 1-high level
* @return  Error code
*/
int GetDI(int id, int block, int[] level);
```

2.3.5.7 Get tool digital input

```
/**
* @brief  Get tool digital input
* @param  [in] id  IO number, range [0~1]
* @param  [in] block  0-blocking, 1-non-blocking
* @param  [out] level 0-low level, 1-high level
* @return  Error code
*/
int GetToolDI(int id, int block, int[] level);
```

2.3.5.8 Get control box analog input

```
/**
* @brief  Get control box analog input
* @param  [in] id  IO number, range [0~1]
* @param  [in] block  0-blocking, 1-non-blocking
* @param  [out] persent Input current or voltage value percentage, range [0~100]␣
↪corresponding to current [0~20mA] or voltage [0~10V]
* @return  Error code
*/
int GetAI(int id, int block, double[] persent)
```

2.3.5.9 Get tool analog input

```
/**
* @brief  Get tool analog input
* @param  [in] id  IO number, range [0]
* @param  [in] block  0-blocking, 1-non-blocking
* @param  [out] persent Input current or voltage value percentage, range [0~100]␣
↪corresponding to current [0~20mA] or voltage [0~10V]
* @return  Error code
*/
int GetToolAI(int id, int block, double[] persent)
```

2.3.5.10 Get robot end-point record button state

```
/**
* @brief  Get robot end-point record button state
* @param  [out] state Button state, 0-pressed, 1-released
* @return  Error code
*/
int GetAxlePointRecordBtnState(int[] state)
```

2.3.5.11 Get robot end DO output state

```
/**
* @brief  Get robot end DO output state
* @param  [out] do_state DO output state, do0~do1 corresponds to bit1~bit2, starting␣
↪from bit0
* @return  Error code
*/
int GetToolDO(int[] do_state)
```

2.3.5.12 Get robot controller DO output state

```
/**
* @brief  Get robot controller DO output state
* @param  [out] do_state_h DO output state, co0~co7 corresponds to bit0~bit7
* @param  [out] do_state_l DO output state, do0~do7 corresponds to bit0~bit7
* @return  Error code
*/
int GetDO(int[] do_state_h, int[] do_state_l)
```

2.3.5.13 Get robot DI/DO state example

```
public static int TestGetDIAI(Robot robot)
{
    int status = 1;
    int smooth = 0;
    int block = 0;
    int[] di =new int[]{0}, tool_di =new int[] {0};
    double[] ai =new double[] {0}, tool_ai = new double[]{0};
    double value = 0.0;

    robot.GetDI(0, block, di);
    System.out.println("di0:"+di[0]);

    robot.GetToolDI(1, block, tool_di);
    System.out.println("tool_di1:"+ tool_di[0]);

    robot.GetAI(0, block, ai);
    System.out.println("ai0:"+ ai[0]);
```

(continues on next page)

(continued from previous page)

```
    robot.GetToolAI(0, block, tool_ai);
    System.out.println("tool_ai0:"+ tool_ai[0]);

    int[] _button_state=new int[]{0};
    robot.GetAxlePointRecordBtnState(_button_state);
    System.out.println("_button_state is: "+ _button_state[0]);

    int[] tool_do_state=new int[]{0};
    robot.GetToolDO(tool_do_state);
    System.out.println("tool DO state is: "+ tool_do_state[0]);

    int[] do_state_h=new int[]{0};
    int[] do_state_l=new int[]{0};
    robot.GetDO(do_state_h, do_state_l);
    System.out.println("DO state high is: "+do_state_h[0]+", DO state low is: "+ do_
↪state_l[0]);

    return 0;
}
```

2.3.5.14 Wait for control box digital input

```
/**
* @brief Wait for control box digital input
* @param  [in]  id  IO number, range [0~15]
* @param  [in]  status 0-off, 1-on
* @param  [in]  max_time  Maximum wait time in ms
* @param  [in]  opt  Timeout strategy, 0-program stops and prompts timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
* @return  Error code
*/
int WaitDI(int id, int status, int max_time, int opt);
```

2.3.5.15 Wait for multiple control box digital inputs

```
/**
* @brief Wait for multiple control box digital inputs
* @param  [in] mode 0-multi AND, 1-multi OR
* @param  [in] id  IO number, bit0~bit7 corresponds to DI0~DI7, bit8~bit15 corresponds␣
↪to CI0~CI7
* @param  [in] status 0-off, 1-on
* @param  [in] max_time  Maximum wait time in ms
* @param  [in] opt  Timeout strategy, 0-program stops and prompts timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
* @return  Error code
*/
int WaitMultiDI(int mode, int id, int status, int max_time, int opt);
```

2.3.5.16 Wait for tool digital input

```
/**
* @brief Wait for tool digital input
* @param [in] id IO number, range [0~1]
* @param [in] status 0-off, 1-on
* @param [in] max_time Maximum wait time in ms
* @param [in] opt Timeout strategy, 0-program stops and prompts timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
* @return Error code
*/
int WaitToolDI(int id, int status, int max_time, int opt);
```

2.3.5.17 Wait for control box analog input

```
/**
* @brief Wait for control box analog input
* @param [in] id IO number, range [0~1]
* @param [in] sign 0-greater than, 1-less than
* @param [in] value Input current or voltage value percentage, range [0~100]␣
↪corresponding to current [0~20mA] or voltage [0~10V]
* @param [in] max_time Maximum wait time in ms
* @param [in] opt Timeout strategy, 0-program stops and prompts timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
* @return Error code
*/
int WaitAI(int id, int sign, double value, int max_time, int opt);
```

2.3.5.18 Wait for tool analog input

```
/**
* @brief Wait for tool analog input
* @param [in] id IO number, range [0]
* @param [in] sign 0-greater than, 1-less than
* @param [in] value Input current or voltage value percentage, range [0~100]␣
↪corresponding to current [0~20mA] or voltage [0~10V]
* @param [in] max_time Maximum wait time in ms
* @param [in] opt Timeout strategy, 0-program stops and prompts timeout, 1-ignore␣
↪timeout and continue, 2-wait indefinitely
* @return Error code
*/
int WaitToolAI(int id, int sign, double value, int max_time, int opt);
```

2.3.5.19 Wait for digital/analog input signal example

```
public static int TestWaitDIAI(Robot robot)
{
    int rtn=-1;

    int status = 1;
    int smooth = 0;
    int block = 0;
    int di = 0, tool_di = 0;
    double ai = 0.0, tool_ai = 0.0;
    double value = 0.0;

    rtn = robot.WaitDI(0, 1, 1000, 1);
    System.out.println("WaitDI over; rtn is: "+ rtn);

    robot.WaitMultiDI(1, 3, 3, 1000, 1);
    System.out.println("WaitDI over; rtn is: "+ rtn);

    robot.WaitToolDI(1, 1, 1000, 1);
    System.out.println("WaitDI over; rtn is: " + rtn);

    robot.WaitAI(0, 0, 50, 1000, 1);
    System.out.println("WaitDI over; rtn is: " + rtn);

    robot.WaitToolAI(0, 0, 50, 1000, 1);
    System.out.println("WaitDI over; rtn is: " + rtn);
    return 0;
}
```

2.3.5.20 Set Whether Control Box DO Output Resets After Stop/Pause

```
/**
* @brief Set whether control box DO output resets after stop/pause
* @param resetFlag 0-Do not reset; 1-Reset
* @param reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetCtlBoxDO(int resetFlag, int reloadFlag)
```

2.3.5.21 Set Whether Control Box AO Output Resets After Stop/Pause

```
/**
* @brief Set whether control box AO output resets after stop/pause
* @param resetFlag 0-Do not reset; 1-Reset
* @param reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetCtlBoxAO(int resetFlag, int reloadFlag)
```

2.3.5.22 Set Whether End Tool DO Output Resets After Stop/Pause

```
/**
* @brief Set whether end tool DO output resets after stop/pause
* @param resetFlag 0-Do not reset; 1-Reset
* @param reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetAxleDO(int resetFlag, int reloadFlag)
```

2.3.5.23 Set Whether End Tool AO Output Resets After Stop/Pause

```
/**
* @brief Set whether end tool AO output resets after stop/pause
* @param resetFlag 0-Do not reset; 1-Reset
* @param reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetAxleAO(int resetFlag, int reloadFlag)
```

2.3.5.24 Set Whether Extended DO Output Resets After Stop/Pause

```
/**
* @brief Set whether extended DO output resets after stop/pause
* @param resetFlag 0-Do not reset; 1-Reset
* @param reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetExtDO(int resetFlag, int reloadFlag)
```

2.3.5.25 Set Whether Extended AO Output Resets After Stop/Pause

```
/**
* @brief Set whether extended AO output resets after stop/pause
* @param resetFlag 0-Do not reset; 1-Reset
* @param reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetExtAO(int resetFlag, int reloadFlag)
```

2.3.5.26 Set Whether SmartTool Output Resets After Stop/Pause

```
/**
* @brief Set whether SmartTool output resets after stop/pause
* @param resetFlag 0-Do not reset; 1-Reset
* @param reloadFlag Whether to reload after pause resume, 0-Do not load; 1-Load
* @return Error code
*/
public int SetOutputResetSmartToolDO(int resetFlag, int reloadFlag)
```

2.3.5.27 Code Example for Setting Output Reset After Lua Program Stop/Pause

```
public static void TestDOReset(Robot robot)
{
    for (int i = 0; i < 16; i++)
    {
        robot.SetDO(i, 1, 0, 0);
        robot.Sleep(200);
    }
    int resetFlag = 1;
    int resumeReloadFlag = 1;
    int rtn = robot.SetOutputResetCtlBoxDO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetCtlBoxAO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetAxleDO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetAxleAO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetExtDO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetExtAO(resetFlag, resumeReloadFlag);
    robot.SetOutputResetSmartToolDO(resetFlag, resumeReloadFlag);
    robot.ProgramLoad("/fruser/test.lua");
    robot.ProgramRun();
    robot.Sleep(2000);
    robot.PauseMotion();
    robot.Sleep(2000);
    robot.ResumeMotion();
    robot.Sleep(2000);
    robot.CloseRPC();
}
```

2.3.6 Robot common settings

2.3.6.1 Set tool reference point - six point method

```
/**
* @brief Set tool reference point - six point method
* @param [in] point_num Point number, range [1~6]
* @return Error code
*/
int SetToolPoint(int point_num);
```

2.3.6.2 Calculate tool coordinate system - six point method

```
/**
* @brief Calculate tool coordinate system
* @param [out] tcp_pose Tool coordinate system
* @return Error code
*/
int ComputeTool(DescPose tcp_pose);
```

2.3.6.3 Set tool reference point - four point method

```
/**
* @brief Set tool reference point - four point method
* @param [in] point_num Point number, range [1~4]
* @return Error code
*/
int SetTcp4RefPoint(int point_num);
```

2.3.6.4 Calculate tool coordinate system - four point method

```
/**
* @brief Calculate tool coordinate system
* @param [out] tcp_pose Tool coordinate system
* @return Error code
*/
int ComputeTcp4(DescPose tcp_pose);
```

2.3.6.5 Calculate tool coordinate system based on point information

```
/**
* @brief Calculate tool coordinate system based on point information
* @param [in] method Calculation method; 0-four point method; 1-six point method
* @param [in] pos Joint position group, array length is 4 for four point method, 6 for six point method
* @param [out] tool_pose Output tool coordinate system
* @return Error code
*/
int ComputeToolCoordWithPoints(int method, JointPos[] pos, DescPose tool_pose);
```

2.3.6.6 Set tool coordinate system

```
/**
* @brief Set tool coordinate system
* @param [in] id Coordinate system number, range [0~14]
* @param [in] coord Tool center point pose relative to end flange center
* @param [in] type 0-tool coordinate system, 1-sensor coordinate system
* @param [in] install Installation position, 0-robot end, 1-external to robot
* @param [in] toolID Tool ID
* @param [in] loadNum Load number
* @return Error code
*/
int SetToolCoord(int id, DescPose coord, int type, int install, int toolID, int loadNum);
```

2.3.6.7 Set tool coordinate system list

```
/**
* @brief Set tool coordinate system list
* @param [in] id Coordinate system number, range [0~14]
* @param [in] coord Tool center point pose relative to end flange center
* @param [in] type 0-tool coordinate system, 1-sensor coordinate system
* @param [in] install Installation position, 0-robot end, 1-external to robot
* @param [in] loadNum Load number
* @return Error code
*/
int SetToolList(int id, DescPose coord, int type, int install, int loadNum);
```

2.3.6.8 Get current tool coordinate system

```
/**
* @brief Get current tool coordinate system
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] desc_pos Tool coordinate system pose
* @return Error code
*/
int GetTCPOffset(int flag, DescPose desc_pos);
```

2.3.6.9 Robot tool coordinate system operation code example

```
public static int TestTCPCompute(Robot robot)
{
    DescPose p1Desc=new DescPose(186.331, 487.913, 209.850, 149.030, 0.688, -114.347);
    JointPos p1Joint=new JointPos(-127.876, -75.341, 115.417, -122.741, -59.820, 74.300);

    DescPose p2Desc=new DescPose(69.721, 535.073, 202.882, -144.406, -14.775, -89.012);
    JointPos p2Joint=new JointPos(-101.780, -69.828, 110.917, -125.740, -127.841, 74.
↪300);

```

(continues on next page)

(continued from previous page)

```
    DescPose p3Desc=new DescPose(146.861, 578.426, 205.598, 175.997, -36.178, -93.437);
    JointPos p3Joint=new JointPos(-112.851, -60.191, 86.566, -80.676, -97.463, 74.300);

    DescPose p4Desc=new DescPose(136.284, 509.876, 225.613, 178.987, 1.372, -100.696);
    JointPos p4Joint=new JointPos(-116.397, -76.281, 113.845, -128.611, -88.654, 74.299);

    DescPose p5Desc=new DescPose(138.395, 505.972, 298.016, 179.134, 2.147, -101.110);
    JointPos p5Joint=new JointPos(-116.814, -82.333, 109.162, -118.662, -88.585, 74.302);

    DescPose p6Desc=new DescPose(105.553, 454.325, 232.017, -179.426, 0.444, -99.952);
    JointPos p6Joint=new JointPos(-115.649, -84.367, 122.447, -128.663, -90.432, 74.303);

    ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
    DescPose offdese=new DescPose(0, 0, 0, 0, 0, 0);

    JointPos[] posJ = { p1Joint , p2Joint , p3Joint , p4Joint , p5Joint , p6Joint };
    DescPose coordRtn =new DescPose() {};
    int rtn = robot.ComputeToolCoordWithPoints(1, posJ, coordRtn);

    robot.MoveJ(p1Joint, p1Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetToolPoint(1);
    robot.MoveJ(p2Joint, p2Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetToolPoint(2);
    robot.MoveJ(p3Joint, p3Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetToolPoint(3);
    robot.MoveJ(p4Joint, p4Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetToolPoint(4);
    robot.MoveJ(p5Joint, p5Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetToolPoint(5);
    robot.MoveJ(p6Joint, p6Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetToolPoint(6);
    rtn = robot.ComputeTool(coordRtn);
    robot.SetToolList(3, coordRtn, 0, 0, 0);

    robot.MoveJ(p1Joint, p1Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(1);
    robot.MoveJ(p2Joint, p2Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(2);
    robot.MoveJ(p3Joint, p3Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(3);
    robot.MoveJ(p4Joint, p4Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(4);
    rtn = robot.ComputeTcp4(coordRtn);

    robot.SetToolCoord(4, coordRtn, 0, 0, 1, 0);

    DescPose getCoord = new DescPose(){};
    rtn = robot.GetTCPOffset(0, getCoord);
    return 0;
}
```

2.3.6.10 Set external tool reference point - six point method

```
/**
* @brief Set external tool reference point - three point method
* @param [in] point_num Point number, range [1~3]
* @return Error code
*/
int SetExTCPPoint(int point_num);
```

2.3.6.11 Calculate external tool coordinate system - six point method

```
/**
* @brief Calculate external tool coordinate system - three point method
* @param [out] tcp_pose External tool coordinate system
* @return Error code
*/
int ComputeExTCF(DescPose tcp_pose);
```

2.3.6.12 Set external tool coordinate system

```
/**
* @brief Set external tool coordinate system
* @param [in] id Coordinate system number, range [0~14]
* @param [in] etcp Tool center point pose relative to end flange center
* @param [in] etool To be determined
* @return Error code
*/
int SetExToolCoord(int id, DescPose etcp, DescPose etool);
```

2.3.6.13 Set external tool coordinate system list

```
/**
* @brief Set external tool coordinate system list
* @param [in] id Coordinate system number, range [0~14]
* @param [in] etcp Tool center point pose relative to end flange center
* @param [in] etool To be determined
* @return Error code
*/
int SetExToolList(int id, DescPose etcp, DescPose etool);
```

2.3.6.14 Robot external tool coordinate system operation code example

```
public static int TestExtCoord(Robot robot)
{
    DescPose p1Desc=new DescPose(-89.606, 779.517, 193.516, 178.000, 0.476, -92.484);
    JointPos p1Joint=new JointPos(-108.145, -50.137, 85.818, -125.599, -87.946, 74.329);

    DescPose p2Desc=new DescPose(-24.656, 850.384, 191.361, 177.079, -2.058, -95.355);
    JointPos p2Joint=new JointPos(-111.024, -41.538, 69.222, -114.913, -87.743, 74.329);

    DescPose p3Desc=new DescPose(-99.813, 766.661, 241.878, -176.817, 1.917, -91.604);
    JointPos p3Joint=new JointPos(-107.266, -56.116, 85.971, -122.560, -92.548, 74.331);

    robot.GetForwardKin(p1Joint,p1Desc);
    robot.GetForwardKin(p2Joint,p2Desc);
    robot.GetForwardKin(p3Joint,p3Desc);

    ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
    DescPose offdese=new DescPose(0, 0, 0, 0, 0, 0);

    DescPose[] posTCP = { p1Desc , p2Desc , p3Desc };
    DescPose coordRtn = new DescPose();

    robot.MoveJ(p1Joint, p1Desc, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetExTCPPoint(1);
    robot.MoveJ(p2Joint, p2Desc, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetExTCPPoint(2);
    robot.MoveJ(p3Joint, p3Desc, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetExTCPPoint(3);
    int rtn = robot.ComputeExTCF(coordRtn);

    robot.SetExToolCoord(1, coordRtn, offdese);
    robot.SetExToolList(1, coordRtn, offdese);
    return 0;
}
```

2.3.6.15 Set workpiece reference point - three point method

```
/**
* @brief Set workpiece reference point - three point method
* @param [in] point_num Point number, range [1~3]
* @return Error code
*/
int SetWObjCoordPoint(int point_num);
```

2.3.6.16 Calculate workpiece coordinate system

```
/**
* @brief Calculate workpiece coordinate system
* @param [in] method Calculation method 0: origin-x axis-z axis 1: origin-x axis-xy plane
* @param [in] refFrame Reference coordinate system
* @param [out] wobj_pose Workpiece coordinate system
* @return Error code
*/
int ComputeWObjCoord(int method, int refFrame, DescPose wobj_pose);
```

2.3.6.17 Set workpiece coordinate system

```
/**
* @brief Set workpiece coordinate system
* @param [in] id Coordinate system number, range [1~15]
* @param [in] coord Workpiece coordinate system pose relative to end flange center
* @param [in] refFrame Reference coordinate system
* @return Error code
*/
int SetWObjCoord(int id, DescPose coord, int refFrame);
```

2.3.6.18 Set workpiece coordinate system list

```
/**
* @brief Set workpiece coordinate system list
* @param [in] id Coordinate system number, range [1~15]
* @param [in] coord Workpiece coordinate system pose relative to end flange center
* @param [in] refFrame Reference coordinate system
* @return Error code
*/
int SetWObjList(int id, DescPose coord, int refFrame);
```

2.3.6.19 Calculate workpiece coordinate system based on point information

```
/**
* @brief Calculate workpiece coordinate system based on point information
* @param [in] method Calculation method; 0: origin-x axis-z axis 1: origin-x axis-xy␣
↪plane
* @param [in] pos Three TCP position groups
* @param [in] refFrame Reference coordinate system
* @param [in] tcp_pose Output workpiece coordinate system
* @return Error code
*/
int ComputeWObjCoordWithPoints(int method, DescPose[] pos, int refFrame,DescPose tcp_
↪pose);
```

2.3.6.20 Get current workpiece coordinate system

```
/**
* @brief Get current workpiece coordinate system
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] desc_pos Workpiece coordinate system pose
* @return Error code
*/
int GetWObjOffset(int flag, DescPose desc_pos);
```

2.3.6.21 Workpiece coordinate system operation code example

```
public static int TestWobjCoord(Robot robot)
{
    DescPose p1Desc=new DescPose(-89.606, 779.517, 193.516, 178.000, 0.476, -92.484);
    JointPos p1Joint=new JointPos(-108.145, -50.137, 85.818, -125.599, -87.946, 74.329);

    DescPose p2Desc=new DescPose(-24.656, 850.384, 191.361, 177.079, -2.058, -95.355);
    JointPos p2Joint=new JointPos(-111.024, -41.538, 69.222, -114.913, -87.743, 74.329);

    DescPose p3Desc=new DescPose(-99.813, 766.661, 241.878, -176.817, 1.917, -91.604);
    JointPos p3Joint=new JointPos(-107.266, -56.116, 85.971, -122.560, -92.548, 74.331);

    robot.GetForwardKin(p1Joint,p1Desc);
    robot.GetForwardKin(p2Joint,p2Desc);
    robot.GetForwardKin(p3Joint,p3Desc);

    ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
    DescPose offdese=new DescPose(0, 0, 0, 0, 0, 0);

    DescPose[] posTCP =new DescPose[]{p1Desc , p2Desc , p3Desc };
    DescPose coordRtn =new DescPose();
    int rtn = robot.ComputeWObjCoordWithPoints(1, posTCP, 0, coordRtn);

    robot.MoveJ(p1Joint, p1Desc, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetWObjCoordPoint(1);
    robot.MoveJ(p2Joint, p2Desc, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetWObjCoordPoint(2);
    robot.MoveJ(p3Joint, p3Desc, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetWObjCoordPoint(3);
    rtn = robot.ComputeWObjCoord(1, 0, coordRtn);

    robot.SetWObjCoord(1, coordRtn, 0);
    robot.SetWObjList(1, coordRtn, 0);

    DescPose getWobjDesc = new DescPose();
    rtn = robot.GetWObjOffset(0, getWobjDesc);
    return 0;
}
```

2.3.6.22 Set global speed

```
/**
* @brief Set global speed
* @param [in] vel Speed percentage, range [0~100]
* @return Error code
*/
int SetSpeed(int vel);
```

2.3.6.23 Set robot acceleration

```
/**
* @brief Set robot acceleration
* @param [in] acc Robot acceleration percentage
* @return Error code
*/
int SetOaccScale(double acc);
```

2.3.6.24 Get robot default speed

```
/**
* @brief Get robot default speed
* @return List[0]:int Error code; List[1]: double vel Speed, unit mm/s
*/
List<Number> GetDefaultTransVel();
```

2.3.6.25 Set end load weight

```
/**
* @brief Set end load weight
* @param [in] loadNum Load number
* @param [in] weight Load weight, unit kg
* @return Error code
*/
int SetLoadWeight(int loadNum,double weight);
```

2.3.6.26 Set end load center of mass coordinates

```
/**
* @brief Set end load center of mass coordinates
* @param [in] coord Center of mass coordinates, unit mm
* @return Error code
*/
int SetLoadCoord(DescTran coord);
```

2.3.6.27 Set End Effector Load Center of Mass Coordinates

```
/**
 * @brief  Set end effector load center of mass coordinates
 * @param  [in] loadNum Load number
 * @param  [in] coord Center of mass coordinates, unit: mm
 * @return  Error code
 */
public int SetLoadCoord(int loadNum, DescTran coord)
```

2.3.6.28 Get current load weight

```
/**
* @brief Get current load weight
* @param [in] flag 0-blocking, 1-non-blocking
* @return List[0]:int Error code; List[1]: double weight Load weight, unit kg
*/
List<Number> GetTargetPayload(int flag);
```

2.3.6.29 Get current load center of mass

```
/**
* @brief Get current load center of mass
* @param [in] flag 0-blocking, 1-non-blocking
* @param [out] cog Load center of mass, unit mm
* @return Error code
*/
int GetTargetPayloadCog(int flag, DescTran cog);
```

2.3.6.30 Set robot installation method

```
/**
* @brief Set robot installation method
* @param [in] install Installation method, 0-standard, 1-side, 2-inverted
* @return Error code
*/
int SetRobotInstallPos(int install);
```

2.3.6.31 Set robot installation angle

```
/**
* @brief Set robot installation angle, free installation
* @param [in] yangle Tilt angle
* @param [in] zangle Rotation angle
* @return Error code
*/
int SetRobotInstallAngle(double yangle, double zangle);
```

2.3.6.32 Get robot installation angle

```
/**
* @brief Get robot installation angle
* @return List[0]:Error code; List[1]:double yangle Tilt angle; List[2]:double zangle␣
↪Rotation angle
*/
public List<Number> GetRobotInstallAngle()
```

2.3.6.33 Set system variable value

```
/**
* @brief Set system variable value
* @param [in] id Variable number, range [1~20]
* @param [in] value Variable value
* @return Error code
*/
int SetSysVarValue(int id, double value);
```

2.3.6.34 Get system variable value

```
/**
* @brief Get system variable value
* @param [in] id System variable number, range [1~20]
* @return List[0]:Error code; List[1]:double value System variable value
*/
List<Number> GetSysVarValue(int id);
```

2.3.6.35 Robot common settings code example

```
public static int TestLoadInstall(Robot robot)
{
    for (int i = 1; i < 100; i++)
    {
        robot.SetSpeed(i);
        robot.SetOaccScale(i);
        robot.Sleep(30);
    }

    List<Number> defaultVel=new ArrayList<>();

    defaultVel=robot.GetDefaultTransVel();
    System.out.println("GetDefaultTransVel is:"+ defaultVel.get(1));

    for (int i = 1; i < 21; i++)
    {
        robot.SetSysVarValue(i, i + 0.5);
        robot.Sleep(100);
```

(continues on next page)

(continued from previous page)

```
    }

    for (int i = 1; i < 21; i++)
    {
        float value = 0;
        defaultVel=robot.GetSysVarValue(i);
        System.out.println("sys value :"+i+", is :"+defaultVel.get(1));
        robot.Sleep(100);
    }

    robot.SetLoadWeight(0, 2.5);

    DescTran loadCoord = new DescTran();
    loadCoord.x = 3.0;
    loadCoord.y = 4.0;
    loadCoord.z = 5.0;
    robot.SetLoadCoord(loadCoord);

    robot.Sleep(1000);

    List<Number> getLoad = new ArrayList<>();

    getLoad=robot.GetTargetPayload(0);

    DescTran getLoadTran =new DescTran();
    robot.GetTargetPayloadCog(0, getLoadTran);
    System.out.println("get load is:"+getLoad.get(1)+", get load cog is: "+getLoadTran.x+
↪","+getLoadTran.y+","+getLoadTran.z);

    robot.SetRobotInstallPos(0);
    robot.SetRobotInstallAngle(15.0, 25.0);

    List<Number> angle=new ArrayList<>();
    angle=robot.GetRobotInstallAngle();
    System.out.println("GetRobotInstallAngle x:"+angle.get(1)+";  y:"+angle.get(2));

    robot.CloseRPC();
    return 0;
}
```

2.3.6.36 Joint friction compensation switch

```
/**
* @brief Joint friction compensation switch
* @param [in] state 0-off, 1-on
* @return Error code
*/
int FrictionCompensationOnOff(int state);
```

2.3.6.37 Set joint friction compensation coefficient - standard installation

```
/**
* @brief Set joint friction compensation coefficient - standard installation
* @param [in] coeff Six joint compensation coefficients, range [0~1]
* @return Error code
*/
int SetFrictionValue_level(Object[] coeff);
```

2.3.6.38 Set joint friction compensation coefficient - side installation

```
/**
* @brief Set joint friction compensation coefficient - side installation
* @param [in] coeff Six joint compensation coefficients, range [0~1]
* @return Error code
*/
int SetFrictionValue_wall(Object[] coeff);
```

2.3.6.39 Set joint friction compensation coefficient - inverted installation

```
/**
* @brief Set joint friction compensation coefficient - inverted installation
* @param [in] coeff Six joint compensation coefficients, range [0~1]
* @return Error code
*/
int SetFrictionValue_ceiling(Object[] coeff);
```

2.3.6.40 Set joint friction compensation coefficient - free installation

```
/**
* @brief Set joint friction compensation coefficient - free installation
* @param [in] coeff Six joint compensation coefficients, range [0~1]
* @return Error code
*/
int SetFrictionValue_freedom(Object[] coeff);
```

2.3.6.41 Robot set joint friction compensation code example

```
public static int TestFriction(Robot robot)
{

    Object[] lcoeff = { 0.9,0.9,0.9,0.9,0.9,0.9 };
    Object[] wcoeff = { 0.4,0.4,0.4,0.4,0.4,0.4 };
    Object[] ccoeff = { 0.6,0.6,0.6,0.6,0.6,0.6 };
    Object[] fcoeff = { 0.5,0.5,0.5,0.5,0.5,0.5 };

    int rtn = robot.FrictionCompensationOnOff(1);
```

(continues on next page)

(continued from previous page)

```
    System.out.println("FrictionCompensationOnOff rtn is:"+ rtn);

    rtn = robot.SetFrictionValue_level(lcoeff);
    System.out.println("SetFrictionValue_level rtn is:"+ rtn);

    rtn = robot.SetFrictionValue_wall(wcoeff);
    System.out.println("SetFrictionValue_wall rtn is:"+ rtn);

    rtn = robot.SetFrictionValue_ceiling(ccoeff);
    System.out.println("SetFrictionValue_ceiling rtn is:"+ rtn);

    rtn = robot.SetFrictionValue_freedom(fcoeff);
    System.out.println("SetFrictionValue_freedom rtn is:"+ rtn);

    robot.CloseRPC();
    return 0;
}
```

2.3.6.42 Query robot error code

```
/**
 * @brief Query robot error code
 * @param [out] maincode Main error code
 * @param [out] subcode Sub error code
 * @return Error code
 */
int GetRobotErrorCode(int[] maincode, int[] subcode);
```

2.3.6.43 Error state clear

```
/**
* @brief Error state clear
* @return Error code
*/
int ResetAllError();
```

2.3.6.44 Robot fault state get and clear error code example

```
public static int TestGetError(Robot robot)
{
    int[] maincode={0}, subcode={0};
    robot.GetRobotErrorCode(maincode, subcode);

    robot.ResetAllError();

    robot.Sleep(1000);

```

(continues on next page)

(continued from previous page)

```
    robot.GetRobotErrorCode(maincode, subcode);
    return 0;
}
```

2.3.6.45 Set wide voltage control box temperature and fan speed monitoring parameters

New in version Java: SDK-v1.0.6-3.8.3

```
/**
* @brief Set wide voltage control box temperature and fan speed monitoring parameters
* @param [in] enable 0-disable monitoring; 1-enable monitoring
* @param [in] period Monitoring period(s), range 1-100
* @return Error code
*/
int SetWideBoxTempFanMonitorParam(int enable, int period);
```

2.3.6.46 Get wide voltage control box temperature and fan speed monitoring parameters

New in version Java: SDK-v1.0.6-3.8.3

```
/**
* @brief Get wide voltage control box temperature and fan speed monitoring parameters
* @return List[0]-Error code,List[1]-enable 0-disable monitoring; 1-enable monitoring,
↪List[2]-period Monitoring period(s), range 1-100
*/
List<Number> GetWideBoxTempFanMonitorParam()
```

2.3.6.47 Wide voltage control box temperature and fan current state get code example

```
public static void TestWideVoltageCtrlBoxtemp(Robot robot)
{
    robot.SetWideBoxTempFanMonitorParam(1, 2);
    List<Number> list=robot.GetWideBoxTempFanMonitorParam();
    System.out.println("GetWideBoxTempFanMonitorParam enable is:"+list.get(1)+", period␣
↪is:"+list.get(2));
    ROBOT_STATE_PKG pkg=new ROBOT_STATE_PKG();
    for (int i = 0; i < 100; i++)
    {
        pkg=robot.GetRobotRealTimeState();
        System.out.println("robot ctrl box temp is:"+pkg.wideVoltageCtrlBoxTemp+",fan␣
↪current is:"+pkg.wideVoltageCtrlBoxFanCurrent);
        robot.Sleep(100);
    }

    int rtn = robot.SetWideBoxTempFanMonitorParam(0, 2);

    list=robot.GetWideBoxTempFanMonitorParam();
    for (int i = 0; i < 100; i++)
```

(continues on next page)

(continued from previous page)

```
    {
        pkg=robot.GetRobotRealTimeState();
        System.out.println("robot ctrl box temp is:"+pkg.wideVoltageCtrlBoxTemp+" ,fan current is:"+pkg.wideVoltageCtrlBoxFanCurrent);
        robot.Sleep(100);
    }

    robot.CloseRPC();
    robot.Sleep(2000);
    return 0;
}
```

2.3.6.48 Set Focus Calibration Point

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Set focus calibration point
* @param [in] pointNum Focus calibration point number (1-8)
* @param [in] point    Calibration point coordinates
* @return Error code
*/
int SetFocusCalibPoint(int pointNum, DescPose point)
```

2.3.6.49 Compute Focus Calibration Result

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Compute focus calibration result
* @param [in] pointNum  Number of calibration points
* @param [in] resultPos Calibration result XYZ
* @param [out] accuracy  Calibration accuracy error
* @return Error code
*/
int ComputeFocusCalib(int pointNum, DescTran resultPos, double[] accuracy)
```

2.3.6.50 Start focus following

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Start focus following
* @param [in] kp       Proportional parameter (default: 50.0)
* @param [in] kpredict Feedforward parameter (default: 19.0)
* @param [in] aMax     Maximum angular acceleration limit (default: 1440°/s²)
* @param [in] vMax     Maximum angular velocity limit (default: 180°/s)
* @param [in] type     Lock X-axis direction (0-Reference input vector; 1-Horizontal; 2-Vertical)
```

(continues on next page)

(continued from previous page)

```
* @return Error code
*/
int FocusStart(double kp, double kpredict, double aMax, double vMax, int type)
```

2.3.6.51 Stop Focus Following

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Stop focus following
* @return Error code
*/
int FocusEnd()
```

2.3.6.52 Set Focus Position

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Set focus position
* @param pos Focus position XYZ
* @return Error code
*/
public int SetFocusPosition(DescTran pos)
```

2.3.6.53 Focus Following Code Example

```
public static void TestFocus(Robot robot){
    DescPose p1Desc=new DescPose(186.331, 487.913, 209.850, 149.030, 0.688, -114.347);
    JointPos p1Joint = new JointPos(-127.876, -75.341, 115.417, -122.741, -59.820, 74.
↪300);

    DescPose p2Desc = new DescPose(69.721, 535.073, 202.882, -144.406, -14.775, -89.012);
    JointPos p2Joint = new JointPos(-101.780, -69.828, 110.917, -125.740, -127.841, 74.
↪300);

    DescPose p3Desc = new DescPose(146.861, 578.426, 205.598, 175.997, -36.178, -93.437);
    JointPos p3Joint = new JointPos(-112.851, -60.191, 86.566, -80.676, -97.463, 74.300);

    DescPose p4Desc = new DescPose(136.284, 509.876, 225.613, 178.987, 1.372, -100.696);
    JointPos p4Joint = new JointPos(-116.397, -76.281, 113.845, -128.611, -88.654, 74.
↪299);

    DescPose p5Desc = new DescPose(138.395, 505.972, 298.016, 179.134, 2.147, -101.110);
    JointPos p5Joint = new JointPos(-116.814, -82.333, 109.162, -118.662, -88.585, 74.
↪302);

    DescPose p6Desc = new DescPose(105.553, 454.325, 232.017, -179.426, 0.444, -99.952);
```

(continues on next page)

(continued from previous page)

```
    JointPos p6Joint = new JointPos(-115.649, -84.367, 122.447, -128.663, -90.432, 74.303);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 100, 0, 0, 0);

    robot.GetForwardKin(p1Joint, p1Desc);
    robot.GetForwardKin(p2Joint,  p2Desc);
    robot.GetForwardKin(p3Joint,  p3Desc);
    robot.GetForwardKin(p4Joint,  p4Desc);
    robot.GetForwardKin(p5Joint,  p5Desc);
    robot.GetForwardKin(p6Joint,  p6Desc);

    robot.MoveJ(p1Joint, p1Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(1);
    robot.MoveJ(p2Joint, p2Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(2);
    robot.MoveJ(p3Joint, p3Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(3);
    robot.MoveJ(p4Joint, p4Desc, 0, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.SetTcp4RefPoint(4);

    DescPose coordRtn = new DescPose(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
    int rtn = robot.ComputeTcp4( coordRtn);

    robot.SetToolCoord(1, coordRtn, 0, 0, 1, 0);

    robot.GetForwardKin(p1Joint, p1Desc);
    robot.GetForwardKin(p2Joint, p2Desc);
    robot.GetForwardKin(p3Joint, p3Desc);

    robot.SetFocusCalibPoint(1, p1Desc);
    robot.SetFocusCalibPoint(2, p2Desc);
    robot.SetFocusCalibPoint(3, p3Desc);

    DescTran resultPos = new DescTran(0.0, 0.0, 0.0);
    double[] accuracy = {0.0};
    rtn = robot.ComputeFocusCalib(3,  resultPos,  accuracy);
    rtn = robot.SetFocusPosition(resultPos);

    robot.GetForwardKin(p5Joint,  p5Desc);
    robot.GetForwardKin(p6Joint,  p6Desc);

    robot.MoveL(p5Joint, p5Desc, 1, 0, 10, 100, 100, -1, 0, exaxisPos, 0, 1, offdese,0,100);
    robot.MoveL(p6Joint, p6Desc, 1, 0, 10, 100, 100, -1, 0, exaxisPos, 0, 1, offdese,0,100);

    robot.FocusStart(50, 19, 710, 90, 0);
    robot.MoveL(p5Joint, p5Desc, 1, 0, 10, 100, 100, -1, 0, exaxisPos, 0, 1, offdese,0,100);
    robot.MoveL(p6Joint, p6Desc, 1, 0, 10, 100, 100, -1, 0, exaxisPos, 0, 1, offdese,0,
```

(continues on next page)

(continued from previous page)

```
↪100);
    robot.FocusEnd();
}
```

2.3.6.54 Joint Torque Sensor Sensitivity Calibration Function Enabled

```
/**
* @brief Joint torque sensor sensitivity calibration function enabled
* @param status 0-Disabled; 1-Enabled
* @return Error code
*/
public int JointSensitivityEnable(int status)
```

2.3.6.55 Joint Torque Sensor Sensitivity Data Acquisition

```
/**
* @brief Joint Torque Sensor Sensitivity Data Collection
* @return Error code
*/
public int JointSensitivityCollect()
```

2.3.6.56 Get Joint Torque Sensor Sensitivity Calibration Results

```
/**
* @brief Get joint torque sensor sensitivity calibration results
* @param calibResult j1~j6 joint sensitivity [0-1]
* @param linearity j1~j6 joint linearity [0-1]
* @return Error code
*/
public int JointSensitivityCalibration(double[] calibResult, double[] linearity)
```

2.3.6.57 Get Joint Torque Sensor Hysteresis Error

```
/**
* @brief Get joint torque sensor hysteresis error
* @param hysteresisError j1~j6 joint hysteresis error
* @return Error code
*/
public int JointHysteresisError(double[] hysteresisError);
```

2.3.6.58 Get Joint Torque Sensor Repeatability

```
/**
* @brief Get joint torque sensor repeatability
* @param repeatability j1~j6 joint torque sensor repeatability
* @return Error code
*/
public int JointRepeatability(double[] repeatability);
```

2.3.6.59 Set Joint Force Sensor Parameters

```
/**
* @brief Set joint force sensor parameters
* @param required M J1-J6 mass coefficient []
* @param required B J1-J6 damping coefficient []
* @param required K J1-J6 stiffness coefficient []
* @param default threshold Force control threshold, Nm
* @param default sensitivity Sensitivity, Nm/V, []
* @param default setZeroFlag Function enable flag; 0-Off; 1-On; 2-Record zero point at␣
↪position 1; 3-Record zero point at position 2
* @return Error code
*/
public int SetAdmittanceParams(double[] M, double[] B, double[] K, double[] threshold,␣
↪double[] sensitivity, int setZeroFlag);
```

2.3.6.60 Joint Torque Sensor Sensitivity Auto-Calibration Code Example

```
public static void TestSensitivityCalib(Robot robot)
{
    int rtn = robot.JointSensitivityEnable(0);
    rtn = robot.JointSensitivityEnable(1);
    System.out.printf("JointSensitivityEnable rtn is %d\n", rtn);
    JointPos curJPos = new JointPos();
    robot.GetActualJointPosDegree(curJPos);
    ExaxisPos epos = new ExaxisPos(0,0,0,0);
    DescPose offset_pos =new DescPose(0,0,0,0,0,0 );
    JointPos jointPos1 = new JointPos(curJPos.J1, 0, 0, -90, 0.02, curJPos.J6);
    DescPose descPos1 = new DescPose();
    robot.GetForwardKin(jointPos1, descPos1);
    robot.MoveJ(jointPos1, descPos1, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(200);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 1 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos2 =new JointPos( curJPos.J1, -30, 0, -90, 0.02, curJPos.J6 );
    DescPose descPos2 =new DescPose();
    robot.GetForwardKin(jointPos2, descPos2);
    robot.MoveJ(jointPos2, descPos2, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
```

(continues on next page)

(continued from previous page)

```
    System.out.printf("JointSensitivityCollect 2 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos3 = new JointPos( curJPos.J1, -60, 0, -90, 0.02, curJPos.J6 );
    DescPose descPos3 =new DescPose();
    robot.GetForwardKin(jointPos3, descPos3);
    robot.MoveJ(jointPos3, descPos3, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 3 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos4 = new JointPos(curJPos.J1, -90, 0, -90, 0.02, curJPos.J6);
    DescPose descPos4 = new DescPose();
    robot.GetForwardKin(jointPos4, descPos4);
    robot.MoveJ(jointPos4, descPos4, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 4 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos5 = new JointPos(curJPos.J1, -120, 0, -90, 0.02, curJPos.J6);
    DescPose descPos5 = new DescPose();
    robot.GetForwardKin(jointPos5, descPos5);
    robot.MoveJ(jointPos5, descPos5, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 5 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos6 = new JointPos(curJPos.J1, -150, 0, -90, 0.02, curJPos.J6);
    DescPose descPos6 = new DescPose();
    robot.GetForwardKin(jointPos6, descPos6);
    robot.MoveJ(jointPos6, descPos6, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 6 rtn is %d\n", rtn);
    robot.Sleep(100);
    JointPos jointPos7 = new JointPos(curJPos.J1, -180, 0, -90, 0.02, curJPos.J6);
    DescPose descPos7 = new DescPose();
    robot.GetForwardKin(jointPos7, descPos7);
    robot.MoveJ(jointPos7, descPos7, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 7 rtn is %d\n", rtn);
    robot.Sleep(100);
    // Reverse stroke
    robot.MoveJ(jointPos6, descPos6, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 8 rtn is %d\n", rtn);
    robot.Sleep(100);
    robot.MoveJ(jointPos5, descPos5, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 9 rtn is %d\n", rtn);
```

(continues on next page)

(continued from previous page)

```
    robot.Sleep(100);
    robot.MoveJ(jointPos4, descPos4, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 10 rtn is %d\n", rtn);
    robot.Sleep(100);
    robot.MoveJ(jointPos3, descPos3, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 11 rtn is %d\n", rtn);
    robot.Sleep(100);
    robot.MoveJ(jointPos2, descPos2, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(100);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 12 rtn is %d\n", rtn);
    robot.Sleep(100);
    robot.MoveJ(jointPos1, descPos1, 0, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.Sleep(200);
    rtn = robot.JointSensitivityCollect();
    System.out.printf("JointSensitivityCollect 13 rtn is %d\n", rtn);
    robot.Sleep(100);
    double[] calibResult =new double[6];
    double[] linearity = new double[6];
    rtn = robot.JointSensitivityCalibration(calibResult, linearity);
    System.out.printf("JointSensitivityCalibration rtn is %d\n", rtn);
    rtn = robot.JointSensitivityEnable(0);
    System.out.printf("JointSensitivityEnable rtn is %d\n", rtn);
    System.out.printf("jointSensor Calib result is %f %f %f %f %f %f\njointSensor␣
↪linearity is %f %f %f %f %f %f\n",
            calibResult[0], calibResult[1], calibResult[2],
            calibResult[3], calibResult[4], calibResult[5],
            linearity[0], linearity[1], linearity[2],
            linearity[3], linearity[4], linearity[5]);
    double[] hysteresisError = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    rtn = robot.JointHysteresisError(hysteresisError);
    System.out.printf("JointHysteresisError result is %f %f %f %f %f %f\n",
            hysteresisError[0], hysteresisError[1], hysteresisError[2],
            hysteresisError[3], hysteresisError[4], hysteresisError[5]);
    double[] repeatability = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    rtn = robot.JointRepeatability(repeatability);
    System.out.printf("JointRepeatability result is %f %f %f %f %f %f\n",
            repeatability[0], repeatability[1], repeatability[2],
            repeatability[3], repeatability[4], repeatability[5]);
    double[] M = {1.0, 1.0, 1.0, 1.0, 1.0, 1.0};
    double[] B = {1.0, 1.0, 1.0, 1.0, 1.0, 1.0};
    double[] K = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    double[] threshold = {1.0, 1.0, 1.0, 1.0, 1.0, 1.0};
    int setZeroFlag = 1;
    rtn = robot.SetAdmittanceParams(M, B, K, threshold, calibResult, setZeroFlag);
    System.out.printf("SetAdmittanceParams rtn is %d\n", rtn);
}
```

2.3.6.61 Retrieve the number of error frames for the robot's 8 slave ports

```
/**
* @brief Retrieve error frame counts for the robot's 8 slave ports
* @param  inRecvErr Input received error frame count
* @param  inCRCErr Input CRC error frame count
* @param  inTransmitErr Input transmitted error frame count
* @param  inLinkErr Input link error frame count
* @param  outRecvErr Output received error frame count
* @param  outCRCErr Output CRC error frame count
* @param  outTransmitErr Output transmitted error frame count
* @param  outLinkErr Output link error frame count
* @return Error code
*/
public int GetSlavePortErrCounter(int[] inRecvErr, int[] inCRCErr, int[] inTransmitErr,␣
↪int[] inLinkErr, int[] outRecvErr, int[] outCRCErr, int[] outTransmitErr, int[]␣
↪outLinkErr)
```

2.3.6.62 Reset slave port error frames

```
/**
* @brief Clears error frames for slave port
* @param slaveID Slave ID (0~7)
* @return Error code
*/
public int SlavePortErrCounterClear(int slaveID)
```

2.3.6.63 Example for retrieving slave port error frame codes

```
public static void TestSlavePortErr(Robot robot)
{
    ROBOT_STATE_PKG pkg =new ROBOT_STATE_PKG();
    int[] inRecvErr =new int[8];
    int[] inCRCErr =new int[8];
    int[] inTransmitErr =new int[8];
    int[] inLinkErr =new int[8];
    int[] outRecvErr =new int[8];
    int[] outCRCErr =new int[8];
    int[] outTransmitErr =new int[8];
    int[] outLinkErr =new int[8];
    robot.GetSlavePortErrCounter(inRecvErr,  inCRCErr, inTransmitErr,␣
↪inLinkErr,outRecvErr, outCRCErr, outTransmitErr, outLinkErr);
    for (int i = 0; i < 8; i++)
    {
        if (inRecvErr[i] != 0)
        {
            System.out.printf("inRecvErr %d is %d\n", i, inRecvErr[i]);
        }
        if (inCRCErr[i] != 0)
        {
```

(continues on next page)

(continued from previous page)

```
            System.out.printf("inRecvErr %d is %d\n", i, inCRCErr[i]);
        }
        if (inTransmitErr[i] != 0)
        {
            System.out.printf("inRecvErr %d is %d\n", i, inTransmitErr[i]);
        }
        if (inLinkErr[i] != 0)
        {
            System.out.printf("inRecvErr %d is %d\n", i, inLinkErr[i]);
        }
        if (outRecvErr[i] != 0)
        {
            System.out.printf("outRecvErr %d is %d\n", i, outRecvErr[i]);
        }
        if (outCRCErr[i] != 0)
        {
            System.out.printf("outCRCErr %d is %d\n", i, outCRCErr[i]);
        }
        if (outTransmitErr[i] != 0)
        {
            System.out.printf("outTransmitErr %d is %d\n", i,␣
↪outTransmitErr[i]);
        }
        if (outLinkErr[i] != 0)
        {
            System.out.printf("outLinkErr %d is %d\n", i, outLinkErr[i]);
        }
    }
    System.out.printf("others has no err!\n");
    for (int i = 0; i < 8; i++)
    {
        robot.SlavePortErrCounterClear(i);
    }
    robot.CloseRPC();
}
```

2.3.6.64 Set speed feedforward coefficients for each axis

```
/**
* @brief Sets the feedforward ratio for each axis
* @param  radio The feedforward ratio for each axis
* @return Error code
*/
public int SetVelFeedForwardRatio(double[] radio)
```

2.3.6.65 Retrieve speed feedforward ratios for each axis

```
/**
* @brief Retrieve speed feedforward ratios for each axis
* @param  radio Speed feedforward coefficients for each axis
* @return Error code
*/
public int GetVelFeedForwardRatio(double[] radio)
```

2.3.6.66 Robot Speed Feedforward Coefficient Code Example

```
public static void TestVelFeedForwardRatio(Robot robot)
{
    double[] setRadio =new double[] { 1.0, 1.0, 1.0, 1.0, 1.0, 1.0 };
    robot.SetVelFeedForwardRatio(setRadio);
    double[] getRadio = new double[]{ 0.0 };
    robot.GetVelFeedForwardRatio(getRadio);
    System.out.printf(" %f %f %f %f %f %f\n", getRadio[0], getRadio[1],␣
↪getRadio[2], getRadio[3], getRadio[4], getRadio[5]);
    robot.CloseRPC();
}
```

2.3.6.67 Photoelectric Sensor TCP Calibration - Compute Tool RPY

```
/**
* @brief Photoelectric sensor TCP calibration - compute tool RPY
* @param  Btool Robot Cartesian position
* @param  Etool Current tool coordinate coefficient value
* @param  sensor Current sensor coordinate coefficient value (not yet available)
* @param  radius Circular motion radius mm (not yet available)
* @param  dz Movement distance along the negative direction of the base coordinate␣
↪system Z-axis; when dz = 10000, the function directly returns tool RPY
* @param  TCPRPY Tool RPY value
* @return Error code
*/
public int TCPComputeRPY(DescPose Btool, DescPose Etool, DescPose sensor, double radius,␣
↪double dz, Rpy TCPRPY)
```

2.3.6.68 Photoelectric Sensor TCP Calibration - Compute Tool XYZ

```
/**
* @brief Photoelectric sensor TCP calibration - compute tool XYZ
* @param  select 0-Compute tool TCP; 1-Compute sensor origin; 2-Compute sensor␣
↪orientation; 3-Directly return tool TCP; 4-Record current workpiece coordinate system␣
↪and tool coordinate system
* @param  originDirection 0-X direction; 1-Y direction; 2-Z direction
* @param  pos1 Robot Cartesian position 1
* @param  pos2 Robot Cartesian position 2
```

(continues on next page)

(continued from previous page)

```
* @param  pos3 Robot Cartesian position 3
* @param  pos4 Robot Cartesian position 4
* @param  TCP Tool XYZ value
* @return Error code
*/
public int TCPComputeXYZ(int select, double originDirection, DescTran pos1, DescTran␣
↪pos2, DescTran pos3, DescTran pos4, DescTran TCP)
```

2.3.6.69 Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position

```
/**
* @brief Photoelectric sensor TCP calibration - start recording flange center position
* @return Error code
*/
public int TCPRecordFlangePosStart()
```

2.3.6.70 Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position

```
/**
* @brief Photoelectric sensor TCP calibration - stop recording flange center position
* @return Error code
*/
public int TCPRecordFlangePosEnd()
```

2.3.6.71 Photoelectric Sensor TCP Calibration - Get Tool Center Point Position

```
/**
* @brief Photoelectric sensor TCP calibration - get tool center point position
* @param  TCP Tool center point position (x,y,z)
* @return Error code
*/
public int TCPGetRecordFlangePos(DescTran TCP)
```

2.3.6.72 Photoelectric Sensor TCP Calibration

```
/**
* @brief Photoelectric sensor TCP calibration
* @param luaPath Automatic calibration Lua program path: QX version robot - "/fruser/FR_
↪CalibrateTheToolTcp.lua"; LA version robot - "/usr/local/etc/controller/lua/FR_
↪CalibrateTheToolTcp.lua"
* @param offset Teach point offset (x,y,z) mm
* @param TCP Calibrated tool coordinate system (x,y,z,rx,ry,rz)
* @return Error code
*/
public int PhotoelectricSensorTCPCalibration(String luaPath, DescTran offset, DescPose␣
↪TCP)
```

2.3.6.73 Photoelectric Sensor TCP Calibration Code Example

```
public static void TestPhotoelectricSensorTCPCalib(Robot robot)
{
    DescTran offset = new DescTran(10.0, 10.0, 3.0 );
    DescPose TCP = new DescPose();
    int rtn = robot.PhotoelectricSensorTCPCalibration("/fruser/FR_CalibrateTheToolTcp.lua
↪", offset, TCP);
    System.out.printf("PhotoelectricSensorTCPCalibration rtn is %d %f %f %f %f %f %f \n",
↪ rtn, TCP.tran.x, TCP.tran.y, TCP.tran.z, TCP.rpy.rx, TCP.rpy.ry, TCP.rpy.rz);
    robot.CloseRPC();
    robot.Sleep(9999999);
}
```

2.3.7 Robot safety settings

2.3.7.1 Set collision level

```
/**
* @brief Set collision level
* @param  [in]  mode  0-level, 1-percentage
* @param  [in]  level Collision threshold. Level range: [1-10 corresponds to levels 1-10,
↪ 100-disabled]. Percentage range: [0~10 corresponds to 0% - 100%]
* @param  [in]  config 0-Do not update config file, 1-Update config file
* @return  Error code
*/
int SetAnticollision(int mode, Object[] level, int config);
```

2.3.7.2 Set post-collision strategy

```
/**
* @brief  Set post-collision strategy
* @param  [in] strategy  0-Report error and stop, 1-Continue running
* @param  [in] safeTime  Safe stop time [1000 - 2000]ms
* @param  [in] safeDistance  Safe stop distance [1-150]mm
* @param  [in] safetyMargin  J1-J6 safety coefficients [1-10]
* @return  Error code
*/
int SetCollisionStrategy(int strategy, int safeTime, int safeDistance, int␣
↪safetyMargin[]);
```

2.3.7.3 Custom collision detection threshold function start

New in version Java: SDK-v1.0.3-3.8.0

```
/**
* @brief  Start custom collision detection threshold function. Set collision detection thresholds for joints and TCP
* @param  [in] flag 1-Joint detection only; 2-TCP detection only; 3-Both joint and TCP detection
* @param  [in] jointDetectionThreshould  Joint collision detection thresholds for J1-J6
* @param  [in] tcpDetectionThreshould  TCP collision detection thresholds for xyzabc
* @param  [in] block 0-Non-blocking; 1-Blocking
* @return  Error code
*/
public int CustomCollisionDetectionStart(int flag, double[] jointDetectionThreshould, double[] tcpDetectionThreshould, int block);
```

2.3.7.4 Custom collision detection threshold function end

New in version Java: SDK-v1.0.3-3.8.0

```
/**
* @brief  End custom collision detection threshold function
* @return  Error code
*/
public int CustomCollisionDetectionEnd();
```

2.3.7.5 Robot collision level setting code example

```
public static int TestCollision(Robot robot)
{
    int mode = 0;
    int config = 1;
    Object[] level1 = new Object[]{ 1.0,2.0,3.0,4.0,5.0,6.0 };
    Object[] level2 = new Object[]{ 50.0,20.0,30.0,40.0,50.0,60.0 };

    int rtn = robot.SetAnticollision(mode, level1, config);
    System.out.println("SetAnticollision mode 0 rtn is: "+ rtn);
    mode = 1;
    rtn = robot.SetAnticollision(mode, level2, config);
    System.out.println("SetAnticollision mode 1 rtn is :"+ rtn);

    JointPos p1Joint=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos p2Joint=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);

    DescPose p1Desc=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose p2Desc=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);

    ExaxisPos exaxisPos=new ExaxisPos(0.0, 0.0, 0.0, 0.0);
    DescPose offdese=new DescPose(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
    robot.MoveL(p2Joint, p2Desc, 0, 0, 100, 100, 100, 2,0, exaxisPos, 0, 0, offdese,0,
```

(continues on next page)

(continued from previous page)

```
↪10);
    robot.ResetAllError();
    int[] safety = new int[]{ 5,5,5,5,5,5 };
    rtn = robot.SetCollisionStrategy(3, 1000, 150, 250, safety);
    System.out.println("SetCollisionStrategy rtn is:"+ rtn);

    double[] jointDetectionThreshould = new double[]{ 0.1, 0.1, 0.1, 0.1, 0.1, 0.1 };
    double[] tcpDetectionThreshould =new double[] { 60,60,60,60,60,60 };
    rtn = robot.CustomCollisionDetectionStart(3, jointDetectionThreshould,␣
↪tcpDetectionThreshould, 0);
    System.out.println("CustomCollisionDetectionStart rtn is :"+ rtn);

    robot.MoveL(p1Joint, p1Desc, 0, 0, 100, 100, 100, -1,0, exaxisPos, 0, 0, offdese,0,
↪10);
    robot.MoveL(p2Joint, p2Desc, 0, 0, 100, 100, 100, -1,0, exaxisPos, 0, 0, offdese,0,
↪10);
    rtn = robot.CustomCollisionDetectionEnd();
    System.out.println("CustomCollisionDetectionEnd rtn is: "+ rtn);
    return 0;
}
```

2.3.7.6 Set positive limit

```
/**
* @brief  Set positive limit
* @param  [in] limit Six joint positions in deg
* @return  Error code
*/
int SetLimitPositive(Object[] limit);
```

2.3.7.7 Set negative limit

```
/**
* @brief  Set negative limit
* @param  [in] limit Six joint positions in deg
* @return  Error code
*/
int SetLimitNegative(Object[] limit);
```

2.3.7.8 Get joint soft limit angles

```
/**
* @brief  Get joint soft limit angles
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] negative  Negative limit angles in deg
* @param  [out] positive  Positive limit angles in deg
* @return  Error code
*/
int GetJointSoftLimitDeg(int flag, Object[] negative, Object[] positive);
```

2.3.7.9 Robot limit setting code example

```
public static int TestLimit(Robot robot)
{
    Object[] plimit =new Object[] { 170.0,80.0,150.0,80.0,170.0,160.0 };
    robot.SetLimitPositive(plimit);
    Object[] nlimit =new Object[] { -170.0,-260.0,-150.0,-260.0,-170.0,-160.0 };
    robot.SetLimitNegative(nlimit);

    Object[] neg_deg =new Object[] {0, 0 , 0, 0, 0, 0}, pos_deg = new Object[]{0, 0 , 0,␣
↪0, 0, 0};
    robot.GetJointSoftLimitDeg(1,  neg_deg,  pos_deg);
    System.out.println("neg limit deg:"+ neg_deg[0]+","+ neg_deg[1]+","+ neg_deg[2]+","+␣
↪neg_deg[3]+","+ neg_deg[4]+","+ neg_deg[5]);
    System.out.println("pos limit deg:"+pos_deg[0]+","+ pos_deg[1]+","+ pos_deg[2]+","+␣
↪pos_deg[3]+","+ pos_deg[4]+","+pos_deg[5]);
    return 0;
}
```

2.3.7.10 Set robot collision detection method

Changed in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Set robot collision detection method
* @param [in] method Collision detection method: 0-Current mode; 1-Dual encoder; 2-Both␣
↪current and dual encoder
* @param [in] thresholdMode Collision level threshold method: 0-Fixed threshold; 1-
↪Custom collision detection threshold
* @return Error code
*/
int SetCollisionDetectionMethod(int method,int thresholdMode)
```

2.3.7.11 Set static collision detection on/off

```
/**
* @brief Set static collision detection on/off
* @param  [in] status 0-Off; 1-On
* @return  Error code
*/
public int SetStaticCollisionOnOff(int status)
```

2.3.7.12 Robot collision detection method code example

```
public static int TestCollisionMethod(Robot robot)
{
    int rtn = robot.SetCollisionDetectionMethod(0);

    rtn = robot.SetStaticCollisionOnOff(1);
    System.out.println("SetStaticCollisionOnOff On rtn is:"+ rtn);
    robot.Sleep(5000);
    rtn = robot.SetStaticCollisionOnOff(0);
    System.out.println("SetStaticCollisionOnOff Off rtn is:"+ rtn);

    robot.CloseRPC();
    return 0;
}
```

2.3.7.13 Joint torque power detection

```
/**
* @brief Joint torque power detection
* @param  [in] status 0-Off; 1-On
* @param  [in] power Set maximum power (W)
* @return  Error code
*/
public int SetPowerLimit(int status, double power)
```

2.3.7.14 Joint torque power detection code example

```
public static int TestPowerLimit(Robot robot)
{
    robot.DragTeachSwitch(1);
    robot.SetPowerLimit(1, 200);
    List<Number> joint_toq=new ArrayList<>();
    joint_toq=robot.GetJointTorques(1);

    int count = 100;
    robot.ServoJTStart(); //   #servoJT start
    int error = 0;
    while (count > 0)
```

(continues on next page)

(continued from previous page)

```
    {
        count = count - 1;
        robot.Sleep(1);
    }
    error = robot.ServoJTEnd();
    robot.DragTeachSwitch(0);

    robot.CloseRPC();
    return 0;
}
```

2.3.8 Robot status inquiry

2.3.8.1 Get current joint positions (degrees)

```
/**
* @brief  Get current joint positions (degrees)
* @param  [out] jPos Six joint positions in deg
* @return  Error code
*/
int GetActualJointPosDegree(JointPos jPos);
```

2.3.8.2 Get joint feedback speed (deg/s)

```
/**
* @brief Get joint feedback speed (deg/s)
* @param [out] speed Six joint speeds
* @return Error code
*/
int GetActualJointSpeedsDegree(Object[] speed);
```

2.3.8.3 Get joint feedback acceleration

```
/**
* @brief  Get joint feedback acceleration (deg/s^2)
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] acc Six joint accelerations
* @return  Error code
*/
public int GetActualJointAccDegree(int flag, Object[] acc)
```

2.3.8.4 Get TCP command composite speed

```
/**
* @brief  Get TCP command speed
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] tcp_speed Linear speed
* @param  [out] ori_speed Orientation speed
* @return  Error code
*/
public int GetTargetTCPCompositeSpeed(int flag, double tcp_speed, double ori_speed)
```

2.3.8.5 Get TCP feedback composite speed

```
/**
* @brief  Get TCP feedback composite speed
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] tcp_speed Linear speed
* @param  [out] ori_speed Orientation speed
* @return  Error code
*/
public int GetActualTCPCompositeSpeed(int flag, double tcp_speed, double ori_speed)
```

2.3.8.6 Get TCP command speed

```
/**
* @brief  Get TCP command speed
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] speed [x,y,z,rx,ry,rz] speed
* @return  Error code
*/
public int GetTargetTCPSpeed(int flag, Object[] speed)
```

2.3.8.7 Get TCP feedback speed

```
/**
* @brief  Get TCP feedback speed
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] speed [x,y,z,rx,ry,rz] speed
* @return  Error code
*/
public int GetActualTCPSpeed(int flag, Object[] speed)
```

2.3.8.8 Get current tool pose

```
/**
* @brief  Get current tool pose
* @param  [out] desc_pos  Tool pose
* @return  Error code
*/
int GetActualTCPPose(DescPose desc_pos);
```

2.3.8.9 Get current tool coordinate system number

```
/**
* @brief  Get current tool coordinate system number
* @param  [in] flag  0-Blocking, 1-Non-blocking
* @param  [out] id  Tool coordinate system number
* @return  Error code
*/
int GetActualTCPNum(int flag, int[] id)
```

2.3.8.10 Get current workpiece coordinate system number

```
/**
* @brief  Get current workpiece coordinate system number
* @param  [in] flag  0-Blocking, 1-Non-blocking
* @param  [out] id  Workpiece coordinate system number
* @return  Error code
*/
public int GetActualWObjNum(int flag, int[] id)
```

2.3.8.11 Get current end flange pose

```
/**
* @brief  Get current end flange pose
* @param  [in] flag  0-Blocking, 1-Non-blocking
* @param  [out] desc_pos  Flange pose
* @return  Error code
*/
public int GetActualToolFlangePose(int flag, DescPose desc_pos)
```

2.3.8.12 Get current joint torque

```
/**
* @brief Get current joint torque
* @param  [in]  flag 0-Blocking, 1-Non-blocking
* @param  [out]  torques Joint torques
* @return  Error code
*/
int GetJointTorques(int flag, Object[] torques);
```

2.3.8.13 Get system time

```
/**
* @brief  Get system time
* @return  List[0]:int Error code; List[1]:double t_ms in ms
*/
List<Number> GetSystemClock();
```

2.3.8.14 Check if robot motion is completed

```
/**
* @brief  Check if robot motion is completed
* @param   [out] state  0-Not completed, 1-Completed
* @return  Error code
*/
public int GetRobotMotionDone(int[] state)
```

2.3.8.15 Query robot motion queue buffer length

```
/**
* @brief  Query robot motion queue buffer length
* @param   [out] len  Buffer length
* @return  Error code
*/
public int GetMotionQueueLength(int[] len)
```

2.3.8.16 Get robot emergency stop state

```
/**
* @brief Get robot emergency stop state
* @param [out] state Emergency stop state, 0-Normal, 1-Emergency stop
* @return Error code
*/
public int GetRobotEmergencyStopState(int[] state)
```

2.3.8.17 Get SDK-robot communication state

```
/**
* @brief Get SDK-robot communication state
* @return state Communication state, 0-Normal, 1-Abnormal
*/
public int GetSDKComState()
```

2.3.8.18 Get safety stop signal

```
/**
* @brief Get safety stop signal
* @param  [out] si0_state Safety stop signal SI0, 0-Invalid, 1-Valid
* @param  [out] si1_state Safety stop signal SI1, 0-Invalid, 1-Valid
* @return Error code
*/
public int GetSafetyStopState(int[] si0_state, int[] si1_state)
```

2.3.8.19 Get robot joint driver temperature (°C)

```
/**
* @brief Get robot joint driver temperature (°C)
* @param  [out] temperature Temperature
* @return Error code
*/
public int GetJointDriverTemperature(double[] temperature)
```

2.3.8.20 Get robot joint driver torque (Nm)

```
/**
* @brief Get robot joint driver torque (Nm)
* @param  [out] torque Torque
* @return Error code
*/
public int GetJointDriverTorque(double[] torque)
```

2.3.8.21 Get robot real-time state structure

```
/**
* @brief Get robot real-time state structure
* @return Real-time state structure
*/
public ROBOT_STATE_PKG GetRobotRealTimeState()
```

2.3.8.22 Robot status inquiry code example

```
public static int TestGetStatus(Robot robot)
{
    List<Number> angle=new ArrayList<>();
    angle=robot.GetRobotInstallAngle();
    System.out.println("yangle:"+angle.get(1)+",zangle:"+angle.get(2));

    JointPos j_deg =new JointPos(){};
    robot.GetActualJointPosDegree( j_deg);

    Object[] jointSpeed =new Object[] { 0,0,0,0,0,0 };
    robot.GetActualJointSpeedsDegree(jointSpeed);

    Object[] jointAcc = new Object[]{0,0,0,0,0,0 };
    robot.GetActualJointAccDegree(0, jointAcc);

    double tcp_speed = 0.0;
    double ori_speed = 0.0;
    robot.GetTargetTCPCompositeSpeed(0, tcp_speed, ori_speed);

    robot.GetActualTCPCompositeSpeed(0, tcp_speed, ori_speed);

    Object[] targetSpeed =new Object[] { 0,0,0,0,0,0 };
    robot.GetTargetTCPSpeed(0, targetSpeed);

    Object[] actualSpeed =new Object[] {0,0,0,0,0,0 };
    robot.GetActualTCPSpeed(0, actualSpeed);

    DescPose tcp = new DescPose(){};
    robot.GetActualTCPPose(tcp);

    DescPose flange = new DescPose(){};
    robot.GetActualToolFlangePose(0, flange);

    int[] id = {};
    robot.GetActualTCPNum(0, id);

    robot.GetActualWObjNum(0, id);

    List<Number> jtorque=new ArrayList<>();
    jtorque=robot.GetJointTorques(0);

    List<Number> t_ms = new ArrayList<>();
    t_ms=robot.GetSystemClock();

    List<Integer> config = new ArrayList<>();
    config=robot.GetRobotCurJointsConfig();

    int motionDone = 0;
    robot.GetRobotMotionDone(motionDone);

    int[] len ={0 };
```

(continues on next page)

(continued from previous page)

```
    robot.GetMotionQueueLength(len);

    int[] emergState = {0};
    robot.GetRobotEmergencyStopState(emergState);

    int comstate = 0;
    comstate=robot.GetSDKComState();

    int[] si0_state=new int[]{0}, si1_state=new int[]{0};
    robot.GetSafetyStopState(si0_state, si1_state);

    double[] temp =new double[] { 0,0,0,0,0,0 };
    robot.GetJointDriverTemperature(temp);

    double[] torque = new double[]{ 0,0,0,0,0,0 };
    robot.GetJointDriverTorque(torque);

    ROBOT_STATE_PKG pkg=new ROBOT_STATE_PKG();
    pkg=robot.GetRobotRealTimeState();

    return 0;
}
```

2.3.8.23 Inverse kinematics calculation

```
/**
* @brief  Inverse kinematics calculation
* @param  [in] type 0-Absolute pose (base frame), 1-Incremental pose (base frame), 2-
↪Incremental pose (tool frame)
* @param  [in] desc_pos Cartesian pose
* @param  [in] config Joint space configuration, [-1]-Solve with reference to current␣
↪joint position, [0~7]-Solve with specific joint space configuration
* @param  [out] joint_pos Joint position
* @return  Error code
*/
int GetInverseKin(int type, DescPose desc_pos, int config, JointPos joint_pos);
```

2.3.8.24 Inverse kinematics calculation (reference position)

```
/**
* @brief  Inverse kinematics calculation with reference joint position
* @param  [in] posMode 0-Absolute pose, 1-Relative pose-base frame, 2-Relative pose-tool␣
↪frame
* @param  [in] desc_pos Cartesian pose
* @param  [in] joint_pos_ref Reference joint position
* @param  [out] joint_pos Joint position
* @return  Error code
*/
```

(continues on next page)

(continued from previous page)

```
int GetInverseKinRef(int posMode, DescPose desc_pos, JointPos joint_pos_ref, JointPos␣
↪joint_pos);
```

2.3.8.25 Inverse Kinematics Solution, Cartesian Space Includes Extended Axis Position

```
/**
* @brief Inverse kinematics solution, Cartesian space includes extended axis position
* @param  type 0-Absolute pose (base coordinate system), 1-Incremental pose (base␣
↪coordinate system), 2-Incremental pose (tool coordinate system)
* @param  desc_pos Cartesian pose
* @param  exaxis Extended axis position
* @param  tool Tool number
* @param  workPiece Workpiece number
* @param  joint_pos Joint position
* @return Error code
*/
public int GetInverseKinExaxis(int type, DescPose desc_pos, ExaxisPos exaxis, int tool,␣
↪int workPiece, JointPos joint_pos)
```

2.3.8.26 Inverse Kinematics Solution Including Extended Axis Position Code Example

```
public static void  TestInverseKinExaxis(Robot robot)
{
    DescPose desc=new DescPose(99.957, -0.002, 29.994, -176.569, -6.757, -167.462);
    ExaxisPos exaxis=new ExaxisPos(100.0, 0.0, 0.0, 0.0);
    JointPos jointPos =new JointPos();
    DescPose offsetPos =new DescPose();
    ROBOT_STATE_PKG pkg=robot.GetRobotRealTimeState();
    int toolnum = pkg.tool;
    int workPcsNum = pkg.user;
    robot.GetInverseKinExaxis(0, desc, exaxis, toolnum, workPcsNum, jointPos);
    System.out.printf("GetInverseKinExaxis joint is %f, %f, %f, %f, %f, %f\n", jointPos.
↪J1, jointPos.J2, jointPos.J3, jointPos.J4, jointPos.J5, jointPos.J6);
    robot.ExtAxisMove(exaxis, 100, -1);
    robot.MoveJ(jointPos, desc, toolnum, workPcsNum, 100.0, 100.0, 100.0, exaxis, -1, 0,␣
↪offsetPos);
    robot.CloseRPC();
    robot.Sleep(9999999);
}
```

2.3.8.27 Check if inverse kinematics has solution

```
/**
* @brief  Check if inverse kinematics has solution with reference joint position
* @param  [in] posMode 0-Absolute pose, 1-Relative pose-base frame, 2-Relative pose-tool frame
* @param  [in] desc_pos Cartesian pose
* @param  [in] joint_pos_ref Reference joint position
* @return  Error code  List[0]:Error code; List[1]: int hasResult 0-No solution, 1-Solution exists
*/
List<Integer> GetInverseKinHasSolution(int posMode, DescPose desc_pos, JointPos joint_pos_ref);
```

2.3.8.28 Forward kinematics calculation

```
/**
* @brief  Forward kinematics calculation
* @param  [in] joint_pos Joint position
* @param  [out] desc_pos Cartesian pose
* @return  Error code
*/
int GetForwardKin(JointPos joint_pos, DescPose desc_pos);
```

2.3.8.29 Robot forward/inverse kinematics calculation code example

```
public static int TestInverseKin(Robot robot)
{
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    DescPose desc_pos1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);

    JointPos inverseRtn = new JointPos(){};

    robot.GetInverseKin(0, desc_pos1, -1, inverseRtn);
    robot.GetInverseKinRef(0, desc_pos1, j1, inverseRtn);

    int hasResut = 0;
    robot.GetInverseKinHasSolution(0, desc_pos1, j1);

    DescPose forwordResult = new DescPose(){};
    robot.GetForwardKin(j1, forwordResult);

    return 0;
}
```

2.3.8.30 Query robot teaching management point data

```
/**
* @brief Query robot teaching management point data
* @param [in]  name  Point name
* @return  List[0]:Error code; List[1] - List[20] : Point data double[20]{x,y,z,rx,ry,rz,
↪j1,j2,j3,j4,j5,j6,tool,wobj,speed,acc,e1,e2,e3,e4}
*/
List<Number> GetRobotTeachingPoint(String name);
```

2.3.8.31 Get robot DH parameter compensation values

```
/**
* @brief Get robot DH parameter compensation values
* @param dhCompensation Robot DH parameter compensation values (mm) [cmpstD1,cmpstA2,
↪cmpstA3,cmpstD4,cmpstD5,cmpstD6]
* @return Error code
*/
public int GetDHCompensation(Object[] dhCompensation)
```

2.3.8.32 Get controller SN code

New in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief Get controller SN code
* @param [out] SNCode Controller SN code
* @return Error code
*/
int GetRobotSN(String[] SNCode);
```

2.3.8.33 Query robot teaching management point data code example

```
public static int TestGetTeachPoint(Robot robot)
{
    String name = "P1";
    List<Number> data=new ArrayList<>();
    data = robot.GetRobotTeachingPoint(name);
    System.out.println(name+" name is: "+data.get(0));
    for (int i = 0; i < 20; i++)
    {
        System.out.println("data is: "+ data.get(i+1));
    }

    int[] que_len = {0};
    int rtn = robot.GetMotionQueueLength(que_len);
    System.out.println("GetMotionQueueLength rtn is:"+rtn+", queue length is:"+ que_
↪len[0]);

```

(continues on next page)

(continued from previous page)

```
    Object[] dh = new Object[]{ 0,0,0,0,0,0 };
    int retval = 0;
    retval = robot.GetDHCompensation(dh);
    System.out.println("retval is: "+retval);

    String[] SN = new String[]{""};
    robot.GetRobotSN(SN);
    System.out.println("robot SN is "+SN[0]);
    return 0;
}
```

2.3.8.34 Get Tool Coordinate System by ID

New in version Java: SDK-v1.0.9-3.8.6

```
/**
* @brief Get Tool Coordinate System by ID
* @param [in] id Tool coordinate system ID
* @param [out] coord Coordinate system value
* @return Error code
*/
int GetToolCoordWithID(int id, DescPose coord)
```

2.3.8.35 Get Work Object Coordinate System by ID

New in version Java: SDK-v1.0.9-3.8.6

```
/**
* @brief Get Work Object Coordinate System by ID
* @param [in]  id Work object coordinate system ID
* @param [out] coord Coordinate system value
* @return Error code
*/
public int GetWObjCoordWithID(int id, DescPose coord)
```

2.3.8.36 Get External Tool Coordinate System by ID

New in version Java: SDK-v1.0.9-3.8.6

```
/**
* @brief Get External Tool Coordinate System by ID
* @param [in]  id External tool coordinate system ID
* @param [out] coord Coordinate system value
* @return Error code
*/
public int GetExToolCoordWithID(int id, DescPose coord)
```

2.3.8.37 Get Extended Axis Coordinate System by ID

New in version Java: SDK-v1.0.9-3.8.6

```
/**
* @brief Get Extended Axis Coordinate System by ID
* @param [in]  id Extended axis coordinate system ID
* @param [out] coord Coordinate system value
* @return Error code
*/
public int GetExAxisCoordWithID(int id, DescPose coord)
```

2.3.8.38 Get Current Tool Coordinate System

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Get Current Tool Coordinate System
 * @param [out] coord Coordinate system value
 * @return Error code
 */
public int GetCurToolCoord(DescPose coord)
```

2.3.8.39 Get Current Work Object Coordinate System

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Get Current Work Object Coordinate System
 * @param [out] coord Coordinate system value
 * @return Error code
 */
public int GetCurWObjCoord(DescPose coord)
```

2.3.8.40 Get Current External Tool Coordinate System

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Get Current External Tool Coordinate System
 * @param  [out] coord Coordinate system value
 * @return Error code
 */
public int GetCurExToolCoord(DescPose coord)
```

2.3.8.41 Get Current Extended Axis Coordinate System

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Get Current Extended Axis Coordinate System
 * @param [out] coord Coordinate system value
 * @return Error code
 */
public int GetCurExAxisCoord(DescPose coord)
```

2.3.8.42 Get Robot Coordinate System and Payload Code Example

```
public static void TestCoord(Robot robot)
{
    int id = 1;
    int rtn = 0;
    DescPose toolCoord = new DescPose();
    DescPose extoolCoord = new DescPose();
    DescPose wobjCoord = new DescPose();
    DescPose exAxisCoord = new DescPose();


    robot.GetCurToolCoord(toolCoord);// Tool
    System.out.println("GetToolCoord:"+id+","+
            toolCoord.tran.x+","+ toolCoord.tran.y+","+ toolCoord.tran.z+","+
            toolCoord.rpy.rx+","+ toolCoord.rpy.ry+","+ toolCoord.rpy.rz);


    robot.GetCurWObjCoord(toolCoord);// Work object
    System.out.println("GetCurWObjCoord:"+id+","+
            toolCoord.tran.x+","+ toolCoord.tran.y+","+ toolCoord.tran.z+","+
            toolCoord.rpy.rx+","+ toolCoord.rpy.ry+","+ toolCoord.rpy.rz);

    robot.GetCurExToolCoord(toolCoord);// External tool
    System.out.println("GetCurExToolCoord:"+id+","+
            toolCoord.tran.x+","+ toolCoord.tran.y+","+ toolCoord.tran.z+","+
            toolCoord.rpy.rx+","+ toolCoord.rpy.ry+","+ toolCoord.rpy.rz);


    robot.GetCurExAxisCoord(toolCoord);// Extended axis
    System.out.println("GetCurExAxisCoord:"+id+","+ // Corrected print string for␣
↪consistency
            toolCoord.tran.x+","+ toolCoord.tran.y+","+ toolCoord.tran.z+","+
            toolCoord.rpy.rx+","+ toolCoord.rpy.ry+","+ toolCoord.rpy.rz);


    List<Number> weightT = new ArrayList<>();// Center of gravity
    DescTran cogT=new DescTran();
    weightT=robot.GetTargetPayload(0);
    robot.GetTargetPayloadCog(0,cogT);
    System.out.println("GetTargetPayload :"+weightT.get(1).doubleValue()+", "+
```

(continues on next page)

(continued from previous page)

```
            cogT.x+", "+cogT.y+", "+cogT.z);


    robot.GetToolCoordWithID(id, toolCoord);
    System.out.println("GetToolCoordWithID:"+id+","+
            toolCoord.tran.x+","+ toolCoord.tran.y+","+ toolCoord.tran.z+","+
            toolCoord.rpy.rx+","+ toolCoord.rpy.ry+","+ toolCoord.rpy.rz);

    robot.GetWObjCoordWithID(id, wobjCoord);
    System.out.println("GetWObjCoordWithID "+id+", "+
            wobjCoord.tran.x+","+ wobjCoord.tran.y+","+ wobjCoord.tran.z+","+
            wobjCoord.rpy.rx+","+ wobjCoord.rpy.ry+","+ wobjCoord.rpy.rz);


    robot.GetExToolCoordWithID(id, extoolCoord);// External tool
    System.out.println("GetExToolCoordWithID :"+ id+","+
            extoolCoord.tran.x+","+ extoolCoord.tran.y+","+ extoolCoord.tran.z+","+
            extoolCoord.rpy.rx+","+ extoolCoord.rpy.ry+","+ extoolCoord.rpy.rz);

    robot.GetExAxisCoordWithID(id, exAxisCoord);// Extended axis
    System.out.println("GetExAxisCoordWithID "+id+","+
            exAxisCoord.tran.x+","+ exAxisCoord.tran.y+","+ exAxisCoord.tran.z+","+
            exAxisCoord.rpy.rx+","+ exAxisCoord.rpy.ry+","+ exAxisCoord.rpy.rz);


    double[] weight = new double[1];// Payload center of gravity
    DescTran getCog = new DescTran();
    robot.GetTargetPayloadWithID(id, weight, getCog);
    System.out.println("GetTargetPayloadWithID :"+ id+","+ weight[0]+","+
            getCog.x+","+ getCog.y+","+ getCog.z);

    DescPose coordSet0 = new DescPose(0, 0, 0, 0, 0, 0);
    DescPose coordSet = new DescPose(1, 2, 3, 4, 5, 6);
    DescPose etcp = new DescPose(10, 20, 30, 40, 50, 60);
    DescPose etool = new DescPose(0.1, 0.2, 0.3, 0.4, 0.5, 0.6);
    DescTran cog = new DescTran(1, 2, 3);

    robot.SetToolCoord(id, coordSet, 0, 0, 1, 0);
    robot.Sleep(100);
    robot.SetWObjCoord(id, coordSet, 0);
    robot.Sleep(100);
    robot.ExtAxisActiveECoordSys(id, 1, coordSet, 1); // Apply calibration result to␣
↪extended axis coordinate system
    robot.Sleep(100);
    rtn = robot.SetExToolCoord(id, etcp, etool);
    robot.Sleep(100);
    rtn = robot.SetLoadWeight(id, 1.5);
    robot.Sleep(500);
    rtn = robot.SetLoadCoord(id, cog);
    robot.Sleep(100);
}
```

2.3.9 Robot trajectory playback

2.3.9.1 Set TPD trajectory recording parameters

```
/**
* @brief  Set TPD trajectory recording parameters
* @param  [in] type  Recording data type, 1-Joint position
* @param  [in] name  Trajectory file name
* @param  [in] period_ms  Data sampling period, fixed value 2ms/4ms/8ms
* @param  [in] di_choose  DI selection, bit0~bit7 correspond to controller DI0~DI7, bit8~bit9 correspond to end DI0~DI1, 0-Not selected, 1-Selected
* @param  [in] do_choose  DO selection, bit0~bit7 correspond to controller DO0~DO7, bit8~bit9 correspond to end DO0~DO1, 0-Not selected, 1-Selected
* @return  Error code
*/
int SetTPDParam(int type, String name, int period_ms, int di_choose, int do_choose);
```

2.3.9.2 Start TPD trajectory recording

```
/**
* @brief  Start TPD trajectory recording
* @param  [in] type  Recording data type, 1-Joint position
* @param  [in] name  Trajectory file name
* @param  [in] period_ms  Data sampling period, fixed value 2ms/4ms/8ms
* @param  [in] di_choose  DI selection, bit0~bit7 correspond to controller DI0~DI7, bit8~bit9 correspond to end DI0~DI1, 0-Not selected, 1-Selected
* @param  [in] do_choose  DO selection, bit0~bit7 correspond to controller DO0~DO7, bit8~bit9 correspond to end DO0~DO1, 0-Not selected, 1-Selected
* @return  Error code
*/
int SetTPDStart(int type, String name, int period_ms, int di_choose, int do_choose);
```

2.3.9.3 Stop TPD trajectory recording

```
/**
* @brief  Stop TPD trajectory recording
* @return  Error code
*/
int SetWebTPDStop();
```

2.3.9.4 Delete TPD trajectory recording

```
/**
* @brief  Delete TPD trajectory recording
* @param  [in] name  Trajectory file name
* @return  Error code
*/
int SetTPDDelete(string name);
```

2.3.9.5 TPD trajectory preloading

```
/**
* @brief  Trajectory preloading
* @param  [in] name  Trajectory file name
* @return  Error code
*/
int LoadTPD(String name);
```

2.3.9.6 TPD trajectory playback

```
/**
* @brief  Trajectory playback
* @param  [in] name  Trajectory file name
* @param  [in] blend 0-No smoothing, 1-Smoothing
* @param  [in] ovl  Speed scaling percentage, range [0~100]
* @return  Error code
*/
int MoveTPD(String name, int blend, double ovl);
```

2.3.9.7 Get TPD starting pose

```
/**
* @brief Get trajectory starting pose
* @param [in] name  Trajectory file name (without extension)
* @param [out] desc_pose  Retrieved trajectory starting pose
* @return Error code
*/
int GetTPDStartPose(String name, DescPose desc_pose);
```

2.3.9.8 Robot TPD trajectory recording code example

```
public static int TestTPD(Robot robot)
{
    int type = 1;
    String name = "tpd2025";
    int period_ms = 4;
    int di_choose = 0;
    int do_choose = 0;

    robot.SetTPDParam(type, name, period_ms, di_choose, do_choose);

    robot.Mode(1);
    robot.Sleep(1000);
    robot.DragTeachSwitch(1);
    robot.SetTPDStart(type, name, period_ms, di_choose, do_choose);
    robot.Sleep(10000);
    robot.SetWebTPDStop();
    robot.DragTeachSwitch(0);

    double ovl = 100.0;
    int blend = 0;

    DescPose start_pose =new DescPose() {};

    int rtn = robot.LoadTPD(name);
    System.out.println("LoadTPD rtn is:"+ rtn);

    robot.GetTPDStartPose(name, start_pose);
    robot.MoveCart(start_pose, 0, 0, 100, 100, ovl, -1, -1);
    robot.Sleep(1000);

    rtn = robot.MoveTPD(name, blend, ovl);
    System.out.println("MoveTPD rtn is: "+ rtn);
    robot.Sleep(5000);

    robot.SetTPDDelete(name);
    return 0;
}
```

2.3.9.9 Trajectory preprocessing

```
/**
* @brief External trajectory file preprocessing
* @param [in] name Trajectory file name
* @param [in] ovl Speed scaling percentage, range [0~100]
* @param [in] opt 1-Control point (default)
* @return Error code
*/
int LoadTrajectoryJ(String name, double ovl, int opt);
```

2.3.9.10 Trajectory playback

```
/**
* @brief External trajectory file playback
* @return Error code
*/
int MoveTrajectoryJ();
```

2.3.9.11 Get trajectory starting pose

```
/**
* @brief Get trajectory starting pose
* @param [in] name Trajectory file name
* @param [out] desc_pose Retrieved trajectory starting pose
* @return Error code
*/
int GetTrajectoryStartPose(String name, DescPose desc_pose);
```

2.3.9.12 Get trajectory point number

```
/**
* @brief  Get trajectory point number
* @return  Error code
*/
public int GetTrajectoryPointNum(int pnum)
```

2.3.9.13 Set trajectory playback speed

```
/**
* @brief  Set trajectory playback speed
* @param  [in] ovl Speed percentage
* @return  Error code
*/
public int SetTrajectoryJSpeed(double ovl)
```

2.3.9.14 Set force/torque during trajectory playback

```
/**
* @brief  Set force/torque during trajectory playback
* @param  [in] ft Force and torque in three directions (unit: N and Nm)
* @return  Error code
*/
public int SetTrajectoryJForceTorque(ForceTorque ft)
```

2.3.9.15 Set x-direction force during trajectory playback

```
/**
* @brief Set x-direction force during trajectory playback
* @param [in] fx x-direction force (unit: N)
* @return Error code
*/
int SetTrajectoryJForceFx(double fx);
```

2.3.9.16 Set y-direction force during trajectory playback

```
/**
* @brief Set y-direction force during trajectory playback
* @param [in] fy y-direction force (unit: N)
* @return Error code
*/
int SetTrajectoryJForceFy(double fy);
```

2.3.9.17 Set z-direction force during trajectory playback

```
/**
* @brief Set z-direction force during trajectory playback
* @param [in] fz z-direction force (unit: N)
* @return Error code
*/
int SetTrajectoryJForceFz(double fz);
```

2.3.9.18 Set x-axis torque during trajectory playback

```
/**
* @brief Set x-axis torque during trajectory playback
* @param [in] tx x-axis torque (unit: Nm)
* @return Error code
*/
int SetTrajectoryJTorqueTx(double tx);
```

2.3.9.19 Set y-axis torque during trajectory playback

```
/**
* @brief Set y-axis torque during trajectory playback
* @param [in] ty y-axis torque (unit: Nm)
* @return Error code
*/
int SetTrajectoryJTorqueTy(double ty);
```

2.3.9.20 Set z-axis torque during trajectory playback

```
/**
* @brief Set z-axis torque during trajectory playback
* @param [in] tz z-axis torque (unit: Nm)
* @return Error code
*/
int SetTrajectoryJTorqueTz(double tz);
```

2.3.9.21 Upload trajectory J file

```
/**
* @brief Upload trajectory J file
* @param [in] filePath Full path of trajectory file (e.g., C://test/testJ.txt)
* @return Error code
*/
int TrajectoryJUpLoad(String filePath);
```

2.3.9.22 Delete trajectory J file

```
/**
* @brief Delete trajectory J file
* @param [in] fileName File name (e.g., testJ.txt)
* @return Error code
*/
int TrajectoryJDelete(String fileName);
```

2.3.9.23 Robot trajectory J file playback code example

```
public static int TestTraj(Robot robot)
{
    int rtn = robot.TrajectoryJUpLoad("D://zUP/traj.txt");
    System.out.println("Upload TrajectoryJ A :"+ rtn);

    String traj_file_name = "/fruser/traj/traj.txt";
    rtn = robot.LoadTrajectoryJ(traj_file_name, 100, 1);
    System.out.println("LoadTrajectoryJ:"+traj_file_name+", rtn is:"+ rtn);

    DescPose traj_start_pose=new DescPose(0,0,0,0,0,0);
    rtn = robot.GetTrajectoryStartPose(traj_file_name, traj_start_pose);

    robot.Sleep(1000);

    ExaxisPos epos=new ExaxisPos(0,0,0,0);
    DescPose po=new DescPose(0,0,0,0,0,0);
    robot.SetSpeed(50);
    robot.MoveCart(traj_start_pose, 0, 0, 100, 100, 100, -1, -1);

```

(continues on next page)

(continued from previous page)

```
    int traj_num = 0;
    rtn = robot.GetTrajectoryPointNum(traj_num);

    rtn = robot.SetTrajectoryJSpeed(50.0);
    System.out.println("SetTrajectoryJSpeed is:"+ rtn);

    ForceTorque traj_force=new ForceTorque(0,0,0,0,0,0);
    traj_force.fx = 10;
    rtn = robot.SetTrajectoryJForceTorque(traj_force);
    System.out.println("SetTrajectoryJForceTorque rtn is: "+ rtn);

    rtn = robot.SetTrajectoryJForceFx(10.0);
    System.out.println("SetTrajectoryJForceFx rtn is:"+ rtn);

    rtn = robot.SetTrajectoryJForceFy(0.0);
    System.out.println("SetTrajectoryJForceFy rtn is:"+ rtn);

    rtn = robot.SetTrajectoryJForceFz(0.0);
    System.out.println("SetTrajectoryJForceFz rtn is: "+ rtn);

    rtn = robot.SetTrajectoryJTorqueTx(10.0);
    System.out.println("SetTrajectoryJTorqueTx rtn is: "+ rtn);

    rtn = robot.SetTrajectoryJTorqueTy(10.0);
    System.out.println("SetTrajectoryJTorqueTy rtn is:"+ rtn);

    rtn = robot.SetTrajectoryJTorqueTz(10.0);
    System.out.println("SetTrajectoryJTorqueTz rtn is:"+ rtn);

    rtn = robot.MoveTrajectoryJ();
    System.out.println("MoveTrajectoryJ rtn is: "+ rtn);

    return 0;
}
```

2.3.9.24 Trajectory preprocessing (lookahead)

New in version Java: SDK-v1.0.3-3.8.0

```
/**
* @brief Trajectory preprocessing (lookahead)
* @param [in] name  Trajectory file name
* @param [in] mode  Sampling mode: 0-No sampling; 1-Equal interval sampling; 2-Equal␣
↪error limit sampling
* @param [in] errorLim  Error limit (effective when using linear fitting)
* @param [in] type  Smoothing method: 0-Bezier smoothing
* @param [in] precision  Smoothing precision (effective when using Bezier smoothing)
* @param [in] vamx  Maximum speed setting (mm/s)
* @param [in] amax  Maximum acceleration setting (mm/s^2)
* @param [in] jmax  Maximum jerk setting (mm/s^3)
* @return Error code
```

(continues on next page)

(continued from previous page)

```
*/
int LoadTrajectoryLA(String name, int mode, double errorLim, int type, double precision,␣
↪double vamx, double amax, double jmax);
```

2.3.9.25 Trajectory playback (lookahead)

New in version Java: SDK-v1.0.3-3.8.0

```
/**
* @brief Trajectory playback (lookahead)
* @return Error code
*/
int MoveTrajectoryLA();
```

2.3.9.26 Trajectory playback (lookahead) code example

```
public static int TestLoadTrajLA(Robot robot)
{
    int rtn = robot.TrajectoryJUpLoad("D://zUP/traj.txt");

    String traj_file_name = "/fruser/traj/traj.txt";
    rtn = robot.LoadTrajectoryLA(traj_file_name, 1, 2, 0, 2, 100, 200, 1000);

    DescPose traj_start_pose=new DescPose(0,0,0,0,0,0);
    rtn = robot.GetTrajectoryStartPose(traj_file_name, traj_start_pose);

    robot.Sleep(1000);
    robot.SetSpeed(50);
    robot.MoveCart(traj_start_pose, 0, 0, 100, 100, 100, -1, -1);

    rtn = robot.MoveTrajectoryLA();

    robot.CloseRPC();
    return 0;
}
```

2.3.10 Robot WebAPP program usage

2.3.10.1 Set default program to load automatically on startup

```
/**
* @brief  Set default program to load automatically on startup
* @param  [in] flag  0-Do not load default program automatically, 1-Load default program␣
↪automatically
* @param  [in] program_name Program name and path, e.g., "/fruser/movej.lua" ("/fruser/"␣
↪is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the fixed path for␣
↪LA)
* @return  Error code
```

(continues on next page)

(continued from previous page)

```
*/
int LoadDefaultProgConfig(int flag, String program_name);
```

2.3.10.2 Load specified program

```
/**
* @brief  Load specified program
* @param  [in] program_name Program name and path, e.g., "/fruser/movej.lua" ("/fruser/" is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the fixed path for LA)
* @return  Error code
*/
int ProgramLoad(String program_name);
```

2.3.10.3 Get loaded program name

```
/**
* @brief  Get loaded program name
* @param  [out] program_name program_name[0]: Program name and path, e.g., "/fruser/movej.lua" ("/fruser/" is the fixed path for QX, and "/usr/local/etc/controller/lua/" is the fixed path for LA)
* @return  Error code
*/
int GetLoadedProgram(String[] program_name);
```

2.3.10.4 Get current program execution line number

```
/**
* @brief  Get current program execution line number
* @param  [out] List[0]: Error code; List[1]: int line - Line number
* @return  Error code
*/
List<Integer> GetCurrentLine();
```

2.3.10.5 Run currently loaded program

```
/**
* @brief  Run currently loaded program
* @return  Error code
*/
int ProgramRun();
```

2.3.10.6 Pause current running program

```
/**
* @brief  Pause current running program
* @return  Error code
*/
int PauseMotion();
```

2.3.10.7 Resume paused program

```
/**
* @brief  Resume paused program
* @return  Error code
*/
int ResumeMotion();
```

2.3.10.8 Stop current running program

```
/**
* @brief  Stop current running program
* @return  Error code
*/
int StopMotion();
```

2.3.10.9 Get robot program execution state

```
/**
* @brief  Get robot program execution state
* @param   [out] state 1-Program stopped/no program running, 2-Program running, 3-
↪Program paused
* @return  Error code
*/
public int GetProgramState(int[] state)
```

2.3.10.10 Robot LUA program operation code example

```
public static int TestLuaOp(Robot robot)
{
    String program_name = "/fruser/Text1.lua";
    String[] loaded_name = new String[]{""};
    int[] state=new int[]{0};
    List<Integer> line=new ArrayList<>();

    robot.Mode(0);
    robot.LoadDefaultProgConfig(0, program_name);
    robot.ProgramLoad(program_name);
    robot.ProgramRun();
```

(continues on next page)

(continued from previous page)

```
    robot.Sleep(1000);
    robot.ProgramPause();
    robot.GetProgramState(state);
    System.out.println("program state:"+ state[0]);
    line=robot.GetCurrentLine();
    System.out.println("current line:"+ line);
    robot.GetLoadedProgram(loaded_name);
    System.out.println("program name:"+ loaded_name[0]);
    robot.Sleep(1000);
    robot.ProgramResume();
    robot.Sleep(1000);
    robot.ProgramStop();
    robot.Sleep(1000);

    robot.CloseRPC();
    return 0;
}
```

2.3.10.11 Download Lua program

```
/**
* @brief Download program
* @param [in] fileName Lua file name to download ("test.lua" or "test.tar.gz")
* @param [in] savePath Local save path ("D://Down/")
* @return Error code
*/
int LuaDownLoad(String fileName, String savePath);
```

2.3.10.12 Delete Lua program

```
/**
* @brief Delete program
* @param [in] fileName Program name to delete ("test.lua")
* @return Error code
*/
int LuaDelete(String fileName);
```

2.3.10.13 Get all current Lua file names

```
/**
* @brief Get all current Lua file names
* @param [out] luaNames Program name list
* @return Error code
*/
int GetLuaList(List<String> luaNames);
```

2.3.10.14 Upload Lua program

```
/**
* @brief Upload program
* @param [in] filePath Local Lua file path (".../test.lua" or ".../test.tar.gz")
* @param [out] errStr Error message
* @return Error code
*/
int LuaUpload(String filePath, String errStr);
```

2.3.10.15 Robot LUA file upload/download code example

```
public static int TestLUAUpDownLoad(Robot robot)
{
    List<String> luaNames=new ArrayList<>();
    int rtn = robot.GetLuaList(luaNames);
    System.out.println("res is: "+rtn);
    System.out.println("size is: "+luaNames.size());
    for (int it =1; it < luaNames.size(); it++)
    {
        System.out.println(luaNames.get(it));
    }

    rtn = robot.LuaDownLoad("test.lua", "D://zDOWN/");
    System.out.println("LuaDownLoad rtn is:"+rtn);

    rtn = robot.LuaUpload("D://zUP/XG.lua","");
    System.out.println("LuaUpload rtn is:"+ rtn);

    rtn = robot.LuaDelete("XG.lua");
    System.out.println("LuaDelete rtn is:"+ rtn);

    return 0;
}
```

2.3.11 Robot peripherals

2.3.11.1 Configure gripper

```
/**
* @brief  Configure gripper
* @param  [in] config .company  Gripper manufacturer, 1-Robotiq, 2-Huiling, 3-Tianji, 4-
↪Dahuan, 5-Zhixing
* @param  [in] config .device  Device number, Robotiq(0-2F-85 series), Huiling(0-NK␣
↪series,1-Z-EFG-100), Tianji(0-TEG-110), Dahuan(0-PGI-140), Zhixing(0-CTPM2F20)
* @param  [in] config .softvesion  Software version, not currently used, default 0
* @param  [in] config .bus  Device bus position at end, not currently used, default 0
* @return  Error code
*/
int SetGripperConfig(DeviceConfig config);
```

2.3.11.2 Get gripper configuration

```
/**
* @brief  Get gripper configuration
* @param  [out] config .company  Gripper manufacturer, 1-Robotiq, 2-Huiling, 3-Tianji, 4-Dahuan, 5-Zhixing
* @param  [out] config .device  Device number, Robotiq(0-2F-85 series), Huiling(0-NK series,1-Z-EFG-100), Tianji(0-TEG-110), Dahuan(0-PGI-140), Zhixing(0-CTPM2F20)
* @param  [out] config .softvesion  Software version, not currently used, default 0
* @param  [out] config .bus  Device bus position at end, not currently used, default 0
* @return  Error code
*/
int GetGripperConfig(DeviceConfig config);
```

2.3.11.3 Activate gripper

```
/**
* @brief  Activate gripper
* @param  [in] index  Gripper number
* @param  [in] act  0-reset, 1-activate
* @return  Error code
*/
int ActGripper(int index, int act);
```

2.3.11.4 Control gripper

```
/**
* @brief  Control gripper
* @param  [in] index  Gripper number
* @param  [in] pos  Position percentage, range [0~100]
* @param  [in] vel  Speed percentage, range [0~100]
* @param  [in] force  Torque percentage, range [0~100]
* @param  [in] max_time  Maximum wait time, range [0~30000], unit ms
* @param  [in] block  0-blocking, 1-non-blocking
* @param  [in] type Gripper type, 0-parallel gripper; 1-rotary gripper
* @param  [in] rotNum Rotation turns
* @param  [in] rotVel Rotation speed percentage [0-100]
* @param  [in] rotTorque Rotation torque percentage [0-100]
* @return Error code
*/
int MoveGripper(int index, int pos, int vel, int force, int max_time, int block, int type, double rotNum, int rotVel, int rotTorque);
```

2.3.11.5 Get gripper motion status

```
/**
* @brief  Get gripper motion status
* @return List[0]:Error code; List[1] : fault  0-no error, 1-error; List[2]: staus  0-motion not completed, 1-motion completed
*/
List<Integer> GetGripperMotionDone();
```

2.3.11.6 Get gripper activation status

```
/**
* @brief  Get gripper activation status
* @return  List[0]:Error code; List[1] : fault  0-no error, 1-error; List[2]: status bit0~bit15 correspond to gripper numbers 0~15, bit=0 not activated, bit=1 activated
*/
List<Number> GetGripperActivateStatus()
```

2.3.11.7 Get gripper position

```
/**
* @brief  Get gripper position
* @return  List[0]:Error code; List[1] : fault  0-no error, 1-error; List[2]: position Position percentage, range 0~100%
*/
List<Number> GetGripperCurPosition()
```

2.3.11.8 Get gripper speed

```
/**
* @brief  Get gripper speed
* @return  List[0]:Error code; List[1] : fault  0-no error, 1-error; List[2]: speed Speed percentage, range 0~100%
*/
List<Number> GetGripperCurSpeed()
```

2.3.11.9 Get gripper current

```
/**
* @brief  Get gripper current
* @return  List[0]:Error code; List[1] : fault  0-no error, 1-error; List[2]: current Current percentage, range 0~100%
*/
List<Number> GetGripperCurCurrent()
```

2.3.11.10 Get gripper voltage

```
/**
* @brief  Get gripper voltage
* @return List[0]:Error code; List[1] : fault  0-no error, 1-error; List[2]:voltage Voltage, unit 0.1V
*/
List<Number> GetGripperVoltage()
```

2.3.11.11 Get gripper temperature

```
/**
* @brief  Get gripper temperature
* @return List[0]:Error code; List[1] : fault  0-no error, 1-error; List[2]:temp Temperature, unit °C
*/
List<Number> GetGripperTemp()
```

2.3.11.12 Calculate pre-grasp point - vision

```
/**
* @brief Calculate pre-grasp point - vision
* @param [in] desc_pos  Grasp point cartesian pose
* @param [in] zlength   z-axis offset
* @param [in] zangle    Rotation offset around z-axis
* @param [out] pre_pos  Pre-grasp point
* @return Error code
*/
int ComputePrePick(DescPose desc_pos, double zlength, double zangle, DescPose pre_pos);
```

2.3.11.13 Calculate retreat point - vision

```
/**
* @brief Calculate retreat point - vision
* @param [in] desc_pos  Grasp point cartesian pose
* @param [in] zlength   z-axis offset
* @param [in] zangle    Rotation offset around z-axis
* @param [out] post_poss Retreat point
* @return Error code
*/
int ComputePostPick(DescPose desc_pos, double zlength, double zangle, DescPose post_pos);
```

2.3.11.14 Robot gripper operation code example

```
public static int TestGripper(Robot robot)
{
    int company = 4;
    int device = 0;
    int softversion = 0;
    int bus = 2;
    int index = 2;
    int act = 0;
    int max_time = 30000;
    int block = 0;

    int current_pos = 0;
    int current = 0;
    int voltage = 0;
    int temp = 0;
    int speed = 0;

    DeviceConfig cnn=new DeviceConfig(company,device,softversion,bus);
    robot.SetGripperConfig(cnn);
    robot.GetGripperConfig(cnn);

    robot.ActGripper(index, act);
    robot.Sleep(1000);
    act = 1;
    robot.ActGripper(index, act);
    robot.Sleep(1000);

    robot.MoveGripper(index, 100, 50, 50, max_time, block, 0, 0, 0, 0);
    robot.Sleep(1000);
    robot.MoveGripper(index, 0, 50, 0, max_time, block, 0, 0, 0, 0);

    List<Integer> stat=new ArrayList<>();
    stat=robot.GetGripperMotionDone();

    List<Number> list=new ArrayList<>();
    list=robot.GetGripperActivateStatus();

    list=robot.GetGripperCurPosition();

    list=robot.GetGripperCurCurrent();

    list=robot.GetGripperVoltage();

    list=robot.GetGripperTemp();

    list=robot.GetGripperCurSpeed();

    int retval = 0;
    DescPose prepick_pose = new DescPose(){};
    DescPose postpick_pose = new DescPose(){};

```

(continues on next page)

(continued from previous page)

```
    DescPose p1Desc=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose p2Desc=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);

    retval = robot.ComputePrePick(p1Desc, 10, 0, prepick_pose);

    retval = robot.ComputePostPick(p2Desc, -10, 0, postpick_pose);
    return 0;
}
```

2.3.11.15 Get rotary gripper rotation turns

```
/**
* @brief  Get rotary gripper rotation turns
* @return List[0]:Error code List[1]: 0-no error, 1-error List[2]:Rotation turns
*/
List<Number> GetGripperRotNum();
```

2.3.11.16 Get rotary gripper rotation speed percentage

```
/**
* @brief  Get rotary gripper rotation speed percentage
* @return List[0]:Error code List[1]: 0-no error, 1-error List[2]:Rotation speed␣
↪percentage
*/
List<Number> GetGripperRotSpeed();
```

2.3.11.17 Get rotary gripper rotation torque percentage

```
/**
* @brief  Get rotary gripper rotation torque percentage
* @return List[0]:Error code List[1]: 0-no error, 1-error List[2]:Rotation torque␣
↪percentage
*/
List<Number> GetGripperRotTorque();
```

2.3.11.18 Code example for getting rotary gripper status

```
public static int TestRotGripperState(Robot robot)
{
    int fault = 0;
    List<Number> rotNum=new ArrayList<>();
    List<Number> rotSpeed=new ArrayList<>();
    List<Number> rotTorque=new ArrayList<>();

    rotNum=robot.GetGripperRotNum();
    rotSpeed=robot.GetGripperRotSpeed();
```

(continues on next page)

(continued from previous page)

```
    rotTorque=robot.GetGripperRotTorque();
    System.out.println("gripper rot num :"+rotNum.get(2)+ ", gripper rotSpeed :
↪"+rotSpeed.get(2)+",gripper rotTorque : "+rotTorque.get(2));

    return 0;
}
```

2.3.11.19 Conveyor start/stop

```
/**
* @brief  Conveyor start/stop
* @param  [in] status Status, 1-start, 0-stop
* @return  Error code
*/
int ConveyorStartEnd(int status);
```

2.3.11.20 Record IO detection point

```
/**
* @brief  Record IO detection point
* @return  Error code
*/
int ConveyorPointIORecord();
```

2.3.11.21 Record point A

```
/**
* @brief  Record point A
* @return  Error code
*/
int ConveyorPointARecord();
```

2.3.11.22 Record reference point

```
/**
* @brief  Record reference point
* @return  Error code
*/
int ConveyorRefPointRecord();
```

2.3.11.23 Record point B

```
/**
* @brief  Record point B
* @return Error code
*/
int ConveyorPointBRecord();
```

2.3.11.24 Conveyor workpiece IO detection

```
/**
* @brief Conveyor workpiece IO detection
* @param [in] max_t Maximum detection time, unit ms
* @return Error code
*/
int ConveyorIODetect(int max_t);
```

2.3.11.25 Get object current position

```
/**
* @brief Get object current position
* @param [in] mode 1-tracking grasp, 2-tracking motion, 3-TPD tracking
* @return Error code
*/
int ConveyorGetTrackData(int mode);
```

2.3.11.26 Conveyor tracking start

```
/**
* @brief Conveyor tracking start
* @param [in] status Status, 1-start, 0-stop
* @return Error code
*/
int ConveyorTrackStart(int status);
```

2.3.11.27 Conveyor tracking stop

```
/**
* @brief Conveyor tracking stop
* @return Error code
*/
int ConveyorTrackEnd();
```

2.3.11.28 Conveyor parameter configuration

Changed in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief  Conveyor parameter configuration
* @param [in] encChannel Encoder channel 1~2
* @param [in] resolution Pulses per encoder revolution
* @param [in] lead Conveyor travel distance per encoder revolution
* @param [in] wpAxis Workpiece coordinate system number For tracking motion function, set to 0 for tracking grasp and TPD tracking
* @param [in] vision Whether vision is configured  0 no  1 yes
* @param [in] speedRadio Speed ratio  For conveyor tracking grasp option (1-100) Default 1 for other options
* @param [in] followType Tracking motion type, 0-tracking motion; 1-chasing inspection motion
* @param [in] startDis Required for chasing grasp, tracking start distance, -1: auto calculate (automatically chase after workpiece reaches below robot), unit mm, default 0
* @param [in] endDis Required for chasing grasp, tracking end distance, unit mm, default 100
* @return Error code
*/
int ConveyorSetParam(int encChannel, int resolution, double lead, int wpAxis, int vision, double speedRadio, int followType, int startDis, int endDis);
```

2.3.11.29 Set conveyor grasp point compensation

```
/**
* @brief Set conveyor grasp point compensation
* @param [in] cmp Compensation position double[3]{x, y, z}
* @return Error code
*/
int ConveyorCatchPointComp(Object[] cmp);
```

2.3.11.30 Conveyor linear motion

```
/**
* @brief Linear motion
* @param [in] name Motion point description
* @param [in] tool Tool coordinate number, range [0~14]
* @param [in] wobj Workpiece coordinate number, range [0~14]
* @param [in] vel Speed percentage, range [0~100]
* @param [in] acc Acceleration percentage, range [0~100], not currently available
* @param [in] ovl Speed scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius (non-blocking), unit mm
* @return Error code
*/
int ConveyorTrackMoveL(String name, int tool, int wobj, double vel, double acc, double ovl, double blendR);
```

2.3.11.31 Conveyor communication input detection

New in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief Conveyor communication input detection
* @param [in] timeout Wait timeout time ms
* @return Error code
*/
int ConveyorComDetect(int timeout);
```

2.3.11.32 Conveyor communication input detection trigger

New in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief Conveyor communication input detection trigger
* @param [in] timeout Wait timeout time ms
* @return Error code
*/
int ConveyorComDetectTrigger();
```

2.3.11.33 Robot conveyor operation example program

```
public static int TestConveyor(Robot robot)
{
    int retval = 0;

    retval = robot.ConveyorStartEnd(1);

    retval = robot.ConveyorPointIORecord();

    retval = robot.ConveyorPointARecord();

    retval = robot.ConveyorRefPointRecord();

    retval = robot.ConveyorPointBRecord();

    retval = robot.ConveyorStartEnd(0);

    retval = 0;

    retval = robot.ConveyorSetParam(1,10000,200,0,0,20,0,0,100);

    Object[] cmp = new Object[]{ 0.0, 0.0, 0.0 };
    retval = robot.ConveyorCatchPointComp(cmp);

    int index = 1;
    int max_time = 30000;
    int block = 0;
    retval = 0;
```

(continues on next page)

(continued from previous page)

```
    DescPose p1Desc=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose p2Desc=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);


    retval = robot.MoveCart(p1Desc, 1, 0, 100.0, 100.0, 100.0, -1.0, -1);

    retval = robot.WaitMs(1);

    retval = robot.ConveyorTrackStart(1);

    retval = robot.ConveyorTrackMoveL("cvrCatchPoint", 1, 0, 100, 100, 100, -1.0);

    retval = robot.MoveGripper(index, 51, 40, 30, max_time, block, 0, 0, 0, 0);

    retval = robot.ConveyorTrackMoveL("cvrRaisePoint", 1, 0, 100, 100, 100, -1.0);

    retval = robot.ConveyorTrackEnd();

    robot.MoveCart(p2Desc, 1, 0, 100.0, 100.0, 100.0, -1.0, -1);

    retval = robot.MoveGripper(index, 100, 40, 10, max_time, block, 0, 0, 0, 0);

    return 0;
}
```

2.3.11.34 End sensor configuration

```
/**
* @brief End sensor configuration
* @param [in] config idCompany Manufacturer, 18-JUNKONG; 25-HUIDE
* @param [in] config idDevice Type, 0-JUNKONG/RYR6T.V1.0
* @param [in] config idSoftware Software version, 0-J1.0/HuiDe1.0(not currently␣
↪available)
* @param [in] config idBus Mount position, 1-end port 1; 2-end port 2...8-end port 8(not␣
↪currently available)
* @return Error code
*/
int AxleSensorConfig(DeviceConfig config);
```

2.3.11.35 Get end sensor configuration

```
/**
* @brief Get end sensor configuration
* @param [out] config idCompany Manufacturer, 18-JUNKONG; 25-HUIDE
* @param [out] config idDevice Type, 0-JUNKONG/RYR6T.V1.0
* @return Error code
*/
int AxleSensorConfigGet(DeviceConfig config);
```

2.3.11.36 End sensor activation

```
/**
* @brief End sensor activation
* @param [in] actFlag 0-reset; 1-activate
* @return Error code
*/
int AxleSensorActivate(int actFlag);
```

2.3.11.37 End sensor register write

```
/**
* @brief End sensor register write
* @param [in] devAddr  Device address number 0-255
* @param [in] regHAddr Register address high 8 bits
* @param [in] regLAddr Register address low 8 bits
* @param [in] regNum  Register count 0-255
* @param [in] data1 Write register value 1
* @param [in] data2 Write register value 2
* @param [in] isNoBlock 0-blocking; 1-non-blocking
* @return Error code
*/
int AxleSensorRegWrite(int devAddr, int regHAddr, int regLAddr, int regNum, int data1,␣
↪int data2, int isNoBlock);
```

2.3.11.38 End sensor code example

```
public static int TestAxleSensor(Robot robot)
{
    DeviceConfig con=new DeviceConfig(18,0,0,1);
    robot.AxleSensorConfig(con);
    int company = -1;
    int type = -1;
    robot.AxleSensorConfigGet(con);

    int rtn = robot.AxleSensorActivate(1);

    robot.Sleep(1000);

    rtn = robot.AxleSensorRegWrite(1, 4, 6, 1, 0, 0, 0);
    return 0;
}
```

2.3.11.39 Get robot peripheral protocol

```
/**
* @brief Get robot peripheral protocol
* @return List[0]:Error code; List[1] : int protocol Robot peripheral protocol number␣
↪4096-extension axis control card; 4097-ModbusSlave; 4098-ModbusMaster
*/
List<Integer> GetExDevProtocol();
```

2.3.11.40 Set robot peripheral protocol

```
/**
* @brief Set robot peripheral protocol
* @param [in] protocol Robot peripheral protocol number 4096-extension axis control card;
↪ 4097-ModbusSlave; 4098-ModbusMaster
* @return Error code
*/
int SetExDevProtocol(int protocol);
```

2.3.11.41 Set robot peripheral protocol example program

```
public static int TestExDevProtocol(Robot robot)
{
    int protocol = 4096;
    int rtn = robot.SetExDevProtocol(protocol);
    List<Integer> integer=new ArrayList<>();
    integer = robot.GetExDevProtocol();

    return 0;
}
```

2.3.11.42 Get end communication parameters

```
/**
* @brief Get end communication parameters
* @param [out] param End communication parameters
* @return Error code
*/
int GetAxleCommunicationParam(AxleComParam param)
```

2.3.11.43 Set end communication parameters

```
/**
* @brief Set end communication parameters
* @param [in] param End communication parameters
* @return Error code
*/
int SetAxleCommunicationParam(AxleComParam param)
```

2.3.11.44 Set end file transfer type

```
/**
* @brief Set end file transfer type
* @param [in] type 1-MCU upgrade file; 2-LUA file
* @return  Error code
*/
public int SetAxleFileType(int type)
```

2.3.11.45 Set enable end LUA execution

```
/**
* @brief Set enable end LUA execution
* @param [in] enable 0-disable; 1-enable
* @return  Error code
*/
public int SetAxleLuaEnable(int enable)
```

2.3.11.46 End LUA file error recovery

```
/**
* @brief End LUA file error recovery
* @param [in] status 0-no recovery; 1-recover
* @return  Error code
*/
public int SetRecoverAxleLuaErr(int status)
```

2.3.11.47 Get end LUA execution enable status

```
/**
* @brief Get end LUA execution enable status
* @param [out] status[0]: 0-disabled; 1-enabled
* @return  Error code
*/
int GetAxleLuaEnableStatus(int[] status)
```

2.3.11.48 Set end LUA end device enable type

```
/**
* @brief Set end LUA end device enable type
* @param forceSensorEnable Force sensor enable status, 0-disable; 1-enable
* @param gripperEnable Gripper enable status, 0-disable; 1-enable
* @param IOEnable IO device enable status, 0-disable; 1-enable
* @return  Error code
*/
public int SetAxleLuaEnableDeviceType(int forceSensorEnable, int gripperEnable, int␣
↪IOEnable)
```

2.3.11.49 Get end LUA end device enable type

```
/**
 * @brief Get end LUA end device enable type
 * @param enable enable[0]:forceSensorEnable Force sensor enable status, 0-disable; 1-
↪enable
 * @param enable enable[1]:gripperEnable Gripper enable status, 0-disable; 1-enable
 * @param enable enable[2]:IOEnable IO device enable status, 0-disable; 1-enable
 * @return  Error code
 */
public int GetAxleLuaEnableDeviceType(int[] enable)
```

2.3.11.50 Get currently configured end devices

```
/**
 * @brief Get currently configured end devices
 * @param forceSensorEnable Force sensor enable device number 0-disabled; 1-enabled
 * @param gripperEnable Gripper enable device number, 0-disabled; 1-enabled
 * @param IODeviceEnable IO device enable device number, 0-disabled; 1-enabled
 * @return  Error code
 */
public int GetAxleLuaEnableDevice(int[] forceSensorEnable, int[] gripperEnable, int[]␣
↪IODeviceEnable)
```

2.3.11.51 Set enable gripper action control function

```
/**
 * @brief Set enable gripper action control function
 * @param id Gripper device number
 * @param func func[0]-gripper enable; func[1]-gripper initialization; 2-position set; 3-
↪speed set; 4-torque set; 6-read gripper status; 7-read initialization status; 8-read␣
↪error code; 9-read position; 10-read speed; 11-read torque
 * @return  Error code
 */
public int SetAxleLuaGripperFunc(int id, int[] func)
```

2.3.11.52 Get enable gripper action control function

```
/**
 * @brief Get enable gripper action control function
 * @param id Gripper device number
 * @param func func[0]-gripper enable; func[1]-gripper initialization; 2-position set; 3-speed set; 4-torque set; 6-read gripper status; 7-read initialization status; 8-read error code; 9-read position; 10-read speed; 11-read torque
 * @return  Error code
 */
public int GetAxleLuaGripperFunc(int id, int[] func)
```

2.3.11.53 Robot Ethercat slave file write

```
/**
 * @brief Robot Ethercat slave file write
 * @param type Slave file type, 1-upgrade slave file; 2-upgrade slave configuration file
 * @param slaveID Slave number
 * @param fileName Upload file name
 * @return  Error code
 */
public int SlaveFileWrite(int type, int slaveID, String fileName)
```

2.3.11.54 Upload end Lua open protocol file

```
/**
 * @brief Upload end Lua open protocol file
 * @param filePath Local lua file path name ".../AXLE_LUA_End_DaHuan.lua"
 * @return Error code
 */
public int AxleLuaUpload(String filePath)
```

2.3.11.55 Robot Ethercat slave enter boot mode

```
/**
 * @brief Robot Ethercat slave enter boot mode
 * @return  Error code
 */
public int SetSysServoBootMode()
```

2.3.11.56 Robot end LUA file operation code example

```
public static int TestAxleLua(Robot robot)
{
    robot.AxleLuaUpload("D://zUP/AXLE_LUA_End_DaHuan.lua");

    AxleComParam param=new AxleComParam(7, 8, 1, 0, 5, 3, 1);
    robot.SetAxleCommunicationParam(param);

    robot.GetAxleCommunicationParam(param);

    robot.SetAxleLuaEnable(1);
    int[] luaEnableStatus = new int[]{0};
    robot.GetAxleLuaEnableStatus(luaEnableStatus);
    robot.SetAxleLuaEnableDeviceType(0, 1, 0);

    int forceEnable = 0;
    int gripperEnable = 0;
    int ioEnable = 0;
    int [] enable=new int[]{0,0,0};
    robot.GetAxleLuaEnableDeviceType(enable);

    int[] func = { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
    robot.SetAxleLuaGripperFunc(1, func);
    int[] getFunc = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
    robot.GetAxleLuaGripperFunc(1, getFunc);
    int[] getforceEnable = { 0,0,0,0,0,0,0,0};
    int[] getgripperEnable = { 0,0,0,0,0,0,0,0};
    int[] getioEnable = { 0,0,0,0,0,0,0,0};
    robot.GetAxleLuaEnableDevice(getforceEnable, getgripperEnable, getioEnable);
    for (int i = 0; i < 8; i++)
    {
        System.out.println(getforceEnable[i]);
    }
    System.out.println("getgripperEnable status : ");
    for (int i = 0; i < 8; i++)
    {
        System.out.println(getgripperEnable[i]);
    }
    System.out.println("getioEnable status : ");
    for (int i = 0; i < 8; i++)
    {
        System.out.println(getioEnable[i]);
```

(continues on next page)

(continued from previous page)

```
    }
    robot.ActGripper(1, 0);
    robot.Sleep(2000);
    robot.ActGripper(1, 1);
    robot.Sleep(2000);
    robot.MoveGripper(1, 90, 10, 100, 50000, 0, 0, 0, 0, 0);
    int pos = 0;
    while (true)
    {
        ROBOT_STATE_PKG pkg=new ROBOT_STATE_PKG();
        pkg=robot.GetRobotRealTimeState();
        System.out.println("gripper pos is:"+pkg.gripper_position);
        robot.Sleep(100);
    }

}
```

2.3.11.57 Get SmartTool button status

New in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief Get SmartTool button status
* @param [out] state SmartTool handle button status;(bit0:0-communication normal; 1-communication lost; bit1-undo operation; bit2-clear program; bit3-A key; bit4-B key; bit5-C key; bit6-D key; bit7-E key; bit8-IO key; bit9-manual/auto; bit10-start)
* @return Error code
*/
int GetSmarttoolBtnState(int[] state)
```

2.3.11.58 SmartTool button code example

```
public static void main(String[] args)
{
    Robot robot = new Robot();
    robot.SetReconnectParam(true, 100, 500);//Set reconnection times and interval
    robot.LoggerInit(FrLogType.DIRECT, FrLogLevel.INFO, "D://log", 10, 10);
    int rtn = robot.RPC("192.168.58.2");
    if (rtn == 0) {
        System.out.println("rpc connection success");
    } else {
        System.out.println("rpc connection fail");
        return;
    }

    int[] state = {0};
    while (true)
    {
        robot.GetSmarttoolBtnState(state);
```

(continues on next page)

(continued from previous page)

```
18
19          String binaryString = String.format("%32s", Integer.toBinaryString(state[0])).
    ↪replace(' ', '0');
20          System.out.println("GetSmarttoolBtnState:"+binaryString);
21          robot.Sleep(100);
22      }
23  }
```

2.3.11.59 Upload Open Protocol Lua File

New in version Java: SDK-v1.0.8-3.8.5

```
1  /**
2  * @brief Upload Open Protocol Lua File
3  * @param  filePath Local open protocol lua file path name
4  * @return Error code
5  */
6  public int OpenLuaUpload(String filePath)
```

2.3.11.60 Get Slave Board Parameters

New in version Java: SDK-v1.0.8-3.8.5

```
1  /**
2  * @brief  Get Slave Board Parameters
3  * @param  type  0-Ethercat, 1-CClink, 3-Ethercat, 4-EIP
4  * @param  version  Protocol version
5  * @param  connState  0-Disconnected 1-Connected
6  * @return  Error code
7  */
8  public int GetFieldBusConfig(int[] type, int[] version, int[] connState)
```

2.3.11.61 Write Slave DO

New in version Java: SDK-v1.0.8-3.8.5

```
1  /**
2  * @brief  Write Slave DO
3  * @param   DOIndex  DO number
4  * @param   wirteNum  Number to write
5  * @param   status Value to write, max 8
6  * @return  Error code
7  */
8  public int FieldBusSlaveWriteDO(int DOIndex, int wirteNum, int[] status)
```

2.3.11.62 Write Slave AO

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Write Slave AO
* @param  AOIndex  AO number
* @param  wirteNum  Number to write
* @param  status Value to write, max 8
* @return  Error code
*/
public int FieldBusSlaveWriteAO(int AOIndex, int wirteNum, int[] status)
```

2.3.11.63 Read Slave DI

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Read Slave DI
* @param  DOIndex  DI number
* @param  readNum  Number to read
* @param  status Read value, max 8
* @return  Error code
*/
public int FieldBusSlaveReadDI(int DOIndex, int readNum, int[] status)
```

2.3.11.64 Read Slave AI

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief  Read Slave AI
* @param  AIIndex  AI number
* @param  readNum  Number to read
* @param  status Read value, max 8
* @return  Error code
*/
public int FieldBusSlaveReadAI(int AIIndex, int readNum, double[] status)
```

2.3.11.65 Wait for Extended DI Input

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Wait for Extended DI Input
* @param  DIIndex DI number
* @param  status 0-Low level; 1-High level
* @param  waitMs Max waiting time (ms)
* @return Error code
*/
public int FieldBusSlaveWaitDI(int DIIndex, int status, int waitMs)
```

2.3.11.66 Wait for Extended AI Input

New in version Java: SDK-v1.0.8-3.8.5

```java
/**
* @brief Wait for Extended AI Input
* @param  AIIndex AI number
* @param  waitType 0-Greater than; 1-Less than
* @param  value AI value
* @param  waitMs Max waiting time (ms)
* @return Error code
*/
public int FieldBusSlaveWaitAI(int AIIndex, int waitType, double value, int waitMs)
```

2.3.11.67 Slave Mode Related Interface Command Code Example

```java
public static void testFieldBusBoard(Robot robot)
{
    //Upload and load open protocol file
    robot.OpenLuaUpload("D://zUP/1111/CtrlDev_field.lua");
    robot.Sleep(2000);
    robot.SetCtrlOpenLUAName(3, "CtrlDev_field.lua");
    robot.UnloadCtrlOpenLUA(3);
    robot.LoadCtrlOpenLUA(3);
    robot.Sleep(8000);
    int[] type=new int[1];
    int[] version=new int[1];
    int[] connState=new int[1];
    //Get protocol type, software version, and PLC connection status of the slave board
    robot.GetFieldBusConfig(type, version, connState);
    System.out.println("type is: "+type[0]+", version is : "+version[0]+", connState is␣
↪: "+connState[0]);
    //Write DO0 = 1, DO1 = 0, DO2 = 1
    int[] ctrl =new int[8];
    ctrl[0] = 1;
    ctrl[1] = 0;
    ctrl[2] = 1;
    robot.FieldBusSlaveWriteDO(0, 3, ctrl);
    //Write AO2 = 0x1000
    int[] ctrlAO =new int[8];
    ctrlAO[0] = 0x1000;
    robot.FieldBusSlaveWriteAO(2, 1, ctrlAO);
    int[] DI=new int[4];
    double[] AI=new double[3];
    //Loop monitor DI0~DI3 AI0~AI2
    for (int i = 0; i < 100; i++)
    {
        robot.FieldBusSlaveReadDI(0, 4, DI);
        System.out.println("DI0 is: "+DI[0]+", DI1 is: "+DI[1]+",DI2 is: "+DI[2]+",DI3␣
↪is: "+DI[3]);
        robot.FieldBusSlaveReadAI(0, 3, AI);
        System.out.println("AI0 is: "+AI[0]+ ",AI1 is: "+AI[1]+",AI2 is: "+AI[2]);
```

(continues on next page)

(continued from previous page)

```
        robot.Sleep(10);
    }
    //Wait for DI0 to be 1, wait time 100ms, and print result
    int ret = robot.FieldBusSlaveWaitDI(0, 1, 100);
    System.out.println("FieldBusSlaveWaitDI result is: "+ ret);
    //Wait for AI0 to be greater than 400, wait time 100ms, and print result
    ret = robot.FieldBusSlaveWaitAI(0,0,400.00,100);
    System.out.println("FieldBusSlaveWaitAI result is: "+ ret);
    robot.CloseRPC();
}
```

2.3.11.68 Control Array Sucker

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Control Array Sucker
* @param  slaveID Slave ID
* @param  len Length
* @param  ctrlValue Control value 1-Suction at max vacuum; 2-Suction at set vacuum; 3-
↪Stop suction
* @return Error code
*/
public int SetSuckerCtrl(int slaveID, int len, int[] ctrlValue)
```

2.3.11.69 Get Array Sucker Status

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Get Array Sucker Status
* @param  slaveID Slave ID
* @param  state Adsorption state 0-Release object; 1-Workpiece detected and adsorbed␣
↪successfully; 2-No object adsorbed; 3-Object detached
* @param  pressValue Current vacuum degree Unit kpa
* @param  error Sucker current error code
* @return Error code
*/
public int GetSuckerState(int slaveID, int[] state, int[] pressValue, int[] error)
```

2.3.11.70 Wait for Sucker Status

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief Wait for Sucker Status
* @param  slaveID Slave ID
* @param  state Adsorption state 0-Release object; 1-Workpiece detected and adsorbed successfully; 2-No object adsorbed; 3-Object detached
* @param  ms Max waiting time
* @return Error code
*/
public int WaitSuckerState(int slaveID, int state, int ms)
```

2.3.11.71 Array Sucker Control Command Code Example

```
public static void testSucker(Robot robot)
{
    //Upload and load open protocol file
    robot.OpenLuaUpload("C//project/peripheralSDK/CtrlDev_sucker.lua");
    robot.Sleep(2000);
    robot.UnloadCtrlOpenLUA(1);
    robot.LoadCtrlOpenLUA(1);
    robot.Sleep(1000);
    //Control sucker in broadcast mode, adsorb with maximum capacity
    int[] ctrl = {1};
    robot.SetSuckerCtrl(0, 1, ctrl);
    int[] state=new int[1];
    int[] pressVlaue=new int[1];
    int[] error=new int[1];
    //Loop monitor status of sucker 1 and sucker 12
    for (int i = 0; i < 100; i++)
    {
        robot.GetSuckerState(1, state,pressVlaue, error);
        System.out.println("sucker1 state is:"+state[0]+",pressVlaue is:"+pressVlaue[0]+",error num is"+error[0]);
        robot.GetSuckerState(12, state, pressVlaue, error);
        System.out.println("sucker12 state is :"+state[0]+", pressVlaue is:"+pressVlaue[0]+",error num is:"+error[0]);
        robot.Sleep(100);
    }
    //Wait for sucker 1 to be in adsorbed state, wait time 100ms
    int ret = robot.WaitSuckerState(1, 1, 100);
    System.out.println("WaitSuckerState result is:"+ ret);
    //Unicast mode to turn off sucker 1 and 12
    ctrl[0] = 3;
    robot.SetSuckerCtrl(1, 1, ctrl);
    robot.SetSuckerCtrl(12, 1, ctrl);
    robot.CloseRPC();
}
```

2.3.11.72 Laser Peripheral On/Off Function

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser peripheral on/off function
 * @param [in] OnOff 0-off 1-on
 * @param [in] weldId Weld seam ID, default is 0
 * @return Error code
 */
public int LaserTrackingLaserOnOff(int OnOff, int weldId)
```

2.3.11.73 Laser Tracking Start/Stop Function

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser tracking start/stop function
 * @param [in] OnOff 0-stop 1-start
 * @param [in] coordId Laser peripheral tool coordinate system number
 * @return Error code
 */
public int LaserTrackingTrackOnOff(int OnOff, int coordId)
```

2.3.11.74 Laser Positioning - Fixed Direction

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser positioning - fixed direction
 * @param [in] direction 0-x+ 1-x- 2-y+ 3-y- 4-z+ 5-z-
 * @param [in] vel Speed in %
 * @param [in] distance Maximum positioning distance in mm
 * @param [in] timeout Positioning timeout in ms
 * @param [in] posSensorNum Laser calibrated tool coordinate number
 * @return Error code
 */
public int LaserTrackingSearchStart_xyz(int direction, int vel, int distance, int␣
↪timeout, int posSensorNum)
```

2.3.11.75 Laser Positioning - Arbitrary Direction

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser positioning - arbitrary direction
 * @param [in] directionPoint XYZ coordinates of the positioning input point
 * @param [in] vel Speed in %
 * @param [in] distance Maximum positioning distance in mm
 * @param [in] timeout Positioning timeout in ms
```

(continues on next page)

(continued from previous page)

```
 * @param [in] posSensorNum Laser calibrated tool coordinate number
 * @return Error code
 */
public int LaserTrackingSearchStart_point(DescTran directionPoint, int vel, int distance,
↪ int timeout, int posSensorNum)
```

2.3.11.76 Laser Positioning Stop

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser positioning stop
 * @return Error code
 */
public int LaserTrackingSearchStop()
```

2.3.11.77 Laser IP Configuration

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser IP configuration
 * @param [in] ip IP address of the laser peripheral
 * @param [in] port Port number of the laser peripheral
 * @return Error code
 */
public int LaserTrackingSensorConfig(String ip, int port)
```

2.3.11.78 Laser Peripheral Sampling Period Configuration

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser peripheral sampling period configuration
 * @param [in] period Laser peripheral sampling period in ms
 * @return Error code
 */
public int LaserTrackingSensorSamplePeriod(int period)
```

2.3.11.79 Laser Peripheral Driver Loading

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser peripheral driver loading
 * @param [in] type Laser peripheral driver protocol type 101-Ruiniu 102-Chuangxiang 103-
↪Quanshi 104-Tongzhou 105-Aotai
 * @return Error code
```

(continues on next page)

(continued from previous page)

```
 */
public int LoadPosSensorDriver(int type)
```

2.3.11.80 Laser Peripheral Driver Unloading

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser peripheral driver unloading
 * @return Error code
 */
public int UnLoadPosSensorDriver()
```

2.3.11.81 Laser Weld Seam Trajectory Recording

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser weld seam trajectory recording
 * @param [in] status 0-stop recording 1-real-time tracking 2-start recording
 * @param [in] delayTime Delay time in ms
 * @return Error code
 */
public int LaserSensorRecord1(int status, int delayTime)
```

2.3.11.82 Laser Weld Seam Trajectory Replay

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser weld seam trajectory replay
 * @param [in] delayTime Delay time in ms
 * @param [in] speed Speed in %
 * @return Error code
 */
public int LaserSensorReplay(int delayTime, double speed)
```

2.3.11.83 Laser Tracking Replay

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser tracking replay
 * @return Error code
 */
public int MoveLTR()
```

2.3.11.84 Laser Weld Seam Trajectory Recording and Replay

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Laser Weld Seam Trajectory Replay
 * @param delayMode Mode 0-Delay Time 1-Delay Distance
 * @param delayTime Delay time in milliseconds (ms)
 * @param delayDisExAxisNum Extended Axis Number
 * @param delayDis Delay distance in millimeters (mm)
 * @param sensitivePara Compensation Sensitivity Coefficient
 * @param trackMode Fixed-point Tracking Type. 0-Extended Axis Asynchronous Motion; 1-Robot
 * @param triggerMode Fixed-point Tracking Trigger Method. 0-Tracking Duration; 1-IO
 * @param runTime Robot Fixed-point Tracking Duration in seconds (s)
 * @param speed Speed in percentage (%)
 * @return Error Code
 */
public int LaserSensorRecordandReplay(int delayMode, int delayTime, int delayDisExAxisNum, double delayDis, double sensitivePara, int trackMode, int triggerMode, double runTime, double speed)
```

2.3.11.85 Move to Laser Record Start Point

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Move to laser record start point
 * @param [in] moveType 0-PTP 1-LIN
 * @param [in] ovl Speed in %
 * @return Error code
 */
public int MoveToLaserRecordStart(int moveType, double ovl)
```

2.3.11.86 Move to Laser Record End Point

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Move to laser record end point
 * @param [in] moveType 0-PTP 1-LIN
 * @param [in] ovl Speed in %
 * @return Error code
 */
public int MoveToLaserRecordEnd(int moveType, double ovl)
```

2.3.11.87 Move to Laser Sensor Positioning Point

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Move to laser sensor positioning point
 * @param [in] moveFlag Motion type: 0-PTP; 1-LIN
 * @param [in] ovl Speed scaling factor, 0-100
 * @param [in] dataFlag Weld seam cache data selection: 0-execute planning data; 1-execute recorded data
 * @param [in] plateType Plate type: 0-corrugated plate; 1-corrugated cardboard; 2-fence plate; 3-oil drum; 4-corrugated shell steel
 * @param [in] trackOffectType Laser sensor offset type: 0-no offset; 1-base coordinate system offset; 2-tool coordinate system offset; 3-laser sensor raw data offset
 * @param [in] offset Offset value
 * @return Error code
 */
public int MoveToLaserSeamPos(int moveFlag, double ovl, int dataFlag, int plateType, int trackOffectType, DescPose offset)
```

2.3.11.88 Get Laser Sensor Positioning Point Coordinate Information

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Get laser sensor positioning point coordinate information
 * @param [in] trackOffectType Laser sensor offset type: 0-no offset; 1-base coordinate system offset; 2-tool coordinate system offset; 3-laser sensor raw data offset
 * @param [in] offset Offset value
 * @param [out] jPos Joint position [°]
 * @param [out] descPos Cartesian position [mm]
 * @param [out] tool Tool coordinate system
 * @param [out] user Workpiece coordinate system
 * @param [out] exaxis Extended axis position [mm]
 * @return Error code
 */
public int GetLaserSeamPos(int trackOffectType, DescPose offset, JointPos jPos, DescPose descPos, int[] tool, int[] user, ExaxisPos exaxis)
```

2.3.11.89 Laser Peripheral Sensor Parameter Configuration and Debugging Code Example

```
public static void testLaserConfig(Robot robot)
{
    robot.LaserTrackingSensorConfig("192.168.58.20", 5020);

    robot.LaserTrackingSensorSamplePeriod(20);

    robot.LoadPosSensorDriver(101);
    robot.LaserTrackingLaserOnOff(0,0);

    robot.Sleep(3000);
```

(continues on next page)

(continued from previous page)

```

    robot.LaserTrackingLaserOnOff(1, 0);

    robot.CloseRPC();
}
```

2.3.11.90 Laser Trajectory Scanning and Trajectory Replay Code Example

```
public static void testLaserRecordAndReplay(Robot robot)
{
    // Upload and load open protocol file
    robot.OpenLuaUpload("D://zUP/CtrlDev_laser_ruiniu-0117.lua");
    robot.Sleep(2000);
    robot.SetCtrlOpenLUAName(0, "CtrlDev_laser_ruiniu-0117.lua");
    robot.UnloadCtrlOpenLUA(0);
    robot.LoadCtrlOpenLUA(0);
    robot.Sleep(8000);

    for (int i=0;i<10;++i){
        JointPos startjointPos=new JointPos(56.205, -117.951, 141.872, -118.149, -94.217, -122.176);
        DescPose startdescPose=new DescPose(-97.552, -282.855, 26.675, 174.182, -1.338, -91.707);
        ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
        DescPose offdese=new DescPose(0, 0, 0, 0, 0, 0);
        robot.MoveL(startjointPos, startdescPose, 1, 0, 100, 100, 100, -1, 0,exaxisPos, 0, 0, offdese, 0,1, 1);

        robot.LaserSensorRecord1(2, 10);

        JointPos endjointPos=new JointPos(68.809, -87.100, 121.120, -127.233, -95.038, -109.555);
        DescPose enddescPose=new DescPose(-103.555, -464.234, 13.076, 174.179, -1.344, -91.709);
        robot.MoveL(endjointPos, enddescPose, 1, 0, 50, 100, 100, -1,0, exaxisPos, 0, 0, offdese, 0,1, 1);

        robot.LaserSensorRecord1(0, 10);

        robot.MoveToLaserRecordStart(1, 30);

        robot.LaserSensorReplay(10, 100);

        robot.MoveLTR();

        robot.LaserSensorRecord1(0, 10);
    }

    robot.CloseRPC();
}
```

2.3.11.91 Laser Positioning and Real-time Tracking Code Example

```
public static void testLasertrack(Robot robot)
{
    // Upload and load open protocol file
    robot.OpenLuaUpload("D://zUP/CtrlDev_laser_ruiniu-0117.lua");
    robot.Sleep(2000);
    robot.SetCtrlOpenLUAName(0, "CtrlDev_laser_ruiniu-0117.lua");
    robot.UnloadCtrlOpenLUA(0);
    robot.LoadCtrlOpenLUA(0);
    robot.Sleep(8000);
    for(int i=0;i<10;++i){
        JointPos startjointPos=new JointPos(56.205, -117.951, 141.872, -118.149, -94.217, -122.176);
        DescPose startdescPose=new DescPose(-97.552, -282.855, 26.675, 174.182, -1.338, -91.707);
        ExaxisPos exaxisPos=new ExaxisPos(0, 0, 0, 0);
        DescPose offdese=new DescPose(0, 0, 0, 0, 0, 0);
        DescTran directionPoint=new DescTran();
        robot.MoveL(startjointPos, startdescPose, 1, 0, 100, 100, 100, -1, 0,exaxisPos,0, 0, offdese, 0,1, 1);

        robot.LaserTrackingSearchStart_xyz(3, 100, 300, 1000, 3);
        robot.LaserTrackingSearchStop();

        //robot.GetRobotTeachingPoint(name, data);
        robot.MoveToLaserSeamPos(1, 30, 0, 0, 0, offdese);
        //printf("%f, %f, %f,%f, %f, %f,%f, %f, %f,%f, %f, %f\n", data[0], data[1],data[2], data[3], data[4], data[5], data[6], data[7], data[8], data[9], data[10],data[11]);

        robot.LaserTrackingTrackOnOff(1, 3);
        //robot.LaserTrackingTrackOn(3);
        JointPos endjointPos=new JointPos(68.809,-87.100,121.120,-127.233,-95.038,-109.555);
        DescPose enddescPose=new DescPose(-103.555,-464.234,13.076,174.179,-1.344,-91.709);
        robot.MoveL(endjointPos, enddescPose, 1, 0, 20, 100, 100, -1, 0,exaxisPos, 0, 0,offdese, 0,1, 1);

        robot.LaserTrackingTrackOnOff(0, 3);
        System.out.println("Current iteration: "+(i+1));
    }
    robot.CloseRPC();
}
```

2.3.11.92 Extended Axis and Robot Synchronized Laser Tracking Code Example

```java
public static void testLasertrackandExitAxis(Robot robot)
{
    ExaxisPos startexaxisPos =new ExaxisPos( 0,0,0,0 );
    ExaxisPos seamexaxisPos = new ExaxisPos(-10,0,0,0 );
    ExaxisPos endexaxisPos = new ExaxisPos(-30, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0 );
    JointPos seamjointPos=new JointPos(0, 0, 0, 0, 0, 0);
    DescPose seamdescPose=new DescPose(0, 0, 0, 0, 0, 0);

    for(int i =0;i<10;++i) {
        // Move to the starting point for positioning
        JointPos startjointPos = new JointPos(58.337, -119.628, 146.037, -116.358, -92.
↪224, -117.654);
        DescPose startdescPose = new DescPose(-53.375, -255.363, 0.919, 178.054, 1.077, -
↪94.026);
        robot.ExtAxisSyncMoveJ(startjointPos, startdescPose, 1, 0, 100, 100, 100,␣
↪startexaxisPos, -1, 0, offdese);

        System.out.println("11111");
        // Start positioning along -y direction
        int ret = robot.LaserTrackingSearchStart_xyz(3, 100, 300, 1000, 2);
        robot.LaserTrackingSearchStop();
        System.out.println("2222");
        int[] tool = new int[1];
        int[] user = new int[1];
        robot.GetLaserSeamPos(0, offdese, seamjointPos, seamdescPose, tool, user,␣
↪startexaxisPos);
        System.out.println(seamjointPos.J1 + ", " + seamjointPos.J2 + ", " +
                seamjointPos.J3 + ", " + seamjointPos.J4 + ", " +
                seamjointPos.J5 + ", " + seamjointPos.J6 + ", " +
                seamdescPose.tran.x + ", " + seamdescPose.tran.y + ", " +
                seamdescPose.tran.z + ", " + seamdescPose.rpy.rx + ", " +
                seamdescPose.rpy.ry + ", " + seamdescPose.rpy.rz);
        // If positioning is successful
        if (ret == 0) {
            // Robot and extended axis synchronously move to the positioning point
            robot.ExtAxisSyncMoveJ(seamjointPos, seamdescPose, 1, 0, 100, 100, 100,␣
↪seamexaxisPos, -1, 0, offdese);

            // Start laser tracking along the positioning point and synchronously move␣
↪with extended axis
            System.out.println("3333");
            robot.LaserTrackingTrackOnOff(1, 2);
            JointPos endjointPos = new JointPos(70.580, -90.918, 126.593, -125.154, -92.
↪162, -105.403);
            DescPose enddescPose = new DescPose(-53.375, -419.020, 0.920, 178.054, 1.076,
↪ -94.026);
            robot.ExtAxisSyncMoveL(endjointPos, enddescPose, 1, 0, 20, 100, 100, -1,␣
↪endexaxisPos, 0, offdese);
            ;
            // Stop tracking
```

(continues on next page)

(continued from previous page)

```
            robot.LaserTrackingTrackOnOff(0, 2);
            System.out.println("44444");
        }
        System.out.println("Current run count: "+i);
    }
    robot.CloseRPC();
}
```

2.3.12 Robot Force Control

2.3.12.1 Configure Force Sensor

```
/**
* @brief  Configure force sensor
* @param  config company: Force sensor manufacturer, 17-Kunwei Technology, 19-Aerospace 11th Academy, 20-ATI Sensor, 21-Zhongke Midian, 22-Weihang Minxin, 23-NBIT, 24-Xinjingcheng (XJC), 26-NSR
* @param  config device: Device number, Kunwei (0-KWR75B), Aerospace 11th Academy (0-MCS6A-200-4), ATI (0-AXIA80-M8), Zhongke Midian (0-MST2010), Weihang Minxin (0-WHC6L-YB-10A), NBIT (0-XLH93003ACS), Xinjingcheng XJC (0-XJC-6F-D82), NSR (0-NSR-FTSensorA)
* @param  config softvesion: Software version number, not currently used, default is 0
* @param  config bus: Device bus location, not currently used, default is 0
* @return  Error code
*/
int FT_SetConfig(DeviceConfig config);
```

2.3.12.2 Get Force Sensor Configuration

```
/**
* @brief Get force sensor configuration
* @param [out] config company: Force sensor manufacturer, 17-Kunwei Technology, 19-Aerospace 11th Academy, 20-ATI Sensor, 21-Zhongke Midian, 22-Weihang Minxin
* @param [out] config device: Device number, Kunwei (0-KWR75B), Aerospace 11th Academy (0-MCS6A-200-4), ATI (0-AXIA80-M8), Zhongke Midian (0-MST2010), Weihang Minxin (0-WHC6L-YB-10A)
* @param [out] config softvesion: Software version number, not currently used, default is 0
* @param [out] config bus: Device bus location, not currently used, default is 0
* @return Error code
*/
int FT_GetConfig(DeviceConfig config);
```

2.3.12.3 Activate Force Sensor

```
/**
* @brief  Activate force sensor
* @param  [in] act  0-Reset, 1-Activate
* @return  Error code
*/
int FT_Activate(int act);
```

2.3.12.4 Zero Calibration for Force Sensor

```
/**
* @brief  Zero calibration for force sensor
* @param  [in] act  0-Remove zero point, 1-Zero calibration
* @return  Error code
*/
int FT_SetZero(int act);
```

2.3.12.5 Set Force Sensor Reference Coordinate System

```
/**
* @brief  Set force sensor reference coordinate system
* @param  [in] type  0-Tool coordinate system, 1-Base coordinate system, 2-Free␣
↪coordinate system
* @param  [in] coord  Free coordinate system value
* @return  Error code
*/
int FT_SetRCS(int type, DescPose coord);
```

2.3.12.6 Set Load Weight Under Force Sensor

```
/**
* @brief Set load weight under force sensor
* @param [in] weight Load weight in kg
* @return Error code
*/
int SetForceSensorPayLoad(double weight);
```

2.3.12.7 Set Load Center of Gravity Under Force Sensor

```
/**
* @brief Set load center of gravity under force sensor
* @param [in] cog Load center of gravity in mm
* @return Error code
*/
int SetForceSensorPayLoadCog(DescTran cog);
```

2.3.12.8 Get Load Weight Under Force Sensor

```
/**
* @brief Get load weight under force sensor
* @return List[0]: Error code; List[1]: weight Load weight in kg
*/
List<Number> GetForceSensorPayLoad();
```

2.3.12.9 Get Load Center of Gravity Under Force Sensor

```
/**
* @brief Get load center of gravity under force sensor
* @param [out] cog Load center of gravity in mm
* @return Error code
*/
int GetForceSensorPayLoadCog(DescTran cog);
```

2.3.12.10 Automatic Zero Calibration for Force Sensor

```
/**
* @brief Automatic zero calibration for force sensor
* @param [in] massCenter Sensor mass (kg) and center of gravity (mm)
* @return Error code
*/
int ForceSensorAutoComputeLoad(MassCenter massCenter);
```

2.3.12.11 Get Force/Torque Data in Reference Coordinate System

```
/**
* @brief  Get force/torque data in reference coordinate system
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] ft  Force/torque, fx,fy,fz,tx,ty,tz
* @return  Error code
*/
int FT_GetForceTorqueRCS(int flag, ForceTorque ft);
```

2.3.12.12 Get Raw Force/Torque Data from Force Sensor

```
/**
* @brief  Get raw force/torque data from force sensor
* @param  [in] flag 0-Blocking, 1-Non-blocking
* @param  [out] ft  Force/torque, fx,fy,fz,tx,ty,tz
* @return  Error code
*/
int FT_GetForceTorqueOrigin(int flag, ForceTorque ft);
```

2.3.12.13 Force Sensor Configuration and Automatic Zero Calibration Example

```
public static int TestFTInit(Robot robot)
{
    DescTran tr1=new DescTran(0,0,0);
    robot.SetForceSensorPayload(0);
    robot.SetForceSensorPayloadCog(tr1);

    int company = 24;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    int index = 1;

    DeviceConfig con=new DeviceConfig(company,device,softversion,bus);
    robot.FT_SetConfig(con);
    robot.Sleep(1000);
    robot.FT_GetConfig(con);
    robot.Sleep(1000);

    robot.FT_Activate(0);
    robot.Sleep(1000);
    robot.FT_Activate(1);
    robot.Sleep(1000);

    robot.Sleep(1000);
    robot.FT_SetZero(0);
    robot.Sleep(1000);

    ForceTorque ft=new ForceTorque(0,0,0,0,0,0);
    robot.FT_GetForceTorqueOrigin(0, ft);
    robot.FT_SetZero(1);
    robot.Sleep(1000);

    DescPose ftCoord = new DescPose();
    robot.FT_SetRCS(0, ftCoord);

    robot.SetForceSensorPayload(0.824);

    DescTran tr=new DescTran(0.778, 2.554, 48.765);
    robot.SetForceSensorPayloadCog(tr);
    List<Number> weight = new ArrayList<>();
    double x = 0, y = 0, z = 0;
    weight=robot.GetForceSensorPayload();
    robot.GetForceSensorPayloadCog(tr);
    tr.x=0;
    tr.y=0;
    tr.z=0;
    robot.SetForceSensorPayload(0);
    robot.SetForceSensorPayloadCog(tr);

    double computeWeight = 0;
    DescTran tran = new DescTran();
```

(continues on next page)

(continued from previous page)

```
    MassCenter mass=new MassCenter();
    mass.weight=weight.get(1).doubleValue();
    mass.cog=tran;
    robot.ForceSensorAutoComputeLoad(mass);
    return 0;
}
```

2.3.12.14 Load Weight Identification Record

```
/**
* @brief  Load weight identification record
* @param  [in] id  Sensor coordinate system number, range [1~14]
* @return  Error code
*/
int FT_PdIdenRecord(int id);
```

2.3.12.15 Load Weight Identification Calculation

```
/**
* @brief  Load weight identification calculation
* @return  List[0]: Error code; List[1]: double weight Load weight in kg
*/
List<Number> FT_PdIdenCompute();
```

2.3.12.16 Load Center of Gravity Identification Record

```
/**
* @brief  Load center of gravity identification record
* @param  [in] id  Sensor coordinate system number, range [1~14]
* @param  [in] index Point number, range [1~3]
* @return  Error code
*/
int FT_PdCogIdenRecord(int id, int index);
```

2.3.12.17 Load Center of Gravity Identification Calculation

```
/**
* @brief  Load center of gravity identification calculation
* @param  [out] cog  Load center of gravity in mm
* @return  Error code
*/
int FT_PdCogIdenCompute(DescTran cog);
```

2.3.12.18 Force Sensor Load Identification Example

```
public static int TestFTLoadCompute(Robot robot)
{
    DescTran tr1=new DescTran(0,0,0);
    robot.SetForceSensorPayload(0);
    robot.SetForceSensorPayloadCog(tr1);

    int company = 24;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    int index = 1;

    DeviceConfig con=new DeviceConfig(company, device, softversion, bus);
    robot.FT_SetConfig(con);
    robot.Sleep(1000);
    robot.FT_GetConfig(con);
    robot.Sleep(1000);

    robot.FT_Activate(0);
    robot.Sleep(1000);
    robot.FT_Activate(1);
    robot.Sleep(1000);

    robot.Sleep(1000);
    robot.FT_SetZero(0);
    robot.Sleep(1000);

    ForceTorque ft=new ForceTorque(0,0,0,0,0,0);
    robot.FT_GetForceTorqueOrigin(0, ft);
    robot.FT_SetZero(1);
    robot.Sleep(1000);

    DescPose tcoord = new DescPose();
    tcoord.tran.z = 35.0;
    robot.SetToolCoord(10, tcoord, 1, 0, 0, 0);

    robot.FT_PdIdenRecord(10);
    robot.Sleep(1000);

    List<Number> weight =new ArrayList<>();
    weight=robot.FT_PdIdenCompute();

    DescPose desc_p1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_p2=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose desc_p3=new DescPose(-327.622, 402.230, 320.402, -178.067, 2.127, -46.207);

    robot.MoveCart(desc_p1, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
    robot.Sleep(1000);
    robot.FT_PdCogIdenRecord(10, 1);
    robot.MoveCart(desc_p2, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
    robot.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
    robot.FT_PdCogIdenRecord(10, 2);
    robot.MoveCart(desc_p3, 0, 0, 100.0, 100.0, 100.0, -1.0, -1);
    robot.Sleep(1000);
    robot.FT_PdCogIdenRecord(10, 3);
    robot.Sleep(1000);
    DescTran cog=new DescTran(0,0,0);
    robot.FT_PdCogIdenCompute(cog);

    robot.CloseRPC();
    return 0;
}
```

2.3.12.19 Collision Guard

```
/**
* @brief  Collision guard
* @param  [in] flag 0-Disable collision guard, 1-Enable collision guard
* @param  [in] sensor_id Force sensor number
* @param  [in] select  Select which degrees of freedom to detect collision, 0-Disable, 1-Enable
* @param  [in] ft  Collision force/torque, fx,fy,fz,tx,ty,tz
* @param  [in] max_threshold Maximum threshold
* @param  [in] min_threshold Minimum threshold
* @note   Force/torque detection range: (ft-min_threshold, ft+max_threshold)
* @return  Error code
*/
int FT_Guard(int flag, int sensor_id, Object[] select, ForceTorque ft, Object[] max_threshold, Object[] min_threshold);
```

2.3.12.20 Collision Guard Example

```
public static void main(String[] args)
{
    Robot robot = new Robot();
    robot.SetReconnectParam(true,20,500);//Set reconnection times and interval
    robot.LoggerInit(FrLogType.DIRECT, FrLogLevel.INFO, "D://log", 10, 10);
    int rtn = robot.RPC("192.168.58.2");
    if(rtn == 0)
    {
        System.out.println("rpc connection success");
    }
    else
    {
        System.out.println("rpc connection fail");
        return ;
    }
    byte flag = 1;
    byte sensor_id = 8;
    Object[] select = { 1, 0, 0, 0, 0, 0 };//Only enable collision guard on x-axis
```

(continues on next page)

(continued from previous page)

```
    Object[] max_threshold = { 5.0, 0.01, 0.01, 0.01, 0.01, 0.01 };
    Object[] min_threshold = { 3.0, 0.01, 0.01, 0.01, 0.01, 0.01 };

    ForceTorque ft = new ForceTorque(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
    DescPose  desc_p1, desc_p2, desc_p3;
    desc_p1 = new DescPose(-14.404,-455.283,319.847,-172.935,25.141,-68.097);
    desc_p2 = new DescPose(-107.999,-599.174,285.939,153.472,12.686,-71.284);
    desc_p3 = new DescPose(6.586,-704.897,309.638,178.909,-27.759,-70.479);

    int rtn =  robot.FT_Guard(flag, sensor_id, select, ft, max_threshold, min_threshold);
    System.out.println("FT_Guard start rtn {rtn}");
    robot.MoveCart(desc_p1, 0, 0, 20, 100.0f, 100.0f, -1.0f, -1);
    robot.MoveCart(desc_p2, 0, 0, 20, 100.0f, 100.0f, -1.0f, -1);
    robot.MoveCart(desc_p3, 0, 0, 20, 100.0f, 100.0f, -1.0f, -1);
}
```

2.3.12.21 Constant Force Control

```
/**
* @brief  Constant force control
* @param  [in] flag 0-Disable constant force control, 1-Enable constant force control
* @param  [in] sensor_id Force sensor number
* @param  [in] select  Select which degrees of freedom to detect, 0-Disable, 1-Enable
* @param  [in] ft  Force/torque, fx,fy,fz,tx,ty,tz
* @param  [in] ft_pid Force PID parameters, torque PID parameters
* @param  [in] adj_sign Adaptive start/stop control, 0-Disable, 1-Enable
* @param  [in] ILC_sign ILC start/stop control, 0-Stop, 1-Training, 2-Operation
* @param  [in] max_dis Maximum adjustment distance in mm
* @param  [in] max_ang Maximum adjustment angle in deg
* @param  [in] filter_Sign Filter enable flag, 0-Disable, 1-Enable, default is disabled
* @param  [in] posAdapt_sign Posture compliance enable flag, 0-Disable, 1-Enable,␣
↪default is disabled
* @param  [in] isNoBlock Blocking flag, 0-Blocking, 1-Non-blocking
* @return  Error code
*/
int FT_Control(int flag, int sensor_id, Object[] select, ForceTorque ft, Object[] ft_pid,
↪ int adj_sign, int ILC_sign, double max_dis, double max_ang, int filter_Sign, int␣
↪posAdapt_sign, int isNoBlock);
```

2.3.12.22 Constant Force Control

```
/**
* @brief Constant Force Control
* @param  flag 0-Disable constant force control, 1-Enable constant force control
* @param  sensor_id Force sensor ID
* @param  select  Collision detection for six degrees of freedom, 0-Disable, 1-Enable
* @param  ft  Collision force/torque, fx, fy, fz, tx, ty, tz
* @param  ft_pid Force PID parameters, Torque PID parameters
* @param  adj_sign Adaptive start/stop control, 0-Disable, 1-Enable
```

(continues on next page)

(continued from previous page)

```
* @param  ILC_sign ILC start/stop control: 0-Stop, 1-Training, 2-Operational
* @param  max_dis Maximum adjustment distance, unit mm
* @param  max_ang Maximum adjustment angle, unit deg
* @param  M Mass parameter for rx, ry [0.1-10], default 2
* @param  B Damping parameter for rx, ry [0.1-50], default 8
* @param  threshold Activation threshold for rx, ry [0-10], default 0.2
* @param  adjustCoeff Torque adjustment coefficient for rx, ry [0-1], default 1
* @param  polishRadio Polishing radius, in mm
* @param  filter_Sign Filter enable flag: 0-Off; 1-On (default off)
* @param  posAdapt_sign Pose adaptation enable flag: 0-Off; 1-On (default off)
* @param  isNoBlock Blocking flag: 0-Blocking; 1-Non-blocking
* @return Error code
*/
public int FT_Control(int flag, int sensor_id, int[] select, ForceTorque ft, double[] ft_
↪pid, int adj_sign, int ILC_sign, double max_dis, double max_ang,double[] M,double[] B,␣
↪double[] threshold, double[] adjustCoeff, double polishRadio,int filter_Sign, int␣
↪posAdapt_sign, int isNoBlock)
```

2.3.12.23 Example Code for Constant Force Control with Damping

```
public static int TestFTControlWithAdjustCoeff(Robot robot)
{
    int sensor_id = 10;
    int[] select = { 0,0,1,0,0,0 };
    double[] ft_pid = { 0.0008, 0.0, 0.0, 0.0, 0.0, 0.0 };
    int adj_sign = 0;
    int ILC_sign = 0;
    double max_dis = 1000.0;
    double max_ang = 20;
    ForceTorque ft = new ForceTorque(0.0,0,0,0,0,0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);
    JointPos j1=new JointPos(80.765, -98.795, 106.548, -97.734, -89.999, 94.842);
    JointPos j2=new JointPos(43.067, -84.429, 92.620, -98.175, -90.011, 57.144);
    DescPose desc_p1=new DescPose(5.009, -547.463, 262.053, -179.999, -0.019, 75.923);
    DescPose desc_p2=new DescPose(-347.966, -547.463, 262.048, -180.000, -0.019, 75.923);
    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    double[] M = { 2.0, 2.0 };
    double[] B = { 15.0, 15.0 };
    double[] threshold = {1.0, 1.0};
    double[] adjustCoeff = {1.0, 0.8};
    double polishRadio = 0.0;
    int filter_Sign = 0;
    int posAdapt_sign = 1;
    int isNoBlock;
    ft.fz = -10.0;
    while(true)
    {
        int rtn = robot.FT_Control(1, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign,␣
↪max_dis, max_ang, M, B, threshold, adjustCoeff, 0, 0, 1, 0);
        System.out.printf("FT_Control start rtn is %d\n", rtn);
```

(continues on next page)

(continued from previous page)

```
        robot.MoveL(j1, desc_p1, 1, 0, 100.0, 100.0, 100.0, -1.0, 0, epos, 0, 0, offset_
↪pos, 0,0, 0,10);
        robot.MoveL(j2, desc_p2, 1, 0, 100.0, 100.0, 100.0, -1.0, 0, epos, 0, 0, offset_
↪pos, 0,0, 0,10);
        rtn = robot.FT_Control(0, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_
↪dis, max_ang, M, B, threshold, adjustCoeff, 0, 0, 1, 0);
        System.out.printf("FT_Control end rtn is %d\n", rtn);
    }
}
```

2.3.12.24 Rotational Insertion

```
/**
* @brief Rotational Insertion
* @param rcs Reference coordinate system, 0 - Tool coordinate system, 1 - Base␣
↪coordinate system
* @param angVelRot Rotational angular velocity, unit deg/s
* @param ft Force/Torque threshold, fx,fy,fz,tx,ty,tz, range [0~100]
* @param max_angle Maximum rotation angle, unit deg
* @param orn Force/Torque direction, 1 - Along Z-axis direction, 2 - Around Z-axis␣
↪direction
* @param max_angAcc Maximum rotational acceleration, unit deg/s^2, currently unused,␣
↪default is 0
* @param rotorn Rotation direction, 1 - Clockwise, 2 - Counterclockwise
* @param strategy Processing strategy for undetected force/torque, 0 - Error; 1 -␣
↪Warning, continue motion
* @return Error code
*/
public int FT_RotInsertion(int rcs, double angVelRot, double ft, double max_angle, int␣
↪orn, double max_angAcc, int rotorn, int strategy)
```

2.3.12.25 Robot Force Sensor Rotational Insertion Code Example

```
public static int TestMove(Robot robot)
{
    int rtn=-1;
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    JointPos j3=new JointPos(-29.777, -84.536, 109.275, -114.075, -86.655, 74.257);
    JointPos j4=new JointPos(-31.154, -95.317, 94.276, -88.079, -89.740, 74.256);
    DescPose desc_pos1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_pos2=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.
↪869);
    DescPose desc_pos3=new DescPose(-487.434, 154.362, 308.576, 176.600, 0.268, -14.061);
    DescPose desc_pos4=new DescPose(-443.165, 147.881, 480.951, 179.511, -0.775, -15.
↪409);
    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);
    int tool = 0;
```

(continues on next page)

(continued from previous page)

```
    int user = 0;
    double vel = 100.0;
    double acc = 100.0;
    double ovl = 100.0;
    double oacc = 100.0;
    double blendT = 0.0;
    double blendR = 0.0;
    int flag = 0;
    int search = 0;
    int blendMode = 0;
    int velAccMode = 0;
    robot.SetSpeed(20);
    rtn = robot.MoveJ(j1, desc_pos1, tool, user, vel, acc, ovl, epos, blendT, flag,␣
↪offset_pos);
    System.out.printf("movej errcode:%d\n", rtn);
    rtn = robot.MoveL(j2, desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, epos,␣
↪search, flag, offset_pos, oacc, velAccMode,0,10);
    System.out.printf("movel errcode:%d\n", rtn);
    rtn = robot.MoveC(j3, desc_pos3, tool, user, vel, acc, epos, flag, offset_pos, j4,␣
↪desc_pos4, tool, user, vel, acc, epos, flag, offset_pos, ovl, blendR, oacc,␣
↪velAccMode);
    System.out.printf("movec errcode:%d\n", rtn);
    rtn = robot.MoveJ(j2, desc_pos2, tool, user, vel, acc, ovl, epos, blendT, flag,␣
↪offset_pos);
    System.out.printf("movej errcode:%d\n", rtn);
    rtn = robot.Circle(j3, desc_pos3, tool, user, vel, acc, epos, j1, desc_pos1, tool,␣
↪user, vel, acc, epos, ovl, flag, offset_pos, oacc, -1, velAccMode);
    System.out.printf("circle errcode:%d\n", rtn);
    rtn = robot.MoveCart(desc_pos4, tool, user, vel, acc, ovl, blendT, -1);
    System.out.printf("MoveCart errcode:%d\n", rtn);
    rtn = robot.MoveJ(j1, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos);
    System.out.printf("movej errcode:%d\n", rtn);
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, blendMode, epos,␣
↪search, flag, offset_pos, -1, velAccMode,0,10);
    System.out.printf("movel errcode:%d\n", rtn);
    rtn = robot.MoveC(desc_pos3, tool, user, vel, acc, epos, flag, offset_pos, desc_pos4,
↪ tool, user, vel, acc, epos, flag, offset_pos, ovl, blendR, -1, velAccMode);
    System.out.printf("movec errcode:%d\n", rtn);
    rtn = robot.MoveJ(j2, tool, user, vel, acc, ovl, epos, blendT, flag, offset_pos);
    System.out.printf("movej errcode:%d\n", rtn);
    rtn = robot.Circle(desc_pos3, tool, user, vel, acc, epos, desc_pos1, tool, user, vel,
↪ acc, epos, ovl, flag, offset_pos, oacc, blendR, -1, velAccMode);
    System.out.printf("circle errcode:%d\n", rtn);
    return 0;
}
```

2.3.12.26 Constant Force Control Example

```
public static int TestFTControl(Robot robot)
{
    DescTran tr1=new DescTran(0,0,0);
    robot.SetForceSensorPayload(0);
    robot.SetForceSensorPayloadCog(tr1);

    int company = 24;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    int index = 1;
    DeviceConfig con=new DeviceConfig(company, device, softversion, bus);

    robot.FT_SetConfig(con);
    robot.Sleep(1000);
    robot.FT_GetConfig(con);
    robot.Sleep(1000);

    robot.FT_Activate(0);
    robot.Sleep(1000);
    robot.FT_Activate(1);
    robot.Sleep(1000);

    robot.Sleep(1000);
    robot.FT_SetZero(0);
    robot.Sleep(1000);

    int sensor_id = 1;
    Object[] select =new Object[] { 0,0,1,0,0,0 };
    Object[] ft_pid =new Object[]{ 0.0005,0.0,0.0,0.0,0.0,0.0 };
    int adj_sign = 0;
    int ILC_sign = 0;
    double max_dis = 100.0;
    double max_ang = 0.0;

    ForceTorque ft=new ForceTorque(0,0,0,0,0,0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);
    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose desc_p1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_p2=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);
    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);

    ft.fz = -10.0;

    int rtn = robot.MoveJ(j1, desc_p1, 0, 0, 100.0, 180.0, 100.0, epos, -1.0, 0, offset_
↪pos);
    robot.FT_Control(1, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis, max_
↪ang,0,0,0);
    rtn = robot.MoveJ(j2, desc_p2, 0, 0, 100.0, 180.0, 100.0, epos, -1.0, 0, offset_pos);
    robot.FT_Control(0, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis, max_
```

(continues on next page)

(continued from previous page)

```
↪ang,0,0,0);
    return 0;
}
```

2.3.12.27 Compliance Control Start

```
/**
* @brief  Compliance control start
* @param  [in] p Position adjustment coefficient or compliance coefficient
* @param  [in] force Compliance activation force threshold in N
* @return  Error code
*/
int FT_ComplianceStart(double p, double force);
```

2.3.12.28 Compliance Control Stop

```
/**
* @brief  Compliance control stop
* @return  Error code
*/
int FT_ComplianceStop();
```

2.3.12.29 Compliance Control Example

```
public static int TestCompliance(Robot robot)
{
    DescTran tr1=new DescTran(0,0,0);
    robot.SetForceSensorPayload(0);
    robot.SetForceSensorPayloadCog(tr1);

    int company = 24;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    int index = 1;

    DeviceConfig con=new DeviceConfig(company, device, softversion, bus);
    robot.FT_SetConfig(con);
    robot.Sleep(1000);
    robot.FT_GetConfig(con);

    robot.Sleep(1000);

    robot.FT_Activate(0);
    robot.Sleep(1000);
    robot.FT_Activate(1);
    robot.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
    robot.Sleep(1000);
    robot.FT_SetZero(0);
    robot.Sleep(1000);

    int flag = 1;
    int sensor_id = 1;
    Object[] select =new Object[] { 1,1,1,0,0,0 };
    Object[] ft_pid =new Object[] { 0.0005,0.0,0.0,0.0,0.0,0.0 };
    int adj_sign = 0;
    int ILC_sign = 0;
    double max_dis = 100.0;
    double max_ang = 0.0;

    ForceTorque ft=new ForceTorque(0,0,0,0,0,0);
    DescPose  offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);


    JointPos j1=new JointPos(-11.904, -99.669, 117.473, -108.616, -91.726, 74.256);
    JointPos j2=new JointPos(-45.615, -106.172, 124.296, -107.151, -91.282, 74.255);
    DescPose desc_p1=new DescPose(-419.524, -13.000, 351.569, -178.118, 0.314, 3.833);
    DescPose desc_p2=new DescPose(-321.222, 185.189, 335.520, -179.030, -1.284, -29.869);

    ft.fx = -10.0;
    ft.fy = -10.0;
    ft.fz = -10.0;
    robot.FT_Control(flag, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);
    double p = 0.00005;
    double force = 30.0;
    int rtn = robot.FT_ComplianceStart(p, force);

    int count = 15;
    while (count>0)
    {
        robot.MoveL(j1, desc_p1, 0, 0, 100.0, 180.0, 100.0, -1.0,0, epos, 0, 1, offset_
↪pos,0,10);
        robot.MoveL(j2, desc_p2, 0, 0, 100.0, 180.0, 100.0, -1.0,0, epos, 0, 0, offset_
↪pos,0,10);
        count -= 1;
    }
    robot.FT_ComplianceStop();
    flag = 0;
    robot.FT_Control(flag, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis,␣
↪max_ang, 0, 0, 0);

    robot.CloseRPC();
    return 0;
}
```

2.3.12.30 Load Identification Initialization

```
/**
* @brief Load identification initialization
* @return Error code
*/
int LoadIdentifyDynFilterInit();
```

2.3.12.31 Load Identification Variable Initialization

```
/**
* @brief Load identification variable initialization
* @return Error code
*/
int LoadIdentifyDynVarInit();
```

2.3.12.32 Load Identification Main Program

```
/**
* @brief Load identification main program
* @param [in] joint_torque Joint torque
* @param [in] joint_pos Joint position
* @param [in] t Sampling period
* @return Error code
*/
int LoadIdentifyMain(Object[] joint_torque, Object[] joint_pos, double t);
```

2.3.12.33 Get Load Identification Result

```
/**
* @brief Get load identification result
* @param [in] gain
* @return List[0]: Error code; List[1]: double weight Load weight; List[2]: x Load␣
↪center of gravity X in mm; List[3]: y Load center of gravity Y in mm; List[2]: z Load␣
↪center of gravity Z in mm
*/
List<Number> LoadIdentifyGetResult(Object[] gain);
```

2.3.12.34 Robot Load Identification Example

```
public static int TestIdentify(Robot robot)
{
    int retval = 0;

    retval = robot.LoadIdentifyDynFilterInit();

    retval = robot.LoadIdentifyDynVarInit();
```

(continues on next page)

(continued from previous page)

```
    JointPos posJ = new JointPos(0,0,0,0,0,0);
    DescPose posDec = new DescPose(0,0,0,0,0,0);
    List<Number> joint_toq=new ArrayList<>();
    robot.GetActualJointPosDegree( posJ);
    posJ.J2 = posJ.J2 + 10;
    joint_toq=robot.GetJointTorques(0);

    Object[] gain =new Object[] { 0,0.05,0,0,0,0,0,0.02,0,0,0,0 };
    double weight = 0;
    DescTran load_pos=new DescTran(0,0,0);
    List<Number> num=new ArrayList<>();
    num = robot.LoadIdentifyGetResult(gain);

    robot.CloseRPC();
    return 0;

}
```

2.3.12.35 Force Sensor Assisted Dragging

Changed in version Java: SDK-v1.0.6-3.8.3

```
/**
* @brief Force sensor assisted dragging
* @param [in] status Control status, 0-Disable; 1-Enable
* @param [in] asaptiveFlag Adaptive enable flag, 0-Disable; 1-Enable
* @param [in] interfereDragFlag Interference zone dragging flag, 0-Disable; 1-Enable
* @param [in] ingularityConstraintsFlag Singularity strategy, 0-Avoid; 1-Traverse
* @param [in] forceCollisionFlag Robot collision detection flag during assisted dragging;
↪ 0-Disable; 1-Enable
* @param [in] M Inertia coefficient
* @param [in] B Damping coefficient
* @param [in] K Stiffness coefficient
* @param [in] F Dragging six-dimensional force threshold
* @param [in] Fmax Maximum dragging force limit in Nm
* @param [in] Vmax Maximum joint speed limit in °/s
* @return Error code
*/
int EndForceDragControl(int status, int asaptiveFlag, int interfereDragFlag,int␣
↪ingularityConstraintsFlag, int forceCollisionFlag, Object[] M, Object[] B, Object[] K,␣
↪Object[] F, double Fmax, double Vmax)
```

2.3.12.36 Get Force Sensor Dragging Switch Status

```
/**
* @brief Get force sensor dragging switch status
* @return List[0]: Error code; List[1]: dragState Force sensor assisted dragging control status, 0-Disable; 1-Enable; List[1]: sixDimensionalDragState Six-dimensional force assisted dragging control status, 0-Disable; 1-Enable
*/
List<Integer> GetForceAndTorqueDragState();
```

2.3.12.37 Force Sensor Auto-Enable After Error Clearance

```
/**
* @brief Force sensor auto-enable after error clearance
* @param [in] status Control status, 0-Disable; 1-Enable
* @return Error code
*/
int SetForceSensorDragAutoFlag(int status)
```

2.3.12.38 Force Sensor Assisted Dragging Example

```
public static int TestEndForceDragCtrl(Robot robot)
{
    DescTran tr1=new DescTran(0,0,0);
    robot.SetForceSensorPayload(0);
    robot.SetForceSensorPayloadCog(tr1);

    robot.SetForceSensorDragAutoFlag(1);

    Object[] M =new Object[] { 15.0, 15.0, 15.0, 0.5, 0.5, 0.1 };
    Object[] B =new Object[] { 150.0, 150.0, 150.0, 5.0, 5.0, 1.0 };
    Object[] K =new Object[] { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
    Object[] F =new Object[] { 10.0, 10.0, 10.0, 1.0, 1.0, 1.0 };
    robot.EndForceDragControl(1, 0, 0, 0, M, B, K, F, 50, 100);

    robot.Sleep(10000);

    int dragState = 0;
    int sixDimensionalDragState = 0;
    List<Integer> state=new ArrayList<>();
    state=robot.GetForceAndTorqueDragState();

    robot.EndForceDragControl(0, 0, 0, 0, M, B, K, F, 50, 100);
    return 0;
}
```

2.3.12.39 Set Six-Dimensional Force and Joint Impedance Hybrid Dragging Switch and Parameters

```
/**
* @brief Set six-dimensional force and joint impedance hybrid dragging switch and␣
↪parameters
* @param [in] status Control status, 0-Disable; 1-Enable
* @param [in] impedanceFlag Impedance enable flag, 0-Disable; 1-Enable
* @param [in] lamdeGain Dragging gain
* @param [in] KGain Stiffness gain
* @param [in] BGain Damping gain
* @param [in] dragMaxTcpVel Maximum end-effector linear velocity limit during dragging
* @param [in] dragMaxTcpOriVel Maximum end-effector angular velocity limit during␣
↪dragging
* @return Error code
*/
int ForceAndJointImpedanceStartStop(int status, int impedanceFlag, Object[] lamdeGain,␣
↪Object[] KGain, Object[] BGain, double dragMaxTcpVel, double dragMaxTcpOriVel);
```

2.3.12.40 Six-Dimensional Force and Joint Impedance Hybrid Dragging Example

```
public static int TestForceAndJointImpedance(Robot robot)
{
    robot.DragTeachSwitch(1);
    Object[] lamdeDain =new Object[] { 3.0, 2.0, 2.0, 2.0, 2.0, 3.0 };
    Object[] KGain = new Object[]{ 0, 0, 0, 0, 0, 0 };
    Object[] BGain =new Object[] { 150, 150, 150, 5.0, 5.0, 1.0 };
    int rtn = robot.ForceAndJointImpedanceStartStop(1, 0, lamdeDain, KGain, BGain, 1000.
↪0, 180.0);

    robot.Sleep(10000);

    robot.DragTeachSwitch(0);
    rtn = robot.ForceAndJointImpedanceStartStop(0, 0, lamdeDain, KGain, BGain, 1000.0,␣
↪180.0);

    robot.CloseRPC();
    return 0;
}
```

2.3.12.41 Example Program

```
private static void TestUDPWireSearch(Robot robot)
{
    UDPComParam param = new UDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100,␣
↪10,0);
    robot.ExtDevSetUDPComParam(param);//UDP extended axis communication

    robot.SetWireSearchExtDIONum(0, 0);

    int rtn0, rtn1, rtn2 = 0;
```

(continues on next page)

(continued from previous page)

```
    ExaxisPos exaxisPos = new ExaxisPos(0.0, 0.0, 0.0, 0.0);
    DescPose offdese = new DescPose(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);

    DescPose descStart = new DescPose(-158.767, -510.596, 271.709, -179.427, -0.745, -137.349);
    JointPos jointStart = new JointPos(61.667, -79.848, 108.639, -119.682, -89.700, -70.985);

    DescPose descEnd = new DescPose(0.332, -516.427, 270.688, 178.165, 0.017, -119.989);
    JointPos jointEnd = new JointPos(79.021, -81.839, 110.752, -118.298, -91.729, -70.981);

    robot.MoveL(jointStart, descStart, 1, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 0, 100);
    robot.MoveL(jointEnd, descEnd, 1, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 0,␣
↪100);

    DescPose descREF0A = new DescPose(-66.106, -560.746, 270.381, 176.479, -0.126, -126.745);
    JointPos jointREF0A = new JointPos(73.531, -75.588, 102.941, -116.250, -93.347, -69.689);

    DescPose descREF0B = new DescPose(-66.109, -528.440, 270.407, 176.479, -0.129, -126.744);
    JointPos jointREF0B = new JointPos(72.534, -79.625, 108.046, -117.379, -93.366, -70.687);

    DescPose descREF1A = new DescPose(72.975, -473.242, 270.399, 176.479, -0.129, -126.744);
    JointPos jointREF1A = new JointPos(87.169, -86.509, 115.710, -117.341, -92.993, -56.034);

    DescPose descREF1B = new DescPose(31.355, -473.238, 270.405, 176.480, -0.130, -126.745);
    JointPos jointREF1B = new JointPos(82.117, -87.146, 116.470, -117.737, -93.145, -61.090);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF0A, descREF0A, 1, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 0, 100);  //Starting point
    robot.MoveL(jointREF0B, descREF0B, 1, 0, 10, 100, 100, -1, exaxisPos, 1, 0, offdese,␣
↪0, 100);  //Direction point
    rtn1 = robot.WireSearchWait("REF0");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF1A, descREF1A, 1, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 0, 100);  //Starting point
    robot.MoveL(jointREF1B, descREF1B, 1, 0, 10, 100, 100, -1, exaxisPos, 1, 0, offdese,␣
↪0, 100);  //Direction point
    rtn1 = robot.WireSearchWait("REF1");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);
```

(continues on next page)

(continued from previous page)

```
    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF0A, descREF0A, 1, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 0, 100);  //Starting point
    robot.MoveL(jointREF0B, descREF0B, 1, 0, 10, 100, 100, -1, exaxisPos, 1, 0, offdese,0, 100);  //Direction point
    rtn1 = robot.WireSearchWait("RES0");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF1A, descREF1A, 1, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 0, 100);  //Starting point
    robot.MoveL(jointREF1B, descREF1B, 1, 0, 10, 100, 100, -1, exaxisPos, 1, 0, offdese,0, 100);  //Direction point

    rtn1 = robot.WireSearchWait("RES1");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    String[] varNameRef = {"REF0", "REF1", "#", "#", "#", "#"};
    String[] varNameRes = {"RES0", "RES1", "#", "#", "#", "#"};
    int offectFlag = 0;
    //DescPose offectPos = new DescPose(0, 0, 0, 0, 0, 0);
    DescOffset offset = new DescOffset();
    rtn0 = robot.GetWireSearchOffset(0, 0, varNameRef, varNameRes, offset);
    robot.PointsOffsetEnable(0, offset.offset);
    robot.MoveL(jointStart, descStart, 1, 0, 100, 100, 100, -1, exaxisPos, 0, 0, offdese, 0, 100);
    robot.MoveL(jointEnd, descEnd, 1, 0, 100, 100, 100, -1, exaxisPos, 1, 0, offdese, 0,100);
    robot.PointsOffsetDisable();
}
```

2.3.12.42 Impedance Control Start/Stop

```
/**
* @brief Impedance Control Start/Stop
* @param [in] status 0: disable; 1-enable
* @param [in] workSpace 0-joint space; 1-Cartesian space
* @param [in] forceThreshold Trigger force threshold (N)
* @param [in] m Mass parameter
* @param [in] b Damping parameter
* @param [in] k Stiffness parameter
* @param [in] maxV Max linear velocity (mm/s)
* @param [in] maxVA Max linear acceleration (mm/s2)
* @param [in] maxW Max angular velocity (°/s)
* @param [in] maxWA Max angular acceleration (°/s2)
* @return Error code
*/
public int ImpedanceControlStartStop(int status, int workSpace, double[] forceThreshold,double[] m, double[] b, double[] k, double maxV, double maxVA, double maxW, double maxWA)
```

2.3.12.43 Robot Impedance Control Start/Stop Code Example

```
public static int TestImpedanceControl(Robot robot)
{
    JointPos j1=new JointPos(102.622, -135.990, 120.769, -73.950, -90.848, 35.507);
    JointPos j2=new JointPos(93.674, -80.062, 82.947, -92.199, -90.967, 26.559);
    DescPose desc_pos1=new DescPose(136.552, -149.799, 449.532, 179.817, -1.172, 157.
↪123);
    DescPose desc_pos2=new DescPose(136.540, -561.048, 449.542, 179.819, -1.172, 157.
↪122);
    DescPose offset_pos=new DescPose(0, 0, 0, 0, 0, 0);
    ExaxisPos epos=new ExaxisPos(0, 0, 0, 0);
    int tool = 0;
    int user = 0;
    double vel = 100.0;
    double acc = 200.0;
    double ovl = 100.0;
    double blendT = -1.0;
    double blendR = -1.0;
    int flag = 0;
    int search = 0;
    robot.SetSpeed(20);
    int company = 17;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    DeviceConfig con=new DeviceConfig(company, device, softversion, bus);
    robot.FT_SetConfig(con);
    robot.Sleep(1000);
    robot.FT_GetConfig(con);
    System.out.println("FT config:"+con.company+","+con.device+","+con.softwareVersion+
↪"con"+ con.bus);
    robot.Sleep(1000);
    robot.FT_Activate(0);
    robot.Sleep(1000);
    robot.FT_Activate(1);
    robot.Sleep(1000);
    robot.Sleep(1000);
    robot.FT_SetZero(0);
    robot.Sleep(1000);
    robot.FT_SetZero(1);
    robot.Sleep(1000);
    double[] forceThreshold = { 30,30,30,5,5,5 };
    double[] m= { 0.1,0.1,0.1,0.02,0.02,0.02 };
    double[] b = { 1,1,1,0.08,0.08,0.08 };
    double[] k = { 0,0,0,0,0,0 };
    int rtn = robot.ImpedanceControlStartStop(1, 1, forceThreshold, m, b, k, 1000, 500,␣
↪100, 100);
    System.out.println("ImpedanceControlStartStop errcode:"+ rtn);
    rtn = robot.MoveL(desc_pos1, tool, user, vel, acc, ovl, blendR, 0, epos, search,␣
↪flag, offset_pos, -1,0,-1, 1);
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, 0, epos, search,␣
↪flag, offset_pos, -1,0,-1, 1);
```

(continues on next page)

(continued from previous page)

```
    rtn = robot.MoveL(desc_pos1, tool, user, vel, acc, ovl, blendR, 0, epos, search, flag, offset_pos, -1,0,-1, 1);
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, 0, epos, search, flag, offset_pos, -1,0,-1, 1);
    rtn = robot.MoveL(desc_pos1, tool, user, vel, acc, ovl, blendR, 0, epos, search, flag, offset_pos, -1,0,-1, 1);
    rtn = robot.MoveL(desc_pos2, tool, user, vel, acc, ovl, blendR, 0, epos, search, flag, offset_pos, -1,0,-1, 1);
    System.out.println("movel errcode:"+ rtn);
    robot.ImpedanceControlStartStop(0, 1, forceThreshold, m, b, k, 1000, 500, 100, 100);
    robot.CloseRPC();
    return 0;
}
```

2.3.12.44 Enable Torque Compensation Function and Compensation Coefficients

```
/**
* @brief Enable Torque Compensation Function and Compensation Coefficients
* @param  status Switch, 0-Disable; 1-Enable
* @param  torqueCoeff J1-J6 Torque compensation coefficients [0-1]
* @return Error code
*/
public int SerCoderCompenParams(int status, double[] torqueCoeff)
```

2.3.13 Extended Axis

2.3.13.1 Set 485 Extended Axis Parameters

```
/**
* @brief Set 485 extended axis parameters
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [in] param 485 extended axis parameters
* @return Error code
*/
int AuxServoSetParam(int servoId, Axis485Param param)
```

2.3.13.2 Get 485 Extended Axis Parameters

```
/**
* @brief Get 485 extended axis configuration parameters
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [out] param 485 extended axis parameters
* @return Error code
*/
int AuxServoGetParam(int servoId, Axis485Param param);
```

2.3.13.3 Set 485 Extended Axis Enable/Disable

```
/**
* @brief Set 485 extended axis enable/disable
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [in] status Enable status, 0-disable, 1-enable
* @return Error code
*/
int AuxServoEnable(int servoId, int status);
```

2.3.13.4 Set 485 Extended Axis Control Mode

```
/**
* @brief Set 485 extended axis control mode
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [in] mode Control mode, 0-position mode, 1-speed mode
* @return Error code
*/
int AuxServoSetControlMode(int servoId, int mode);
```

2.3.13.5 Set 485 Extended Axis Target Position (Position Mode)

```
/**
* @brief Set 485 extended axis target position (position mode)
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [in] pos Target position, mm or °
* @param [in] speed Target speed, mm/s or °/s
* @param [in] acc Acceleration percentage [0-100]
* @return Error code
*/
int AuxServoSetTargetPos(int servoId, double pos, double speed, double acc);
```

2.3.13.6 Set 485 Extended Axis Target Torque (Torque Mode) - Not Yet Available

```
/**
* @brief Set 485 extended axis target torque (torque mode) - Not yet available
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [in] torque Target torque, Nm
* @return Error code
*/
int AuxServoSetTargetTorque(int servoId, double torque);
```

2.3.13.7 Set 485 Extended Axis

```
/**
* @brief Set 485 extended axis homing
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [in] mode Homing mode, 1-current position homing; 2-negative limit homing; 3-positive limit homing
* @param [in] searchVel Homing speed, mm/s or °/s
* @param [in] latchVel Latch speed, mm/s or °/s
* @param [in] acc Acceleration percentage [0-100]
* @return Error code
*/
int AuxServoHoming(int servoId, int mode, double searchVel, double latchVel);
```

2.3.13.8 Clear 485 Extended Axis Error Information

```
/**
* @brief Clear 485 extended axis error information
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @return Error code
*/
int AuxServoClearError(int servoId);
```

2.3.13.9 Get 485 Extended Axis Servo Status

```
/**
* @brief Get 485 extended axis servo status
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [in] servoErrCode Servo driver error code
* @param [in] servoState Servo driver status bit0:0-disabled; 1-enabled; bit1:0-not moving; 1-moving; bit4 0-positioning not completed; 1-positioning completed; bit5:0-homing not completed; 1-homing completed
* @param [in] servoPos Servo current position mm or °
* @param [in] servoSpeed Servo current speed mm/s or °/s
* @param [in] servoTorque Servo current torque Nm
* @return Error code
*/
int AuxServoGetStatus(int servoId, int[] servoErrCode, int[] servoState, double[] servoPos, double[] servoSpeed, double[] servoTorque)
```

2.3.13.10 Set 485 Extended Axis Target Speed (Speed Mode)

```
/**
* @brief Set 485 extended axis target speed (speed mode)
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @param [in] speed Target speed, mm/s or °/s
* @param [in] acc Acceleration percentage [0-100]
* @return Error code
*/
int AuxServoSetTargetSpeed(int servoId, double speed, double acc);
```

2.3.13.11 Set Axis Number for Status Feedback of 485 Extended Axis

```
/**
* @brief Set axis number for status feedback of 485 extended axis
* @param [in] servoId Servo driver ID, range [1-16], corresponding to slave ID
* @return Error code
*/
int AuxServosetStatusID(int servoId);
```

2.3.13.12 Set 485 Extended Axis Motion Acceleration/Deceleration

```
/**
* @brief Set 485 extended axis motion acceleration/deceleration
* @param [in] acc 485 extended axis motion acceleration
* @param [in] dec 485 extended axis motion deceleration
* @return Error code
*/
int AuxServoSetAcc(double acc, double dec)
```

2.3.13.13 Set 485 Extended Axis Emergency Stop Acceleration/Deceleration

```
/**
* @brief Set 485 extended axis emergency stop acceleration/deceleration
* @param [in] acc 485 extended axis emergency stop acceleration
* @param [in] dec 485 extended axis emergency stop deceleration
* @return Error code
*/
int AuxServoSetEmergencyStopAcc(double acc, double dec)
```

2.3.13.14 Get 485 Extended Axis Motion Acceleration/Deceleration

```
/**
* @brief Get 485 extended axis motion acceleration/deceleration
* @return List[0]: Error code; List[1]: 485 extended axis motion acceleration; List[2]: 485 extended axis motion deceleration
*/
List<Number> AuxServoGetAcc()
```

2.3.13.15 Get 485 Extended Axis Emergency Stop Acceleration/Deceleration

```
/**
* @brief Get 485 extended axis emergency stop acceleration/deceleration
* @return List[0]: Error code; List[1]: 485 extended axis emergency stop acceleration; List[2]: 485 extended axis emergency stop deceleration
*/
List<Number> AuxServoGetEmergencyStopAcc()
```

2.3.13.16 Extended Axis Control Code Example

```
public static int Test485Auxservo(Robot robot)
{
    Axis485Param ax=new Axis485Param(1, 1, 1, 131072, 15.45);
    int retval = robot.AuxServoSetParam(1, ax);

    Axis485Param ax2=new Axis485Param();
    retval = robot.AuxServoGetParam(1, ax2);

    ax.servoCompany=10;
    ax.servoModel=11;
    ax.servoSoftVersion=12;
    ax.servoResolution=13;
    ax.axisMechTransRatio=14;

    retval = robot.AuxServoSetParam(1, ax);

    retval = robot.AuxServoGetParam(1,ax2);

    ax.servoCompany=1;
    ax.servoModel=1;
    ax.servoSoftVersion=1;
    ax.servoResolution=131072;
    ax.axisMechTransRatio=36;

    retval = robot.AuxServoSetParam(1, ax);
    robot.Sleep(3000);

    robot.AuxServoSetAcc(3000, 3000);
    robot.AuxServoSetEmergencyStopAcc(5000, 5000);
    robot.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
    double emagacc = 0, acc = 0;
    double emagdec = 0, dec = 0;

    List<Number> aux=new ArrayList<>();

    aux=robot.AuxServoGetEmergencyStopAcc();
    aux=robot.AuxServoGetAcc();

    robot.AuxServoSetControlMode(1, 0);
    robot.Sleep(2000);

    retval = robot.AuxServoEnable(1, 0);
    robot.Sleep(1000);
    int[] servoerrcode =new int[]{0};
    int[] servoErrCode=new int[]{0};
    int[] servoState=new int[]{0};
    double[] servoPos=new double[]{0};
    double[] servoSpeed=new double[]{0};
    double[] servoTorque=new double[]{0};
    retval = robot.AuxServoGetStatus(1, servoErrCode, servoState, servoPos, servoSpeed,␣
↪servoTorque);
    robot.Sleep(1000);;

    retval = robot.AuxServoEnable(1, 1);
    robot.Sleep(1000);
    retval = robot.AuxServoGetStatus(1, servoErrCode, servoState, servoPos, servoSpeed,␣
↪servoTorque);
    robot.Sleep(1000);

    retval = robot.AuxServoHoming(1, 1, 5, 1,100);
    robot.Sleep(3000);

    retval = robot.AuxServoSetTargetPos(1, 200, 30,100);
    robot.Sleep(1000);
    retval = robot.AuxServoGetStatus(1, servoErrCode, servoState, servoPos, servoSpeed,␣
↪servoTorque);
    robot.Sleep(8000);


    robot.AuxServoSetControlMode(1, 1);
    robot.Sleep(2000);

    robot.AuxServoEnable(1, 0);
    robot.Sleep(1000);
    robot.AuxServoEnable(1, 1);
    robot.Sleep(1000);
    robot.AuxServoSetTargetSpeed(1, 100, 80);

    robot.Sleep(5000);
    robot.AuxServoSetTargetSpeed(1, 0, 80);

    robot.CloseRPC();
```

(continues on next page)

(continued from previous page)

```
    return 0;
}
```

2.3.13.17 UDP Extended Axis Communication Parameter Configuration

Changed in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief UDP extended axis communication parameter configuration
* @param [in] param Communication parameters
* @return Error code
*/
int ExtDevSetUDPComParam(UDPComParam param);
```

2.3.13.18 Get UDP Extended Axis Communication Parameter Configuration

```
/**
* @brief Get UDP extended axis communication parameters
* @param [out] param Communication parameters
* @return Error code
*/
int ExtDevGetUDPComParam(UDPComParam param);
```

2.3.13.19 Load UDP Communication

```
/**
* @brief Load UDP communication
* @return Error code
*/
int ExtDevLoadUDPDriver();
```

2.3.13.20 Unload UDP Communication

```
/**
* @brief Unload UDP communication
* @return Error code
*/
int ExtDevUnloadUDPDriver();
```

2.3.13.21 Reconnect After UDP Extended Axis Communication Exception Disconnection

```
/**
* @brief Reconnect after UDP extended axis communication exception disconnection
* @return Error code
*/
int ExtDevUDPClientComReset();
```

2.3.13.22 Close Communication After UDP Extended Axis Communication Exception Disconnection

```
/**
* @brief Close communication after UDP extended axis communication exception␣
↪disconnection
* @return Error code
*/
int ExtDevUDPClientComClose();
```

2.3.13.23 UDP Extended Axis Parameter Configuration

```
/**
* @brief UDP extended axis parameter configuration
* @param [in] axisID Axis number
* @param [in] axisType Extended axis type 0-translation; 1-rotation
* @param [in] axisDirection Extended axis direction 0-forward; 1-reverse
* @param [in] axisMax Extended axis maximum position mm
* @param [in] axisMin Extended axis minimum position mm
* @param [in] axisVel Speed mm/s
* @param [in] axisAcc Acceleration mm/s2
* @param [in] axisLead Lead mm
* @param [in] encResolution Encoder resolution
* @param [in] axisOffect Weld starting point extended axis offset
* @param [in] axisCompany Driver manufacturer 1-Hechuan; 2-Huichuan; 3-Panasonic
* @param [in] axisModel Driver model 1-Hechuan-SV-XD3EA040L-E, 2-Hechuan-SV-X2EA150A-A,␣
↪1-Huichuan-SV620PT5R4I, 1-Panasonic-MADLN15SG, 2-Panasonic-MSDLN25SG, 3-Panasonic-
↪MCDLN35SG
* @param [in] axisEncType Encoder type 0-incremental; 1-absolute
* @return Error code
*/
int ExtAxisParamConfig(int axisID, int axisType, int axisDirection, double axisMax,␣
↪double axisMin, double axisVel, double axisAcc, double axisLead, int encResolution,␣
↪double axisOffect, int axisCompany, int axisModel, int axisEncType);
```

2.3.13.24 Set Extended Axis Installation Position

```
/**
* @brief Set extended axis installation position
* @param [in] installType 0-robot installed on external axis, 1-robot installed outside external axis
* @return Error code
*/
int SetRobotPosToAxis(int installType);
```

2.3.13.25 Set Extended Axis System DH Parameter Configuration

```
/**
* @brief Set extended axis system DH parameter configuration
* @param [in]  axisConfig External axis configuration, 0-single degree of freedom linear slide, 1-two degree of freedom L-type positioner, 2-three degree of freedom, 3-four degree of freedom, 4-single degree of freedom positioner
* @param [in]  axisDHd1 External axis DH parameter d1 mm
* @param [in]  axisDHd2 External axis DH parameter d2 mm
* @param [in]  axisDHd3 External axis DH parameter d3 mm
* @param [in]  axisDHd4 External axis DH parameter d4 mm
* @param [in]  axisDHa1 External axis DH parameter a1 mm
* @param [in]  axisDHa2 External axis DH parameter a2 mm
* @param [in]  axisDHa3 External axis DH parameter a3 mm
* @param [in]  axisDHa4 External axis DH parameter a4 mm
* @return Error code
*/
int SetAxisDHParaConfig(int axisConfig, double axisDHd1, double axisDHd2, double axisDHd3, double axisDHd4, double axisDHa1, double axisDHa2, double axisDHa3, double axisDHa4);
```

2.3.13.26 UDP Extended Axis Enable

```
/**
* @brief UDP extended axis enable
* @param [in] axisID Axis number [1-4]
* @param [in] status 0-disable; 1-enable
* @return Error code
*/
int ExtAxisServoOn(int axisID, int status);
```

2.3.13.27 UDP Extended Axis Homing

```
/**
* @brief UDP extended axis homing
* @param [in] axisID Axis number [1-4]
* @param [in] mode Homing method 0-current position homing, 1-negative limit homing, 2-positive limit homing
* @param [in] searchVel Homing speed (mm/s)
* @param [in] latchVel Homing latch speed (mm/s)
* @return Error code
*/
int ExtAxisSetHoming(int axisID, int mode, double searchVel, double latchVel);
```

2.3.13.28 UDP Extended Axis Jog Start

```
/**
* @brief UDP extended axis jog start
* @param [in] axisID Axis number [1-4]
* @param [in] direction Rotation direction 0-reverse; 1-forward
* @param [in] vel Speed (mm/s)
* @param [in] acc Acceleration (mm/s2)
* @param [in] maxDistance Maximum jog distance
* @return Error code
*/
int ExtAxisStartJog(int axisID, int direction, double vel, double acc, double maxDistance);
```

2.3.13.29 UDP Extended Axis Jog Stop

```
/**
* @brief UDP extended axis jog stop
* @param [in] axisID Axis number [1-4]
* @return Error code
*/
int ExtAxisStopJog(int axisID);
```

2.3.13.30 UDP Extended Axis Configuration and Jog Code Example

```
public static int TestUDPAxis(Robot robot)//UDP
{
    UDPComParam para1=new UDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100, 5, 1);
    int rtn = robot.ExtDevSetUDPComParam(para1);
    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;
    UDPComParam para2=new UDPComParam(ip, port, period, lossPkgTime, lossPkgNum, disconnectTime, reconnectEnable, reconnectPeriod, reconnectNum,0);
```

(continues on next page)

(continued from previous page)

```
    rtn = robot.ExtDevGetUDPComParam(para2);

    robot.ExtDevLoadUDPDriver();

    rtn = robot.ExtAxisServoOn(1, 1);
    rtn = robot.ExtAxisServoOn(2, 1);
    robot.Sleep(3000);

    robot.ExtAxisSetHoming(1, 0, 10, 2);
    robot.Sleep(3000);
    rtn = robot.ExtAxisSetHoming(2, 0, 10, 2);

    robot.Sleep(4000);

    rtn = robot.SetRobotPosToAxis(1);
    rtn = robot.SetAxisDHParaConfig(10, 20, 0, 0, 0, 0, 0, 0, 0);
    rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905, 262144, 200,␣
↪1, 0, 0);
    rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444, 262144, 200,␣
↪1, 0, 0);

    robot.Sleep(4000);
    robot.ExtAxisStartJog(1, 0, 10, 10, 30);
    robot.Sleep(4000);
    robot.ExtAxisStopJog(1);
    robot.Sleep(4000);
    robot.ExtAxisServoOn(1, 0);

    robot.Sleep(4000);
    robot.ExtAxisStartJog(2, 0, 10, 10, 30);
    robot.Sleep(4000);
    robot.ExtAxisStopJog(2);
    robot.Sleep(4000);
    robot.ExtAxisServoOn(2, 0);
    robot.Sleep(4000);
    robot.ExtDevUnloadUDPDriver();

    return 0;
}
```

2.3.13.31 Set Extended Axis Coordinate System Reference Point - Four-Point Method

```
/**
* @brief Set extended axis coordinate system reference point - four-point method
* @param [in]  pointNum Point number [1-4]
* @return Error code
*/
int ExtAxisSetRefPoint(int pointNum);
```

2.3.13.32 Calculate Extended Axis Coordinate System - Four-Point Method

```
/**
* @brief Calculate extended axis coordinate system - four-point method
* @param [out]  coord Coordinate system value
* @return Error code
*/
int ExtAxisComputeECoordSys(DescPose coord);
```

2.3.13.33 Positioner Coordinate System Reference Point Setting

```
/**
* @brief Positioner coordinate system reference point setting
* @param [in]  pointNum Point number [1-4]
* @return Error code
*/
int PositionorSetRefPoint(int pointNum);
```

2.3.13.34 Positioner Coordinate System Calculation - Four-Point Method

```
/**
* @brief Positioner coordinate system calculation - four-point method
* @param [out] coord Coordinate system value
* @return Error code
*/
int PositionorComputeECoordSys(DescPose coord);
```

2.3.13.35 Set Calibration Reference Point Pose in Extended Axis End Coordinate System

```
/**
* @brief Set calibration reference point pose in extended axis end coordinate system
* @param [in] pos Pose value
* @return Error code
*/
int SetRefPointInExAxisEnd(DescPose pos);
```

2.3.13.36 Apply Extended Axis Coordinate System

```
/**
* @brief Apply extended axis coordinate system
* @param [in]  applyAxisId Extended axis number bit0-bit3 corresponds to extended axis␣
↪number 1-4, e.g., to apply extended axis 1 and 3, it is 0b 0000 0101; which is 5
* @param [in]  axisCoordNum Extended axis coordinate system number
* @param [in]  coord Coordinate system value
* @param [in]  calibFlag Calibration flag 0-no, 1-yes
* @return Error code
*/
```

(continues on next page)

(continued from previous page)

```
int ExtAxisActiveECoordSys(int applyAxisId, int axisCoordNum, DescPose coord, int␣
↪calibFlag);
```

2.3.13.37 Get Extended Axis Coordinate System

New in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Get extended axis coordinate system
* @param [out] coord Extended axis coordinate system
* @return Error code
*/
int ExtAxisGetCoord(DescPose coord);
```

2.3.13.38 Extended Axis Coordinate System Calibration Code Example

```
public static int TestUDPAxisCalib(Robot robot)
{
    UDPComParam para1=new UDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100, 5,␣
↪1);

    int rtn = robot.ExtDevSetUDPComParam(para1);
    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;
    UDPComParam para2=new UDPComParam(ip, port, period, lossPkgTime, lossPkgNum,␣
↪disconnectTime, reconnectEnable, reconnectPeriod, reconnectNum,0);

    rtn = robot.ExtDevGetUDPComParam(para2);

    robot.ExtDevLoadUDPDriver();

    rtn = robot.ExtAxisServoOn(1, 1);
    rtn = robot.ExtAxisServoOn(2, 1);

    robot.Sleep(4000);

    rtn = robot.SetRobotPosToAxis(1);
    rtn = robot.SetAxisDHParaConfig(1, 128.5, 206.4,  0, 0, 0, 0, 0, 0);
    rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905, 262144, 200,␣
↪1, 0, 0);
    rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444, 262144, 200,␣
↪1, 0, 0);

    DescPose toolCoord=new DescPose(0, 0, 210, 0, 0, 0);
    robot.SetToolCoord(1, toolCoord, 0, 0, 1, 0);

    JointPos jSafe=new JointPos(115.193, -96.149, 92.489, -87.068, -89.15, -83.488);
    JointPos j1=new JointPos(117.559, -92.624, 100.329, -96.909, -94.057, -83.488);
```

(continues on next page)

(continued from previous page)

```
    JointPos j2=new JointPos(112.239, -90.096, 99.282, -95.909, -89.824, -83.488);
    JointPos j3=new JointPos(110.839, -83.473, 93.166, -89.22, -90.499, -83.487);
    JointPos j4=new JointPos(107.935, -83.572, 95.424, -92.873, -87.933, -83.488);

    DescPose descSafe =new DescPose(0,0,0,0,0,0);
    DescPose desc1 = new DescPose(0,0,0,0,0,0);
    DescPose desc2 = new DescPose(0,0,0,0,0,0);
    DescPose desc3 = new DescPose(0,0,0,0,0,0);
    DescPose desc4 = new DescPose(0,0,0,0,0,0);
    ExaxisPos exaxisPos =new ExaxisPos(0,0,0,0);
    DescPose offdese =new DescPose(0, 0, 0, 0, 0, 0);

    robot.GetForwardKin(jSafe, descSafe);
    robot.MoveJ(jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.Sleep(2000);

    robot.GetForwardKin(j1, desc1);
    robot.MoveJ(j1, desc1, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.Sleep(2000);

    DescPose actualTCPPos =new DescPose(0,0,0,0,0,0);

    robot.GetActualTCPPose(actualTCPPos);
    robot.SetRefPointInExAxisEnd(actualTCPPos);
    rtn = robot.PositionorSetRefPoint(1);
    robot.Sleep(2000);

    robot.MoveJ(jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.ExtAxisStartJog(1, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.GetForwardKin(j2, desc2);
    rtn = robot.MoveJ(j2, desc2, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    rtn = robot.PositionorSetRefPoint(2);
    robot.Sleep(2000);

    robot.MoveJ(jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.ExtAxisStartJog(1, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.GetForwardKin(j3, desc3);
    robot.MoveJ(j3, desc3, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    rtn = robot.PositionorSetRefPoint(3);
    robot.Sleep(2000);

    robot.MoveJ(jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    robot.ExtAxisStartJog(1, 0, 50, 50, 10);
    robot.Sleep(1000);
    robot.ExtAxisStartJog(2, 0, 50, 50, 10);
    robot.Sleep(1000);
```

(continues on next page)

(continued from previous page)

```
    robot.GetForwardKin(j4, desc4);
    robot.MoveJ(j4, desc4, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    rtn = robot.PositionorSetRefPoint(4);
    robot.Sleep(2000);

    DescPose axisCoord = new DescPose();
    robot.PositionorComputeECoordSys(axisCoord);
    robot.MoveJ(jSafe, descSafe, 1, 0, 100, 100, 100, exaxisPos, -1, 0, offdese);
    rtn = robot.ExtAxisActiveECoordSys(3, 1, axisCoord, 1);

    robot.CloseRPC();
    return 0;
}
```

2.3.13.39 UDP Extended Axis Motion

Changed in version Java: SDK-v1.0.7-3.8.4

```
/**
*@ brief UDP Extended Axis Motion
*@ param [in] pos target location
*@ param [in] ovl speed percentage
*@ param [in] blend smoothing parameter (mm or ms)
*@ return error code
*/
int ExtAxisMove(ExaxisPos pos, double ovl, double blend);
```

2.3.13.40 UDP Extended Axis Motion Code Example

```
public static int TestUDPAxisCalib(Robot robot)
{
    ExaxisPos exaxisPos = new ExaxisPos( 20, 0, 0, 0 );
    robot.ExtAxisMove(exaxisPos,40);
    robot.CloseRPC();
    return 0;
}
```

2.3.13.41 UDP Extended Axis and Robot Joint Motion Synchronous Motion

```
/**
* @brief UDP extended axis and robot joint motion synchronous motion
* @param [in] joint_pos  Target joint position, unit deg
* @param [in] desc_pos   Target Cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Speed percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not yet available
* @param [in] ovl  Speed scaling factor, range [0~100]
```

(continues on next page)

(continued from previous page)

```
* @param [in] epos  Extended axis position, unit mm
* @param [in] blendT [-1.0]-motion(blocking), [0~500.0]-smoothing time (non-blocking), unit ms
* @param [in] offset_flag  0-no offset, 1-offset in base coordinate system/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] ffset_pos  Pose offset
* @return Error code
*/
int ExtAxisSyncMoveJ(JointPos joint_pos, DescPose desc_pos, int tool, int user, double vel, double acc, double ovl, ExaxisPos epos, double blendT, int offset_flag, DescPose offset_pos);
```

2.3.13.42 UDP extended axis and robot joint motion synchronous motion (automatic forward kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief UDP extended axis and robot joint motion synchronous motion (automatic forward kinematics calculation)
* @param [in] joint_pos  Target joint position, unit deg
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] epos  Extended axis position, unit mm
* @param [in] blendT [-1.0]-move to position (blocking), [0~500.0]-smoothing time (non-blocking), unit ms
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @return Error code
*/
int ExtAxisSyncMoveJ(JointPos joint_pos, int tool, int user, double vel, double acc, double ovl, ExaxisPos epos, double blendT, int offset_flag, DescPose offset_pos)
```

2.3.13.43 UDP Extended Axis and Robot Joint Motion Synchronous Motion Code Example

```
public int testSyncMoveJ(Robot robot)
{
    //1. Calibrate and apply robot tool coordinate system. You can use the four-point or six-point method for tool coordinate system calibration and application. The interfaces involved are as follows:
    //  int SetToolPoint(int point_num); //Set tool reference point - six-point method
    //  int ComputeTool(ref DescPose tcp_pose); //Calculate tool coordinate system
    //  int SetTcp4RefPoint(int point_num);  //Set tool reference point - four-point method
    //  int ComputeTcp4(ref DescPose tcp_pose);  //Calculate tool coordinate system -
```

(continues on next page)

(continued from previous page)

```
↪four-point method
    //  int SetToolCoord(int id, DescPose coord, int type, int install); //Set and apply␣
↪tool coordinate system
    //  int SetToolList(int id, DescPose coord, int type, int install);  //Set and apply␣
↪tool coordinate system list
    //2. Set UDP communication parameters and load UDP communication
    UDPComParam param=new UDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10,
↪0);
    robot.ExtDevSetUDPComParam(param);
    robot.ExtDevLoadUDPDriver();
    //3. Set extended axis parameters, including extended axis type, extended axis␣
↪driver parameters, and extended axis DH parameters
    robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner␣
↪and DH parameters
    robot.SetRobotPosToAxis(1); //Extended axis installation position
    robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //
↪Servo driver parameters. This example is for a single-axis positioner, so only one␣
↪driver parameter needs to be set. If you choose an extended axis type with multiple␣
↪axes, each axis needs to be configured.
    //4. Enable the selected axis and perform homing
    robot.ExtAxisServoOn(1, 0);
    robot.ExtAxisSetHoming(1, 0, 20, 3);
    //5. Perform extended axis coordinate system calibration and application
    DescPose pos = new DescPose(/* Enter your calibration point coordinates */ );
    robot.SetRefPointInExAxisEnd(pos);
    robot.PositionorSetRefPoint(1); /*You need to call this interface 4 times with␣
↪different points to complete the calibration */
    DescPose coord = new DescPose();
    robot.PositionorComputeECoordSys(coord); //Calculate extended axis calibration result
    robot.ExtAxisActiveECoordSys(1, 1, coord, 1); //Apply the calibration result to the␣
↪extended axis coordinate system
    //6. Calibrate the workpiece coordinate system on the extended axis. You will need␣
↪the following interfaces:
    //int SetWObjCoordPoint(int point_num);
    //int ComputeWObjCoord(int method, ref DescPose wobj_pose);
    //int SetWObjCoord(int id, DescPose coord);
    //int SetWObjList(int id, DescPose coord);
    //7. Record your synchronous joint motion starting point
    DescPose startdescPose = new DescPose(/*Enter your coordinates*/ );
    JointPos startjointPos = new JointPos(/*Enter your coordinates*/ );
    ExaxisPos startexaxisPos = new ExaxisPos(/* Enter your extended axis starting point␣
↪coordinates */ );
    //8. Record your synchronous joint motion end point coordinates
    DescPose enddescPose = new DescPose(/*Enter your coordinates*/ );
    JointPos endjointPos = new JointPos(/*Enter your coordinates*/ );
    ExaxisPos endexaxisPos =new ExaxisPos(/* Enter your extended axis end point␣
↪coordinates */);
    //9. Write the synchronous motion program
    //Move to the starting point, assuming the applied tool coordinate system and␣
↪workpiece coordinate system are both 1
    robot.ExtAxisMove(startexaxisPos, 20);
    DescPose offdese = new DescPose( 0, 0, 0, 0, 0, 0 );
```

(continues on next page)

(continued from previous page)

```
    robot.MoveJ(startjointPos, startdescPose, 1, 1, 100, 100, 100, startexaxisPos, 0, 0,␣
↪offdese);
    //Start synchronous motion
    robot.ExtAxisSyncMoveJ(endjointPos, enddescPose, 1, 1, 100, 100, 100, endexaxisPos, -
↪1, 0, offdese);
    robot.MoveJ(startjointPos, 1, 1, 100, 100, 100, startexaxisPos, 0, 0, offdese);
    //Start synchronous motion
    robot.ExtAxisSyncMoveJ(endjointPos, 1, 1, 100, 100, 100, endexaxisPos, -1, 0,␣
↪offdese);

    robot.CloseRPC();
    return 0;
}
```

2.3.13.44 UDP Extended Axis and Robot Linear Motion Synchronous Motion

```
/**
* @brief UDP extended axis and robot linear motion synchronous motion
* @param [in] joint_pos  Target joint position, unit deg
* @param [in] desc_pos   Target Cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Speed percentage, range [0~100]
* @param [in] acc  Acceleration percentage, range [0~100], not yet available
* @param [in] ovl  Speed scaling factor, range [0~100]
* @param [in] blendR [-1.0]-motion(blocking), [0~1000.0]-smoothing radius (non-blocking),
↪ unit mm
* @param [in] epos  Extended axis position, unit mm
* @param [in] offset_flag  0-no offset, 1-offset in base coordinate system/workpiece␣
↪coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @return Error code
*/
int ExtAxisSyncMoveL(JointPos joint_pos, DescPose desc_pos, int tool, int user, double␣
↪vel, double acc, double ovl, double blendR, ExaxisPos epos, int offset_flag, DescPose␣
↪offset_pos);
```

2.3.13.45 UDP extended axis and robot linear motion synchronous motion (automatic inverse kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief UDP extended axis and robot linear motion synchronous motion (automatic inverse␣
↪kinematics calculation)
* @param [in] desc_pos  Target cartesian pose
* @param [in] tool  Tool coordinate number, range [0~14]
* @param [in] user  Workpiece coordinate number, range [0~14]
* @param [in] vel  Velocity percentage, range [0~100]
```

(continues on next page)

(continued from previous page)

```
* @param [in] acc  Acceleration percentage, range [0~100], not open yet
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius␣
↪(non-blocking), unit mm
* @param [in] epos  Extended axis position, unit mm
* @param [in] offset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-
↪offset in tool coordinate system
* @param [in] offset_pos  Pose offset
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based␣
↪on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
int ExtAxisSyncMoveL(DescPose desc_pos, int tool, int user, double vel, double acc,␣
↪double ovl, double blendR, ExaxisPos epos, int offset_flag, DescPose offset_pos,int␣
↪config)
```

2.3.13.46 UDP Extended Axis and Robot Linear Motion Synchronous Motion Code Example

```
public int testSyncMoveL(Robot robot)
{
    //1. Calibrate and apply robot tool coordinate system. You can use the four-point or␣
↪six-point method for tool coordinate system calibration and application. The␣
↪interfaces involved are as follows:
    //  int SetToolPoint(int point_num); //Set tool reference point - six-point method
    //  int ComputeTool(ref DescPose tcp_pose); //Calculate tool coordinate system
    //  int SetTcp4RefPoint(int point_num);  //Set tool reference point - four-point␣
↪method
    //  int ComputeTcp4(ref DescPose tcp_pose);  //Calculate tool coordinate system -␣
↪four-point method
    //  int SetToolCoord(int id, DescPose coord, int type, int install); //Set and apply␣
↪tool coordinate system
    //  int SetToolList(int id, DescPose coord, int type, int install);  //Set and apply␣
↪tool coordinate system list
    //2. Set UDP communication parameters and load UDP communication
    UDPComParam param=new UDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10,
↪0);
    robot.ExtDevSetUDPComParam(param);
    robot.ExtDevLoadUDPDriver();
    //3. Set extended axis parameters, including extended axis type, extended axis␣
↪driver parameters, and extended axis DH parameters
    robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner␣
↪and DH parameters
    robot.SetRobotPosToAxis(1); //Extended axis installation position
    robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //
↪Servo driver parameters. This example is for a single-axis positioner, so only one␣
↪driver parameter needs to be set. If you choose an extended axis type with multiple␣
↪axes, each axis needs to be configured.
    //4. Enable the selected axis and perform homing
    robot.ExtAxisServoOn(1, 0);
    robot.ExtAxisSetHoming(1, 0, 20, 3);
```

(continues on next page)

(continued from previous page)

```
    //5. Perform extended axis coordinate system calibration and application
    DescPose pos = new DescPose(/* Enter your calibration point coordinates */ );
    robot.SetRefPointInExAxisEnd(pos);
    robot.PositionorSetRefPoint(1); /*You need to call this interface 4 times with␣
↪different points to complete the calibration */
    DescPose coord = new DescPose();
    robot.PositionorComputeECoordSys(coord); //Calculate extended axis calibration result
    robot.ExtAxisActiveECoordSys(1, 1, coord, 1); //Apply the calibration result to the␣
↪extended axis coordinate system
    //6. Calibrate the workpiece coordinate system on the extended axis. You will need␣
↪the following interfaces:
    //int SetWObjCoordPoint(int point_num);
    //int ComputeWObjCoord(int method, ref DescPose wobj_pose);
    //int SetWObjCoord(int id, DescPose coord);
    //int SetWObjList(int id, DescPose coord);
    //7. Record your synchronous joint motion starting point
    DescPose startdescPose = new DescPose(/*Enter your coordinates*/ );
    JointPos startjointPos = new JointPos(/*Enter your coordinates*/ );
    ExaxisPos startexaxisPos = new ExaxisPos(/* Enter your extended axis starting point␣
↪coordinates */ );
    //8. Record your synchronous joint motion end point coordinates
    DescPose enddescPose = new DescPose(/*Enter your coordinates*/ );
    JointPos endjointPos = new JointPos(/*Enter your coordinates*/ );
    ExaxisPos endexaxisPos =new ExaxisPos(/* Enter your extended axis end point␣
↪coordinates */);
    //9. Write the synchronous motion program
    //Move to the starting point, assuming the applied tool coordinate system and␣
↪workpiece coordinate system are both 1
    robot.ExtAxisMove(startexaxisPos, 20);
    DescPose offdese = new DescPose( 0, 0, 0, 0, 0, 0 );
    robot.MoveJ(startjointPos, startdescPose, 1, 1, 100, 100, 100, startexaxisPos, 0, 0,␣
↪offdese);
    //Start synchronous motion
    robot.ExtAxisSyncMoveL(endjointPos, enddescPose, 1, 1, 100, 100, 100, 0,␣
↪endexaxisPos, 0, offdese);
    robot.MoveJ(startjointPos, 1, 1, 100, 100, 100, startexaxisPos, 0, 0, offdese);
    //Start synchronous motion
    robot.ExtAxisSyncMoveL(enddescPose, 1, 1, 100, 100, 100, 0, endexaxisPos, 0, offdese,
↪-1);
    robot.CloseRPC();
    return 0;
}
```

2.3.13.47 UDP Extended Axis and Robot Arc Motion Synchronous Motion

```
/**
* @brief UDP extended axis and robot arc motion synchronous motion
* @param [in] joint_pos_p  Path point joint position, unit deg
* @param [in] desc_pos_p   Path point Cartesian pose
* @param [in] ptool  Tool coordinate number, range [0~14]
* @param [in] puser  Workpiece coordinate number, range [0~14]
* @param [in] pvel  Speed percentage, range [0~100]
* @param [in] pacc  Acceleration percentage, range [0~100], not yet available
* @param [in] epos_p  Intermediate point extended axis position, unit mm
* @param [in] poffset_flag  0-no offset, 1-offset in base coordinate system/workpiece␣
↪coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos_p  Pose offset
* @param [in] joint_pos_t  Target point joint position, unit deg
* @param [in] desc_pos_t   Target point Cartesian pose
* @param [in] ttool  Tool coordinate number, range [0~14]
* @param [in] tuser  Workpiece coordinate number, range [0~14]
* @param [in] tvel  Speed percentage, range [0~100]
* @param [in] tacc  Acceleration percentage, range [0~100], not yet available
* @param [in] epos_t  Extended axis position, unit mm
* @param [in] toffset_flag  0-no offset, 1-offset in base coordinate system/workpiece␣
↪coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos_t  Pose offset
* @param [in] ovl  Speed scaling factor, range [0~100]
* @param [in] blendR [-1.0]-motion(blocking), [0~1000.0]-smoothing radius (non-blocking),
↪ unit mm
* @return Error code
*/
int ExtAxisSyncMoveC(JointPos joint_pos_p, DescPose desc_pos_p, int ptool, int puser,␣
↪double pvel, double pacc, ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p,␣
↪JointPos joint_pos_t, DescPose desc_pos_t, int ttool, int tuser, double tvel, double␣
↪tacc, ExaxisPos epos_t, int toffset_flag, DescPose offset_pos_t, double ovl, double␣
↪blendR);
```

2.3.13.48 UDP extended axis and robot circular motion synchronous motion (automatic inverse kinematics calculation)

New in version Java: SDK-v1.0.8-3.8.5

```
/**
* @brief UDP extended axis and robot circular motion synchronous motion (automatic␣
↪inverse kinematics calculation)
* @param [in] desc_pos_p  Path point cartesian pose
* @param [in] ptool  Tool coordinate number, range [0~14]
* @param [in] puser  Workpiece coordinate number, range [0~14]
* @param [in] pvel  Velocity percentage, range [0~100]
* @param [in] pacc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos_p  Extended axis position, unit mm
* @param [in] poffset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-
↪offset in tool coordinate system
* @param [in] offset_pos_p  Pose offset
```

(continues on next page)

(continued from previous page)

```
* @param [in] desc_pos_t  Target point cartesian pose
* @param [in] ttool  Tool coordinate number, range [0~14]
* @param [in] tuser  Workpiece coordinate number, range [0~14]
* @param [in] tvel  Velocity percentage, range [0~100]
* @param [in] tacc  Acceleration percentage, range [0~100], not open yet
* @param [in] epos_t  Extended axis position, unit mm
* @param [in] toffset_flag  0-no offset, 1-offset in base/workpiece coordinate system, 2-offset in tool coordinate system
* @param [in] offset_pos_t  Pose offset
* @param [in] ovl  Velocity scaling factor, range [0~100]
* @param [in] blendR [-1.0]-move to position (blocking), [0~1000.0]-smoothing radius (non-blocking), unit mm
* @param [in] config Inverse kinematics joint space configuration, [-1]-calculate based on current joint position, [0~7]-solve according to specific joint space configuration
* @return Error code
*/
int ExtAxisSyncMoveC(DescPose desc_pos_p, int ptool, int puser, double pvel, double pacc, ExaxisPos epos_p, int poffset_flag, DescPose offset_pos_p, DescPose desc_pos_t, int ttool, int tuser, double tvel, double tacc, ExaxisPos epos_t, int toffset_flag, DescPose offset_pos_t, double ovl, double blendR,int config)
```

2.3.13.49 UDP Extended Axis and Robot Arc Motion Synchronous Motion Code Example

```
public int testSyncMoveC(Robot robot)
{
    //1. Calibrate and apply robot tool coordinate system. You can use the four-point or six-point method for tool coordinate system calibration and application. The interfaces involved are as follows:
    //  int SetToolPoint(int point_num); //Set tool reference point - six-point method
    //  int ComputeTool(ref DescPose tcp_pose); //Calculate tool coordinate system
    //  int SetTcp4RefPoint(int point_num);  //Set tool reference point - four-point method
    //  int ComputeTcp4(ref DescPose tcp_pose);  //Calculate tool coordinate system - four-point method
    //  int SetToolCoord(int id, DescPose coord, int type, int install); //Set and apply tool coordinate system
    //  int SetToolList(int id, DescPose coord, int type, int install);  //Set and apply tool coordinate system list
    //2. Set UDP communication parameters and load UDP communication
    UDPComParam param=new UDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10, 0);
    robot.ExtDevSetUDPComParam(param);
    robot.ExtDevLoadUDPDriver();
    //3. Set extended axis parameters, including extended axis type, extended axis driver parameters, and extended axis DH parameters
    robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0); //Single-axis positioner and DH parameters
    robot.SetRobotPosToAxis(1); //Extended axis installation position
    robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0); //Servo driver parameters. This example is for a single-axis positioner, so only one
```

(continues on next page)

(continued from previous page)

```
↪driver parameter needs to be set. If you choose an extended axis type with multiple␣
↪axes, each axis needs to be configured.
    //4. Enable the selected axis and perform homing
    robot.ExtAxisServoOn(1, 0);
    robot.ExtAxisSetHoming(1, 0, 20, 3);
    //5. Perform extended axis coordinate system calibration and application
    DescPose pos = new DescPose(/* Enter your calibration point coordinates */ );
    robot.SetRefPointInExAxisEnd(pos);
    robot.PositionorSetRefPoint(1); /*You need to call this interface 4 times with␣
↪different points to complete the calibration */
    DescPose coord = new DescPose();
    robot.PositionorComputeECoordSys(coord); //Calculate extended axis calibration result
    robot.ExtAxisActiveECoordSys(1, 1, coord, 1); //Apply the calibration result to the␣
↪extended axis coordinate system
    //6. Calibrate the workpiece coordinate system on the extended axis. You will need␣
↪the following interfaces:
    //int SetWObjCoordPoint(int point_num);
    //int ComputeWObjCoord(int method, ref DescPose wobj_pose);
    //int SetWObjCoord(int id, DescPose coord);
    //int SetWObjList(int id, DescPose coord);
    //7. Record your synchronous arc motion starting point
    DescPose startdescPose = new DescPose(/*Enter your coordinates*/ );
    JointPos startjointPos = new JointPos(/*Enter your coordinates*/ );
    ExaxisPos startexaxisPos = new ExaxisPos(/* Enter your extended axis starting point␣
↪coordinates */ );
    //8. Record your synchronous arc motion end point coordinates
    DescPose enddescPose = new DescPose(/*Enter your coordinates*/ );
    JointPos endjointPos = new JointPos(/*Enter your coordinates*/ );
    ExaxisPos endexaxisPos = new ExaxisPos(/* Enter your extended axis end point␣
↪coordinates */ );
    //9. Record your synchronous arc motion intermediate point coordinates
    DescPose middescPose = new DescPose(/*Enter your coordinates*/ );
    JointPos midjointPos =new JointPos(/*Enter your coordinates*/ );
    ExaxisPos midexaxisPos = new ExaxisPos(/* Enter the extended axis coordinates when␣
↪the robot is at the arc intermediate point */ );
    //10. Write the synchronous motion program
    //Move to the starting point, assuming the applied tool coordinate system and␣
↪workpiece coordinate system are both 1
    robot.ExtAxisMove(startexaxisPos, 20);
    DescPose offdese = new DescPose( 0, 0, 0, 0, 0, 0 );
    robot.MoveJ(startjointPos, startdescPose, 1, 1, 100, 100, 100, startexaxisPos, 0, 0,␣
↪offdese);
    //Start synchronous motion
    robot.ExtAxisSyncMoveC(midjointPos, middescPose, 1, 1, 100, 100, midexaxisPos, 0,␣
↪offdese, endjointPos, enddescPose, 1, 1, 100, 100, endexaxisPos, 0, offdese, 100, 0);
    robot.MoveJ(startjointPos, 1, 1, 100, 100, 100, startexaxisPos, 0, 0, offdese);
    //Start synchronous motion
    robot.ExtAxisSyncMoveC(middescPose, 1, 1, 100, 100, midexaxisPos, 0, offdese,␣
↪enddescPose, 1, 1, 100, 100, endexaxisPos, 0, offdese, 100, 0,-1);
    robot.CloseRPC();
    return 0;
}
```

2.3.13.50 Set Extended DO

```
/**
* @brief Set extended DO
* @param [in] DONum DO number
* @param [in] bOpen Switch true-on; false-off
* @param [in] smooth Whether to smooth
* @param [in] block Whether to block
* @return Error code
*/
int SetAuxDO(int DONum, boolean bOpen, boolean smooth, boolean block);
```

2.3.13.51 Set Extended AO

```
/**
* @brief Set extended AO
* @param [in] AONum AO number
* @param [in] value Analog value [0-4095]
* @param [in] block Whether to block
* @return Error code
*/
int SetAuxAO(int AONum, double value, boolean block);
```

2.3.13.52 Set Extended DI Input Filter Time

```
/**
* @brief Set extended DI input filter time
* @param [in] filterTime Filter time (ms)
* @return  Error code
*/
int SetAuxDIFilterTime(int filterTime);
```

2.3.13.53 Set Extended AI Input Filter Time

```
/**
* @brief Set extended AI input filter time
* @param [in] AONum AO number
* @param [in] filterTime Filter time (ms)
* @return Error code
*/
int SetAuxAIFilterTime(int AONum, int filterTime);
```

2.3.13.54 Wait for Extended DI Input

```
/**
* @brief Wait for extended DI input
* @param [in] DINum DI number
* @param [in] bOpen Switch 0-off; 1-on
* @param [in] time Maximum wait time (ms)
* @param [in] errorAlarm Whether to continue motion
* @return Error code
*/
int WaitAuxDI(int DINum, boolean bOpen, int time, boolean errorAlarm);
```

2.3.13.55 Wait for Extended AI Input

```
/**
* @brief Wait for extended AI input
* @param [in] AINum AI number
* @param [in] sign 0-greater than; 1-less than
* @param [in] value AI value
* @param [in] time Maximum wait time (ms)
* @param [in] errorAlarm Whether to continue motion
* @return Error code
*/
int WaitAuxAI(int AINum, int sign, int value, int time, boolean errorAlarm);
```

2.3.13.56 Get Extended DI Value

```
/**
* @brief Get extended DI value
* @param [in] DINum DI number
* @param [in] isNoBlock Whether to block
* @return List[0]: Error code; List[1] : isOpen 0-off; 1-on
*/
List<Integer> GetAuxDI(int DINum, boolean isNoBlock)
```

2.3.13.57 Get Extended AI Value

```
/**
* @brief Get extended AI value
* @param [in] AINum AI number
* @param [in] isNoBlock Whether to block
* @return List[0]: Error code; List[1] : value Input value
*/
List<Integer> GetAuxAI(int AINum, boolean isNoBlock);
```

2.3.13.58 Extended IO Code Example

```
public static int TestAuxDOAO(Robot robot)
{
    for (int i = 0; i < 128; i++)
    {
        robot.SetAuxDO(i, true, false, true);
        robot.Sleep(100);
    }
    for (int i = 0; i < 128; i++)
    {
        robot.SetAuxDO(i, false, false, true);
        robot.Sleep(100);
    }

    for (int i = 0; i < 409; i++)
    {
        robot.SetAuxAO(0, i * 10, true);
        robot.SetAuxAO(1, 4095 - i * 10, true);
        robot.SetAuxAO(2, i * 10, true);
        robot.SetAuxAO(3, 4095 - i * 10, true);
        robot.Sleep(10);
    }

    robot.SetAuxDIFilterTime(10);
    robot.SetAuxAIFilterTime(0, 10);


    int curValue = -1;
    List<Integer> liter=new ArrayList<>();
    for (int i = 0; i < 4; i++)
    {
        liter = robot.GetAuxAI(i, true);
    }

    robot.WaitAuxDI(1, false, 1000, false);
    robot.WaitAuxAI(1, 1, 132, 1000, false);

    robot.CloseRPC();
    return 0;
}
```

2.3.13.59 Movable Device Enable

```
/**
* @brief Movable device enable
* @param [in] enable false-disable; true-enable
* @return Error code
*/
int TractorEnable(Boolean enable);
```

2.3.13.60 Movable Device Homing

```
/**
* @brief Movable device homing
* @return Error code
*/
int TractorHoming();
```

2.3.13.61 Movable Device Linear Motion

```
/**
* @brief Movable device linear motion
* @param [in] distance Linear motion distance (mm)
* @param [in] vel Linear motion speed percentage (0-100)
* @return Error code
*/
int TractorMoveL(double distance, double vel);
```

2.3.13.62 Movable Device Arc Motion

```
/**
* @brief Movable device arc motion
* @param [in] radio Arc motion radius (mm)
* @param [in] angle Arc motion angle (°)
* @param [in] vel Linear motion speed percentage (0-100)
* @return Error code
*/
int TractorMoveC(double radio, double angle, double vel);
```

2.3.13.63 Movable Device Stop Motion

```
/**
* @brief Movable device stop motion
* @return Error code
*/
int TractorStop();
```

2.3.13.64 Movable Device Code Example

```
public static void main(String[] args)
{
    Robot robot = new Robot();
    robot.SetReconnectParam(true,20,500);//Set reconnection count and interval
    robot.LoggerInit(FrLogType.DIRECT, FrLogLevel.INFO, "D://log", 10, 10);
    int rtn = robot.RPC("192.168.58.2");
    if(rtn == 0)
    {
```

(continues on next page)

(continued from previous page)

```
        System.out.println("RPC connection success");
    }
    else
    {
        System.out.println("RPC connection fail");
        return ;
    }
    UDPComParam param = new UDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10);
    robot.ExtDevSetUDPComParam(param);//UDP extended axis communication
    robot.ExtAxisParamConfig(1, 0, 0, 50000, -50000, 1000, 1000, 6.280, 16384, 200, 0, 0, 0);
    robot.ExtAxisParamConfig(2, 0, 0, 50000, -50000, 1000, 1000, 6.280, 16384, 200, 0, 0, 0);
    robot.SetAxisDHParaConfig(5, 0, 0, 0, 0, 0, 0, 0, 0);

    robot.TractorEnable(false);
    robot.Sleep(2000);
    robot.TractorEnable(true);
    robot.Sleep(2000);
    robot.TractorHoming();

    robot.Sleep(2000);
    robot.TractorMoveL(100, 20);
    robot.Sleep(5000);
    robot.TractorMoveL(-100, 20);
    robot.Sleep(5000);
    robot.TractorMoveC(300, 90, 20);
    robot.Sleep(2000);
    robot.TractorStop();//Stop the device
    robot.TractorMoveC(300, -90, 20);
}
```

2.3.14 Robot Welding

2.3.14.1 Set welding process curve parameters

```
/**
* @brief Set welding process curve parameters
* @param [in] id Welding process ID (1-99)
* @param [in] param Welding process parameters
* @return Error code
*/
int WeldingSetProcessParam(int id, WeldingProcessParam param);
```

2.3.14.2 Get welding process curve parameters

```
/**
* @brief Get welding process curve parameters
* @param [in] id Welding process ID (1-99)
* @param [out] param Welding process parameters
* @return Error code
*/
int WeldingGetProcessParam(int id, WeldingProcessParam param);
```

2.3.14.3 Set welding current to analog output relation

```
/**
* @brief Set welding current to analog output relation
* @param [in] relation Relation value
* @return Error code
*/
int WeldingSetCurrentRelation(WeldCurrentAORelation relation);
```

2.3.14.4 Set welding voltage to analog output relation

```
/**
* @brief Set welding voltage to analog output relation
* @param [in] relation Welding voltage-analog output relation value
* @return Error code
*/
int WeldingSetVoltageRelation(WeldVoltageAORelation relation);
```

2.3.14.5 Get welding current to analog output relation

```
/**
* @brief Get welding current to analog output relation
* @param [out] relation Relation value
* @return Error code
*/
int WeldingGetCurrentRelation(WeldCurrentAORelation relation);
```

2.3.14.6 Get welding voltage to analog output relation

```
/**
* @brief Get welding voltage to analog output relation
* @param [out] relation Welding voltage-analog output relation value
* @return Error code
*/
int WeldingGetVoltageRelation(WeldVoltageAORelation relation);
```

2.3.14.7 Set welding current

```
/**
* @brief Set welding current
* @param [in] ioType Control IO type: 0-Controller IO; 1-Extension IO
* @param [in] current Welding current value (A)
* @param [in] AOIndex Controller analog output port (0-1)
* @param [in] blend Smoothing: 0-No smoothing; 1-Smoothing
* @return Error code
*/
int WeldingSetCurrent(int ioType, double current, int AOIndex, int blend);
```

2.3.14.8 Set welding voltage

```
/**
* @brief Set welding voltage
* @param [in] ioType Control IO type: 0-Controller IO; 1-Extension IO
* @param [in] voltage Welding voltage value (V)
* @param [in] AOIndex Controller analog output port (0-1)
* @param [in] blend Smoothing: 0-No smoothing; 1-Smoothing
* @return Error code
*/
int WeldingSetVoltage(int ioType, double voltage, int AOIndex, int blend);
```

2.3.14.9 Set weaving parameters

Changed in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Set weaving parameters
* @param [in] weaveNum Weaving parameter configuration ID
* @param [in] weaveType Weaving type:
*   0-Horizontal triangular wave
*   1-Vertical L-shaped triangular wave
*   2-Clockwise circular
*   3-Counter-clockwise circular
*   4-Horizontal sine wave
*   5-Vertical L-shaped sine wave
*   6-Vertical triangular wave
*   7-Vertical sine wave
* @param [in] weaveFrequency Weaving frequency (Hz)
* @param [in] weaveIncStayTime Wait mode:
*   0-Cycle excludes wait time
*   1-Cycle includes wait time
* @param [in] weaveRange Weaving amplitude (mm)
* @param [in] weaveLeftRange Vertical triangular wave left chord length (mm)
* @param [in] weaveRightRange Vertical triangular wave right chord length (mm)
* @param [in] additionalStayTime Vertical triangular wave apex dwell time (ms)
* @param [in] weaveLeftStayTime Left dwell time (ms)
* @param [in] weaveRightStayTime Right dwell time (ms)
```

(continues on next page)

(continued from previous page)

```
* @param [in] weaveCircleRadio Circular weaving callback ratio (0-100%)
* @param [in] weaveStationary Position wait:
*   0-Continue moving during wait
*   1-Stationary during wait
* @param [in] weaveYawAngle Weaving direction azimuth angle (around Z-axis) in °
* @param [in] weaveRotAngle Weaving direction azimuth angle (around X-axis) in °
* @return Error code
*/
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);
```

2.3.14.10 Welding parameter setting code example

```
public static int TestSetWeldParam(Robot robot)
{
    WeldingProcessParam para1 = new WeldingProcessParam(177, 27, 1000, 178, 28, 176, 26,␣
↪1000);
    WeldingProcessParam para2 = new WeldingProcessParam(188, 28, 555, 199, 29, 133, 23,␣
↪333);

    robot.WeldingSetProcessParam(1, para1);
    robot.WeldingSetProcessParam(2, para2);

    double startCurrent = 0;
    double startVoltage = 0;
    int startTime = 0;
    double weldCurrent = 0;
    double weldVoltage = 0;
    double endCurrent = 0;
    double endVoltage = 0;
    int endTime = 0;

    WeldingProcessParam param = new WeldingProcessParam(startCurrent, startVoltage,␣
↪startTime, weldCurrent, weldVoltage, endCurrent, endVoltage, endTime);
    robot.WeldingGetProcessParam(1, param);
    robot.WeldingGetProcessParam(2, param);

    WeldCurrentAORelation rela1 = new WeldCurrentAORelation(0, 400, 0, 10, 0);
    int rtn = robot.WeldingSetCurrentRelation(rela1);

    WeldVoltageAORelation rela2 = new WeldVoltageAORelation(0, 40, 0, 10, 1);
    rtn = robot.WeldingSetVoltageRelation(rela2);

    double current_min = 0;
    double current_max = 0;
    double vol_min = 0;
    double vol_max = 0;
    double output_vmin = 0;
```

(continues on next page)

(continued from previous page)

```
    double output_vmax = 0;
    int curIndex = 0;
    int volIndex = 0;
    WeldCurrentAORelation rela3 = new WeldCurrentAORelation(current_min, current_max,␣
↪output_vmin, output_vmax, curIndex);
    rtn = robot.WeldingGetCurrentRelation(rela3);

    WeldVoltageAORelation rela4 = new WeldVoltageAORelation(0, 0, 0, 0, 0);
    rtn = robot.WeldingGetVoltageRelation(rela4);

    rtn = robot.WeldingSetCurrent(0, 100, 0, 0);

    robot.Sleep(3000);

    rtn = robot.WeldingSetVoltage(0, 10, 0, 0);

    rtn = robot.WeaveSetPara(0, 0, 2.000000, 0, 10.000000, 0.000000, 0.000000, 0, 0, 0,␣
↪0, 0, 60.000000, 0);

    robot.WeaveOnlineSetPara(0, 0, 1, 0, 20, 0, 0, 0, 0);

    rtn = robot.WeldingSetCheckArcInterruptionParam(1, 200);
    rtn = robot.WeldingSetReWeldAfterBreakOffParam(1, 5.7, 98.2, 0);
    int enable = 0;
    double length = 0;
    double velocity = 0;
    int moveType = 0;
    int checkEnable = 0;
    int arcInterruptTimeLength = 0;
    List<Integer> inter = new ArrayList<>();
    List<Number> num = new ArrayList<>();

    inter = robot.WeldingGetCheckArcInterruptionParam();
    num = robot.WeldingGetReWeldAfterBreakOffParam();

    robot.SetWeldMachineCtrlModeExtDoNum(17);
    for (int i = 0; i < 5; i++) {
        robot.SetWeldMachineCtrlMode(0);
        robot.Sleep(1000);
        robot.SetWeldMachineCtrlMode(1);
        robot.Sleep(1000);
    }
    return 0;
}
```

2.3.14.11 Real-time weaving parameter setting

```
/**
* @brief Real-time weaving parameter setting
* @param [in] weaveNum Weaving parameter configuration ID
* @param [in] weaveType Weaving type:
*   0-Horizontal triangular wave
*   1-Vertical L-shaped triangular wave
*   2-Clockwise circular
*   3-Counter-clockwise circular
*   4-Horizontal sine wave
*   5-Vertical L-shaped sine wave
*   6-Vertical triangular wave
*   7-Vertical sine wave
* @param [in] weaveFrequency Weaving frequency (Hz)
* @param [in] weaveIncStayTime Wait mode:
*   0-Cycle excludes wait time
*   1-Cycle includes wait time
* @param [in] weaveRange Weaving amplitude (mm)
* @param [in] weaveLeftStayTime Left dwell time (ms)
* @param [in] weaveRightStayTime Right dwell time (ms)
* @param [in] weaveCircleRadio Circular weaving callback ratio (0-100%)
* @param [in] weaveStationary Position wait:
*   0-Continue moving during wait
*   1-Stationary during wait
* @return Error code
*/
int WeaveOnlineSetPara(int weaveNum, int weaveType, double weaveFrequency, int␣
↪weaveIncStayTime, double weaveRange, int weaveLeftStayTime, int weaveRightStayTime,␣
↪int weaveCircleRadio, int weaveStationary);
```

2.3.14.12 Set welding arc interruption detection parameters

```
/**
* @brief Set welding arc interruption detection parameters
* @param [in] checkEnable Enable detection: 0-Disable; 1-Enable
* @param [in] arcInterruptTimeLength Arc interruption confirmation duration (ms)
* @return Error code
*/
int WeldingSetCheckArcInterruptionParam(int checkEnable, int arcInterruptTimeLength);
```

2.3.14.13 Get welding arc interruption detection parameters

```
/**
* @brief Get welding arc interruption detection parameters
* @return List[0]: Error code; List[1]: double Detection enable (0-Disable; 1-Enable);␣
↪List[2]: Arc interruption confirmation duration (ms)
*/
List<Integer> WeldingGetCheckArcInterruptionParam();
```

2.3.14.14 Set welding interruption recovery parameters

```
/**
* @brief Set welding interruption recovery parameters
* @param [in] enable Enable welding recovery
* @param [in] length Weld overlap distance (mm)
* @param [in] velocity Robot return speed percentage (0-100)
* @param [in] moveType Robot movement type to restart point: 0-LIN; 1-PTP
* @return Error code
*/
int WeldingSetReWeldAfterBreakOffParam(int enable, double length, double velocity, int␣
↪moveType);
```

2.3.14.15 Get welding interruption recovery parameters

```
/**
* @brief Get welding interruption recovery parameters
* @return List[0]: Error code; List[1]: int Enable welding recovery; List[2]: double␣
↪Weld overlap distance (mm);
* @return List[3]: double Robot return speed percentage (0-100); List[4]: int Robot␣
↪movement type: 0-LIN; 1-PTP
*/
List<Number> WeldingGetReWeldAfterBreakOffParam();
```

2.3.14.16 Set welder control mode extension DO port

```
/**
* @brief Set welder control mode extension DO port
* @param [in] DONum Welder control mode DO port (0-127)
* @return Error code
*/
int SetWeldMachineCtrlModeExtDoNum(int DONum);
```

2.3.14.17 Set welder control mode

```
/**
* @brief Set welder control mode
* @param [in] mode Welder control mode: 0-Unified
* @return Error code
*/
int SetWeldMachineCtrlMode(int mode);
```

2.3.14.18 Welding start

```
/**
* @brief Welding start
* @param [in] ioType IO type: 0-Controller IO; 1-Extension IO
* @param [in] arcNum Welder configuration file ID
* @param [in] timeout Arc start timeout
* @return Error code
*/
int ARCStart(int ioType, int arcNum, int timeout);
```

2.3.14.19 Welding end

```
/**
* @brief Welding end
* @param [in] ioType IO type: 0-Controller IO; 1-Extension IO
* @param [in] arcNum Welder configuration file ID
* @param [in] timeout Arc end timeout
* @return Error code
*/
int ARCEnd(int ioType, int arcNum, int timeout);
```

2.3.14.20 Weaving start

```
/**
* @brief Weaving start
* @param [in] weaveNum Weaving parameter configuration ID
* @return Error code
*/
int WeaveStart(int weaveNum);
```

2.3.14.21 Weaving end

```
/**
* @brief Weaving end
* @param [in] weaveNum Weaving parameter configuration ID
* @return Error code
*/
int WeaveEnd(int weaveNum);
```

2.3.14.22 Forward wire feeding

```
/**
* @brief Forward wire feeding
* @param [in] ioType IO type: 0-Controller IO; 1-Extension IO
* @param [in] wireFeed Wire feed control: 0-Stop; 1-Feed
* @return Error code
*/
int SetForwardWireFeed(int ioType, int wireFeed);
```

2.3.14.23 Reverse wire feeding

```
/**
* @brief Reverse wire feeding
* @param [in] ioType IO type: 0-Controller IO; 1-Extension IO
* @param [in] wireFeed Wire feed control: 0-Stop; 1-Feed
* @return Error code
*/
int SetReverseWireFeed(int ioType, int wireFeed);
```

2.3.14.24 Gas feeding

```
/**
* @brief Gas feeding
* @param [in] ioType IO type: 0-Controller IO; 1-Extension IO
* @param [in] airControl Gas control: 0-Stop; 1-Feed
* @return Error code
*/
int SetAspirated(int ioType, int airControl);
```

2.3.14.25 Set robot to resume welding after interruption

```
/**
* @brief Set robot to resume welding after interruption
* @return Error code
*/
int WeldingStartReWeldAfterBreakOff();
```

2.3.14.26 Set robot to abort welding after interruption

```
/**
* @brief Set robot to abort welding after interruption
* @return Error code
*/
int WeldingAbortWeldAfterBreakOff();
```

2.3.14.27 Robot welding control code example

```
public static int TestWelding(Robot robot)
{
    robot.WeldingSetCurrent(0, 230, 0, 0);
    robot.WeldingSetVoltage(0, 24, 0, 1);

    DescPose p1Desc = new DescPose(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
    JointPos p1Joint = new JointPos(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);

    DescPose p2Desc = new DescPose(-333.302, -435.580, 449.866, -174.997, 2.017, 109.
↪815);
    JointPos p2Joint = new JointPos(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    robot.MoveJ(p1Joint, p1Desc, 13, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
    robot.ARCStart(1, 0, 10000);
    robot.WeaveStart(0);
    robot.MoveL(p2Joint, p2Desc, 13, 0, 20, 100, 100, -1, 0, exaxisPos, 0, 0, offdese, 0,
↪ 10);
    robot.ARCEnd(1, 0, 10000);
    robot.WeaveEnd(0);
    return 0;
}
```

2.3.14.28 Segment welding start

```
/**
* @brief Segment welding start
* @param [in] startDesePos Start point Cartesian position
* @param [in] endDesePos End point Cartesian pose
* @param [in] startJPos Start point joint pose
* @param [in] endJPos End point joint pose
* @param [in] weldLength Weld segment length (mm)
* @param [in] noWeldLength Non-weld segment length (mm)
* @param [in] weldIOType Welding IO type (0-Controller IO; 1-Extension IO)
* @param [in] arcNum Welder configuration file ID
* @param [in] weldTimeout Arc start/end timeout
* @param [in] isWeave Enable weaving
* @param [in] weaveNum Weaving parameter configuration ID
* @param [in] tool Tool number
* @param [in] user Workpiece number
* @param [in] vel Speed percentage (0~100)
* @param [in] acc Acceleration percentage (0~100) (Not currently available)
* @param [in] ovl Speed scaling factor (0~100)
* @param [in] blendR [-1.0]-Move to position (blocking); [0~1000.0]-Smoothing radius␣
↪(non-blocking) in mm
* @param [in] epos External axis position in mm
* @param [in] search 0-No wire search; 1-Wire search
```

(continues on next page)

(continued from previous page)

```
* @param [in] offset_flag 0-No offset; 1-Offset in base/workpiece frame; 2-Offset in tool frame
* @param [in] offset_pos Pose offset
* @return Error code
*/
int SegmentWeldStart(DescPose startDesePos, DescPose endDesePos, JointPos startJPos, JointPos endJPos, double weldLength, double noWeldLength, int weldIOType, int arcNum, int weldTimeout, boolean isWeave, int weaveNum, int tool, int user, double vel, double acc, double ovl, double blendR, ExaxisPos epos, int search, int offset_flag, DescPose offset_pos);
```

2.3.14.29 Robot segment welding code example

```
public static int TestSegWeld(Robot robot)
{
    robot.WeldingSetCurrent(0, 230, 0, 0);
    robot.WeldingSetVoltage(0, 24, 0, 1);

    DescPose p1Desc = new DescPose(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
    JointPos p1Joint = new JointPos(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);

    DescPose p2Desc = new DescPose(-333.302, -435.580, 449.866, -174.997, 2.017, 109.815);
    JointPos p2Joint = new JointPos(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    robot.GetForwardKin(p1Joint, p1Desc);
    robot.GetForwardKin(p2Joint, p2Desc);

    int rtn = robot.SegmentWeldStart(p1Desc, p2Desc, p1Joint, p2Joint, 20, 20, 0, 0, 5000, true, 0, 1, 0, 30, 100, 100, -1, exaxisPos, 0, 0, offdese);
    return 0;
}
```

2.3.14.30 Simulation weaving start

```
/**
* @brief Simulation weaving start
* @param [in] weaveNum Weaving parameter ID
* @return Error code
*/
int WeaveStartSim(int weaveNum);
```

2.3.14.31 Simulation weaving end

```
/**
* @brief Simulation weaving end
* @param [in] weaveNum Weaving parameter ID
* @return Error code
*/
int WeaveEndSim(int weaveNum);
```

2.3.14.32 Start trajectory inspection warning (no movement)

```
/**
* @brief Start trajectory inspection warning (no movement)
* @param [in] weaveNum Weaving parameter ID
* @return Error code
*/
int WeaveInspectStart(int weaveNum);
```

2.3.14.33 End trajectory inspection warning (no movement)

```
/**
* @brief End trajectory inspection warning (no movement)
* @param [in] weaveNum Weaving parameter ID
* @return Error code
*/
int WeaveInspectEnd(int weaveNum);
```

2.3.14.34 Weaving transition start

Changed in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Weaving transition start
* @param [in] weaveChangeFlag 1-Change weaving parameters; 2-Change weaving parameters + welding speed
* @param [in] weaveNum Weaving ID
* @param [in] velStart Welding start speed (cm/min)
* @param [in] velEnd Welding end speed (cm/min)
* @return Error code
*/
int WeaveChangeStart(int weaveChangeFlag, int weaveNum, double velStart, double velEnd);
```

2.3.14.35 Robot weaving transition welding code example

```
public static int TestWeave(Robot robot)
{
    DescPose p1Desc = new DescPose(228.879, -503.594, 453.984, -175.580, 8.293, 171.267);
    JointPos p1Joint = new JointPos(102.700, -85.333, 90.518, -102.365, -83.932, 22.134);

    DescPose p2Desc = new DescPose(-333.302, -435.580, 449.866, -174.997, 2.017, 109.
↪815);
    JointPos p2Joint = new JointPos(41.862, -85.333, 90.526, -100.587, -90.014, 22.135);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    robot.MoveJ(p1Joint, p1Desc, 13, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
    robot.WeaveStartSim(0);
    robot.MoveL(p2Joint, p2Desc, 13, 0, 20, 100, 100, -1, 0, exaxisPos, 0, 0, offdese, 0,
↪ 10);
    robot.WeaveEndSim(0);
    robot.MoveJ(p1Joint, p1Desc, 13, 0, 20, 100, 100, exaxisPos, -1, 0, offdese);
    robot.WeaveInspectStart(0);
    robot.MoveL(p2Joint, p2Desc, 13, 0, 20, 100, 100, -1, 0, exaxisPos, 0, 0, offdese, 0,
↪ 10);
    robot.WeaveInspectEnd(0);

    robot.WeldingSetVoltage(1, 19, 0, 0);
    robot.WeldingSetCurrent(1, 190, 0, 0);
    robot.MoveL(p1Joint, p1Desc, 1, 1, 100, 100, 50, -1, 0, exaxisPos, 0, 0, offdese, 0,␣
↪10);
    robot.ARCStart(1, 0, 10000);
    robot.ArcWeldTraceControl(1, 0, 1, 0.06, 5, 5, 60, 1, 0.06, 5, 5, 80, 0, 0, 4, 1, 10,
↪ 0, 0);
    robot.WeaveStart(0);
    robot.WeaveChangeStart(1, 0, 50, 30);
    robot.MoveL(p2Joint, p2Desc, 1, 1, 100, 100, 1, -1, 0, exaxisPos, 0, 0, offdese, 0,␣
↪10);
    robot.WeaveChangeEnd();
    robot.WeaveEnd(0);
    robot.ArcWeldTraceControl(0, 0, 1, 0.06, 5, 5, 60, 1, 0.06, 5, 5, 80, 0, 0, 4, 1, 10,
↪ 0, 0);
    robot.ARCEnd(1, 0, 10000);
    return 0;
}
```

2.3.14.36 Weaving transition end

New in version Java: SDK-v1.0.2-3.7.9

```
/**
* @brief Weaving transition end
* @return Error code
*/
int WeaveChangeEnd();
```

2.3.14.37 Extension IO - Configure welder gas detection signal

```
/**
* @brief Extension IO - Configure welder gas detection signal
* @param [in] DONum Gas detection signal extension DO number
* @return Error code
*/
int SetAirControlExtDoNum(int DONum);
```

2.3.14.38 Extension IO - Configure welder arc start signal

```
/**
* @brief Extension IO - Configure welder arc start signal
* @param [in] DONum Welder arc start signal extension DO number
* @return Error code
*/
int SetArcStartExtDoNum(int DONum);
```

2.3.14.39 Extension IO - Configure welder reverse wire feed signal

```
/**
* @brief Extension IO - Configure welder reverse wire feed signal
* @param [in] DONum Reverse wire feed signal extension DO number
* @return Error code
*/
int SetWireReverseFeedExtDoNum(int DONum);
```

2.3.14.40 Extension IO - Configure welder forward wire feed signal

```
/**
* @brief Extension IO - Configure welder forward wire feed signal
* @param [in] DONum Forward wire feed signal extension DO number
* @return Error code
*/
int SetWireForwardFeedExtDoNum(int DONum);
```

2.3.14.41 Extension IO - Configure welder arc success signal

```
/**
* @brief Extension IO - Configure welder arc success signal
* @param [in] DINum Arc success signal extension DI number
* @return Error code
*/
int SetArcDoneExtDiNum(int DINum);
```

2.3.14.42 Extension IO - Configure welder ready signal

```
/**
* @brief Extension IO - Configure welder ready signal
* @param [in] DINum Welder ready signal extension DI number
* @return Error code
*/
int SetWeldReadyExtDiNum(int DINum);
```

2.3.14.43 Extension IO - Configure welding interruption recovery signal

```
/**
* @brief Extension IO - Configure welding interruption recovery signal
* @param [in] reWeldDINum Resume welding after interruption signal extension DI number
* @param [in] abortWeldDINum Abort welding after interruption signal extension DI number
* @return Error code
*/
int SetExtDIWeldBreakOffRecover(int reWeldDINum, int abortWeldDINum);
```

2.3.14.44 Set extension IO welding signal code example

```
public static int TestExtDIConfig(Robot robot)
{
    robot.SetArcStartExtDoNum(10);
    robot.SetAirControlExtDoNum(20);
    robot.SetWireForwardFeedExtDoNum(30);
    robot.SetWireReverseFeedExtDoNum(40);

    robot.SetWeldReadyExtDiNum(50);
    robot.SetArcDoneExtDiNum(60);
    robot.SetExtDIWeldBreakOffRecover(70, 80);
    robot.SetWireSearchExtDIONum(0, 1);

    return 0;
}
```

2.3.14.45 Arc tracking control

Changed in version Java: SDK-v1.0.2-3.7.9

```
/**
* @brief Arc tracking control
* @param [in] flag Switch: 0-Off; 1-On
* @param [in] delaytime Lag time in ms
* @param [in] isLeftRight Left/right deviation compensation
* @param [in] klr Left/right adjustment coefficient (sensitivity)
* @param [in] tStartLr Left/right start compensation time in cycles
* @param [in] stepMaxLr Left/right maximum compensation per step in mm
* @param [in] sumMaxLr Left/right total maximum compensation in mm
* @param [in] isUpLow Up/down deviation compensation
* @param [in] kud Up/down adjustment coefficient (sensitivity)
* @param [in] tStartUd Up/down start compensation time in cycles
* @param [in] stepMaxUd Up/down maximum compensation per step in mm
* @param [in] sumMaxUd Up/down total maximum compensation in mm
* @param [in] axisSelect Up/down coordinate system selection: 0-Weaving; 1-Tool; 2-Base
* @param [in] referenceType Up/down reference current setting method: 0-Feedback; 1-
↪Constant
* @param [in] referSampleStartUd Up/down reference current sampling start count␣
↪(feedback) in cycles
* @param [in] referSampleCountUd Up/down reference current sampling cycle count␣
↪(feedback) in cycles
* @param [in] referenceCurrent Up/down reference current in mA
* @param [in] offsetType Offset tracking type: 0-No offset; 1-Sampling; 2-Percentage
* @param [in] offsetParameter Offset parameters:
*   Sampling (offset sampling start time, default one cycle)
*   Percentage (offset percentage (-100 ~ 100))
* @return Error code
*/
int ArcWeldTraceControl(int flag, double delaytime, int isLeftRight, double klr, double␣
↪tStartLr, double stepMaxLr, double sumMaxLr, int isUpLow, double kud, double tStartUd,␣
↪double stepMaxUd, double sumMaxUd, int axisSelect, int referenceType, double␣
↪referSampleStartUd, double referSampleCountUd, double referenceCurrent, int offsetType,
↪ int offsetParameter);
```

2.3.14.46 Arc tracking AI channel selection

```
/**
* @brief Arc tracking AI channel selection
* @param channel Arc tracking AI channel selection [0-3]
* @return Error code
*/
public int ArcWeldTraceExtAIChannelConfig(int channel);
```

2.3.14.47 Arc tracking + multi-layer multi-pass compensation start

```
/**
* @brief Arc tracking + multi-layer multi-pass compensation start
* @return Error code
*/
public int ArcWeldTraceReplayStart();
```

2.3.14.48 Arc tracking + multi-layer multi-pass compensation end

```
/**
* @brief Arc tracking + multi-layer multi-pass compensation end
* @return Error code
*/
public int ArcWeldTraceReplayEnd();
```

2.3.14.49 Offset coordinate transformation - multi-layer multi-pass welding

```
/**
* @brief Offset coordinate transformation - multi-layer multi-pass welding
* @param pointO Reference point Cartesian pose
* @param pointX Reference point X-direction offset point Cartesian pose
* @param pointZ Reference point Z-direction offset point Cartesian pose
* @param dx X-direction offset (mm)
* @param dz Z-direction offset (mm)
* @param dry Rotation around Y-axis offset (°)
* @param offset Calculated offset
* @return Error code
*/
public int MultilayerOffsetTrsfToBase(DescTran pointO, DescTran pointX, DescTran pointZ,␣
↪double dx, double dz, double dry, DescPose offset);
```

2.3.14.50 Multi-layer multi-pass welding arc tracking code example

```
public static int TestArcWeldTrace(Robot robot)
{
    JointPos mulitilineorigin1_joint = new JointPos(-24.090, -63.501, 84.288, -111.940, -
↪93.426, 57.669);
    DescPose mulitilineorigin1_desc = new DescPose(-677.559, 190.951, -1.205, 1.144, -41.
↪482, -82.577);

    DescTran mulitilineX1_desc = new DescTran(0, 0, 0);
    mulitilineX1_desc.x = -677.556;
    mulitilineX1_desc.y = 211.949;
    mulitilineX1_desc.z = -1.206;

    DescTran mulitilineZ1_desc = new DescTran(0, 0, 0);
    mulitilineZ1_desc.x = -677.564;
```

(continues on next page)

(continued from previous page)

```
    mulitilineZ1_desc.y = 190.956;
    mulitilineZ1_desc.z = 19.817;

    JointPos mulitilinesafe_joint = new JointPos(-25.734, -63.778, 81.502, -108.975, -93.
↪392, 56.021);
    DescPose mulitilinesafe_desc = new DescPose(-677.561, 211.950, 19.812, 1.144, -41.
↪482, -82.577);
    JointPos mulitilineorigin2_joint = new JointPos(-29.743, -75.623, 101.241, -116.354,␣
↪-94.928, 55.735);
    DescPose mulitilineorigin2_desc = new DescPose(-563.961, 215.359, -0.681, 2.845, -40.
↪476, -87.443);

    DescTran mulitilineX2_desc = new DescTran(0, 0, 0);
    mulitilineX2_desc.x = -563.965;
    mulitilineX2_desc.y = 220.355;
    mulitilineX2_desc.z = -0.680;

    DescTran mulitilineZ2_desc = new DescTran(0, 0, 0);
    mulitilineZ2_desc.x = -563.968;
    mulitilineZ2_desc.y = 215.362;
    mulitilineZ2_desc.z = 4.331;

    ExaxisPos epos = new ExaxisPos(0, 0, 0, 0);
    DescPose offset = new DescPose(0, 0, 0, 0, 0, 0);

    robot.Sleep(10);
    int error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100,␣
↪100, epos, -1, 0, offset);

    error = robot.MoveL(mulitilineorigin1_joint, mulitilineorigin1_desc, 13, 0, 10, 100,␣
↪100, -1, 0, epos, 0, 0, offset, 0, 100);

    error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
↪epos, -1, 0, offset);

    error = robot.MoveL(mulitilineorigin2_joint, mulitilineorigin2_desc, 13, 0, 10, 100,␣
↪100, -1, 0, epos, 0, 0, offset, 0, 100);

    error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
↪epos, -1, 0, offset);

    error = robot.MoveL(mulitilineorigin1_joint, mulitilineorigin1_desc, 13, 0, 10, 100,␣
↪100, -1, 0, epos, 0, 0, offset, 0, 100);

    error = robot.ARCStart(1, 0, 3000);

    error = robot.WeaveStart(0);

    error = robot.ArcWeldTraceControl(1, 0, 1, 0.06, 5, 5, 50, 1, 0.06, 5, 5, 55, 0, 0,␣
↪4, 1, 10, 0, 0);

    error = robot.MoveL(mulitilineorigin2_joint, mulitilineorigin2_desc, 13, 0, 1, 100,␣
```

(continues on next page)

(continued from previous page)

```
↪100, -1, 0, epos, 0, 0, offset, 0, 100);

    error = robot.ArcWeldTraceControl(0, 0, 1, 0.06, 5, 5, 50, 1, 0.06, 5, 5, 55, 0, 0,␣
↪4, 1, 10, 0, 0);

    error = robot.WeaveEnd(0);

    error = robot.ARCEnd(1, 0, 10000);

    error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
↪epos, -1, 0, offset);

    error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin1_desc.tran, mulitilineX1_
↪desc, mulitilineZ1_desc, 10.0, 0.0, 0.0, offset);

    error = robot.MoveL(mulitilineorigin1_joint, mulitilineorigin1_desc, 13, 0, 10, 100,␣
↪100, -1, 0, epos, 0, 1, offset, 0, 100);

    error = robot.ARCStart(1, 0, 3000);

    error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin2_desc.tran, mulitilineX2_
↪desc, mulitilineZ2_desc, 10, 0, 0, offset);

    error = robot.ArcWeldTraceReplayStart();

    error = robot.MoveL(mulitilineorigin2_joint, mulitilineorigin2_desc, 13, 0, 2, 100,␣
↪100, -1, 0, epos, 0, 1, offset, 0, 100);

    error = robot.ArcWeldTraceReplayEnd();

    error = robot.ARCEnd(1, 0, 10000);

    error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
↪epos, -1, 0, offset);

    error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin1_desc.tran, mulitilineX1_
↪desc, mulitilineZ1_desc, 0, 10, 0, offset);

    error = robot.MoveL(mulitilineorigin1_joint, mulitilineorigin1_desc, 13, 0, 10, 100,␣
↪100, -1, 0, epos, 0, 1, offset, 0, 100);

    error = robot.ARCStart(1, 0, 3000);

    error = robot.MultilayerOffsetTrsfToBase(mulitilineorigin2_desc.tran, mulitilineX2_
↪desc, mulitilineZ2_desc, 0, 10, 0, offset);

    error = robot.ArcWeldTraceReplayStart();

    error = robot.MoveL(mulitilineorigin2_joint, mulitilineorigin2_desc, 13, 0, 2, 100,␣
↪100, -1, 0, epos, 0, 1, offset, 0, 100);

    error = robot.ArcWeldTraceReplayEnd();
```

(continues on next page)

(continued from previous page)

```
94
95      error = robot.ARCEnd(1, 0, 3000);
96
97      error = robot.MoveJ(mulitilinesafe_joint, mulitilinesafe_desc, 13, 0, 10, 100, 100,␣
    ↪epos, -1, 0, offset);
98
99      robot.CloseRPC();
100     return 0;
101 }
```

2.3.14.51 Arc tracking welder current feedback AI channel selection

New in version Java: SDK-v1.0.5-3.8.2

```
1  /**
2  * @brief Arc tracking welder current feedback AI channel selection
3  * @param [in] channel Channel:
4  *   0-Extension AI0
5  *   1-Extension AI1
6  *   2-Extension AI2
7  *   3-Extension AI3
8  *   4-Controller AI0
9  *   5-Controller AI1
10 * @return Error code
11 */
12 int ArcWeldTraceAIChannelCurrent(int channel);
```

2.3.14.52 Arc tracking welder voltage feedback AI channel selection

New in version Java: SDK-v1.0.5-3.8.2

```
1  /**
2  * @brief Arc tracking welder voltage feedback AI channel selection
3  * @param [in] channel Channel:
4  *   0-Extension AI0
5  *   1-Extension AI1
6  *   2-Extension AI2
7  *   3-Extension AI3
8  *   4-Controller AI0
9  *   5-Controller AI1
10 * @return Error code
11 */
12 int ArcWeldTraceAIChannelVoltage(int channel);
```

2.3.14.53 Arc tracking welder current feedback conversion parameters

New in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Arc tracking welder current feedback conversion parameters
* @param [in] AILow AI channel lower limit (default 0V, range [0-10V])
* @param [in] AIHigh AI channel upper limit (default 10V, range [0-10V])
* @param [in] currentLow AI channel lower limit corresponding welder current (default 0V,
↪ range [0-200V])
* @param [in] currentHigh AI channel upper limit corresponding welder current (default␣
↪100V, range [0-200V])
* @return Error code
*/
int ArcWeldTraceCurrentPara(double AILow, double AIHigh, double currentLow, double␣
↪currentHigh);
```

2.3.14.54 Arc tracking welder voltage feedback conversion parameters

New in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Arc tracking welder voltage feedback conversion parameters
* @param [in] AILow AI channel lower limit (default 0V, range [0-10V])
* @param [in] AIHigh AI channel upper limit (default 10V, range [0-10V])
* @param [in] voltageLow AI channel lower limit corresponding welder voltage (default 0V,
↪ range [0-200V])
* @param [in] voltageHigh AI channel upper limit corresponding welder voltage (default␣
↪100V, range [0-200V])
* @return Error code
*/
int ArcWeldTraceVoltagePara(double AILow, double AIHigh, double voltageLow, double␣
↪voltageHigh);
```

2.3.14.55 Arc tracking code example

```
public static void WeldTraceControlWithCtrlBoxAI(Robot robot)
{
    DescPose startdescPose = new DescPose(-473.86, 257.879, -20.849, -37.317, -42.021, 2.
↪543);
    JointPos startjointPos = new JointPos(-43.487, -76.526, 95.568, -104.445, -89.356, 3.
↪72);

    DescPose safedescPose = new DescPose(-504.043, 275.181, 40.908, -28.002, -42.025, -
↪14.044);
    JointPos safejointPos = new JointPos(-39.078, -76.732, 87.227, -99.47, -94.301, 18.
↪714);

    DescPose enddescPose = new DescPose(-499.844, 141.225, 7.72, -34.856, -40.17, 13.13);
    JointPos endjointPos = new JointPos(-31.305, -82.998, 99.401, -104.426, -89.35, 3.
↪696);
```

(continues on next page)

(continued from previous page)

```
    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);
    // Move to safe point
    robot.MoveJ(safejointPos, safedescPose, 1, 0, 5, 20, 100, exaxisPos, -1, 0, offdese);

    WeldCurrentAORelation current = new WeldCurrentAORelation(0, 495, 1, 10, 0);
    WeldVoltageAORelation voltage = new WeldVoltageAORelation(10, 45, 1, 10, 1);
    robot.WeldingSetCurrentRelation(current); // Current to analog output relation
    robot.WeldingSetVoltageRelation(voltage); // Voltage to analog output relation
    robot.WeldingSetVoltage(0, 25, 1, 0); // Set voltage
    robot.WeldingSetCurrent(0, 260, 0, 0); // Set current

    int rtn = robot.ArcWeldTraceAIChannelCurrent(4);
    System.out.println("ArcWeldTraceAIChannelCurrent rtn is " + rtn);

    rtn = robot.ArcWeldTraceAIChannelVoltage(5);
    System.out.println("ArcWeldTraceAIChannelVoltage rtn is " + rtn);

    rtn = robot.ArcWeldTraceCurrentPara(0.0, 5, 0, 500);
    System.out.println("ArcWeldTraceCurrentPara rtn is " + rtn);

    rtn = robot.ArcWeldTraceVoltagePara(1.018, 10, 0, 50);
    System.out.println("ArcWeldTraceVoltagePara rtn is " + rtn);

    robot.MoveJ(startjointPos, startdescPose, 1, 0, 20, 20, 100, exaxisPos, -1, 0,␣
↪offdese);
    robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1,␣
↪10, 0, 0);
    robot.ARCStart(0, 0, 10000);
    robot.WeaveStart(0);
    robot.MoveL(endjointPos, enddescPose, 1, 0, 100, 100, 2, -1, 0, exaxisPos, 0, 0,␣
↪offdese, 0, 10);
    robot.ARCEnd(0, 0, 10000);
    robot.WeaveEnd(0);
    robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1,␣
↪10, 0, 0);

    robot.MoveJ(safejointPos, safedescPose, 1, 0, 20, 20, 100, exaxisPos, -1, 0,␣
↪offdese);
}
```

2.3.14.56 Set Wire Search Extension IO Ports

```
/**
* @brief Sets the wire search extension IO ports
* @param [in] searchDoneDINum Wire search success DO port (0-127)
* @param [in] searchStartDONum Wire search start/stop control DO port (0-127)
* @return Error code
*/
int SetWireSearchExtDIONum(int searchDoneDINum, int searchStartDONum);
```

2.3.14.57 Wire search start

```
/**
* @brief Wire search start
* @param [in] refPos 1-Reference point; 0-Contact point
* @param [in] searchVel Search speed %
* @param [in] searchDis Search distance mm
* @param [in] autoBackFlag Auto return flag: 0-No auto; 1-Auto
* @param [in] autoBackVel Auto return speed %
* @param [in] autoBackDis Auto return distance mm
* @param [in] offectFlag 1-Search with offset; 0-Teach point search
* @return Error code
*/
int WireSearchStart(int refPos, double searchVel, int searchDis, int autoBackFlag,␣
↪double autoBackVel, int autoBackDis, int offectFlag);
```

2.3.14.58 Wire search end

```
/**
* @brief Wire search end
* @param [in] refPos 1-Reference point; 2-Contact point
* @param [in] searchVel Search speed %
* @param [in] searchDis Search distance mm
* @param [in] autoBackFlag Auto return flag: 0-No auto; 1-Auto
* @param [in] autoBackVel Auto return speed %
* @param [in] autoBackDis Auto return distance mm
* @param [in] offectFlag 1-Search with offset; 2-Teach point search
* @return Error code
*/
int WireSearchEnd(int refPos, double searchVel, int searchDis, int autoBackFlag, double␣
↪autoBackVel, int autoBackDis, int offectFlag);
```

2.3.14.59 Calculate wire search offset

```
/**
* @brief Calculate wire search offset
* @param [in] seamType Weld seam type
* @param [in] method Calculation method
* @param [in] varNameRef Reference points 1-6, "#" indicates no point variable
* @param [in] varNameRes Contact points 1-6, "#" indicates no point variable
* @param [out] offset Offset pose [x, y, z, a, b, c] and offset method
* @return Error code
*/
int GetWireSearchOffset(int seamType, int method, String[] varNameRef, String[]␣
↪varNameRes, DescOffset offset);
```

2.3.14.60 Wait for wire search completion

```
/**
* @brief Wait for wire search completion
* @return Error code
*/
int WireSearchWait(String name);
```

2.3.14.61 Write wire search contact point to database

```
/**
* @brief Write wire search contact point to database
* @param [in] varName Contact point name "RES0" ~ "RES99"
* @param [in] pos Contact point data [x, y, z, a, b, c]
* @return Error code
*/
int SetPointToDatabase(String varName, DescPose pos);
```

2.3.14.62 Robot wire search code example

```
public static int TestWireSearch(Robot robot)
{
    DescPose toolCoord = new DescPose(0, 0, 200, 0, 0, 0);
    robot.SetToolCoord(1, toolCoord, 0, 0, 1, 0);
    DescPose wobjCoord = new DescPose(0, 0, 0, 0, 0, 0);
    robot.SetWObjCoord(1, wobjCoord, 0);

    int rtn0, rtn1, rtn2 = 0;
    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    DescPose descStart = new DescPose(216.543, 445.175, 93.465, 179.683, 1.757, -112.
↪641);
    JointPos jointStart = new JointPos(-128.345, -86.660, 114.679, -119.625, -89.219, 74.
```

(continues on next page)

(continued from previous page)

```
↪303);

    DescPose descEnd = new DescPose(111.143, 523.384, 87.659, 179.703, 1.835, -97.750);
    JointPos jointEnd = new JointPos(-113.454, -81.060, 109.328, -119.954, -89.218, 74.
↪302);

    robot.MoveL(jointStart, descStart, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 0, 0,␣
↪offdese, 0, 100);
    robot.MoveL(jointEnd, descEnd, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 0, 0, offdese,␣
↪0, 100);

    DescPose descREF0A = new DescPose(142.135, 367.604, 86.523, 179.728, 1.922, -111.
↪089);
    JointPos jointREF0A = new JointPos(-126.794, -100.834, 128.922, -119.864, -89.218,␣
↪74.302);

    DescPose descREF0B = new DescPose(254.633, 463.125, 72.604, 179.845, 2.341, -114.
↪704);
    JointPos jointREF0B = new JointPos(-130.413, -81.093, 112.044, -123.163, -89.217, 74.
↪303);

    DescPose descREF1A = new DescPose(92.556, 485.259, 47.476, -179.932, 3.130, -97.512);
    JointPos jointREF1A = new JointPos(-113.231, -83.815, 119.877, -129.092, -89.217, 74.
↪303);

    DescPose descREF1B = new DescPose(203.103, 583.836, 63.909, 179.991, 2.854, -103.
↪372);
    JointPos jointREF1B = new JointPos(-119.088, -69.676, 98.692, -121.761, -89.219, 74.
↪303);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF0A, descREF0A, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 0, 0,␣
↪offdese, 0, 10);  // Start point
    robot.MoveL(jointREF0B, descREF0B, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 1, 0,␣
↪offdese, 0, 10);  // Direction point
    rtn1 = robot.WireSearchWait("REF0");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF1A, descREF1A, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 0, 0,␣
↪offdese, 0, 10);  // Start point
    robot.MoveL(jointREF1B, descREF1B, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 1, 0,␣
↪offdese, 0, 10);  // Direction point
    rtn1 = robot.WireSearchWait("REF1");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF0A, descREF0A, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 0, 0,␣
↪offdese, 0, 10);  // Start point
    robot.MoveL(jointREF0B, descREF0B, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 1, 0,␣
↪offdese, 0, 10);  // Direction point
    rtn1 = robot.WireSearchWait("RES0");
```

(continues on next page)

(continued from previous page)

```
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    rtn0 = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0);
    robot.MoveL(jointREF1A, descREF1A, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 0, 0,␣
↪offdese, 0, 10);  // Start point
    robot.MoveL(jointREF1B, descREF1B, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 1, 0,␣
↪offdese, 0, 10);  // Direction point
    rtn1 = robot.WireSearchWait("RES1");
    rtn2 = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0);

    String[] varNameRef = new String[]{"REF0", "REF1", "#", "#", "#", "#"};
    String[] varNameRes = new String[]{"RES0", "RES1", "#", "#", "#", "#"};
    int offectFlag = 0;

    DescPose pos = new DescPose(0, 0, 0, 0, 0, 0);
    DescOffset offectPos = new DescOffset();
    offectPos.offset = pos;
    offectPos.offsetFlag = 0;

    rtn0 = robot.GetWireSearchOffset(0, 0, varNameRef, varNameRes, offectPos);
    robot.PointsOffsetEnable(0, pos);
    robot.MoveL(jointStart, descStart, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 0, 0,␣
↪offdese, 0, 10);
    robot.MoveL(jointEnd, descEnd, 1, 1, 100, 100, 100, -1, 0, exaxisPos, 1, 0, offdese,␣
↪0, 10);
    robot.PointsOffsetDisable();

    robot.CloseRPC();
    return 0;
}
```

2.3.14.63 Set welding voltage gradual change start

New in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Set welding voltage gradual change start
* @param [in] IOType Control type:
*   0-Controller IO
*   1-Digital communication protocol (UDP)
*   2-Digital communication protocol (ModbusTCP)
* @param [in] voltageStart Starting welding voltage (V)
* @param [in] voltageEnd Ending welding voltage (V)
* @param [in] AOIndex Controller AO port number (0-1)
* @param [in] blend Smoothing: 0-No smoothing; 1-Smoothing
* @return Error code
*/
int WeldingSetVoltageGradualChangeStart(int IOType, double voltageStart, double␣
↪voltageEnd, int AOIndex, int blend);
```

2.3.14.64 Set welding voltage gradual change end

New in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Set welding voltage gradual change end
* @return Error code
*/
int WeldingSetVoltageGradualChangeEnd();
```

2.3.14.65 Set welding current gradual change start

New in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Set welding current gradual change start
* @param [in] IOType Control type:
*   0-Controller IO
*   1-Digital communication protocol (UDP)
*   2-Digital communication protocol (ModbusTCP)
* @param [in] currentStart Starting welding current (A)
* @param [in] currentEnd Ending welding current (A)
* @param [in] AOIndex Controller AO port number (0-1)
* @param [in] blend Smoothing: 0-No smoothing; 1-Smoothing
* @return Error code
*/
int WeldingSetCurrentGradualChangeStart(int IOType, double currentStart, double␣
↪currentEnd, int AOIndex, int blend);
```

2.3.14.66 Set welding current gradual change end

New in version Java: SDK-v1.0.5-3.8.2

```
/**
* @brief Set welding current gradual change end
* @return Error code
*/
int WeldingSetCurrentGradualChangeEnd();
```

2.3.14.67 Robot welding current/voltage gradual change code example

```
public static void WeldparamChange(Robot robot)
{
    DescPose startdescPose = new DescPose(-484.707, 276.996, -14.013, -37.657, -40.508, -
↪1.548);
    JointPos startjointPos = new JointPos(-45.421, -75.673, 93.627, -104.302, -87.938, 6.
↪005);

    DescPose enddescPose = new DescPose(-508.767, 137.109, -13.966, -37.639, -40.508, -1.
↪559);
```

(continues on next page)

(continued from previous page)

```
    JointPos endjointPos = new JointPos(-32.768, -80.947, 100.254, -106.201, -87.201, 18.
↪648);

    DescPose safedescPose = new DescPose(-484.709, 294.436, 13.621, -37.660, -40.508, -1.
↪545);
    JointPos safejointPos = new JointPos(-46.604, -75.410, 89.109, -100.003, -88.012, 4.
↪823);

    ExaxisPos exaxisPos = new ExaxisPos(0, 0, 0, 0);
    DescPose offdese = new DescPose(0, 0, 0, 0, 0, 0);

    WeldCurrentAORelation cur = new WeldCurrentAORelation(0, 495, 1, 10, 0);
    WeldVoltageAORelation vol = new WeldVoltageAORelation(10, 45, 1, 10, 1);
    robot.WeldingSetCurrentRelation(cur);
    robot.WeldingSetVoltageRelation(vol);

    robot.WeldingSetVoltage(0, 25, 1, 0);
    robot.WeldingSetCurrent(0, 260, 0, 0);

    robot.MoveJ(safejointPos, safedescPose, 1, 0, 5, 100, 100, exaxisPos, -1, 0,␣
↪offdese);

    robot.WeldingSetCurrentGradualChangeStart(0, 260, 220, 0, 0);
    robot.WeldingSetVoltageGradualChangeStart(0, 25, 22, 1, 0);
    int rtn = robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1,␣
↪0, 4, 1, 10, 0, 0);

    robot.MoveJ(startjointPos, startdescPose, 1, 0, 5, 100, 100, exaxisPos, -1, 0,␣
↪offdese);
    System.out.println("ArcWeldTraceControl rtn is " + rtn);

    robot.ARCStart(0, 0, 10000);
    robot.WeaveStart(0);
    robot.WeaveChangeStart(2, 1, 24, 36);
    robot.MoveL(endjointPos, enddescPose, 1, 0, 100, 100, 2, -1, 0, exaxisPos, 0, 0,␣
↪offdese, 0, 10);
    robot.ARCEnd(0, 0, 10000);
    robot.WeaveChangeEnd();
    robot.WeaveEnd(0);
    robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1,␣
↪10, 0, 0);
    robot.WeldingSetCurrentGradualChangeEnd();
    robot.WeldingSetVoltageGradualChangeEnd();
}
```

2.3.14.68 Set Custom Weaving Parameters

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Set custom weaving parameters
 * @param [in] id Custom weaving number: 0-2
 * @param [in] pointNum Number of weaving points: 0-10
 * @param [in] point Moving endpoint data x,y,z
 * @param [in] stayTime Weaving dwell time in ms
 * @param [in] frequency Weaving frequency in Hz
 * @param [in] incStayType Wait mode: 0-cycle does not include wait time; 1-cycle␣
↪includes wait time
 * @param [in] stationary Weaving position wait: 0-continue motion during wait time; 1-
↪position static during wait time
 * @return  Error code
 */
public int CustomWeaveSetPara(int id, int pointNum, DescTran[] point, double[] stayTime,␣
↪double frequency, int incStayType, int stationary)
```

2.3.14.69 Get Custom Weaving Parameters

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Get custom weaving parameters
 * @param [in] id Custom weaving number: 0-2
 * @param [out] pointNum Number of weaving points: 0-10
 * @param [out] point Moving endpoint data x,y,z
 * @param [out] stayTime Weaving dwell time in ms
 * @param [out] frequency Weaving frequency in Hz
 * @param [out] incStayType Wait mode: 0-cycle does not include wait time; 1-cycle␣
↪includes wait time
 * @param [out] stationary Weaving position wait: 0-continue motion during wait time; 1-
↪position static during wait time
 * @return  Error code
 */
public int CustomWeaveGetPara(int id, int[] pointNum, DescTran[] point, double[]␣
↪stayTime, double[] frequency, int[] incStayType, int[] stationary)
```

2.3.14.70 Custom Weaving Parameters Code Example

```
public static void TestCustomWeaveSetPara(Robot robot)
{
    DescTran[] point = new DescTran[10];
    point[0]=new DescTran();
    point[0].x = -3;
    point[0].y = -3;
    point[0].z = 0;

    point[1]=new DescTran();
```

(continues on next page)

(continued from previous page)

```
    point[1].x = -6;
    point[1].y = 0;
    point[1].z = 0;

    point[2]=new DescTran();
    point[2].x = -3;
    point[2].y = 3;
    point[2].z = 0;

    point[3]=new DescTran();
    point[3].x = 0;
    point[3].y = 0;
    point[3].z = 0;
    point[4]=new DescTran(0,0,0);
    point[5]=new DescTran(0,0,0);
    point[6]=new DescTran(0,0,0);
    point[7]=new DescTran(0,0,0);
    point[8]=new DescTran(0,0,0);
    point[9]=new DescTran(0,0,0);

    double[] stayTime =new double[] { 0,0,0,0,0,0,0,0,0,0 };
    int rtn = robot.CustomWeaveSetPara(2, 4, point, stayTime, 1.000, 0, 0);
    System.out.println("CustomWeaveSetPara rtn is: "+ rtn);
    robot.Sleep(1000);

    int[] pointNum = new int[1];
    double[] frequency=new double[1];
    int[] incStayType=new int[1];
    int[] stationary=new int[1];
    robot.CustomWeaveGetPara(2, pointNum, point, stayTime, frequency, incStayType,␣
↪stationary);
    System.out.println("pointNum is: "+ pointNum[0]);
    for (int i = 0; i < pointNum[0]; i++)
    {
        System.out.println("point: "+i+", "+ point[i].x+", "+ point[i].y+", "+ point[i].
↪z);
    }
    System.out.println("frequency is: "+ frequency[0]+", incStayType is: "+␣
↪incStayType[0]+", stationary is: "+ stationary[0]);

    robot.WeaveSetPara(0, 9, 1.000000, 1, 5.000000,
            6.000000, 5.000000, 50, 100, 100,
            0, 1, 0.000000, 0.000000);

    DescPose desc_p1 =new DescPose(-288.650, 367.807, 288.404, 0.000, -0.001, 0.001 );
    DescPose desc_p2 = new DescPose( -431.714, 367.815, 288.415, 0.001, 0.001, 0.000 );
    DescPose desc_p3 = new DescPose( -348.666, 427.798, 288.404, -0.000, -0.000, 0.001 );
    JointPos j1 = new JointPos( 140.656, -84.560, -91.707, -93.734, 90.000, 50.655 );
    JointPos j2 = new JointPos( 149.873, -98.298, -77.599, -94.103, 90.000, 59.873 );
    JointPos j3 = new JointPos( 139.773, -96.173, -80.014, -93.814, 90.000, 49.772 );

    ExaxisPos epos = new ExaxisPos();
```

(continues on next page)

(continued from previous page)

```
    DescPose offset_pos = new DescPose();

    robot.MoveJ(j1, desc_p1, 3, 0, 100, 100,100, epos, -1, 0, offset_pos);
    robot.WeaveStart(0);
    robot.Circle(j3, desc_p3, 3, 0, 100, 100, epos, j2, desc_p2, 3, 0, 100, 100, epos,␣
↪10, -1, offset_pos,0,-1,0);
    robot.WeaveEnd(0);
    robot.MoveJ(j1, desc_p1, 3, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.WeaveStart(0);
    robot.MoveC(j3, desc_p3, 3, 0, 100, 100, epos, 0, offset_pos, j2, desc_p2, 3, 0, 100,
↪ 100, epos, 0, offset_pos, 10, -1,0);
    robot.WeaveEnd(0);
    robot.MoveJ(j1, desc_p1, 3, 0, 100, 100, 100, epos, -1, 0, offset_pos);
    robot.WeaveStart(0);
    robot.MoveL(j2, desc_p2, 3, 0, 100, 100, 10, -1,epos, 0, 0, offset_pos, 0,0, 100);
    robot.WeaveEnd(0);

    robot.CloseRPC();
}
```

2.3.15 Other interfaces

2.3.15.1 Get SSH public key

```
/**
* @brief Get SSH public key
* @param [out] keygen Public key
* @return Error code
*/
int GetSSHKeygen(String[] keygen)
```

2.3.15.2 Issue SCP command

New in version Java: SDK-v1.0.6-3.8.3

```
/**
* @brief Issue SCP command
* @param [in] mode 0-upload (host->controller), 1-download (controller->host)
* @param [in] sshname Host username
* @param [in] sship Host IP address
* @param [in] usr_file_url Host file path
* @param [in] robot_file_url Robot controller file path
* @return Error code
*/
int SetSSHScpCmd(int mode, String sshname, String sship, String usr_file_url, String␣
↪robot_file_url)
```

2.3.15.3 Calculate MD5 value of specified file

```
/**
* @brief Calculate MD5 value of specified file
* @param [in] file_path File path including filename, default Traj folder path:"/fruser/traj/", e.g. "/fruser/traj/trajHelix_aima_1.txt"
* @param [out] md5 File MD5 value
* @return Error code
*/
int ComputeFileMD5(String file_path, String[] md5)
```

2.3.15.4 Robot SSH and MD5 command code example

```
public static int TestSSHMd5(Robot robot)
{
    String file_path= "/fruser/airlab.lua";
    String[] md5 =new String[]{""};

    String[] ssh_keygen=new String[]{""};
    int retval = robot.GetSSHKeygen(ssh_keygen);
    System.out.println(ssh_keygen[0]);

    String ssh_name = "fr";
    String ssh_ip = "192.168.58.45";
    String ssh_route = "/home/fr";
    String ssh_robot_url = "/root/robot/dhpara.config";
    retval = robot.SetSSHScpCmd(1, ssh_name, ssh_ip, ssh_route, ssh_robot_url);
    System.out.println("SetSSHScpCmd retval is:"+ retval);
    System.out.println("robot url is:"+ ssh_robot_url);

    robot.ComputeFileMD5(file_path, md5);
    System.out.println("md5 is:+"+ md5[0]);
    return 0;
}
```

2.3.15.5 Set robot port 20004 feedback period

```
/**
* @brief Set robot port 20004 feedback period
* @param [in] period Robot port 20004 feedback period(ms)
* @return  Error code
*/
public int SetRobotRealtimeStateSamplePeriod(int period)
```

2.3.15.6 Get robot port 20004 feedback period

```
/**
* @brief  Get robot port 20004 feedback period
* @return  List[0]:Error code; List[1]:Robot port 20004 feedback period(ms)
*/
public List<Integer> GetRobotRealtimeStateSamplePeriod()
```

2.3.15.7 Robot port 20004 status feedback period configuration example

```
public static int TestRealtimePeriod(Robot robot)
{
    robot.SetRobotRealtimeStateSamplePeriod(10);
    List<Integer> getPeriod = new ArrayList<>();
    getPeriod=robot.GetRobotRealtimeStateSamplePeriod();
    robot.Sleep(1000);

    return 0;
}
```

2.3.15.8 Robot software upgrade

```
/**
 * @brief Robot software upgrade
 * @param [in] filePath Full path of software upgrade package
 * @param [in] block Whether to block until upgrade completes true:block; false:non-block
 * @return  Error code
 */
public int SoftwareUpgrade(String filePath, boolean block)
```

2.3.15.9 Get robot software upgrade status

```
/**
* @brief  Get robot software upgrade status
* @return  List[0]:Error code; List[1]:Robot software upgrade status 0-idle or uploading␣
↪upgrade package; 1~100: upgrade completion percentage; -1:upgrade failed; -
↪2:verification failed; -3:version verification failed; -4:decompression failed; -
↪5:user configuration upgrade failed; -6:peripheral configuration upgrade failed; -
↪7:extended axis configuration upgrade failed; -8:robot configuration upgrade failed; -
↪9:DH parameter configuration upgrade failed
*/
public List<Integer> GetSoftwareUpgradeState()
```

2.3.15.10 Robot software upgrade code example

```
public static int TestUpgrade(Robot robot)
{
    robot.SoftwareUpgrade("D://zUP/QNX382/software.tar.gz", false);
    while (true)
    {
        List<Integer> inter=new ArrayList<>();
        inter=robot.GetSoftwareUpgradeState();
        System.out.println("upgrade state is:"+ inter.get(1));
        robot.Sleep(300);
    }
}
```

2.3.15.11 Download point table database

```
/**
* @brief Download point table database
* @param [in] pointTableName Point table name to download    pointTable1.db
* @param [in] saveFilePath Storage path for downloaded point table   C://test/
* @return Error code
*/
int PointTableDownLoad(String pointTableName, String saveFilePath);
```

2.3.15.12 Upload point table database

```
/**
* @brief Upload point table database
* @param [in] pointTableFilePath Full path name of point table to upload   C://test/
↪pointTable1.db
* @return Error code
*/
int PointTableUpLoad(String pointTableFilePath);
```

2.3.15.13 Point table update lua file

```
/**
* @brief Point table update lua file
* @param [in] pointTableName Point table name to switch to   "pointTable1.db", when␣
↪empty "", means updating lua program to initial program without applied point table
* @param [in] luaFileName Lua file name to update   "testPointTable.lua"
* @param [out] errorStr Point table switching error message
* @return Error code
*/
int PointTableUpdateLua(String pointTableName, String luaFileName, String errorStr);
```

2.3.15.14 Robot point table operation code example

```
public static int TestPointTable(Robot robot)
{
    String save_path = "D://zDOWN/";
    String point_table_name = "point_table_FR5.db";
    int rtn = robot.PointTableDownLoad(point_table_name, save_path);

    String upload_path = "D://zUP/point_table_FR5.db";
    rtn = robot.PointTableUpLoad(upload_path);

    String point_tablename = "point_table_FR5.db";
    String lua_name = "airlab.lua";
    String err="";
    rtn = robot.PointTableUpdateLua(point_tablename, lua_name,err);

    robot.CloseRPC();
    return 0;
}
```

2.3.15.15 Controller log download

New in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief Controller log download
* @param [in] savePath Save file path"D://zDown/"
* @return Error code
*/
int RbLogDownload(String savePath);
```

2.3.15.16 All data source download

New in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief All data source download
* @param [in] savePath Save file path"D://zDown/"
* @return Error code
*/
int AllDataSourceDownload(String savePath);
```

2.3.15.17 Data backup package download

New in version Java: SDK-v1.0.4-3.8.1

```
/**
* @brief Data backup package download
* @param [in] savePath Save file path"D://zDown/"
* @return Error code
*/
int DataPackageDownload(String savePath);
```

2.3.15.18 Download controller data code example

```
public static int TestDownLoadRobotData(Robot robot)
{
    int rtn = robot.RbLogDownload("D://zDOWN/");

    rtn = robot.AllDataSourceDownload("D://zDOWN/");

    rtn = robot.DataPackageDownload("D://zDOWN/");
    return 0;
}
```

2.3.15.19 Set Encoder Upgrade

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Set encoder upgrade
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
int SetEncoderUpgrade(String path)
```

2.3.15.20 Set Joint Firmware Upgrade

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Set joint firmware upgrade
* @param [in] type Upgrade file type: 1-Firmware upgrade; 2-Slave configuration file␣
↪upgrade
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
public int SetJointFirmwareUpgrade(int type, String path)
```

2.3.15.21 Set Controller Firmware Upgrade

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Set controller firmware upgrade
* @param [in] type Upgrade file type: 1-Firmware upgrade; 2-Slave configuration file upgrade
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
public int SetCtrlFirmwareUpgrade(int type, String path)
```

2.3.15.22 Set End-Effector Firmware Upgrade

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Set end-effector firmware upgrade
* @param [in] type Upgrade file type: 1-Firmware upgrade; 2-Slave configuration file upgrade
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
public int SetEndFirmwareUpgrade(int type, String path)
```

2.3.15.23 Joint Complete Parameter Configuration Upgrade

New in version Java: SDK-v1.0.7-3.8.4

```
/**
* @brief Joint complete parameter configuration upgrade
* @param [in] path Full local upgrade package path (D://zUP/XXXXX.bin)
* @return Error code
*/
public int JointAllParamUpgrade(String path)
```

2.3.15.24 Robot Slave Firmware Upgrade Code Example

```
public static void TestFirmWareUpgrade(Robot robot)
{
    robot.RobotEnable(0);
    robot.Sleep(200);
    int rtn = robot.JointAllParamUpgrade("D://zUP/standardQX/jointallparametersFR56.0.db");
    System.out.println("robot JointAllParamUpgrade rtn is:"+ rtn);

    rtn = robot.SetCtrlFirmwareUpgrade(2, "D://zUP/upgrade/FAIR_Cobot_Cbd_Asix_V2.0.bin");
    System.out.println("robot SetCtrlFirmwareUpgrade config param rtn is:"+ rtn);
```

(continues on next page)

(continued from previous page)

```

    rtn = robot.SetEndFirmwareUpgrade(2, "D://zUP/upgrade/FAIR_Cobot_Axle_Asix_V2.4.bin
↪");
    System.out.println("robot SetEndFirmwareUpgrade config param rtn is:"+ rtn);

    robot.SetSysServoBootMode();
    rtn = robot.SetCtrlFirmwareUpgrade(1, "D://zUP/standardQX/FR_CTRL_PRIMCU_FV201010_
↪MAIN_U4_T01_20240529.bin");
    System.out.println("robot SetCtrlFirmwareUpgrade rtn is:"+ rtn);

    rtn = robot.SetEndFirmwareUpgrade(1, "D://zUP/standardQX/FR_END_FV201010_MAIN_U01_
↪T01_20250522.bin");
    System.out.println("robot SetEndFirmwareUpgrade rtn is:"+ rtn);

    rtn = robot.SetJointFirmwareUpgrade(1, "D://zUP/standardQX/FR_SERVO_FV502211_MAIN_U7_
↪T07_20250217.bin");
    System.out.println("robot SetJointFirmwareUpgrade rtn is:"+ rtn);

    robot.CloseRPC();
}
```

2.3.15.25 Robot Operating System Upgrade (LA Control Box)

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Robot Operating System Upgrade (LA Control Box)
 * @param [in] filePath Full path of the operating system upgrade package
 * @return Error code
 */
public int KernelUpgrade(String filePath)
```

2.3.15.26 Get Robot Operating System Upgrade Result (LA Control Box)

New in version Java: SDK-v1.0.9-3.8.6

```
/**
 * @brief Get Robot Operating System Upgrade Result (LA Control Box)
 * @param [out] result Upgrade result: 0: Success; -1: Failure
 * @return Error code
 */
public int GetKernelUpgradeResult(int[] result)
```

2.3.15.27 Robot MCU Log Generation

```
/**
* @brief Robot MCU log generation
* @return Error code
*/
public int RobotMCULogCollect()
```

2.3.16 Appendix

2.3.16.1 Source Code Download

Find the "Downloads" section in the FAIRINO documentation (https://fairino-doc-en.readthedocs.io/latest/), click the "Java SDK" button, and then click "FAIRINO Java SDK" on the right-hand page. Wait for the browser to complete the download.



Figure 16.1-1 Java SDK Source Code Download

Extract the compressed package. The file directory is as shown in the figure, which includes:

fairino_Java_SDK_maven: Source code (.java) and library files (.jar) compiled in a Windows system environment.

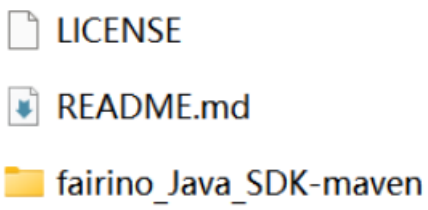


Figure 16.1-2 Java SDK File Directory

Enter the fairino_Java_SDK_maven folder, which contains the following directories:

- lib: Dependency JAR files used in the source code;
- src: Java SDK source code files;
- target: Library files (.jar) generated from the Java SDK source code;



名称	修改日期	类型	大小
.idea	2025/4/27 10:30	文件夹	
lib	2025/4/27 10:30	文件夹	
src	2025/4/27 10:30	文件夹	
target	2025/6/13 17:59	文件夹	
.gitignore	2025/4/27 9:59	Git Ignore 源文件	1 KB
fairino_Java_SDK.iml	2024/10/14 15:16	IML 文件	1 KB
pom.xml	2024/10/14 15:16	SLBrowser HTML D...	4 KB

Figure 16.1-3 Java SDK Source Code and Library File Directory

2.3.16.2 Windows Platform Source Code Compilation

Install and configure the build tool—Maven

Maven download and installation website: Welcome to Apache Maven – Maven

After installation and configuration, the terminal will display the following information when you enter *maven –version*:



Figure 16.2-1 Maven Installation and Configuration

Open a terminal in the Java SDK source code directory and enter *mvn package* to generate the library files (.jar).

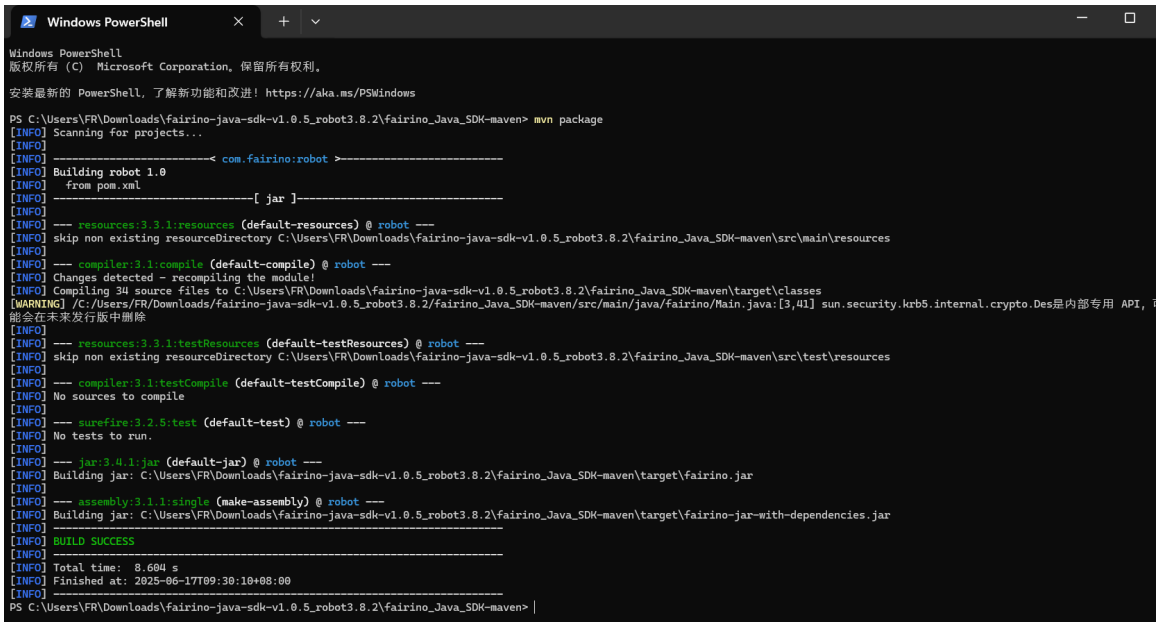


Figure 16.2-2 Compiling Java SDK into Library Files

Locate the "target" folder in the source code directory and find the compiled *fairino-jar-with-dependencies.jar* and *fairino.jar* files, as shown in the figure.

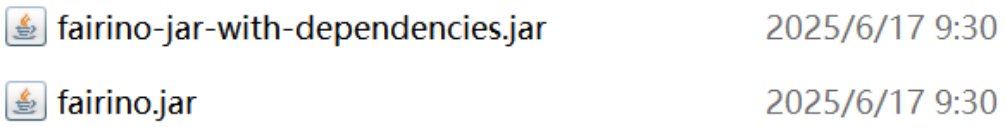


Figure 16.2-3 Generated JAR Files

To use the collaborative robot Java SDK, first open the project in IntelliJ IDEA, then navigate to File -> Project Structure -> Libraries, and add the .jar files generated in the previous step. Use *import fairino.*;* in your files to utilize the generated .jar files.

2.4 Python

This manual is the secondary development interface document of Python.

Important: Robot parameter unit description: The robot position unit is millimeter (mm), and the attitude unit is degree (°).

Important:

1) In code examples that are not specifically stated, the robot has been powered on and enabled by default;
2) All code examples in the documentation default to no interference within the robot's workspace;
3) Please use the data of the on-site robot in the actual use test.

Note: The current document applies to SDK-v2.0.0 version and is backward compatible with v1.x version.

2.4.1 Version Update Description

Version	Date	Update Description
V3.9.3	2026-02-11	1.ServoCart() interface added extended axis parameters 2.SetOutputResetCtlBoxDO() interface added parameter for whether to reload DO state before reset after pause resume 3.SetOutputResetCtlBoxAO() interface added parameter for whether to reload DO state before reset after pause resume 4.SetOutputResetAxleDO() interface added parameter for whether to reload DO state before reset after pause resume 5.SetOutputResetAxleAO() interface added parameter for whether to reload DO state before reset after pause resume 6.SetOutputResetExtDO() interface added parameter for whether to reload DO state before reset after pause resume 7.SetOutputResetExtAO() interface added parameter for whether to reload DO state before reset after pause resume 8.SetOutputResetSmartToolDO() interface added parameter for whether to reload DO state before reset after pause resume 9.Added GetInverseKinExaxis() inverse kinematics solution interface including extended axis position
V3.9.2	2026-01-26	1. Added a processing strategy parameter for undetected force/torque to the FT_RotInsertion() interface 2. Added robot fixed-point tracking related parameters to the LaserSensorRecordandReplay() interface 3. Added the MoveStationary() interface 4. Added the TCPComputeRPY() interface 5. Added the TCPComputeXYZ() interface 6. Added the TCPRecordFlangePosStart() interface 7. Added the TCPRecordFlangePosEnd() interface 8. Added the TCPGetRecordFlangePos() interface 9. Added the PhotoelectricSensorTCPCalibration() interface
V3.9.1	2025-12-25	1. Added oacc speed scaling factor parameter / physical acceleration parameter to the MoveL() interface; 2. Added oacc speed scaling factor parameter / physical acceleration parameter to the MoveC() interface; 3. Optimized parameter descriptions for physical speed and physical acceleration in the Circle() interface; 4. Added FT_Control() overloaded function with rx, ry activation thresholds and torque adjustment coefficient parameters; 5. Added SerCoderCompenParams() interface;
V3.9.0	2025-11-26	1. JointSensitivityCalibration() interface adds j1~j6 joint linearity return 2. Added JointHysteresisError() interface 3. Added JointRepeatability() interface 4. Added SetAdmittanceParams() interface 5. Added MoveToIntersectLineStart() interface

2.4.2 Data structure description

2.4.2.1 Controller status feedback data packet

New in version python: SDK-v2.1.7

Variable	Meaning
program_state	program_running_state, 1-stop; 2-run; 3-pause
robot_state	robot_motion_state, 1-stop; 2-run; 3-pause; 4-drag
main_code	main_fault_code
sub_code	sub_code”
robot_mode	robot_mode, 0-automatic mode; 1-manual mode
jt_cur_pos[i]	Current position of joint, in deg, i:0~5
tl_cur_pos[i]	Tool current position in deg&mm,i:0~5
flange_cur_pos[i]	The end flange is in its current position, in degrees and millimeters. i: 0~5
actual_qd[i]	Robot’s current joint velocity in deg/s,i:0~5
actual_qdd[i]	Current joint acceleration of the robot, in deg/s^2, i:0~5
target_TCP_CmpSpeed[i]	Robot TCP synthesis command speed in mm/s & deg/s,i:0~1
target_TCP_Speed[i]	Robot TCP command speed in mm/s & deg/s,i:0~5
actual_TCP_CmpSpeed[i]	Robot TCP synthesized actual speed in mm/s & deg/s, i:0~1
actual_TCP_Speed[i]	Robot TCP actual speed in mm/s & deg/s,i:0~5
jt_cur_tor[i]	Current torque, unit N-m ,i:0~5
tool	Applied tool coordinate system number
user	Applied workpiece coordinate system number
cl_dgt_output_h	Control Box Digital IO Output 15-8
cl_dgt_output_l	Control Box Digital IO Output 7-0
tl_dgt_output_l	Tool Digital IO Output 7-0, only bit0-bit1 valid
dgt_input_h	Control Box Digital IO Input 15-8
cl_dgt_input_l	Control Box Digital IO Input 7-0
tl_dgt_input_l	Tool digital IO input 7-0, only bit0-bit1 valid
cl_analog_input[i]	Control box analog input,i:0~2
tl_anglog_input	tool_analog_input
ft_sensor_raw_data	torque sensor raw data, unit N&Nm, i:0~5
ft_sensor_data	torque sensor data, unit N&Nm, i:0~5
ft_sensor_active	torque sensor active status, 0-reset, 1-active
EmergencyStop	Emergency stop sign, 0 - emergency stop not pressed, 1 - emergency stop pressed
motion_done	motion_in_place signal,1-in place, 0-not in place
gripper_motiondone	Gripper motion done signal,1-done, 0-not done
mc_queue_len	motion command queue length
collisionState	Collision detection, 1-collision, 0-no collision
trajectory_pnum	trajectory point number
safety_stop0_state	safety stop signal SI0
safety_stop1_state	safety stop signal SI1
gripper_fault_id	error_claw_number
gripper_fault	gripper_fault
gripper_active	gripper_jaw_activity_status, 0-unactivated, 1-activated
gripper_position	Gripper position (percentage)
gripper_speed	Gripper speed (percentage)
gripper_current	gripper_current (percentage)
gripper_tmp	Gripper temperature in °C

continues on next page

Table 1 – continued from previous page

Variable	Meaning
gripper_voltage	gripper_voltage in V
auxState.servoId	485 extended axis, servo drive ID number, i:0~3
auxState.servoErrCode	485 Extended Axis, Servo Drive Error Code, i:0~3
auxState.servoState	485 Extended Axis, Servo Drive State, i:0~3
auxState.servoPos	485 extended axis, servo current position, i:0~3
auxState.servoVel	485 extended axis, servo current speed, i:0~3
auxState.servoTorque	485 Extended axis, servo current torque, i:0~3
extAxisStatus[i].pos	UDP Extension Axis, Position, i:0~3
extAxisStatus[i].vel	UDP Extended Axis,velocity,i:0~3
extAxisStatus[i].errorCode	UDP Extended Axis, Error Code, i:0~3
extAxisStatus[i].ready	UDP extension axis, servo ready, i:0~3
extAxisStatus[i].inPos	UDP Extended Axis, servo in place, i:0~3
extAxisStatus[i].alarm	UDP Extended Axis, Servo Alarm, i:0~3
extAxisStatus[i].flerr	UDP Extended Axis, Follow Error, i:0~3
extAxisStatus[i].nlimit	UDP extension axis, to negative limit, i:0~3
extAxisStatus[i].pLimit	UDP extension axis, to positive limit, i:0~3
extAxisStatus[i].mdbsOffLine	UDP Extension Axis, Drive 485 Bus Offline
extAxisStatus[i].mdbsTimeout	UDP Extension Axis, Control Card and Control Box 485 Communication Timeout
extAxisStatus[i].homingStatus	UDP extension axis, back to zero status
extDIState	Extended Digital Input State
extDOState	Extended Digital Output State
extAIState	Extended Analog Input State
extAOState	Extended Analog Output State
rbtEnableState	robotEnableState
jointDriverTorque	Joint Driver Current Torque
jointDriverTemperature	Joint Driver Current Temperature
year	year
mouth	moon
day	day
hour	hours
minute	minutes
second	seconds
millisecond	milliseconds
softwareUpgradeState	Robot Software Upgrade State
endLuaErrCode	endLUARunningStatus
cl_analog_output[i]	Control box analog output,i:0~1
tl_analog_output	Tool analog output
gripperRotNum	Rotation gripper current rotation number
gripperRotSpeed	Rotation gripper current rotation speed percentage
gripperRotTorque	Rotation gripper current rotation torque percentage
weldingBreakOffState	Welding interruption state
jt_tgt_tor	Joint command torque
smartToolState	The status of the SmartTool handle buttons
wideVoltageCtrlBoxTemp	Wide voltage control box temperature
wideVoltageCtrlBoxFanCurrent	Wide voltage control box fan current(ma)
toolCoord[i]	Tool coordinate system,i:0~5
wobjCoord[i]	Workpiece coordinate system,i:0~5
extoolCoord[i]	External tool coordinate system,i:0~5
exAxisCoord[i]	Extended axis coordinate system,i:0~5

continues on next page

Table 1 – continued from previous page

Variable	Meaning
load	Quality of load
loadCog[i]	Center of mass of load,i:0~2
lastServoTarget[i]	The last ServoJ target position in the queue,i:0 to 5
servoJCmdNum	ServoJ instruction count

2.4.2.2 Status of the servo controller

New in version python: SDK-v2.1.3

Variable	Meaning
servoId	Servo driver ID number
servoErrCode	Servo driver fault code
servoState	Status of the servo driver
servoPos	Current position of the servo
servoVel	Current speed of the servo
servoTorque	Current torque of the servo

2.4.2.3 Extended axis status

New in version python: SDK-v2.1.3

Variable	Meaning
pos	Extended axis position
vel	Extended axis speed
errorCode	Extended shaft fault code
ready	The servo is ready
inPos	Servo in place
alarm	Servo alarm
flerr	Following error
nlimit	To the negative limit
pLimit	To the positive limit position
mdbsOffLine	The 485 bus of the driver is disconnected
mdbsTimeout	The 485 communication between the control card and the control box has timed out
homingStatus	The expansion axis returns to the zero state

2.4.2.4 Welding interruption state

New in version python: SDK-v2.1.3

Variable	Meaning
breakOffState	Welding interruption state
weldArcState	Welding arc interruption state

2.4.2.5 code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
print("program_state:", robot.robot_state_pkg.program_state)
print("robot_state:", robot.robot_state_pkg.robot_state)
print("main_code:", robot.robot_state_pkg.main_code)
print("sub_code:", robot.robot_state_pkg.sub_code)
print("robot_mode:", robot.robot_state_pkg.robot_mode)
print("jt_cur_pos0:", robot.robot_state_pkg.jt_cur_pos[0])
print("jt_cur_pos1:", robot.robot_state_pkg.jt_cur_pos[1])
print("jt_cur_pos2:", robot.robot_state_pkg.jt_cur_pos[2])
print("jt_cur_pos3:", robot.robot_state_pkg.jt_cur_pos[3])
print("jt_cur_pos4:", robot.robot_state_pkg.jt_cur_pos[4])
print("jt_cur_pos5:", robot.robot_state_pkg.jt_cur_pos[5])
print("tl_cur_pos0:", robot.robot_state_pkg.tl_cur_pos[0])
print("tl_cur_pos1:", robot.robot_state_pkg.tl_cur_pos[1])
print("tl_cur_pos2:", robot.robot_state_pkg.tl_cur_pos[2])
print("tl_cur_pos3:", robot.robot_state_pkg.tl_cur_pos[3])
print("tl_cur_pos4:", robot.robot_state_pkg.tl_cur_pos[4])
print("tl_cur_pos5:", robot.robot_state_pkg.tl_cur_pos[5])
print("flange_cur_pos0:", robot.robot_state_pkg.flange_cur_pos[0])
print("flange_cur_pos1:", robot.robot_state_pkg.flange_cur_pos[1])
print("flange_cur_pos2:", robot.robot_state_pkg.flange_cur_pos[2])
print("flange_cur_pos3:", robot.robot_state_pkg.flange_cur_pos[3])
print("flange_cur_pos4:", robot.robot_state_pkg.flange_cur_pos[4])
print("flange_cur_pos5:", robot.robot_state_pkg.flange_cur_pos[5])
print("actual_qd0:", robot.robot_state_pkg.actual_qd[0])
print("actual_qd1:", robot.robot_state_pkg.actual_qd[1])
print("actual_qd2:", robot.robot_state_pkg.actual_qd[2])
print("actual_qd3:", robot.robot_state_pkg.actual_qd[3])
print("actual_qd4:", robot.robot_state_pkg.actual_qd[4])
print("actual_qd5:", robot.robot_state_pkg.actual_qd[5])
print("actual_qdd0:", robot.robot_state_pkg.actual_qdd[0])
print("actual_qdd1:", robot.robot_state_pkg.actual_qdd[1])
print("actual_qdd2:", robot.robot_state_pkg.actual_qdd[2])
print("actual_qdd3:", robot.robot_state_pkg.actual_qdd[3])
print("actual_qdd4:", robot.robot_state_pkg.actual_qdd[4])
print("actual_qdd5:", robot.robot_state_pkg.actual_qdd[5])
print("target_TCP_CmpSpeed0:", robot.robot_state_pkg.target_TCP_CmpSpeed[0])
print("target_TCP_CmpSpeed1:", robot.robot_state_pkg.target_TCP_CmpSpeed[1])
print("target_TCP_Speed0:", robot.robot_state_pkg.target_TCP_Speed[0])
print("target_TCP_Speed1:", robot.robot_state_pkg.target_TCP_Speed[1])
print("target_TCP_Speed2:", robot.robot_state_pkg.target_TCP_Speed[2])
print("target_TCP_Speed3:", robot.robot_state_pkg.target_TCP_Speed[3])
print("target_TCP_Speed4:", robot.robot_state_pkg.target_TCP_Speed[4])
print("target_TCP_Speed5:", robot.robot_state_pkg.target_TCP_Speed[5])
print("actual_TCP_CmpSpeed0:", robot.robot_state_pkg.actual_TCP_CmpSpeed[0])
print("actual_TCP_CmpSpeed1:", robot.robot_state_pkg.actual_TCP_CmpSpeed[1])
print("actual_TCP_Speed0:", robot.robot_state_pkg.actual_TCP_Speed[0])
print("actual_TCP_Speed1:", robot.robot_state_pkg.actual_TCP_Speed[1])
```

(continues on next page)

(continued from previous page)

```
print("actual_TCP_Speed2:", robot.robot_state_pkg.actual_TCP_Speed[2])
print("actual_TCP_Speed3:", robot.robot_state_pkg.actual_TCP_Speed[3])
print("actual_TCP_Speed4:", robot.robot_state_pkg.actual_TCP_Speed[4])
print("actual_TCP_Speed5:", robot.robot_state_pkg.actual_TCP_Speed[5])
print("jt_cur_tor0:", robot.robot_state_pkg.jt_cur_tor[0])
print("jt_cur_tor1:", robot.robot_state_pkg.jt_cur_tor[1])
print("jt_cur_tor2:", robot.robot_state_pkg.jt_cur_tor[2])
print("jt_cur_tor3:", robot.robot_state_pkg.jt_cur_tor[3])
print("jt_cur_tor4:", robot.robot_state_pkg.jt_cur_tor[4])
print("jt_cur_tor5:", robot.robot_state_pkg.jt_cur_tor[5])
print("tool:", robot.robot_state_pkg.tool)
print("user:", robot.robot_state_pkg.user)
print("cl_dgt_output_h:", robot.robot_state_pkg.cl_dgt_output_h)
print("cl_dgt_output_l:", robot.robot_state_pkg.cl_dgt_output_l)
print("tl_dgt_output_l:", robot.robot_state_pkg.tl_dgt_output_l)
print("cl_dgt_input_h:", robot.robot_state_pkg.cl_dgt_input_h)
print("cl_dgt_input_l:", robot.robot_state_pkg.cl_dgt_input_l)
print("tl_dgt_input_l:", robot.robot_state_pkg.tl_dgt_input_l)
print("cl_analog_input0:", robot.robot_state_pkg.cl_analog_input[0])
print("cl_analog_input1:", robot.robot_state_pkg.cl_analog_input[1])
print("tl_anglog_input:", robot.robot_state_pkg.tl_anglog_input)
print("ft_sensor_raw_data0:", robot.robot_state_pkg.ft_sensor_raw_data[0])
print("ft_sensor_raw_data1:", robot.robot_state_pkg.ft_sensor_raw_data[1])
print("ft_sensor_raw_data2:", robot.robot_state_pkg.ft_sensor_raw_data[2])
print("ft_sensor_raw_data3:", robot.robot_state_pkg.ft_sensor_raw_data[3])
print("ft_sensor_raw_data4:", robot.robot_state_pkg.ft_sensor_raw_data[4])
print("ft_sensor_raw_data5:", robot.robot_state_pkg.ft_sensor_raw_data[5])
print("ft_sensor_data0:", robot.robot_state_pkg.ft_sensor_data[0])
print("ft_sensor_data1:", robot.robot_state_pkg.ft_sensor_data[1])
print("ft_sensor_data2:", robot.robot_state_pkg.ft_sensor_data[2])
print("ft_sensor_data3:", robot.robot_state_pkg.ft_sensor_data[3])
print("ft_sensor_data4:", robot.robot_state_pkg.ft_sensor_data[4])
print("ft_sensor_data5:", robot.robot_state_pkg.ft_sensor_data[5])
print("ft_sensor_active:", robot.robot_state_pkg.ft_sensor_active)
print("EmergencyStop:", robot.robot_state_pkg.EmergencyStop)
print("motion_done:", robot.robot_state_pkg.motion_done)
print("gripper_motiondone:", robot.robot_state_pkg.gripper_motiondone)
print("mc_queue_len:", robot.robot_state_pkg.mc_queue_len)
print("collisionState:", robot.robot_state_pkg.collisionState)
print("trajectory_pnum:", robot.robot_state_pkg.trajectory_pnum)
print("safety_stop0_state:", robot.robot_state_pkg.safety_stop0_state)
print("safety_stop1_state:", robot.robot_state_pkg.safety_stop1_state)
print("gripper_fault_id:", robot.robot_state_pkg.gripper_fault_id)
print("gripper_fault:", robot.robot_state_pkg.gripper_fault)
print("gripper_active:", robot.robot_state_pkg.gripper_active)
print("gripper_position:", robot.robot_state_pkg.gripper_position)
print("gripper_speed:", robot.robot_state_pkg.gripper_speed)
print("gripper_current:", robot.robot_state_pkg.gripper_current)
print("gripper_tmp:", robot.robot_state_pkg.gripper_tmp)
print("gripper_voltage:", robot.robot_state_pkg.gripper_voltage)
print("auxState.servoId:", robot.robot_state_pkg.auxState.servoId)
print("auxState.servoErrCode:", robot.robot_state_pkg.auxState.servoErrCode)
```

(continues on next page)

(continued from previous page)

```
print("auxState.servoState:", robot.robot_state_pkg.auxState.servoState)
print("auxState.servoPos:", robot.robot_state_pkg.auxState.servoPos)
print("auxState.servoVel:", robot.robot_state_pkg.auxState.servoVel)
print("auxState.servoTorque:", robot.robot_state_pkg.auxState.servoTorque)
for i in range(4):
    print("extAxisStatus.pos:", i,robot.robot_state_pkg.extAxisStatus[i].pos)
    print("extAxisStatus.vel:", i,robot.robot_state_pkg.extAxisStatus[i].vel)
    print("extAxisStatus.errorCode:", i,robot.robot_state_pkg.extAxisStatus[i].errorCode)
    print("extAxisStatus.ready:", i,robot.robot_state_pkg.extAxisStatus[i].ready)
    print("extAxisStatus.inPos:", i,robot.robot_state_pkg.extAxisStatus[i].inPos)
    print("extAxisStatus.alarm:", i,robot.robot_state_pkg.extAxisStatus[i].alarm)
    print("extAxisStatus.flerr:", i,robot.robot_state_pkg.extAxisStatus[i].flerr)
    print("extAxisStatus.nlimit:", i,robot.robot_state_pkg.extAxisStatus[i].nlimit)
    print("extAxisStatus.pLimit:", i,robot.robot_state_pkg.extAxisStatus[i].pLimit)
    print("extAxisStatus.mdbsOffLine:", i,robot.robot_state_pkg.extAxisStatus[i].
↪mdbsOffLine)
    print("extAxisStatus.mdbsTimeout:", i,robot.robot_state_pkg.extAxisStatus[i].
↪mdbsTimeout)
    print("extAxisStatus.homingStatus:", i,robot.robot_state_pkg.extAxisStatus[i].
↪homingStatus)
for i in range(8):
    print("extDIState:",i, robot.robot_state_pkg.extDIState[i])
    print("extDOState:", i,robot.robot_state_pkg.extDOState[i])
for i in range(4):
    print("extAIState:", i,robot.robot_state_pkg.extAIState[i])
    print("extAOState:", robot.robot_state_pkg.extAOState[i])
print("rbtEnableState:", robot.robot_state_pkg.rbtEnableState)
print("jointDriverTorque0:", robot.robot_state_pkg.jointDriverTorque[0])
print("jointDriverTorque1:", robot.robot_state_pkg.jointDriverTorque[1])
print("jointDriverTorque2:", robot.robot_state_pkg.jointDriverTorque[2])
print("jointDriverTorque3:", robot.robot_state_pkg.jointDriverTorque[3])
print("jointDriverTorque4:", robot.robot_state_pkg.jointDriverTorque[4])
print("jointDriverTorque5:", robot.robot_state_pkg.jointDriverTorque[5])
print("jointDriverTemperature:", robot.robot_state_pkg.jointDriverTemperature[0])
print("jointDriverTemperature:", robot.robot_state_pkg.jointDriverTemperature[1])
print("jointDriverTemperature:", robot.robot_state_pkg.jointDriverTemperature[2])
print("jointDriverTemperature:", robot.robot_state_pkg.jointDriverTemperature[3])
print("jointDriverTemperature:", robot.robot_state_pkg.jointDriverTemperature[4])
print("jointDriverTemperature:", robot.robot_state_pkg.jointDriverTemperature[5])
print("year:", robot.robot_state_pkg.year)
print("mouth:", robot.robot_state_pkg.mouth)
print("day:", robot.robot_state_pkg.day)
print("hour:", robot.robot_state_pkg.hour)
print("minute:", robot.robot_state_pkg.minute)
print("second:", robot.robot_state_pkg.second)
print("millisecond:", robot.robot_state_pkg.millisecond)
print("softwareUpgradeState:", robot.robot_state_pkg.softwareUpgradeState)
print("endLuaErrCode:", robot.robot_state_pkg.endLuaErrCode)
print("cl_analog_output[0]:", robot.robot_state_pkg.cl_analog_output[0])
print("cl_analog_output[1]:", robot.robot_state_pkg.cl_analog_output[1])
print("tl_analog_output:", robot.robot_state_pkg.tl_analog_output)
print("gripperRotNum:", robot.robot_state_pkg.gripperRotNum)
```

(continues on next page)

(continued from previous page)

```
print("gripperRotSpeed:", robot.robot_state_pkg.gripperRotSpeed)
print("gripperRotTorque:", robot.robot_state_pkg.gripperRotTorque)
print("jt_tgt_tor:", robot.robot_state_pkg.jt_tgt_tor)
print("smartToolState:", robot.robot_state_pkg.smartToolState)
print("wideVoltageCtrlBoxTemp:", robot.robot_state_pkg.wideVoltageCtrlBoxTemp)
print("wideVoltageCtrlBoxFanCurrent:", robot.robot_state_pkg.
↪wideVoltageCtrlBoxFanCurrent)
print("toolCoord0:", robot.robot_state_pkg.toolCoord[0])
print("toolCoord1:", robot.robot_state_pkg.toolCoord[1])
print("toolCoord2:", robot.robot_state_pkg.toolCoord[2])
print("toolCoord3:", robot.robot_state_pkg.toolCoord[3])
print("toolCoord4:", robot.robot_state_pkg.toolCoord[4])
print("toolCoord5:", robot.robot_state_pkg.toolCoord[5])
print("wobjCoord0:", robot.robot_state_pkg.wobjCoord[0])
print("wobjCoord1:", robot.robot_state_pkg.wobjCoord[1])
print("wobjCoord2:", robot.robot_state_pkg.wobjCoord[2])
print("wobjCoord3:", robot.robot_state_pkg.wobjCoord[3])
print("wobjCoord4:", robot.robot_state_pkg.wobjCoord[4])
print("wobjCoord5:", robot.robot_state_pkg.wobjCoord[5])
print("extoolCoord0:", robot.robot_state_pkg.extoolCoord[0])
print("extoolCoord1:", robot.robot_state_pkg.extoolCoord[1])
print("extoolCoord2:", robot.robot_state_pkg.extoolCoord[2])
print("extoolCoord3:", robot.robot_state_pkg.extoolCoord[3])
print("extoolCoord4:", robot.robot_state_pkg.extoolCoord[4])
print("extoolCoord5:", robot.robot_state_pkg.extoolCoord[5])
print("exAxisCoord0:", robot.robot_state_pkg.exAxisCoord[0])
print("exAxisCoord1:", robot.robot_state_pkg.exAxisCoord[1])
print("exAxisCoord2:", robot.robot_state_pkg.exAxisCoord[2])
print("exAxisCoord3:", robot.robot_state_pkg.exAxisCoord[3])
print("exAxisCoord4:", robot.robot_state_pkg.exAxisCoord[4])
print("exAxisCoord5:", robot.robot_state_pkg.exAxisCoord[5])
print("load:", robot.robot_state_pkg.load)
print("loadCog0:", robot.robot_state_pkg.loadCog[0])
print("loadCog1:", robot.robot_state_pkg.loadCog[1])
print("loadCog2:", robot.robot_state_pkg.loadCog[2])
print("lastServoTarget0:", robot.robot_state_pkg.lastServoTarget[0])
print("lastServoTarget1:", robot.robot_state_pkg.lastServoTarget[1])
print("lastServoTarget2:", robot.robot_state_pkg.lastServoTarget[2])
print("lastServoTarget3:", robot.robot_state_pkg.lastServoTarget[3])
print("lastServoTarget4:", robot.robot_state_pkg.lastServoTarget[4])
print("lastServoTarget5:", robot.robot_state_pkg.lastServoTarget[5])
print("servoJCmdNum:", robot.robot_state_pkg.servoJCmdNum)
```

2.4.3 Robotics Basics

2.4.3.1 Instantiated Robot

Prototype	`RPC(ip)`
Description	Instantiating a robot object
Required parameter	• `ip`:The IP address of the robot, with a default factory IP of "192.168.58.2"
Optional parameter	NULL
Return value	• Success: Returns a robot object • Failed: The created object will be destroyed

2.4.3.2 Close the RPC connection

Prototype	`CloseRPC()`
Description	Close RPC connection
Mandatory parameters	NULL
Default parameters	NULL
Return Value	NULL

2.4.3.3 Query SDK version number

prototype	`GetSDKVersion()`
Description	Query SDK version number
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Fail-errcode • `sdk`: SDK version number, controller version number

2.4.3.4 Get controller IP

Prototype	`GetControllerIP()`
Description	Query Controller IP
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `ip`: controller IP

2.4.3.5 Controlling the robot into and out of drag-and-drop instructor mode

Prototype	`DragTeachSwitch(state)`
Description	Controls the robot into and out of drag-and-drop demonstration mode.
Mandatory parameter	• `state`: 1-entry into drag-indicator mode, 0-exit from drag-indicator mode
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.3.6 Queries whether the robot is in drag mode

Prototype	`IsInDragTeach()`
Description	Queries whether the robot is in drag-and-drop demonstration mode.
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `state`: 0 - non-drag instructional mode, 1 - drag instructional mode

2.4.3.7 Control robot up-enable or down-enable

prototype	`RobotEnable(state)`
Description	Control robot up-enable or down-enable
Mandatory parameters	• `state`: 1 - up enable, 0 - down enable
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.3.8 Control of robot hand-automatic mode switching

prototype	`Mode(state)`
description	control robot hand auto mode switching
Mandatory parameters	• `state`: 0 - automatic mode, 1 - manual mode
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.3.9 Shut down the robot operating system

New in version python: SDK-v2.1.1

Prototype	`ShutDownRobotOS()`
Description	Shut down the robot operating system
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.3.10 Initialize the log parameters

New in version python: SDK-v2.1.1

Prototype	`LoggerInit(output_model=1, file_path='', file_num=5)`
Description	Initialize the log parameters
Mandatory parameters	NULL
Default parameters	• `output_model`Output mode, 0- Direct output; 1- Buffered output; 2- Asynchronous output, default 1 • `file_path`File path + name, the name must be XXX. The log form, such as/home/fr. / Linux/fairino log. The default path where the program is executed, with the default name fairino_year +month+data.log(e.g., fairino_2024_03_13.log); • `file_num`The number of files for rolling storage ranges from 1 to 20, with a default value of 5. The upper limit for a single file is 50M.
Return Value	Error Code Success-0 Failure- errcode

2.4.3.11 Setting the log filter level

New in version python: SDK-v2.0.2

Prototype	`SetLoggerLevel(lvl=1)`
Description	Set log filter level
Mandatory parameters	NULL
Default Parameters	• `lvl`: filter level value, the smaller the value the less output logs, 1-error, 2-warnning, 3-inform, 4-debug, default value is 1
Return Value	Error Code Success-0 Failure- errcode

2.4.3.12 Robot base control code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
error,version = robot.GetSDKVersion()
print(f"SDK version: {version}")
error,ip = robot.GetControllerIP()
print(f"controller ip: {ip}")
robot.Mode(1)
time.sleep(1)
robot.DragTeachSwitch(state=1)
time.sleep(1)
error,state = robot.IsInDragTeach()
print(f"drag state: {state}")
time.sleep(3)
robot.DragTeachSwitch(state=0)
time.sleep(1)
error,state = robot.IsInDragTeach()
print(f"drag state: {state}")
time.sleep(3)
robot.RobotEnable(0)
time.sleep(3)
robot.RobotEnable(1)
robot.Mode(0)
time.sleep(1)
robot.Mode(1)
robot.CloseRPC()
```

2.4.3.13 Get the software version of the robot

New in version python: SDK-v2.0.1

Prototype	`GetSoftwareVersion()`
Description	Get the software version of the robot
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `robotModel`: robot model • `webVersion`: web version • `controllerVersion`: controller version

2.4.3.14 Getting robot hardware version information

New in version python: SDK-v2.0.1

Prototype	`GetSlaveHardVersion()`
Description	Get robot hardware version information
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `ctrlBoxBoardVersion`: control box version • `driver1Version` • `driver2Version` • `driver3Version` • `driver4Version` • `driver5Version` • `driver6Version` • `endBoardVersion`

2.4.3.15 Getting robot firmware version information

New in version python: SDK-v2.0.1

Prototype	`GetSlaveFirmVersion()`
Description	Get information about the robot's firmware version.
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `ctrlBoxBoardVersion`: control box version • `driver1Version` • `driver2Version` • `driver3Version` • `driver4Version` • `driver5Version` • `driver6Version` • `endBoardVersion`

2.4.3.16 Get the robot software firmware version code sample

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
rtn,robotModel, webversion, controllerVersion = robot.GetSoftwareVersion()
print(f"Getsoftwareversion rtn is: {rtn}")
print(f"robotmodel is: {robotModel}, webversion is: {webversion}, controllerVersion is:
↪{controllerVersion}\n")
```

(continues on next page)

(continued from previous page)

```
rtn,ctrlBoxBoardversion, driver1version, driver2version,driver3version, driver4version,␣
↪driver5version,driver6version, endBoardversion = robot.GetHardwareversion()
print(f"GetHardwareversion rtn is: {rtn}")
print(f"GetHardwareversion get hardware version is: {ctrlBoxBoardversion},
↪{driver1version}, {driver2version}, "
      f"{driver3version}, {driver4version}, {driver5version}, {driver6version},
↪{endBoardversion}\n")
rtn,ctrlBoxBoardversion, driver1version, driver2version,driver3version, driver4version,␣
↪driver5version,driver6version, endBoardversion = robot.GetFirmwareVersion()
print(f"GetFirmwareversion rtn is: {rtn}")
print(f"GetFirmwareversion get firmware version is: {ctrlBoxBoardversion},
↪{driver1version}, {driver2version}, "
      f"{driver3version}, {driver4version}, {driver5version}, {driver6version},
↪{endBoardversion}\n")
robot.CloseRPC()
```

2.4.4 Movement

2.4.4.1 jog point and click

Prototype	`StartJOG(ref,nb,dir,max_dis,vel=20.0,acc=100.0)`
description	jog dot motion
Mandatory parameters	• `ref`: 0 - joint point movement, 2 - base coordinate system point movement, 4 - tool coordinate system point movement, 8 - workpiece coordinate system point movement; • `nb`: 1-1 joint (x-axis), 2-2 joint (y-axis), 3-3 joint (z-axis), 4-4 joint (rx), 5-5 joint (ry), 6-6 joint (rz). • `dir`: 0 - negative direction, 1 - positive direction. • `max_dis`: maximum angle/distance of a single tap in ° or mm;
Default Parameters	• `vel`: percentage of speed, [0 to 100] default 20; • `acc`: acceleration percentage, [0~100] default 100;
Return Value	Error Code Success-0 Failure- errcode

2.4.4.2 jog tap to decelerate and stop

Prototype	`StopJOG(ref)`
description	jog nudging deceleration stop
Mandatory parameters	• `ref`: 1 - joint point stop, 3 - base coordinate system point stop, 5 - tool coordinate system point stop, 9 - workpiece coordinate system point stop
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.3 Immediate stop for jog taps

Prototype	`ImmStopJOG()`
Description	jog nudging stops immediately
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.4 Robot point control code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
for i in range(6):
    robot.StartJOG(0, i + 1, 0, 20.0, 20.0, 30.0)
    time.sleep(1)
    robot.ImmStopJOG()
    time.sleep(1)
for i in range(6):
    robot.StartJOG(2, i + 1, 0, 20.0, 20.0, 30.0)
    time.sleep(1)
    robot.ImmStopJOG()
    time.sleep(1)
for i in range(6):
    robot.StartJOG(4, i + 1, 0, 20.0, 20.0, 30.0)
    time.sleep(1)
    robot.StopJOG(5)
    time.sleep(1)
for i in range(6):
    robot.StartJOG(8, i + 1, 0, 20.0, 20.0, 30.0)
    time.sleep(1)
    robot.StopJOG(9)
    time.sleep(1)
robot.CloseRPC()
```

2.4.4.5 Joint space motion

prototype	`MoveJ(joint_pos, tool, user, desc_pos = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0], vel = 20.0, acc = 0.0, ovl = 100.0, exaxis_pos = [0.0,0.0,0.0,0.0], blendT = -1.0, offset_flag = 0, offset_pos = [0.0,0.0,0.0,0.0,0.0,0.0])`
description	joint space motion
Mandatory parameters	• `joint_pos`: target joint position in [°]; • `tool`: tool number, [0 to 14]; • `user`: artifact number, [0 to 14];
Default parameters	• `desc_pos`: target Cartesian position in [mm][°] Default initial value [0.0,0.0,0.0,0.0,0.0,0.0,0.0], default value calls positive kinematics to solve for the return value. • `vel`: percentage of speed, [0~100] default 20.0. • `acc`: percentage of acceleration, [0~100], not open yet; • `ovl`: velocity scaling factor, [0~100] default 100.0. • `exaxis_pos`: external axis 1 position ~ external axis 4 position Default [0.0,0.0,0.0,0.0]. • `blendT`:[-1.0]-motion in place (blocking), [0~500.0]-smoothing time (non-blocking) in [ms] default -1.0; • `offset_flag`:[0]-no offset, [1]-offset in workpiece/base coordinate system, [2]-offset in tool coordinate system Default 0; • `offset_pos`: position offset in [mm][°] default [0.0,0.0,0.0,0.0,0.0,0.0];
Return Value	Error Code Success-0 Failure- errcode

2.4.4.6 Cartesian Space Linear Motion

New in version python: SDK-v2.1.5

Prototype	`MoveL(desc_pos, tool, user, joint_pos=[0.0, 0.0, 0.0, 0.0, 0. 0, 0.0], vel=20.0, acc=0.0, ovl=100.0,blendR=-1.0, blendMode = 0,exaxis_pos=[0.0, 0.0, 0.0, 0.0], search=0, offset_flag=0, offset_pos=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0],oacc = 100.0, config=-1,velAccParamMode=0,overSpeedStrategy=0,speedPercent=10)`
Description	Cartesian Space Linear Motion
Required Parameters	• `desc_pos`: Target Cartesian pose, unit [mm][°] • `tool`: Tool number, [0~14] • `user`: Workpiece/User number, [0~14]
Default Parameters	• `joint_pos`: Target joint positions, unit [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0], default value is the return value from inverse kinematics • `vel`: Velocity percentage, [0~100] Default 20.0 • `acc`: Acceleration percentage, [0~100], currently unavailable Default 0.0 • `ovl`: Velocity scaling factor, [0~100] Default 100.0 • `blendR`:[-1.0]-Block until motion completes (blocking), [0~1000]-Blend radius (non-blocking), unit [mm] Default -1.0 • `blendMode`: Transition method; 0-Tangent blend 1-Corner blend, default 0 • `exaxis_pos`: External axis 1 position ~ External axis 4 position Default [0.0,0.0,0.0,0.0] • `search`:[0]-No wire search, [1]-Wire search • `offset_flag`:[0]-No offset, [1]-Offset in workpiece/base coordinate system, [2]-Offset in tool coordinate system Default 0 • `offset_pos`: Pose offset value, unit [mm][°] Default [0.0,0.0,0.0,0.0,0.0,0.0] • `oacc`: Acceleration scaling factor [0-100] / Physical acceleration (mm/s^2) Default 100 • `config`: Inverse solution joint space configuration, [-1]-Solve with reference to current joint positions, [0~7]-Solve according to a specific joint space configuration, default -1 • `velAccParamMode`: Velocity/Acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2) Default 0 • `overSpeedStrategy`: Overspeed handling strategy, 0-Strategy disabled; 1-Standard; 2-Stop with error on overspeed; 3-Adaptive speed reduction, default 0 • `speedPercent`: Allowable speed reduction threshold percentage [0-100], default 10%
Return Value	Error code 0-Success Error- errcode

2.4.4.7 Cartesian Space Circular Arc Motion

New in version python: SDK-v2.1.5

Prototype	`MoveC(desc_pos_p, tool_p, user_p, desc_pos_t, tool_t, user_t, joint_pos_p=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0], joint_pos_t=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0],vel_p=20.0, acc_p=100.0, exaxis_pos_p=[0.0, 0.0, 0.0, 0.0], offset_flag_p=0,offset_pos_p=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0],vel_t=20.0, acc_t=100.0, exaxis_pos_t=[0.0, 0.0, 0.0, 0.0], offset_flag_t=0,offset_pos_t=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0],ovl=100.0, blendR=-1.0,oacc=100.0,config=-1, velAccParamMode=0)`
Description	Cartesian Space Circular Arc Motion
Required Parameters	• `desc_pos_p`: Path point Cartesian pose, unit [mm][°] • `tool_p`: Path point tool number, [0~14]; • `user_p`: Path point workpiece/user number, [0~14]; • `desc_pos_t`: Target point Cartesian pose, unit [mm][°]; • `tool_t`: Tool number, [0~14] • `user_t`: Workpiece/User number, [0~14]
Default Parameters	• `joint_pos_p`: Path point joint positions, unit [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0], default value is the return value from inverse kinematics; • `joint_pos_t`: Target point joint positions, unit [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0], default value is the return value from inverse kinematics; • `vel_p`: Path point velocity percentage, [0~100] Default 20.0; • `acc_p`: Path point acceleration percentage, [0~100] Currently unavailable, default 0.0; • `exaxis_pos_p`: Path point external axis 1 position ~ external axis 4 position Default [0.0,0.0,0.0,0.0]; • `offset_flag_p`: Whether the path point is offset [0]-No offset, [1]-Offset in workpiece/base coordinate system, [2]-Offset in tool coordinate system Default 0; • `vel_t`: Target point velocity percentage, [0~100] Default 20.0; • `acc_t`: Target point acceleration percentage, [0~100] Currently unavailable default 0.0; • `exaxis_pos_t`: Target point external axis 1 position ~ external axis 4 position Default [0.0,0.0,0.0,0.0]; • `offset_flag_t`: Whether the target point is offset [0]-No offset, [1]-Offset in workpiece/base coordinate system, [2]-Offset in tool coordinate system Default 0; • `offset_pos_t`: Target point pose offset value, unit [mm][°] Default [0.0,0.0,0.0,0.0,0.0,0.0]; • `ovl:`: Velocity scaling factor, [0~100] Default 100.0; • `blendR`:[-1.0]-Block until motion completes (blocking), [0~1000]-Blend radius (non-blocking), unit [mm] Default -1.0; • `oacc`: Acceleration scaling factor [0-100] / Physical acceleration (mm/s^2) Default 100; • `config`: Inverse solution joint space configuration, [-1]-Solve with reference to current joint positions, [0~7]-Solve according to a specific joint space configuration, default -1; • `velAccParamMode`: Velocity/Acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2) Default 0
Return Value	Error code 0-Success Error- errcode

2.4.4.8 Cartesian Space Full Circle Motion

New in version python: SDK-v2.1.5

Prototype	`Circle(desc_pos_p, tool_p, user_p, desc_pos_t, tool_t, user_t, joint_pos_p=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0],joint_pos_t=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0],vel_p=20.0, acc_p=0.0, exaxis_pos_p=[0.0, 0.0, 0.0, 0.0], vel_t=20.0, acc_t=0.0,exaxis_pos_t=[0.0, 0.0, 0.0, 0.0],ovl=100.0, offset_flag=0, offset_pos=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0], oacc=100.0, blendR=-1,config=-1,velAccParamMode=0)`
Description	Cartesian Space Full Circle Motion
Required Parameters	• `desc_pos_p`: Path point Cartesian pose, unit [mm][°] • `tool_p`: Tool number, [0~14] • `user_p`: Workpiece/User number, [0~14] • `desc_pos_t`: Target point Cartesian pose, unit [mm][°] • `tool_t`: Tool number, [0~14] • `user_t`: Workpiece/User number, [0~14]
Default Parameters	• `joint_pos_p`: Path point joint positions, unit [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0], default value is the return value from inverse kinematics; • `joint_pos_t`: Target point joint positions, unit [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0], default value is the return value from inverse kinematics; • `vel_p`: Velocity percentage, [0~100] Default 20.0; • `acc_p`: Path point acceleration percentage, [0~100] Currently unavailable default 0.0; • `exaxis_pos_p`: Path point external axis 1 position ~ external axis 4 position Default [0.0,0.0,0.0,0.0]; • `vel_t`: Target point velocity percentage, [0~100] Default 20.0; • `acc_t`: Target point acceleration percentage, [0~100] Currently unavailable default 0.0; • `exaxis_pos_t`: Target point external axis 1 position ~ external axis 4 position Default [0.0,0.0,0.0,0.0] • `ovl`: Velocity scaling factor, [0~100] Default 100.0; • `offset_flag`: Whether to offset [0]-No offset, [1]-Offset in workpiece/base coordinate system, [2]-Offset in tool coordinate system Default 0; • `offset_pos`: Pose offset value, unit [mm][°] Default [0.0,0.0,0.0,0.0,0.0,0.0] • `oacc`: Acceleration scaling factor [0-100] / Physical acceleration (mm/s^2), default: 100 • `blendR`:-1Blocking0~1000Blend radius, default: -1 • `config`: Inverse solution joint space configuration, [-1]-Solve with reference to current joint positions, [0~7]-Solve according to a specific joint space configuration, default -1; • `velAccParamMode`: Velocity/Acceleration parameter mode; 0-Percentage; 1-Physical velocity (mm/s) and acceleration (mm/s^2) Default 0
Return Value	Error code 0-Success Error- errcode

2.4.4.9 Point-to-point motion in Cartesian space

Prototype	`MoveCart(desc_pos, tool, user, vel = 20.0, acc = 0.0, ovl = 100.0, blendT = -1.0, config = -1)`
Description	Point-to-point motion in Cartesian space
Mandatory parameters	• `desc_pos`: target Cartesian position; • `tool`: tool number, [0 to 14]; • `user`: artifact number, [0 to 14];
Default parameters	• `vel`: velocity, range [0 to 100], default 20.0; • `acc`: acceleration, range [0-100], not available, default 0.0; • `ovl`: velocity scaling factor, [0 to 100], default 100.0. • `blendT`:[-1.0]-motion in place (blocking), [0~500]-smoothing time (non-blocking) in [ms] default -1.0; • `config`: joint configuration, [-1] - solve with reference to current joint position, [0~7] - solve based on joint configuration default is -1
Return Value	Error Code Success-0 Failure- errcode

2.4.4.10 Sample robot basic motion commands code

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
j1 = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
j2 = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
j3 = [-29.777, -84.536, 109.275, -114.075, -86.655, 74.257]
j4 = [-31.154, -95.317, 94.276, -88.079, -89.740, 74.256]
desc_pos1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
desc_pos2 = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
desc_pos3 = [-487.434, 154.362, 308.576, 176.600, 0.268, -14.061]
desc_pos4 = [-443.165, 147.881, 480.951, 179.511, -0.775, -15.409]
offset_pos = [0.0] * 6
epos = [0.0] * 4
tool = 0
user = 0
vel = 100.0
acc = 100.0
ovl = 100.0
oacc = 100.0
blendT = 0.0
blendR = 0.0
flag = 0
search = 0
blendMode = 0
velAccMode = 0
robot.SetSpeed(20)
rtn = robot.MoveJ(joint_pos=j1, tool=tool, user=user, vel=vel, acc=acc, ovl=ovl, exaxis_
↪pos=epos, blendT=blendT, offset_flag=flag, offset_pos=offset_pos)
print(f"movej errcode:{rtn}")
```

(continues on next page)

(continued from previous page)

```
rtn = robot.MoveL(desc_pos=desc_pos2, tool=tool, user=user, vel=vel, acc=acc, ovl=ovl,␣
↪blendR=blendR, blendMode=blendMode, exaxis_pos=epos, search=search, offset_flag=flag,␣
↪offset_pos=offset_pos,oacc=oacc, velAccParamMode=velAccMode)
print(f"movel errcode:{rtn}")
rtn = robot.MoveC(desc_pos_p=desc_pos3, tool_p=tool, user_p=user, vel_p=vel, acc_p=acc,␣
↪exaxis_pos_p=epos, offset_flag_p=flag, offset_pos_p=offset_pos, desc_pos_t=desc_pos4,␣
↪tool_t=tool, user_t=user, vel_t=vel,acc_t=acc, exaxis_pos_t=epos, offset_flag_t=flag,␣
↪offset_pos_t=offset_pos, ovl=ovl, blendR=blendR, oacc=oacc, velAccParamMode=velAccMode)
print(f"movec errcode:{rtn}")
rtn = robot.MoveJ(joint_pos=j2, tool=tool, user=user, vel=vel, acc=acc, ovl=ovl, exaxis_
↪pos=epos, blendT=blendT, offset_flag=flag, offset_pos=offset_pos)
print(f"movej errcode:{rtn}")
rtn = robot.Circle(desc_pos_p=desc_pos3, tool_p=tool, user_p=user, vel_p=vel, acc_p=acc,␣
↪exaxis_pos_p=epos, desc_pos_t=desc_pos1, tool_t=tool, user_t=user, vel_t=vel, acc_
↪t=acc, exaxis_pos_t=epos, ovl=ovl,offset_flag=flag, offset_pos=offset_pos, oacc=oacc,␣
↪blendR=-1, velAccParamMode=velAccMode)
print(f"circle errcode:{rtn}")
rtn = robot.MoveCart(desc_pos=desc_pos4, tool=tool, user=user, vel=vel, acc=acc,ovl=ovl,␣
↪blendT=blendT, config=-1)
print(f"MoveCart errcode:{rtn}")
rtn = robot.MoveJ(joint_pos=j1, tool=tool, user=user, vel=vel, acc=acc, ovl=ovl, exaxis_
↪pos=epos, blendT=blendT, offset_flag=flag, offset_pos=offset_pos)
print(f"movej errcode:{rtn}")
rtn = robot.MoveL(desc_pos=desc_pos2, tool=tool, user=user, vel=vel, acc=acc, ovl=ovl,␣
↪blendR=blendR, blendMode=blendMode, exaxis_pos=epos, search=search, offset_flag=flag,␣
↪offset_pos=offset_pos, config=-1,velAccParamMode=velAccMode)
print(f"movel errcode:{rtn}")
rtn = robot.MoveC(desc_pos_p=desc_pos3, tool_p=tool, user_p=user, vel_p=vel, acc_p=acc,␣
↪exaxis_pos_p=epos, offset_flag_p=flag, offset_pos_p=offset_pos, desc_pos_t=desc_pos4,␣
↪tool_t=tool, user_t=user, vel_t=vel, acc_t=acc,exaxis_pos_t=epos, offset_flag_t=flag,␣
↪offset_pos_t=offset_pos, ovl=ovl, blendR=blendR, config=-1, velAccParamMode=velAccMode)
print(f"movec errcode:{rtn}")
rtn = robot.MoveJ(joint_pos=j2, tool=tool, user=user, vel=vel, acc=acc, ovl=ovl, exaxis_
↪pos=epos, blendT=blendT, offset_flag=flag, offset_pos=offset_pos)
print(f"movej errcode:{rtn}")
rtn = robot.Circle(desc_pos_p=desc_pos3, tool_p=tool, user_p=user, vel_p=vel, acc_p=acc,␣
↪exaxis_pos_p=epos, desc_pos_t=desc_pos1, tool_t=tool, user_t=user, vel_t=vel, acc_
↪t=acc, exaxis_pos_t=epos, ovl=ovl, offset_flag=flag,offset_pos=offset_pos, oacc=oacc,␣
↪blendR=-1, velAccParamMode=velAccMode)
print(f"circle errcode:{rtn}")
robot.CloseRPC()
return 0
```

2.4.4.11 Spiral motion in Cartesian space

New in version python: SDK-v2.1.7

Prototype	`NewSpiral(desc_pos, tool, user, param, joint_pos = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0], vel = 20.0, acc = 0.0, exaxis_pos = [0.0,0.0,0.0,0.0], ovl = 100.0, offset_flag = 0, offset_pos = [0.0,0.0,0.0,0.0,0.0,0.0], config = -1)`
Description	Spiral motion in Cartesian space
Mandatory parameters	• `desc_pos`: target Cartesian position in [mm][°]; • `tool`: tool number, [0 to 14]. • `user`: artifact number, [0 to 14]. • `param=[circle_num, circle_angle, rad_init, rad_add, rotaxis_add, rot_direction, velAccMode]`: circle_num: number of spiral turns; circle_angle: spiral inclination; rad_init: initial radius of the spiral. rad_add: radius increment; rotaxis_add: rotational direction increment; rot_direction: rotational direction, 0-clockwise, 1-counterclockwise,velAccMode speed acceleration parameter mode: 0- constant angular velocity, 1- constant linear velocity;
Default parameters	• `joint_pos`: target joint position in [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0,0.0], default value calls inverse kinematics to solve for the return value. • `vel`: percentage of speed, [0~100] default 20.0. • `acc`: percentage of acceleration, [0~100] default 100.0. • `exaxis_pos`: external axis 1 position ~ external axis 4 position Default [0.0,0.0,0.0,0.0]. • `ovl`: velocity scaling factor, [0~100] default 100.0. • `offset_flag`:[0]-no offset, [1]-offset in workpiece/base coordinate system, [2]-offset in tool coordinate system Default 0; • `offset_pos`: position offset in [mm][°] default [0.0,0.0,0.0,0.0,0.0,0.0] • `config`: Reverse the joint space configuration, [-1]- calculate based on the current joint position, [0~7]- solve based on the specific joint space configuration, default -1
Return Value	Error Code Success-0 Failure- errcode

2.4.4.12 code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
j = [67.957, -81.482, 87.595, -95.691, -94.899, -9.727]
desc_pos = [-123.142, -551.735, 430.549, 178.753, -4.757, 167.754]
offset_pos1 = [50.0, 0.0, 0.0, -30.0, 0.0, 0.0]
offset_pos2 = [50.0, 0.0, 0.0, -30.0, 0.0, 0.0]
epos = [0.0] * 4
sp = [2, 30.0, 50.0, 10.0, 10.0, 0, 1]  # [circle_num, circle_angle, rad_init, rad_add, rotaxis_add, rot_direction, velAccMode]
```

(continues on next page)

(continued from previous page)

```
tool = 0
user = 0
vel = 30.0
acc = 60.0
ovl = 100.0
blendT = -1.0
flag = 2
robot.SetSpeed(20)
rtn = robot.MoveJ(joint_pos=j, tool=tool, user=user, vel=vel, acc=acc, ovl=ovl, exaxis_
↪pos=epos, blendT=blendT, offset_flag=flag, offset_pos=offset_pos1)
print(f"movej errcode:{rtn}")
rtn = robot.NewSpiral(desc_pos=desc_pos, tool=tool, user=user, vel=vel, acc=acc, exaxis_
↪pos=epos, ovl=ovl, offset_flag=flag, offset_pos=offset_pos2, param=sp)
print(f"newspiral errcode:{rtn}")
robot.CloseRPC()
return 0
```

2.4.4.13 Start of servo motion

Prototype	`ServoMoveStart()`
Description	Servo motion start, used with ServoJ, ServoCart commands
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.14 End of servo motion

Prototype	`ServoMoveEnd()`
Description	End of servo motion, used with ServoJ, ServoCart commands
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.15 Joint space servo mode motion

Prototype	`ServoJ(joint_pos, axisPos, acc = 0.0, vel = 0.0, cmdT = 0.008, filterT = 0.0, gain = 0.0, id=0)`
Description	Joint space servo mode motion
Mandatory parameters	• `joint_pos`: target joint position in [°]; • `axisPos`: external axis position in mm;
Default parameter	• `acc`: acceleration, range [0~100], not open yet, default 0.0. • `vel`: velocity, range [0~100], not open, default 0.0. • `cmdT`: command send cycle, unit s, recommended range [0.001~0.0016], default is 0.008. • `filterT`: filter time in [s], not open, default is 0.0; `filterT`: filter time in [s], not open, default is 0.0. • `gain`: proportional amplifier for target position, not open yet, default 0.0; • `id`: servoJ command ID, default 0.
Return Value	Error Code Success-0 Failure- errcode

2.4.4.16 Example of joint space servo mode motion code

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
j = [0.0] * 6
epos = [0.0] * 4
vel = 0.0
acc = 0.0
cmdT = 0.008
filterT = 0.0
gain = 0.0
flag = 0
count = 500
dt = 0.1
cmdID = 0
ret, j = robot.GetActualJointPosDegree(flag)
if ret == 0:
    cmdID += 1
    robot.ServoMoveStart()
    while count:
        robot.ServoJ(joint_pos=j,axisPos= epos,acc= acc,vel= vel, cmdT=cmdT,␣
↪filterT=filterT, gain=gain, id=cmdID)
        j[4] += dt
        count -= 1
        time.sleep(cmdT)
        rtn,pkg = robot.GetRobotRealTimeState()
        print(f"Servoj Count {pkg.servoJCmdNum}; last pos is {pkg.lastServoTarget[0]},
↪{pkg.lastServoTarget[1]},{pkg.lastServoTarget[2]},{pkg.lastServoTarget[3]},{pkg.
↪lastServoTarget[4]},{pkg.lastServoTarget[5]}")
```

(continues on next page)

(continued from previous page)

```
        if count < 50:
            robot.MotionQueueClear()
            print(f"After queue clear, Servoj Count {pkg.servoJCmdNum}; last pos is {pkg.
↪lastServoTarget[0]},{pkg.lastServoTarget[1]},{pkg.lastServoTarget[2]},{pkg.
↪lastServoTarget[3]},{pkg.lastServoTarget[4]},{pkg.lastServoTarget[5]}")
            break
    robot.ServoMoveEnd()
else:
    print(f"GetActualJointPosDegree errcode:{ret}")
robot.CloseRPC()
```

2.4.4.17 Joint torque control begins

Prototype	`ServoJTStart()`
Description	Joint torque control begins
Mandatory parameters	NULL
Default_parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.18 Joint torque control

Prototype	`ServoJT(torque, interval, checkFlag=0, jPowerLimit=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0], jVelLimit=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0])`
Description	Joint Torque Control
Required Parameters	• **`torque`: Torque for joints j1~j6, unit: Nm** – `interval`: Command cycle, unit: s, range: [0.001~0.008] – `checkFlag`: Detection strategy 0-No restrictions; 1-Power limit; 2-Velocity limit; 3-Both power and velocity limits, default: 0 – `jPowerLimit`: Default parameter jPowerLimit Maximum joint power limit (W), default: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] – `jVelLimit`: Maximum joint velocity (°/s), default: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
Default Parameters	None
Return Value	Error code Success-0 Failure- errcode

2.4.4.19 Joint torque control is completed

Prototype	ServoJTEnd()
Description	Joint torque control is completed
Mandatory parameters	NULL
Default_parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.20 Sample code for joint torque control

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.DragTeachSwitch(1)
# torques = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
error,torques = robot.GetJointTorques(1)
robot.ServoJTStart()
count = 100
while count > 0:
    error = robot.ServoJT(torques, 0.001)
    count -= 1
    time.sleep(0.001)
error = robot.ServoJTEnd()
robot.DragTeachSwitch(0)
robot.CloseRPC()
```

2.4.4.21 Joint Torque Control Code Example with Overspeed Protection

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
robot.ResetAllError()
time.sleep(0.5)
torques = [0.0] * 6
rtn, torques = robot.GetJointTorques(1)
robot.ServoJTStart()
robot.DragTeachSwitch(1)
checkFlag = 3
jPowerLimit = [10.0, 10.0, 10.0, 10.0, 10.0, 10.0]
jVelLimit = [181,80,80,80,80,80]
count = 800000
error = 0
while count > 0:
    torques[2] = torques[2] + 0.01
    error = robot.ServoJT(torques, 0.008, checkFlag, jPowerLimit, jVelLimit)
    print(f"ServoJT rtn is {error}")
    count = count - 1
    time.sleep(0.001)
```

(continues on next page)

(continued from previous page)

```
    rtn,pkg = robot.GetRobotRealTimeState()
    print(f"maincode {pkg.main_code},subcode {pkg.sub_code}")
robot.DragTeachSwitch(0)
error = robot.ServoJTEnd()
```

2.4.4.22 Cartesian Space Servo Mode Motion

Prototype	`ServoCart(mode, desc_pos, exaxis, pos_gain=[1.0, 1.0, 1.0, 1.0, 1.0, 1.0], acc=0.0, vel=0.0, cmdT=0.008,filterT=0.0, gain=0.0)`
Description	Cartesian space servo mode motion
Required Parameters	• `mode`:[0]-Absolute motion (base coordinate system), [1]-Incremental motion (base coordinate system), [2]-Incremental motion (tool coordinate system); • `exaxis`:Extended axis position; • `desc_pos`:Target Cartesian position/Target Cartesian position increment;
Default Parameters	• `pos_gain`:Pose increment proportionality coefficient, only effective in incremental motion, range [0~1], default [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]; • `acc`:Acceleration, range [0~100], temporarily not available, default 0.0; • `vel`:Velocity, range [0~100], temporarily not available, default 0.0; • `cmdT`:Command transmission period, unit s, recommended range [0.001~0.0016], default 0.008; • `filterT`:Filter time, unit [s], temporarily not available, default 0.0; • `gain`:Proportional amplifier for target position, temporarily not available, default 0.0;
Return Value	Error code Success-0 Failure- errcode

2.4.4.23 Cartesian Space Servo Mode Motion Code Example

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
desc_pos_dt = [83.00800, 50.525000, 29.246, 179.629, -7.138, -166.975]
exaxis = [100.0, 0.0, 0.0, 0.0]
pos_gain = [0.0] * 6
mode = 0
vel = 0.0
acc = 0.0
cmdT = 0.001
filterT = 0.0
gain = 0.0
flag = 0
count = 5000
robot.SetSpeed(20)
while count:
    rtn = robot.ServoCart(mode, desc_pos_dt, exaxis, pos_gain, acc, vel, cmdT, filterT,␣
↪gain)
```

(continues on next page)

(continued from previous page)

```
    print(f"ServoCart rtn is {rtn}")
    count -= 1
    desc_pos_dt[0] += 0.01
    exaxis[0] += 0.01
robot.CloseRPC()
return 0
```

2.4.4.24 Start of spline motion

Prototype	`SplineStart()`
Description	Sample movement begins
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.25 Sample motion PTP

Prototype	`SplinePTP(joint_pos, tool, user, desc_pos = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0], vel = 20.0, acc = 100.0, ovl = 100.0)`
Description	Sample Motion PTP
Mandatory parameters	• `joint_pos`: target joint position in [°]; • `tool`: tool number, [0 to 14]; • `user`: artifact number, [0 to 14];
Default parameters	• `desc_pos`: target Cartesian position in [mm][°] Default initial value [0.0,0.0,0.0,0.0,0.0,0.0,0.0], default value calls positive kinematics to solve for the return value. • `vel`: velocity, range [0-100], default 20.0. • `acc`: acceleration, range [0 to 100], default 100.0. • `ovl`: velocity scaling factor, [0 to 100], default 100.0
Return Value	Error Code Success-0 Failure- errcode

2.4.4.26 End of spline motion

Prototype	`SplineEnd()`
Description	End of spline motion
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.27 Spline motion code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
joint_points = [
    [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256],  # j1
    [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255],  # j2
    [-61.954, -84.409, 108.153, -116.316, -91.283, 74.260],  # j3
    [-89.575, -80.276, 102.713, -116.302, -91.284, 74.267]  # j4
]
cart_points = [
    [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833],  # desc_pos1
    [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869],  # desc_pos2
    [-327.622, 402.230, 320.402, -178.067, 2.127, -46.207],  # desc_pos3
    [-104.066, 544.321, 327.023, -177.715, 3.371, -73.818]  # desc_pos4
]
offset_pos = [0] * 6
epos = [0] * 4
tool = user = 0
vel = acc = ovl = 100.0
blendT = -1.0
flag = 0
robot.SetSpeed(20)
err1 = robot.MoveJ(joint_pos=joint_points[0],tool=tool, user=user,vel=vel)
print(f"MoveJ : {err1}")
robot.SplineStart()
robot.SplinePTP(joint_pos=joint_points[0],tool=tool, user=user)
robot.SplinePTP(joint_pos=joint_points[1],tool=tool, user=user)
robot.SplinePTP(joint_pos=joint_points[2],tool=tool, user=user)
robot.SplinePTP(joint_pos=joint_points[3],tool=tool, user=user)
robot.SplineEnd()
robot.CloseRPC()
```

2.4.4.28 New spline movement begins

Changed in version python: SDK-v2.0.3

prototype	`NewSplineStart(type,averageTime=2000)`
Description	New Sample Movement Begins
Mandatory parameters	• `type`: 0 - arc transition, 1 - given point path point
Default Parameters	• `averageTime`: global average articulation time (ms) defaults to 2000
Return Value	Error Code Success-0 Failure- errcode

2.4.4.29 New spline command point

Prototype	`NewSplinePoint(desc_pos,tool,user,lastFlag,joint_pos=[0.0,0.0,0.0,0.0,0.0,0.0], vel = 0.0, acc = 0.0, ovl = 100.0 ,blendR = 0.0)`
description	new spline command point
Mandatory parameters	• `desc_pos`: target Cartesian position in [mm][°]. • `tool`: tool number, [0 to 14]; • `user`: artifact number, [0 to 14]; • `lastFlag`: whether it is the last point, 0 - no, 1 - yes;
Default parameters	• `joint_pos`: target joint position in [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0,0.0], default value calls inverse kinematics to solve for the return value. • `vel`: velocity, range [0~100], not open yet, default is 0.0;; • `acc`: acceleration, range [0 to 100], not open, default 0.0. • `ovl`: velocity scaling factor, [0~100] default 100.0. • `blendR`: [0~1000]-smoothing radius in [mm] default 0.0;
Return Value	Error Code Success-0 Failure- errcode

2.4.4.30 End of new spline movement

Prototype	`NewSplineEnd()`
Description	End of New Sample Campaign
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.31 Example of new spline motion code

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
j1 = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
j2 = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
j3 = [-61.954, -84.409, 108.153, -116.316, -91.283, 74.260]
j4 = [-89.575, -80.276, 102.713, -116.302, -91.284, 74.267]
j5 = [-95.228, -54.621, 73.691, -112.245, -91.280, 74.268]
desc_pos1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
desc_pos2 = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
desc_pos3 = [-327.622, 402.230, 320.402, -178.067, 2.127, -46.207]
desc_pos4 = [-104.066, 544.321, 327.023, -177.715, 3.371, -73.818]
desc_pos5 = [-33.421, 732.572, 275.103, -177.907, 2.709, -79.482]
offset_pos = [0, 0, 0, 0, 0, 0]
epos = [0, 0, 0, 0]
```

(continues on next page)

(continued from previous page)

```
tool = 0
user = 0
vel = 100.0
acc = 100.0
ovl = 100.0
blendT = -1.0
flag = 0
robot.SetSpeed(20)
err1 = robot.MoveJ(joint_pos=j1, tool=tool, user=user, vel=vel)
print(f"movej errcode:{err1}")
robot.NewSplineStart(1, 2000)
robot.NewSplinePoint(desc_pos=desc_pos1, tool=tool, user=user, vel=vel, lastFlag=-1,␣
↪blendR=0)
robot.NewSplinePoint(desc_pos=desc_pos2, tool=tool, user=user, vel=vel, lastFlag=-1,␣
↪blendR=0)
robot.NewSplinePoint(desc_pos=desc_pos3, tool=tool, user=user, vel=vel, lastFlag=-1,␣
↪blendR=0)
robot.NewSplinePoint(desc_pos=desc_pos4, tool=tool, user=user, vel=vel, lastFlag=-1,␣
↪blendR=0)
robot.NewSplinePoint(desc_pos=desc_pos5, tool=tool, user=user, vel=vel, lastFlag=-1,␣
↪blendR=0)
robot.NewSplineEnd()
robot.CloseRPC()
```

2.4.4.32 Robot termination motion

Prototype	`StopMotion()`
Description	Terminate motion, use of terminate motion requires motion instruction to be non-blocking
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.33 Robot pause

Prototype	`PauseMotion()`
Description	Pause motion. Using pause motion requires the motion instruction to be in a non-blocking state
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.34 Robot resume motion

Prototype	`ResumeMotion()`
Description	Resume motion, using resume motion requires the motion instruction to be in a non-blocking state
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.35 Motion pause, resume, and stop code examples

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
j1 =[-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
j5 =[-95.228, -54.621, 73.691, -112.245, -91.280, 74.268]
desc_pos1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
desc_pos5 = [-33.421, 732.572, 275.103, -177.907, 2.709, -79.482]
offset_pos = [0, 0, 0, 0, 0, 0]
epos = [0, 0, 0, 0]
tool = 0
user = 0
vel = 100.0
acc = 100.0
ovl = 100.0
blendT = -1.0
flag = 0
robot.SetSpeed(20)
rtn = robot.MoveJ(joint_pos=j1, tool=tool, user=user, vel=vel)
rtn = robot.MoveJ(joint_pos=j5, tool=tool, user=user, vel=vel, blendT=1)
time.sleep(1)
robot.PauseMotion()
time.sleep(1)
robot.ResumeMotion()
time.sleep(1)
robot.StopMotion()
time.sleep(1)
robot.CloseRPC()
```

2.4.4.36 Overall shift in points begins

Prototype	`PointsOffsetEnable(flag,offset_pos)`
Description	Point Overall Offset Begins
Mandatory parameters	• `flag`: 0 - offset in base or work coordinate system, 2 - offset in tool coordinate system; • `offset_pos`: offset in [mm][°].
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.37 Overall offset of points ends

Prototype	`PointsOffsetDisable()`
Description	End of overall point offset
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.38 Point offset code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
j1 = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
j2 = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
desc_pos1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
desc_pos2 = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
offset_pos = [0, 0, 0, 0, 0, 0]
offset_pos1 = [0, 0, 50, 0, 0, 0]
epos = [0, 0, 0, 0]
tool = 0
user = 0
vel = 100.0
acc = 100.0
ovl = 100.0
blendT = -1.0
flag = 0
robot.SetSpeed(20)
robot.MoveJ(joint_pos=j1,tool=tool, user=user, vel=vel)
robot.MoveJ(joint_pos=j2, tool=tool, user=user, vel=vel)
time.sleep(1)
robot.PointsOffsetEnable(flag=0, offset_pos=offset_pos1)
robot.MoveJ(joint_pos=j1,tool=tool, user=user, vel=vel)
robot.MoveJ(joint_pos=j2, tool=tool, user=user, vel=vel)
robot.PointsOffsetDisable()
robot.CloseRPC()
```

2.4.4.39 Control box motion AO start

New in version python: SDK-v2.0.4

Prototype	`MoveAOStart(AONum,maxTCPSpeed=1000,maxAOPercent=100, zeroZoneCmp=20)`
Description	Control Box Motion AO Start
Mandatory parameters	• `AONum`: control box AO number
Default Parameters	• `maxTCPSpeed`: Maximum TCP speed value [1-5000mm/s], default 1000; • `maxAOPercent`: percentage of AO corresponding to the maximum TCP speed value, default 100%; • `zeroZoneCmp`: deadzone compensation value AO percentage, shaped, default 20%, range [0-100].
Return Value	Error Code Success-0 Failure- errcode

2.4.4.40 End of control box movement AO

New in version python: SDK-v2.0.4

Prototype	`MoveAOStop()`
Description	End of control box motion AO
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.41 End Motion AO Start

New in version python: SDK-v2.0.4

Prototype	`MoveToolAOStart(AONum,maxTCPSpeed=1000,maxAOPercent=100, zeroZoneCmp =20)`
Description	End Motion AO Start
Mandatory parameters	• `AONum`: end AO number
Default Parameters	• `maxTCPSpeed`: Maximum TCP speed value [1-5000mm/s], default 1000; • `maxAOPercent`: percentage of AO corresponding to the maximum TCP speed value, default 100%; • `zeroZoneCmp`: deadzone compensation value AO percentage, shaped, default 20%, range [0-100].
Return Value	Error Code Success-0 Failure- errcode

2.4.4.42 End movement AO end

New in version python: SDK-v2.0.4

Prototype	`MoveToolAOStop()`
Description	end movement AO end
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.43 AO flyshot code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
j1 = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
j2 = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
desc_pos1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
desc_pos2 = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
offset_pos = [0, 0, 0, 0, 0, 0]
offset_pos1 = [0, 0, 50, 0, 0, 0]
epos = [0, 0, 0, 0]
tool = 0
user = 0
vel = 20.0
acc = 20.0
ovl = 100.0
blendT = -1.0
flag = 0
robot.SetSpeed(20)
robot.MoveAOStart(0, 100, 100, 20)
robot.MoveJ(joint_pos=j1,tool=tool, user=user, vel=vel)
robot.MoveJ(joint_pos=j2, tool=tool, user=user, vel=vel)
robot.MoveAOStop()
time.sleep(1)
robot.MoveToolAOStart(0, 100, 100, 20)
robot.MoveJ(joint_pos=j1,tool=tool, user=user, vel=vel)
robot.MoveJ(joint_pos=j2, tool=tool, user=user, vel=vel)
robot.MoveToolAOStop()
robot.CloseRPC()
```

2.4.4.44 Start Ptp motion FIR filtering

New in version python: SDK-v2.1.2

Prototype	`PtpFIRPlanningStart(maxAcc, maxJek)`
Description	Start Ptp motion FIR filtering
Mandatory parameters	• `maxAcc`:Maximum acceleration extremum(deg/s2) • `maxJek`:Unify the extreme values of joint urgency (deg/s3)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.45 Disable Ptp motion FIR filtering

New in version python: SDK-v2.0.7

Prototype	`PtpFIRPlanningEnd()`
Description	Disable Ptp motion FIR filtering
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.46 LIN, ARC motion FIR filtering is started

New in version python: SDK-v2.0.7

Prototype	`LinArcFIRPlanningStart(maxAccLin,maxAccDeg,maxJerkLin, maxJerkDeg)`
Description	LIN, ARC motion FIR filtering is started
Mandatory parameter	• `maxAccLin`:Linear acceleration extremum(mm/s2) • `maxAccDeg`:Angular acceleration extremum(deg/s2) • `maxJerkLin`:Linear plus acceleration extremum(mm/s3) • `maxJerkDeg`:Angle plus acceleration extremum(deg/s3)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.47 Turn off LIN and ARC motion FIR filtering

New in version python: SDK-v2.0.7

Prototype	`LinArcFIRPlanningEnd()`
Description	Turn off LIN and ARC motion FIR filtering
Mandatory parameter	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.48 FIR filtering code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
startjointPos = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
startjointPos = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
midjointPos = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
endjointPos = [-29.777, -84.536, 109.275, -114.075, -86.655, 74.257]
startdescPose = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
middescPose = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
enddescPose = [-487.434, 154.362, 308.576, 176.600, 0.268, -14.061]
exaxisPos = [0.0, 0.0, 0.0, 0.0]
offdese = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
rtn = robot.PtpFIRPlanningStart(1000.0, 1000.0)
print(f"PtpFIRPlanningStart rtn is {rtn}")
error = robot.MoveJ(joint_pos=startjointPos,tool= 0,user= 0,desc_pos=startdescPose,vel=100,acc=100,ovl=100, blendT=-1.0, offset_flag=0)
print(f"MoveJ rtn is {rtn}")
error = robot.MoveJ(joint_pos=endjointPos,tool= 0,user= 0,desc_pos=enddescPose,vel= 100,acc=100,ovl=100, blendT=-1.0, offset_flag=0)
print(f"MoveJ rtn is {rtn}")
robot.PtpFIRPlanningEnd()
print(f"PtpFIRPlanningEnd rtn is {rtn}")
rtn = robot.LinArcFIRPlanningStart(1000, 1000, 1000, 1000)
print(f"LinArcFIRPlanningStart rtn is {rtn}")
error = robot.MoveL(desc_pos=startdescPose,tool= 0,user= 0, joint_pos=startjointPos,vel=100,overSpeedStrategy=1,speedPercent=1)
print(f"MoveL rtn is {rtn}")
error = robot.MoveC(desc_pos_p=middescPose,tool_p= 0,user_p= 0, joint_pos_p=midjointPos,vel_p= 100,desc_pos_t=enddescPose,tool_t= 0,user_t= 0,joint_pos_t=endjointPos,vel_t=100)
print(f"MoveC rtn is {rtn}")
robot.LinArcFIRPlanningEnd()
print(f"LinArcFIRPlanningEnd rtn is {rtn}")
robot.CloseRPC()
```

2.4.4.49 Acceleration smooth on

New in version python: SDK-v2.1.1

Prototype	`AccSmoothStart(saveFlag_flag)`
Description	Acceleration smooth on
Mandatory parameters	• `saveFlag_flag`: Power off and save
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.50 Acceleration smooth closing

New in version python: SDK-v2.1.1

Prototype	`AccSmoothEnd(saveFlag_flag)`
Description	Acceleration smooth closing
Mandatory parameters	• `saveFlag_flag`: Power off and save
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.51 Acceleration smoothing code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
startjointPos = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
endjointPos = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
startdescPose = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
enddescPose = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
exaxisPos = [0.0, 0.0, 0.0, 0.0]
offdese = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
rtn = robot.AccSmoothStart(0)
print(f"AccSmoothStart rtn is {rtn}")
robot.MoveJ(joint_pos=startjointPos,tool= 0,user= 0,vel= 100)
robot.MoveJ(joint_pos=endjointPos,tool= 0,user= 0,vel= 100)
rtn = robot.AccSmoothEnd(0)
print(f"AccSmoothEnd rtn is {rtn}")
```

2.4.4.52 Setting the machine's specified attitude speed on

New in version python: SDK-v2.0.5

Prototype	`AngularSpeedStart(ratio)`
Description	Specifies that attitude speed is on.
Mandatory parameters	• `ratio`: percentage of attitude velocity [0-300]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.53 Specify Attitude Velocity Off

New in version python: SDK-v2.0.5

Prototype	`AngularSpeedEnd()`
Description	Specify Attitude Velocity Off
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.4.54 Robot specified pose velocity code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
startjointPos = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
endjointPos = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
startdescPose = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
enddescPose = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
exaxisPos = [0.0, 0.0, 0.0, 0.0]
offdese = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
rtn = robot.AngularSpeedStart(50)
print(f"AngularSpeedStart rtn is {rtn}")
robot.MoveJ(joint_pos=startjointPos, tool=0,user= 0,vel= 100)
robot.MoveJ(joint_pos=endjointPos, tool=0,user= 0,vel= 100)
rtn = robot.AngularSpeedEnd()
print(f"AngularSpeedEnd rtn is {rtn}")
robot.CloseRPC()
```

2.4.4.55 Odd-position protection on.

New in version python: SDK-v2.0.5

Prototype	`SingularAvoidStart(protectMode, minShoulderPos=100, minElbowPos=50, minWristPos=10)`
Description	Turn on odd-bit posture protection.
Mandatory parameters	• `protectMode`: singular position protection protection mode: 0 - articulated mode; 1 - Cartesian mode
Default Parameters	• `minShoulderPos`: Shoulder singularity adjustment range (mm), default 100.0 • `minElbowPos`: elbow singularity adjustment range (mm), default 50.0 • `minWristPos`: range of wrist singularity adjustment (°), default 10.0
Return Value	• errcode Success-0 Failure- errcode

2.4.4.56 Odd position protection off

New in version python: SDK-v2.0.5

Prototype	`SingularAvoidEnd()`
Description	Turn off odd-position protection
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errcode Success-0 Failure- errcode

2.4.4.57 Example of robot singular pose protection code

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
startjointPos = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
endjointPos = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
startdescPose = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
enddescPose = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
exaxisPos = [0.0, 0.0, 0.0, 0.0]
offdese = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
rtn = robot.SingularAvoidStart(2, 10, 5, 5)
print(f"SingularAvoidStart rtn is {rtn}")
robot.MoveJ(joint_pos=startjointPos, tool=0,user= 0,vel= 100)
robot.MoveJ(joint_pos=endjointPos, tool=0,user= 0,vel= 100)
rtn = robot.SingularAvoidEnd()
print(f"SingularAvoidEnd rtn is {rtn}")
robot.CloseRPC()
```

2.4.4.58 Clear the motor command queue

New in version python: SDK-v2.1.7

Prototype	`MotionQueueClear()`
Description	Clear the motor command queue
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errcode Success-0 Failure- errcode

2.4.4.59 Clear Motion Command Queue

Prototype	`MoveToIntersectLineStart(mainPoint, piecePoint, tool, wobj, vel, acc, ovl, oacc, moveType,mainExaxisPos=[[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]],pieceExaxisPos=[[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]],extAxisFlag=0,exaxisPos=[0.0,0.0,0.0,0.0],moveDirection=0,offset=[0.0,0.0,0.0,0.0,0.0,0.0])`
Description	Clear motion command queue
Required Parameters	• `mainPoint`Cartesian poses of 6 taught points on the main pipe • `piecePoint`Cartesian poses of 6 taught points on the branch pipe • `tool`Tool coordinate system number • `wobj`Workpiece coordinate system number • `vel`Velocity percentage • `acc`Acceleration percentage • `ovl`Velocity scaling factor • `oacc`Acceleration scaling factor • `moveType`Motion type; 0-PTP; 1-LIN • `mainExaxisPos`Extended axis positions for 6 taught points on the main pipe, default [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0 • `pieceExaxisPos`Extended axis positions for 6 taught points on the branch pipe, default [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0 • `extAxisFlag`Whether to enable extended axis; 0-Disable; 1-Enable • `exaxisPos`Start point extended axis position [0.0,0.0,0.0,0.0] • `moveDirection`Motion direction; 0-Clockwise; 1-Counterclockwise • `offset`Offset value
Default Parameters	None
Return Value	• Error code Success-0 Failure- errcode

2.4.4.60 Intersecting Line Motion

Prototype	`MoveIntersectLine(mainPoint, piecePoint, tool, wobj, vel, acc, ovl, oacc, moveDirection,mainExaxisPos=[[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]],pieceExaxisPos=[[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]],extAxisFlag=0,exaxisPos=[[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0]],offset=[0.0,0.0,0.0,0.0,0.0,0.0])`
Description	Intersecting line motion
Required Parameters	• `mainPoint`Cartesian poses of 6 taught points on the main pipe • `piecePoint`Cartesian poses of 6 taught points on the branch pipe • `tool`Tool coordinate system number • `wobj`Workpiece coordinate system number • `vel`Velocity percentage • `acc`Acceleration percentage • `ovl`Velocity scaling factor • `oacc`Acceleration scaling factor • `moveDirection`Motion direction; 0-Clockwise; 1-Counterclockwise • `mainExaxisPos`Extended axis positions for 6 taught points on the main pipe, default [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0 • `pieceExaxisPos`Extended axis positions for 6 taught points on the branch pipe, default [[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0],[0.0,0.0,0.0,0.0 • `extAxisFlag`Whether to enable extended axis; 0-Disable; 1-Enable • `exaxisPos`Start point extended axis position [0.0,0.0,0.0,0.0] • `offset`Offset value
Default Parameters	None
Return Value	• Error code Success-0 Failure- errcode

2.4.4.61 Robot Intersecting Line Motion Code Example

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
mainPoint = [[0.0] * 6 for _ in range(6)]
piecePoint = [[0.0] * 6 for _ in range(6)]
mainExaxisPos = [[0.0] * 4 for _ in range(6)]
pieceExaxisPos = [[0.0] * 4 for _ in range(6)]
extAxisFlag = 1
exaxisPos = [[0.0] * 4 for _ in range(4)]
offset = [0.0, 2.0, 30.0, -2.0, 0.0, 0.0]
mainPoint[0] = [490.004, -383.194, 402.735, -9.332, -1.528, 69.594]
mainPoint[1] = [444.950, -407.117, 389.011, -5.546, -2.196, 65.279]
mainPoint[2] = [445.168, -463.605, 355.759, -1.544, -10.886, 57.104]
mainPoint[3] = [507.529, -485.385, 343.013, -0.786, -4.834, 61.799]
```

(continues on next page)

(continued from previous page)

```
mainPoint[4] = [554.390, -442.647, 367.701, -4.761, -10.181, 64.925]
mainPoint[5] = [532.552, -394.003, 396.467, -13.732, -13.592, 67.411]
mainExaxisPos[0] = [-29.996, 0.000, 0.000, 0.000]
mainExaxisPos[1] = [-29.996, 0.000, 0.000, 0.000]
mainExaxisPos[2] = [-29.996, 0.000, 0.000, 0.000]
mainExaxisPos[3] = [-29.996, 0.000, 0.000, 0.000]
mainExaxisPos[4] = [-29.996, 0.000, 0.000, 0.000]
mainExaxisPos[5] = [-29.996, 0.000, 0.000, 0.000]
piecePoint[0] = [505.571, -192.408, 316.759, 38.098, 37.051, 139.447]
piecePoint[1] = [533.837, -201.558, 332.340, 34.644, 42.339, 137.748]
piecePoint[2] = [530.386, -225.085, 373.808, 35.431, 45.111, 137.560]
piecePoint[3] = [485.646, -229.195, 383.778, 33.870, 45.173, 137.064]
piecePoint[4] = [460.551, -212.161, 354.256, 28.856, 45.602, 135.930]
piecePoint[5] = [474.217, -197.124, 324.611, 42.469, 41.133, 148.167]
pieceExaxisPos[0] = [-29.996, -0.000, 0.000, 0.000]
pieceExaxisPos[1] = [-29.996, -0.000, 0.000, 0.000]
pieceExaxisPos[2] = [-29.996, -0.000, 0.000, 0.000]
pieceExaxisPos[3] = [-29.996, -0.000, 0.000, 0.000]
pieceExaxisPos[4] = [-29.996, -0.000, 0.000, 0.000]
pieceExaxisPos[5] = [-29.996, -0.000, 0.000, 0.000]
exaxisPos[0] = [-29.996, -0.000, 0.000, 0.000]
exaxisPos[1] = [-44.994, 90.000, 0.000, 0.000]
exaxisPos[2] = [-59.992, 0.002, 0.000, 0.000]
exaxisPos[3] = [-44.994, -89.997, 0.000, 0.000]
tool = 2
wobj = 0
vel = 100.0
acc = 100.0
ovl = 12.0
oacc = 12.0
moveType = 1
moveDirection = 1
rtn = robot.MoveToIntersectLineStart(mainPoint=mainPoint, mainExaxisPos=mainExaxisPos,␣
↪piecePoint=piecePoint, pieceExaxisPos=pieceExaxisPos, extAxisFlag=extAxisFlag,
↪exaxisPos=exaxisPos[0], tool=tool, wobj=wobj, vel=vel, acc=acc, ovl=ovl, oacc=oacc,␣
↪moveType=moveType, moveDirection=moveDirection, offset=offset)
print(f"MoveToIntersectLineStart rtn is {rtn}")
rtn = robot.MoveIntersectLine(mainPoint=mainPoint, mainExaxisPos=mainExaxisPos,␣
↪piecePoint=piecePoint, pieceExaxisPos=pieceExaxisPos, extAxisFlag=extAxisFlag,␣
↪exaxisPos=exaxisPos, tool=tool,wobj=wobj, vel=vel, acc=acc, ovl=5.0, oacc=5.0,␣
↪moveDirection=moveDirection, offset=offset)
print(f"MoveIntersectLine rtn is {rtn}")
robot.CloseRPC()
```

2.4.4.62 Stationary Air Motion

Prototype	`MoveStationary()`
Description	Stationary Air Motion
Required Parameters	None
Default Parameters	None
Return Value	• Error code. Success - 0, Failure - errcode

2.4.4.63 Stationary Air Motion Code Example

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
rtn = robot.LaserSensorRecordandReplay(0, 10, 1, 0, 0.1, 1, 0, 10, 100)
print(f"LaserSensorRecordandReplay rtn is {rtn}")
rtn = robot.MoveStationary()
print(f"MoveStationary rtn is {rtn}")
rtn = robot.LaserSensorRecord1(0, 10)
print(f"LaserSensorRecordandReplay rtn is {rtn}")
robot.CloseRPC()
return 0
```

2.4.5 IO

2.4.5.1 Setting the control box digital output

prototype	`SetDO(id, status, smooth=0, block=0)`
Description	Setting the control box digital outputs
Mandatory parameters	• `id`: io number, range [0~15]; • `status`: 0 - off, 1 - on;
Default Parameters	• `smooth`: 0-not smooth, 1-smooth Default 0; • `block`:0-blocking, 1-non-blocking Default 0
Return Value	Error Code Success-0 Failure- errcode

2.4.5.2 Setting Tool Digital Outputs

prototype	`SetToolDO (id, status, smooth=0, block=0)`
Description	Setting the digital output of the tool
Mandatory parameters	• `id`: io number, range [0~1]; • `status`: 0 - off, 1 - on;
Default Parameters	• `smooth`: 0-not smooth, 1-smooth; • `block`: 0-blocking, 1-non-blocking.
Return Value	Error Code Success-0 Failure- errcode

2.4.5.3 Setting the control box analog output

Prototype	`SetAO(id,value,block=0)`
Description	Setting the control box analog output
Mandatory parameters	• `id`: io number, range [0~1]; • `value`: percentage of current or voltage value in the range [0 to 100%] corresponding to current value [0 to 20 mA] or voltage [0 to 10 V];
Default parameters	• `block`:[0]-blocking, [1]-non-blocking Default 0
Return Value	Error Code Success-0 Failure- errcode

2.4.5.4 Setting Tool Analog Outputs

Prototype	`SetToolAO(id,value,block=0)`
Description	Setup Tool Analog Output
Mandatory parameters	• `id`: io number, range [0]; • `value`: percentage of current or voltage value in the range [0 to 100%] corresponding to current value [0 to 20 mA] or voltage [0 to 10 V];
Default parameters	• `block`:[0]-blocking, [1]-non-blocking Default 0
Return Value	Error Code Success-0 Failure- errcode

2.4.5.5 Set digital, analog output code example

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
status = 1
smooth = 0
block = 0
for i in range(16):
    robot.SetDO(i, status, smooth, block)
    time.sleep(0.3)
status = 0
for i in range(16):
    robot.SetDO(i, status, smooth, block)
    time.sleep(0.3)
status = 1
for i in range(2):
    robot.SetToolDO(i, status, smooth, block)
    time.sleep(1)
status = 0
for i in range(2):
    robot.SetToolDO(i, status, smooth, block)
    time.sleep(1)
for i in range(100):
    robot.SetAO(0, i, block)
    time.sleep(0.03)
for i in range(100):
    robot.SetToolAO(0, i, block)
    time.sleep(0.03)
robot.CloseRPC()
```

2.4.5.6 Getting control box digital inputs

prototype	`GetDI(id, block=0)`
Description	Get control box digital inputs
Mandatory parameters	• `id`: io number, range [0~15];
Default Parameters	• `block`: 0-blocking, 1-non-blocking Default 0
Return Value	• errorcode Success-0 Failure- errcode • `di`: 0-low level, 1-high level

2.4.5.7 Get Tool Digital Inputs

prototype	`GetToolDI(id, block=0)`
Description	Get Tool Digital Inputs
Mandatory parameters	• `id`: io number, range [0~1];
Default Parameters	• `block`: 0-blocking, 1-non-blocking Default 0
Return Value	Error Code Success-0 Failure- errcode - `di`: 0 - low level, 1 - high level

2.4.5.8 Getting Control Box Analog Inputs

Prototype	`GetAI(id, block = 0)`
Description	Get control box analog inputs
Mandatory parameters	• `id`: io number, range [0~1];
Default Parameters	• `block`:0-blocking, 1-non-blocking Default 0
Return Value	• errorcode Success-0 Failure- errcode • `value`: Percentage of input current or voltage value, range [0~100] corresponding to current value [0~20mA] or voltage [0~10V].

2.4.5.9 Get Tool Analog Inputs

Prototype	`GetToolAI (id, block = 0)`
Description	Get end analog input
Mandatory parameters	• `id`: io number, range [0];
Default Parameters	• `block`: 0-blocking, 1-non-blocking Default 0
Return Value	• errorcode Success-0 Failure- errcode • `value`: Percentage of input current or voltage value, range [0~100] corresponding to current value [0~20mA] or voltage [0~10V].

2.4.5.10 Obtain the status of the button for recording the end point of the robot

Prototype	`GetAxlePointRecordBtnState()`
Description	Obtain the status of the button for recording the end point of the robot
Mandatory parameters	NULL
Default Parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `buttonstatus`: Button status: 0- Press, 1- release

2.4.5.11 Obtain the DO output status at the end of the robot

Prototype	`GetToolDO()`
Description	Obtain the DO output status at the end of the robot
Mandatory parameters	NULL
Default Parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `do_state`: DO output status: do0 to do1 correspond to bit1 to bit2, starting from bit0

2.4.5.12 Obtain the DO output status of the robot controller

Prototype	`GetDO()`
Description	Obtain the DO output status of the robot controller
Mandatory parameters	NULL
Default Parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `do_state_h`: DO output status: co0 to co7 correspond to bit0 to bit7. do_state_l DO output status: do0 to do7 correspond to bit0 to bit7

2.4.5.13 Get the robot DI, DO status code examples

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
block = 0
error,di = robot.GetDI(0, block)
print(f"di0: {di}")
error,tool_di = robot.GetToolDI(1, block)
print(f"tool_di1: {tool_di}")
error,ai = robot.GetAI(0, block)
```

(continues on next page)

(continued from previous page)

```
print(f"ai0: {ai:.2f}")
error,tool_ai = robot.GetToolAI(0, block)
print(f"tool_ai0: {tool_ai:.2f}")
error,button_state = robot.GetAxlePointRecordBtnState()
print(f"_button_state is: {button_state}")
error,tool_do_state = robot.GetToolDO()
print(f"tool DO state: {tool_do_state}")
error,[do_state_h, do_state_l] = robot.GetDO()
print(f"DO state hight  : {do_state_h}")
print(f"DO state low : {do_state_l}")
robot.CloseRPC()
```

2.4.5.14 Waiting for control box digital inputs

prototype	`WaitDI(id,status,maxtime,opt)`
Description	Waiting for control box digital input
Mandatory parameters	• `id`: io number, range [0~15]; • `status`: 0-off, 1-on; • `maxtime`: maximum waiting time in [ms]; • `opt`: post timeout policy, 0-program stops and prompts for timeout, 1-ignore timeout prompts program to continue execution, 2-always wait
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.5.15 Waiting for control box with multiple digital inputs

prototype	`WaitMultiDI(mode,id,status,maxtime,opt)`
Description	Waiting for control box with multiple digital inputs
Mandatory parameters	• `mode`: [0]-multiple with, [1]-multiple or; • `id`: io number, bit0~bit7 correspond to DI0~DI7, bit8~bit15 correspond to CI0~CI7; • `status`: bit0~bit7 corresponds to DI0~DI7 status, bit8~bit15 corresponds to the status of CI0~CI7 status bits [0]-off, [1]-on; • `maxtime`: maximum waiting time in [ms]; • `opt`: post-timeout policy, 0-program stops and prompts for timeout, 1-ignores timeout prompting program to continue execution, 2-always waits.
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.5.16 Waiting for tool digital inputs

prototype	`WaitToolDI(id,status,maxtime,opt)`
Description	Waiting for end digital input
Mandatory parameters	• `id`: io number, range [0~1]; • `status`: 0-off, 1-on; • `maxtime`: maximum waiting time in [ms]; • `opt`: post timeout policy, 0-program stops and prompts for timeout, 1-ignore timeout prompts program to continue execution, 2-always wait
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.5.17 Waiting for control box analog inputs

prototype	`WaitAI(id,sign,value,maxtime,opt)`
Description	Waiting for control box analog input
Mandatory parameters	• `id`: io number, range [0~1]; • `sign`: 0 - greater than, 1 - less than • `value`: percentage of input current or voltage value, range [0~100] corresponding to current value [0~20mA] or voltage [0~10V]; • `maxtime`: maximum waiting time in [ms]; • `opt`: post timeout policy, 0-program stops and prompts for timeout, 1-ignore timeout prompts program to continue execution, 2-always wait
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.5.18 Waiting for tool analog inputs

prototype	`WaitToolAI(id,sign,value,maxtime,opt)`
Description	Waiting for end analog input
Mandatory parameters	• `id`: io number, range [0]; • `sign`: 0 - greater than, 1 - less than • `value`: percentage of input current or voltage value, range [0~100] corresponding to current value [0~20mA] or voltage [0~10V]; • `maxtime`: maximum waiting time in [ms]; • `opt`: post timeout policy, 0-program stops and prompts for timeout, 1-ignore timeout prompts program to continue execution, 2-always wait
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.5.19 Waiting control box digital, analog input signal code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
status = 1
smooth = 0
block = 0
for i in range(16):
    robot.SetDO(i, status, smooth, block)
    time.sleep(0.3)
status = 0
for i in range(16):
    robot.SetDO(i, status, smooth, block)
    time.sleep(0.3)
status = 1
for i in range(2):
    robot.SetToolDO(i, status, smooth, block)
    time.sleep(1)
status = 0
for i in range(2):
    robot.SetToolDO(i, status, smooth, block)
    time.sleep(1)
for i in range(100):
    robot.SetAO(0, i, block)
    time.sleep(0.03)
for i in range(100):
    robot.SetToolAO(0, i, block)
    time.sleep(0.03)
block = 0
error,di = robot.GetDI(0, block)
print(f"di0: {di}")
error,tool_di = robot.GetToolDI(1, block)
print(f"tool_di1: {tool_di}")
error,ai = robot.GetAI(0, block)
print(f"ai0: {ai:.2f}")
error,tool_ai = robot.GetToolAI(0, block)
print(f"tool_ai0: {tool_ai:.2f}")
error,button_state = robot.GetAxlePointRecordBtnState()
print(f"_button_state is: {button_state}")
error,tool_do_state = robot.GetToolDO()
print(f"tool DO state: {tool_do_state}")
error,[do_state_h, do_state_l] = robot.GetDO()
print(f"DO state hight  : {do_state_h}")
print(f"DO state low : {do_state_l}")
rtn = robot.WaitDI(0, 1, 1000, 1)
print(f"WaitDI over; rtn is: {rtn}")
rtn = robot.WaitMultiDI(1, 3, 3, 1000, 1)
print(f"WaitDI over; rtn is: {rtn}")
rtn = robot.WaitToolDI(1, 1, 1000, 1)
print(f"WaitDI over; rtn is: {rtn}")
rtn = robot.WaitAI(0, 0, 50, 1000, 1)
```

(continues on next page)

(continued from previous page)

```
print(f"WaitDI over; rtn is: {rtn}")
rtn = robot.WaitToolAI(0, 0, 50, 1000, 1)
print(f"WaitDI over; rtn is: {rtn}")
robot.CloseRPC()
```

2.4.5.20 Set Whether Control Box DO Output Resets After Stop/Pause

New in version python: SDK-v2.0.5

Prototype	`SetOutputResetCtlBoxDO(resetFlag,reloadFlag)`
Description	Set whether control box DO output resets after stop/pause
Required Parameters	• `resetFlag`0-Do not reset; 1-Reset • `reloadFlag`Whether to reload after pause resume, 0-Do not load; 1-Load
Default Parameters	None
Return Value	Error code Success-0 Failure- errcode

2.4.5.21 Set Whether Control Box AO Output Resets After Stop/Pause

New in version python: SDK-v2.0.5

Prototype	`SetOutputResetCtlBoxAO(resetFlag,reloadFlag)`
Description	Set whether control box AO output resets after stop/pause
Required Parameters	• `resetFlag`0-Do not reset; 1-Reset • `reloadFlag`Whether to reload after pause resume, 0-Do not load; 1-Load
Default Parameters	None
Return Value	Error code Success-0 Failure- errcode

2.4.5.22 Set Whether End Tool DO Output Resets After Stop/Pause

New in version python: SDK-v2.0.5

Prototype	`SetOutputResetAxleDO(resetFlag,reloadFlag)`
Description	Set whether end tool DO output resets after stop/pause
Required Parameters	• `resetFlag`0-Do not reset; 1-Reset • `reloadFlag`Whether to reload after pause resume, 0-Do not load; 1-Load
Default Parameters	None
Return Value	Error code Success-0 Failure- errcode

2.4.5.23 Set Whether End Tool AO Output Resets After Stop/Pause

New in version python: SDK-v2.0.5

Prototype	`SetOutputResetAxleAO(resetFlag,reloadFlag)`
Description	Set whether end tool AO output resets after stop/pause
Required Parameters	• `resetFlag`0-Do not reset; 1-Reset • `reloadFlag`Whether to reload after pause resume, 0-Do not load; 1-Load
Default Parameters	None
Return Value	Error code Success-0 Failure- errcode

2.4.5.24 Set Whether Extended DO Output Resets After Stop/Pause

New in version python: SDK-v2.0.5

Prototype	`SetOutputResetExtDO (resetFlag,reloadFlag)`
Description	Set whether extended DO output resets after stop/pause
Required Parameters	• `resetFlag`0-Do not reset; 1-Reset • `reloadFlag`Whether to reload after pause resume, 0-Do not load; 1-Load
Default Parameters	None
Return Value	Error code Success-0 Failure- errcode

2.4.5.25 Set Whether Extended AO Output Resets After Stop/Pause

New in version python: SDK-v2.0.5

Prototype	`SetOutputResetExtAO (resetFlag,reloadFlag)`
Description	Set whether extended AO output resets after stop/pause
Required Parameters	• `resetFlag`0-Do not reset; 1-Reset • `reloadFlag`Whether to reload after pause resume, 0-Do not load; 1-Load
Default Parameters	None
Return Value	Error code Success-0 Failure- errcode

2.4.5.26 Set Whether SmartTool Output Resets After Stop/Pause

New in version python: SDK-v2.0.5

Prototype	`SetOutputResetSmartToolDO(resetFlag,reloadFlag)`
Description	Set whether SmartTool output resets after stop/pause
Required Parameters	• `resetFlag`0-Do not reset; 1-Reset • `reloadFlag`Whether to reload after pause resume, 0-Do not load; 1-Load
Default Parameters	None
Return Value	Error code Success-0 Failure- errcode

2.4.5.27 Code Example for Setting Output Reset After Lua Program Stop/Pause

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
for i in range(16):
    robot.SetDO(i, 1, 0, 0)
    time.sleep(0.2)
resetFlag = 0
resumeReloadFlag = 0
rtn = robot.SetOutputResetCtlBoxDO(resetFlag, resumeReloadFlag)
robot.SetOutputResetCtlBoxAO(resetFlag, resumeReloadFlag)
robot.SetOutputResetAxleDO(resetFlag, resumeReloadFlag)
robot.SetOutputResetAxleAO(resetFlag, resumeReloadFlag)
robot.SetOutputResetExtDO(resetFlag, resumeReloadFlag)
robot.SetOutputResetExtAO(resetFlag, resumeReloadFlag)
robot.SetOutputResetSmartToolDO(resetFlag, resumeReloadFlag)
robot.ProgramLoad("/fruser/test.lua")
robot.ProgramRun()
time.sleep(2)
robot.PauseMotion()
time.sleep(2)
robot.ResumeMotion()
time.sleep(2)
robot.CloseRPC()
return 0
```

2.4.6 Common Robot Settings

2.4.6.1 Setting Tool Reference Points - Six-Point Method

Prototype	`SetToolPoint(point_num)`
Description	Setting Tool Reference Points - Six Point Method
Mandatory parameters	• `point_num`: point number, range [1~6]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.2 Calculation tool coordinate system - six-point method

Prototype	`ComputeTool()`
Description	Calculate the tool coordinate system - six-point method (after setting the six tool reference points)
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `tcp_pose=[x,y,z,rx,ry,rz]`: tool coordinate system

2.4.6.3 Setting Tool Reference Points - Four Point Method

Prototype	`SetTcp4RefPoint(point_num)`
Description	Setting Tool Reference Points - Four Point Method
Mandatory parameter	`point_num`: point number, range [1~4]
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `tcp_pose=[x,y,z,rx,ry,rz]`: tool coordinate system

2.4.6.4 Calculation Tool Coordinate System - Four Point Method

prototype	`ComputeTcp4()`
Description	Calculate tool coordinate system - four-point method (after setting the four tool reference points)
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `tcp_pose=[x,y,z,rx,ry,rz]`: tool coordinate system

2.4.6.5 Calculate the tool coordinate system based on the point information

New in version python: SDK-v2.0.8

Prototype	`ComputeToolCoordWithPoints(method, pos)`
Description	Calculate the tool coordinate system based on the point information
Mandatory parameters	• `method`:Calculation method; 0-four point method; One - six point method • `pos`:The array length of the joint position group is 4 in the four-point method and 6 in the six-point method
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `tcp_offset=[x,y,z,rx,ry,rz]`:Tool coordinate system calculated from point information, unit [mm][°]

2.4.6.6 Setting the tool coordinate system

Prototype	`SetToolCoord(id,t_coord,type,install,toolID,loadNum)`
Description	Setting the tool coordinate system
Mandatory parameters	• `id`: coordinate system number, range [1~15]; • `t_coord`: Position of the tool center point relative to the center of the end flange in [mm][°]; • `type`: 0 - tool coordinate system, 1 - sensor coordinate system; • `install`: installation position, 0 - robot end, 1 - robot exterior • `toolID`: tool ID • `loadNum`: load number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.7 Setting the tool coordinate system list

Prototype	`SetToolList(id,t_coord ,type, install, loadNum)`
Description	Set up a list of tool coordinate systems
Mandatory parameters	• `id`: coordinate system number, range [1~15]; • `t_coord`: [x,y,z,rx,ry,rz] Tool center point relative to end flange center position in [mm][°]; • `type`: 0 - tool coordinate system, 1 - sensor coordinate system; • `install`: installation position, 0 - robot end, 1 - robot exterior • `loadNum`: load number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.8 Get the current tool coordinate system

Prototype	`GetTCPOffset(flag=1)`
Description	Get current tool coordinate system
Mandatory parameters	NULL
Default parameters	`flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `tcp_offset=[x,y,z,rx,ry,rz]`: Relative position of the current tool coordinate system in [mm][°]

2.4.6.9 Robot tool coordinate system manipulation code example

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
p1Desc = [186.331, 487.913, 209.850, 149.030, 0.688, -114.347]
p2Desc = [69.721, 535.073, 202.882, -144.406, -14.775, -89.012]
p3Desc = [146.861, 578.426, 205.598, 175.997, -36.178, -93.437]
p4Desc = [136.284, 509.876, 225.613, 178.987, 1.372, -100.696]
p5Desc = [138.395, 505.972, 298.016, 179.134, 2.147, -101.110]
p6Desc = [105.553, 454.325, 232.017, -179.426, 0.444, -99.952]
p1Joint = [-127.876, -75.341, 115.417, -122.741, -59.820, 74.300]
p2Joint = [-101.780, -69.828, 110.917, -125.740, -127.841, 74.300]
p3Joint = [-112.851, -60.191, 86.566, -80.676, -97.463, 74.300]
p4Joint = [-116.397, -76.281, 113.845, -128.611, -88.654, 74.299]
p5Joint = [-116.814, -82.333, 109.162, -118.662, -88.585, 74.302]
p6Joint = [-115.649, -84.367, 122.447, -128.663, -90.432, 74.303]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
posJ = [p1Joint, p2Joint, p3Joint, p4Joint, p5Joint, p6Joint]
rtn,coordRtn = robot.ComputeToolCoordWithPoints(1, posJ)
print(f"ComputeToolCoordWithPoints    {rtn}  coord is {coordRtn[0]} {coordRtn[1]} {coordRtn[2]} {coordRtn[3]} {coordRtn[4]} {coordRtn[5]}")
robot.MoveJ(joint_pos=p1Joint,tool=0, user=0, vel=100)
robot.SetToolPoint(1)
robot.MoveJ(joint_pos=p2Joint,tool=0, user=0, vel=100)
robot.SetToolPoint(2)
robot.MoveJ(joint_pos=p3Joint,tool=0, user=0, vel=100)
robot.SetToolPoint(3)
robot.MoveJ(joint_pos=p4Joint,tool=0, user=0, vel=100)
robot.SetToolPoint(4)
robot.MoveJ(joint_pos=p5Joint,tool=0, user=0, vel=100)
robot.SetToolPoint(5)
robot.MoveJ(joint_pos=p6Joint,tool=0, user=0, vel=100)
robot.SetToolPoint(6)
rtn,coordRtn = robot.ComputeTool()
print(f"6 Point ComputeTool        {rtn}  coord is {coordRtn[0]} {coordRtn[1]}
```

(continues on next page)

(continued from previous page)

```
↪{coordRtn[2]} {coordRtn[3]} {coordRtn[4]} {coordRtn[5]}")
robot.SetToolList(1, coordRtn, 0, 0, 0)
robot.MoveJ(joint_pos=p1Joint,tool=0, user=0, vel=100)
robot.SetTcp4RefPoint(1)
robot.MoveJ(joint_pos=p2Joint,tool=0, user=0, vel=100)
robot.SetTcp4RefPoint(2)
robot.MoveJ(joint_pos=p3Joint,tool=0, user=0, vel=100)
robot.SetTcp4RefPoint(3)
robot.MoveJ(joint_pos=p4Joint,tool=0, user=0, vel=100)
robot.SetTcp4RefPoint(4)
rtn,coordRtn = robot.ComputeTcp4()
print(f"4 Point ComputeTool        {rtn}  coord is {coordRtn[0]} {coordRtn[1]}
↪{coordRtn[2]} {coordRtn[3]} {coordRtn[4]} {coordRtn[5]}")
robot.SetToolCoord(2, coordRtn, 0, 0, 1, 0)
rtn,getCoord = robot.GetTCPOffset(0)
print(f"GetTCPOffset    {rtn}  coord is {getCoord[0]} {getCoord[1]} {getCoord[2]}
↪{getCoord[3]} {getCoord[4]} {getCoord[5]}")
robot.CloseRPC()
```

2.4.6.10 Setting External Tool Reference Points-Six-Point Method

Prototype	`SetExTCPPoint(point_num)`
Description	Setting the external tool reference point - three-point method
Mandatory parameters	• `point_num`: point number, range [1~3]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.11 Calculation of the external tool coordinate system - Six-point method

Prototype	`ComputeExTCF(point_num)`
Description	Calculate external tool coordinate system - three-point method (after setting three reference points)
Mandatory parameter	`point_num`: point number, range [1~3]
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `etcp=[x,y,z,rx,ry,rz]`: external tool coordinate system

2.4.6.12 Setting the external tool coordinate system

Prototype	`SetExToolCoord(id,etcp,etool)`
Description	Setting the external tool coordinate system
Mandatory parameters	• `id`: coordinate system number, range [0~14]; • `etcp`: External tool coordinate system in [mm][°]; • `etool`: end-tool coordinate system in [mm] [°];
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.13 Setting up a list of external tool coordinate systems

Prototype	`SetExToolList(id,etcp ,etool)`
Description	Set the list of external tool coordinate systems
Mandatory parameters	• `id`: coordinate system number, range [0~14]; • `etcp`: External tool coordinate system in [mm][°]; • `etool`: end-tool coordinate system in [mm] [°];
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.14 Example code for robot external tool coordinate system operation

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
p1Desc = [-89.606, 779.517, 193.516, 178.000, 0.476, -92.484]
p1Joint = [-108.145, -50.137, 85.818, -125.599, -87.946, 74.329]
p2Desc = [-24.656, 850.384, 191.361, 177.079, -2.058, -95.355]
p2Joint = [-111.024, -41.538, 69.222, -114.913, -87.743, 74.329]
p3Desc = [-99.813, 766.661, 241.878, -176.817, 1.917, -91.604]
p3Joint = [-107.266, -56.116, 85.971, -122.560, -92.548, 74.331]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
posTCP = [p1Desc, p2Desc, p3Desc]
robot.MoveJ(joint_pos=p1Joint,tool=1, user=0, vel=50)
robot.SetExTCPPoint(1)
robot.MoveJ(joint_pos=p2Joint,tool=1, user=0, vel=50)
robot.SetExTCPPoint(2)
robot.MoveJ(joint_pos=p3Joint,tool=1, user=0, vel=50)
robot.SetExTCPPoint(3)
rtn,coordRtn = robot.ComputeExTCF()
print(f"ComputeExTCF {rtn}  coord is {coordRtn[0]} {coordRtn[1]} {coordRtn[2]}
↪{coordRtn[3]} {coordRtn[4]} {coordRtn[5]}")
```

(continues on next page)

(continued from previous page)

```
robot.SetExToolCoord(1, coordRtn, offdese)
robot.SetExToolList(1, coordRtn, offdese)
robot.CloseRPC()
```

2.4.6.15 Setting the workpiece reference point - three-point method

Prototype	`SetWObjCoordPoint(point_num)`
Description	Setting the workpiece reference point - 3-point method
Mandatory parameter	`point_num`: point number, range [1~3]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.16 Calculation of the workpiece coordinate system - three-point method

Prototype	`ComputeWObjCoord(method, refFrame)`
Description	Calculate the workpiece coordinate system - three-point method (three reference points are set and then calculated);
Mandatory parameters	• `method`: calculation method 0: origin-x-axis-z-axis, 1: origin-x-axis-xy-plane • `refFrame`: reference coordinate system
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `wobj_pose=[x,y,z,rx,ry,rz]`: workpiece coordinate system

2.4.6.17 Setting the workpiece coordinate system

prototype	`SetWObjCoord(id, coord, refFrame)`
Description	Setting the workpiece coordinate system
Mandatory parameters	• `id`: coordinate system number, range [0~14]; • `COORD`: Position of the workpiece in the coordinate system relative to the center of the end flange in [mm][°]. • `refFrame`: reference coordinate system
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.18 Setting the list of workpiece coordinate systems

prototype	`SetWObjList(id, coord, refFrame)`
Description	Set the list of workpiece coordinate systems
Mandatory parameters	• `id`: coordinate system number, range [0~14]; • `COORD`: Position of the workpiece in the coordinate system relative to the center of the end flange in [mm][°]. • `refFrame`: reference coordinate system
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.19 Calculate the workpiece coordinate system based on the point information

New in version python: SDK-v2.0.8

Prototype	`ComputeWObjCoordWithPoints(method, pos, refFrame)`
Description	Calculate the workpiece coordinate system based on the point information
Mandatory parameters	• `method`:Calculation method; 0: origin - X-axis - Z-axis 1: origin - X-axis -xy plane • `pos`:Three TCP location groups • `refFrame`: reference coordinate system
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `wobj_offset=[x,y,z,rx,ry,rz]`:Workpiece coordinate system calculated from point information, unit [mm][°]

2.4.6.20 Get the current workpiece coordinate system

prototype	`GetWObjOffset(flag=1)`
Description	Get current workpiece coordinate system
Mandatory parameters	NULL
Default parameters	`flag`: 0-blocking, 1-non-blocking, default 1
Return Value	• errorcode Success-0 Failure- errcode • `wobj_offset=[x,y,z,rx,ry,rz]`: Relative position of the current workpiece coordinate system in [mm][°]

2.4.6.21 Example of robot workpiece coordinate system manipulation code

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
p1Desc = [-89.606, 779.517, 193.516, 178.000, 0.476, -92.484]
p2Desc = [-24.656, 850.384, 191.361, 177.079, -2.058, -95.355]
p3Desc = [-99.813, 766.661, 241.878, -176.817, 1.917, -91.604]
p1Joint = [-108.145, -50.137, 85.818, -125.599, -87.946, 74.329]
p2Joint = [-111.024, -41.538, 69.222, -114.913, -87.743, 74.329]
p3Joint = [-107.266, -56.116, 85.971, -122.560, -92.548, 74.331]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
posTCP = [p1Desc, p2Desc, p3Desc]
rtn,coordRtn = robot.ComputeWObjCoordWithPoints(1, posTCP, 0)
print(f"ComputeWObjCoordWithPoints    {rtn}  coord is {coordRtn[0]} {coordRtn[1]} {coordRtn[2]} {coordRtn[3]} {coordRtn[4]} {coordRtn[5]}")
robot.MoveJ(joint_pos=p1Joint,tool=1, user=0, vel=100)
robot.SetWObjCoordPoint(1)
robot.MoveJ(joint_pos=p2Joint,tool=1, user=0, vel=100)
robot.SetWObjCoordPoint(2)
robot.MoveJ(joint_pos=p3Joint,tool=1, user=0, vel=100)
robot.SetWObjCoordPoint(3)
rtn,coordRtn = robot.ComputeWObjCoord(1, 0)
print(f"ComputeWObjCoord   {rtn}  coord is {coordRtn[0]} {coordRtn[1]} {coordRtn[2]} {coordRtn[3]} {coordRtn[4]} {coordRtn[5]}")
robot.SetWObjCoord(1, coordRtn, 0)
robot.SetWObjList(1, coordRtn, 0)
rtn,getWobjDesc = robot.GetWObjOffset(0)
print(f"GetWObjOffset    {rtn}  coord is {getWobjDesc[0]} {getWobjDesc[1]} {getWobjDesc[2]} {getWobjDesc[3]} {getWobjDesc[4]} {getWobjDesc[5]}")
robot.CloseRPC()
```

2.4.6.22 Setting the global speed

Prototype	`SetSpeed(vel)`
Description	Set global speed
Mandatory parameter	• `vel`: percentage of speed, range [0~100]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.23 Setting robot acceleration

New in version python: SDK-v2.0.4

Prototype	`SetOaccScale(acc)`
Description	Setting the robot acceleration
Mandatory parameter	• `acc`: percentage of robot acceleration
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.24 Getting the default speed

Prototype	`GetDefaultTransVel()`
Description	Get Default Speed
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `vel`: default speed in [mm/s]

2.4.6.25 Setting the end load weight

Prototype	`SetLoadWeight(loadNum, weight)`
Description	Set the end load weight, incorrect load weight setting may cause the robot to go out of control in drag mode
Mandatory parameters	• `loadNum`:load number • `weight`: unit [kg]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.26 Setting the end load center of mass coordinates

Prototype	`SetLoadCoord(x,y,z,loadNum = 0)`
Description	Set end-load center of mass coordinates, incorrect load center of mass setting may cause robot to go out of control in drag mode
Mandatory parameter	• `x`: center of mass coordinates in [mm]. • `y`: coordinates of the center of mass in [mm]. • `z`: coordinates of the center of mass in [mm]
Default parameters	• `loadNum`: Payload number, 0 by default
Return Value	Error Code Success-0 Failure- errcode

2.4.6.27 Get the weight of the current load

Prototype	`GetTargetPayload(flag=1)`
Description	Get the quality of the current load
Mandatory parameters	NULL
Default parameters	`flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `weight`: current load weight in [kg]

2.4.6.28 Get the center of mass of the current load

Prototype	`GetTargetPayloadCog(flag=1)`
Description	Get the center of mass of the current load
Mandatory parameters	NULL
Default parameters	`flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `cog=[x,y,z]`: coordinates of the current center of mass in [mm]

2.4.6.29 Setting the robot mounting method - fixed mounting

Prototype	`SetRobotInstallPos(method)`
description	set robot mounting method - fixed mounting, wrong mounting method setting can cause robot to lose control in drag mode
Mandatory parameters	• `method`: 0-flat loading, 1-side loading, 2-hanging loading
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.30 Setting the robot mounting angle - free mounting

Prototype	`SetRobotInstallAngle(yangle,zangle)`
Description	Setting the robot mounting angle - free mounting, wrong mounting angle setting can cause the robot to go out of control in drag mode
Mandatory parameters	• `yangle`: angle of inclination; • `zangle`: angle of rotation
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.31 Getting the robot mounting angle

Prototype	`GetRobotInstallAngle()`
Description	Get robot mounting angle
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `[yangle,zangle]`: yangle - angle of inclination, zangle - angle of rotation.

2.4.6.32 Setting system variable values

Prototype	`SetSysVarValue(id,value)`
Description	Setting System Variables
Mandatory parameters	• `id`: variable number, in the range [1~20]. • `value`: variable value
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.33 Getting system variable values

Prototype	`GetSysVarValue(id)`
Description	Get system variable values
Mandatory parameters	• `id`: system variable number, range [1~20]
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `var_value`: system variable value

2.4.6.34 Robot common setup code examples

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
for i in range(1, 100):
    robot.SetSpeed(i)
    robot.SetOaccScale(i)
    time.sleep(0.03)
error,defaultVel = robot.GetDefaultTransVel()
print(f"GetDefaultTransVel is {defaultVel}")
for i in range(1, 21):
```

(continues on next page)

(continued from previous page)

```
    robot.SetSysVarValue(i, i + 0.5)
    time.sleep(0.1)
for i in range(1, 21):
    value = robot.GetSysVarValue(i)
    print(f"sys value {i} is: {value}")
    time.sleep(0.1)
robot.SetLoadWeight(0, 2.5)
robot.SetLoadCoord(3.0,4.0,5.0)
time.sleep(1)
error,getLoad = robot.GetTargetPayload(0)
error,getLoadTran = robot.GetTargetPayloadCog(0)
print(f"get load is {getLoad}; get load cog is {getLoadTran[0]} {getLoadTran[1]}
↪{getLoadTran[2]}")
robot.SetRobotInstallPos(0)
robot.SetRobotInstallAngle(15.0, 25.0)
error,[anglex, angley] = robot.GetRobotInstallAngle()
print(f"GetRobotInstallAngle x: {anglex}; y: {angley}")
robot.CloseRPC()
```

2.4.6.35 Joint Friction Compensation Switch

Prototype	`FrictionCompensationOnOff(state)`
Description	Joint Friction Compensation Switch
Mandatory parameters	• `state`: 0-off, 1-on
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.36 Setting the joint friction compensation coefficients - positive loading

Prototype	`SetFrictionValue_level(coeff)`
Description	Setting the joint friction compensation coefficient - Fixed mounting - Positive mounting
Mandatory parameters	• `coeff=[j1,j2,j3,j4,j5,j6]`: six joint compensation coefficients
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.37 Setting the joint friction compensation coefficient - side mounting

Prototype	`SetFrictionValue_wall(coeff)`
Description	Setting the joint friction compensation coefficient - fixed mounting - side mounting
Mandatory parameters	• `coeff=[j1,j2,j3,j4,j5,j6]`: six joint compensation coefficients
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.38 Setting the Joint Friction Compensation Factor - Inverted

Prototype	`SetFrictionValue_ceiling(coeff)`
Description	Setting the joint friction compensation coefficient - fixed mounting - inverted mounting
Mandatory parameters	• `coeff=[j1,j2,j3,j4,j5,j6]`: six joint compensation coefficients
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.39 Setting the joint friction compensation factor - free mounting

Prototype	`SetFrictionValue_freedom(coeff)`
Description	Setting the joint friction compensation factor - free mounting
Mandatory parameters	• `coeff=[j1,j2,j3,j4,j5,j6]`: six joint compensation coefficients
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.40 Robot setup joint friction compensation code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
lcoeff = [0.9, 0.9, 0.9, 0.9, 0.9, 0.9]
wcoeff = [0.4, 0.4, 0.4, 0.4, 0.4, 0.4]
ccoeff = [0.6, 0.6, 0.6, 0.6, 0.6, 0.6]
fcoeff = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5]
rtn = robot.FrictionCompensationOnOff(1)
print(f"FrictionCompensationOnOff rtn is {rtn}")
rtn = robot.SetFrictionValue_level(lcoeff)
print(f"SetFrictionValue_level rtn is {rtn}")
rtn = robot.SetFrictionValue_wall(wcoeff)
print(f"SetFrictionValue_wall rtn is {rtn}")
rtn = robot.SetFrictionValue_ceiling(ccoeff)
print(f"SetFrictionValue_ceiling rtn is {rtn}")
```

(continues on next page)

(continued from previous page)

```
rtn = robot.SetFrictionValue_freedom(fcoeff)
print(f"SetFrictionValue_freedom rtn is {rtn}")
robot.CloseRPC()
```

2.4.6.41 Query Robot Error Code

Prototype	`GetRobotErrorCode()`
Description	Query Robot Error Code
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `[maincode subcode]`: robot error code, maincode - main error code, subcode - suberror code

2.4.6.42 Error state clearing

Prototype	`ResetAllError()`
Description	Error state clearing, only resettable errors can be cleared
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.43 Robot fault state acquisition and clearing error code examples

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
p1Joint = [-108.145, -50.137, 85.818, -125.599, -87.946, 74.329]
robot.MoveJ(joint_pos=p1Joint, tool=5, user=2, vel=50)
time.sleep(1)
error,[maincode, subcode] = robot.GetRobotErrorCode()
print(f"robot maincode is {maincode}; subcode is {subcode}")
time.sleep(1)
robot.ResetAllError()
time.sleep(1)
error,[maincode, subcode] = robot.GetRobotErrorCode()
print(f"robot maincode is {maincode}; subcode is {subcode}")
robot.CloseRPC()
```

2.4.6.44 Set the monitoring parameters for the temperature and fan speed of the wide-voltage control box

New in version python: SDK-v2.1.3

Prototype	`SetWideBoxTempFanMonitorParam(enable, period)`
Description	Set the monitoring parameters for the temperature and fan speed of the wide-voltage control box
Mandatory parameters	• `enable`: 0- Monitoring is not enabled; 1- Enable monitoring • `period`: Monitoring period (s), ranging from 1 to 100
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.45 Obtain the monitoring parameters of the temperature and fan speed of the wide-voltage control box

New in version python: SDK-v2.1.3

Prototype	`GetWideBoxTempFanMonitorParam()`
Description	Obtain the monitoring parameters of the temperature and fan speed of the wide-voltage control box
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `enable`: 0- Monitoring is not enabled; 1- Enable monitoring • `period`: Monitoring period (s), ranging from 1 to 100

2.4.6.46 Sample code for obtaining wide voltage control box temperature and fan current status

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.SetWideBoxTempFanMonitorParam(1, 2)
error, enable, period = robot.GetWideBoxTempFanMonitorParam()
print(f"GetWideBoxTempFanMonitorParam enable is:{enable},period is:{period}")
for i in range(100):
    error, pkg = robot.GetRobotRealTimeState()
    print(f"robot ctrl box temp is:{pkg.wideVoltageCtrlBoxTemp},fan current is:{pkg.
↪wideVoltageCtrlBoxFanCurrent}")
    time.sleep(0.1)
rtn = robot.SetWideBoxTempFanMonitorParam(0, 2)
print(f"SetWideBoxTempFanMonitorParam rtn is:{rtn}")
error, enable, period = robot.GetWideBoxTempFanMonitorParam()
print(f"GetWideBoxTempFanMonitorParam enable is:{enable},period is:{period}")
```

(continues on next page)

(continued from previous page)

```
for i in range(100):
    error, pkg = robot.GetRobotRealTimeState()
    print(f"robot ctrl box temp is:{pkg.wideVoltageCtrlBoxTemp},fan current is:{pkg.
↪wideVoltageCtrlBoxFanCurrent}")
    time.sleep(0.1)
robot.CloseRPC()
```

2.4.6.47 Sets the focus point

New in version python: SDK-v2.1.4

Prototype	`SetFocusCalibPoint(pointNum, point)`
Description	Sets the focus point
Mandatory parameters	• `pointNum`The number of the focus calibration point 1-8 • `point`Coordinate the points
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.48 Calculate the focus calibration result

New in version python: SDK-v2.1.4

Prototype	`ComputeFocusCalib(pointNum)`
Description	Calculate the focus calibration result
Mandatory parameters	• `pointNum`Number of calibration points
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `resultPos`Calibration result XYZ • `accuracy`Calibration accuracy error

2.4.6.49 Enable focus following

New in version python: SDK-v2.1.4

Prototype	`FocusStart(kp=50.0, kpredic=19.0, aMax=1440, vMax=180, type=0)`
Description	Enable focus following
Mandatory parameters	NULL
Default parameters	• `kp`Scale parameter. Default is 50.0 • `kpredic`Feedforward parameter, default 19.0 • `aMax`Maximum angular acceleration limit, default 1440°/s^2 • `vMax`Maximum angular speed limit, default 180°/s • `type`Lock X-axis pointing (0-reference input vector; 1-level; 2- vertical), 0 by default
Return Value	Error Code Success-0 Failure- errcode

2.4.6.50 Stop focusing following

New in version python: SDK-v2.1.4

Prototype	`FocusEnd()`
Description	Stop focusing following
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.51 Set the focus coordinates

New in version python: SDK-v2.1.4

Prototype	`SetFocusPosition(pos)`
Description	Set the focus coordinates
Mandatory parameters	• `pos`Focus coordinate XYZ
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.52 Robot focus following code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
p1Desc = [186.331, 487.913, 209.850, 149.030, 0.688, -114.347]
p1Joint = [-127.876, -75.341, 115.417, -122.741, -59.820, 74.300]
p2Desc = [69.721, 535.073, 202.882, -144.406, -14.775, -89.012]
p2Joint = [-101.780, -69.828, 110.917, -125.740, -127.841, 74.300]
p3Desc = [146.861, 578.426, 205.598, 175.997, -36.178, -93.437]
p3Joint = [-112.851, -60.191, 86.566, -80.676, -97.463, 74.300]
```

(continues on next page)

(continued from previous page)

```
p4Desc = [136.284, 509.876, 225.613, 178.987, 1.372, -100.696]
p4Joint = [-116.397, -76.281, 113.845, -128.611, -88.654, 74.299]
p5Desc = [138.395, 505.972, 298.016, 179.134, 2.147, -101.110]
p5Joint = [-116.814, -82.333, 109.162, -118.662, -88.585, 74.302]
p6Desc = [105.553, 454.325, 232.017, -179.426, 0.444, -99.952]
p6Joint = [-115.649, -84.367, 122.447, -128.663, -90.432, 74.303]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 100, 0, 0, 0]
robot.MoveJ(joint_pos=p1Joint,tool=0,user=0,vel=100,acc=100,ovl=100,exaxis_pos=exaxisPos,
↪blendT=-1,offset_flag=0,offset_pos=offdese)
robot.SetTcp4RefPoint(1)
robot.MoveJ(joint_pos=p2Joint,tool=0,user=0,vel=100,acc=100,ovl=100,exaxis_pos=exaxisPos,
↪blendT=-1,offset_flag=0,offset_pos=offdese)
robot.SetTcp4RefPoint(2)
robot.MoveJ(joint_pos=p3Joint,tool=0,user=0,vel=100,acc=100,ovl=100,exaxis_pos=exaxisPos,
↪blendT=-1,offset_flag=0,offset_pos=offdese)
robot.SetTcp4RefPoint(3)
robot.MoveJ(joint_pos=p4Joint,tool=0,user=0,vel=100,acc=100,ovl=100,exaxis_pos=exaxisPos,
↪blendT=-1,offset_flag=0,offset_pos=offdese)
robot.SetTcp4RefPoint(4)
rtn,coordRtn = robot.ComputeTcp4()
print(f"4 Point ComputeTool {rtn} coord is {coordRtn[0]} {coordRtn[1]} {coordRtn[2]} "
      f"{coordRtn[3]} {coordRtn[4]} {coordRtn[5]}")
robot.SetToolCoord(1, coordRtn, 0, 0, 1, 0)
error, p1Desc = robot.GetForwardKin(p1Joint)
error, p2Desc = robot.GetForwardKin(p2Joint)
error, p3Desc = robot.GetForwardKin(p3Joint)
robot.SetFocusCalibPoint(1, p1Desc)
robot.SetFocusCalibPoint(2, p2Desc)
robot.SetFocusCalibPoint(3, p3Desc)
rtn, resultPos, accuracy = robot.ComputeFocusCalib(pointNum=3)
print(f"ComputeFocusCalib coord is {rtn} {resultPos[0]} {resultPos[1]} {resultPos[2]}␣
↪accuracy is {accuracy}")
rtn = robot.SetFocusPosition(resultPos)
error, p5Desc = robot.GetForwardKin(p5Joint)
error, p6Desc = robot.GetForwardKin(p6Joint)
robot.MoveL(desc_pos=p5Desc,tool=1,user=0,vel=10,acc=100,ovl=100,blendR=-1,blendMode=0,
↪exaxis_pos=exaxisPos,search=0,offset_flag=1,offset_pos=offdese)
robot.MoveL(desc_pos=p6Desc,tool=1,user=0,vel=10,acc=100,ovl=100,blendR=-1,blendMode=0,
↪exaxis_pos=exaxisPos,search=0,offset_flag=1,offset_pos=offdese)
robot.FocusStart(50, 19, 710, 90, 0)
robot.MoveL(desc_pos=p5Desc,tool=1,user=0,vel=10,acc=100,ovl=100,blendR=-1,blendMode=0,
↪exaxis_pos=exaxisPos,search=0,offset_flag=1,offset_pos=offdese)
robot.MoveL(desc_pos=p6Desc,tool=1,user=0,vel=10,acc=100,ovl=100,blendR=-1,blendMode=0,
↪exaxis_pos=exaxisPos,search=0,offset_flag=1,offset_pos=offdese)
robot.FocusEnd()
robot.CloseRPC()
```

2.4.6.53 Joint torque sensor sensitivity calibration function is enabled

New in version python: SDK-v2.1.7

Prototype	`JointSensitivityEnable(status)`
Description	Joint torque sensor sensitivity calibration function is enabled
Mandatory parameters	• `status`0- closed; 1- On
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.54 Joint torque sensor sensitivity data acquisition

New in version python: SDK-v2.1.7

Prototype	`JointSensitivityCollect()`
Description	Joint torque sensor sensitivity data acquisition
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.55 The sensitivity calibration results of the joint torque sensor were obtained

New in version python: SDK-v2.1.7

Prototype	`JointSensitivityCalibration()`
Description	The sensitivity calibration results of the joint torque sensor were obtained
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `calibResult`j1~j6 Joint sensitivity [0-1] • `linearity`nj1~j6 joint linearity[0-1]

2.4.6.56 Get Joint Torque Sensor Hysteresis Error

Prototype	`JointHysteresisError()`
Description	Get joint torque sensor hysteresis error
Required Parameters	None
Default Parameters	None
Return Value	• Error code Success-0 Failure- errcode • `hysteresisError`j1~j6 joint hysteresis error

2.4.6.57 Get Joint Torque Sensor Repeatability

Prototype	`JointRepeatability()`
Description	Get joint torque sensor repeatability
Required Parameters	None
Default Parameters	None
Return Value	• Error code Success-0 Failure- errcode • `repeatability`j1~j6 joint repeatability

2.4.6.58 Set Joint Force Sensor Parameters

Prototype	`SetAdmittanceParams(M, B, K, threshold, sensitivity, setZeroFlag)`
Description	Set joint force sensor parameters
Required Parameters	• `M`J1-J6 mass coefficient [0.001 ~ 10] • `B`J1-J6 damping coefficient [0.001 ~ 10] • `K`J1-J6 stiffness coefficient [0.001 ~ 10] • `threshold`Force control threshold, Nm • `sensitivity`Sensitivity, Nm/V, [0 ~ 10] • `setZeroFlag`Function enable flag; 0-Off; 1-On; 2-Record zero point at position 1; 3-Record zero point at position 2
Default Parameters	None
Return Value	• Error code Success-0 Failure- errcode

2.4.6.59 Sample code for automatic calibration of joint torque sensor sensitivity

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
rtn = robot.JointSensitivityEnable(0)
rtn = robot.JointSensitivityEnable(1)
print(f"JointSensitivityEnable rtn is {rtn}")
curJPos = [0.0] * 6
rtn, curJPos = robot.GetActualJointPosDegree(0)
epos = [0.0] * 4
offset_pos = [0.0] * 6
jointPos1 = [curJPos[0], 0.0, 0.0, -90.0, 0.02, curJPos[5]]
descPos1 = [0.0] * 6
rtn, descPos1 = robot.GetForwardKin(jointPos1)
robot.MoveJ(joint_pos=jointPos1, desc_pos=descPos1, tool=0, user=0, vel=100, acc=100,␣
↪ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.2)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 1 rtn is {rtn}")
```

(continues on next page)

(continued from previous page)

```
time.sleep(0.1)
jointPos2 = [curJPos[0], -30.0, 0.0, -90.0, 0.02, curJPos[5]]
descPos2 = [0.0] * 6
rtn, descPos2 = robot.GetForwardKin(jointPos2)
robot.MoveJ(joint_pos=jointPos2, desc_pos=descPos2, tool=0, user=0, vel=100, acc=100,␣
↪ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 2 rtn is {rtn}")
time.sleep(0.1)
jointPos3 = [curJPos[0], -60.0, 0.0, -90.0, 0.02, curJPos[5]]
descPos3 = [0.0] * 6
rtn, descPos3 = robot.GetForwardKin(jointPos3)
robot.MoveJ(joint_pos=jointPos3, desc_pos=descPos3, tool=0, user=0, vel=100, acc=100,␣
↪ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 3 rtn is {rtn}")
time.sleep(0.1)
jointPos4 = [curJPos[0], -90.0, 0.0, -90.0, 0.02, curJPos[5]]
descPos4 = [0.0] * 6
rtn, descPos4 = robot.GetForwardKin(jointPos4)
robot.MoveJ(joint_pos=jointPos4, desc_pos=descPos4, tool=0, user=0, vel=100, acc=100,␣
↪ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 4 rtn is {rtn}")
time.sleep(0.1)
jointPos5 = [curJPos[0], -120.0, 0.0, -90.0, 0.02, curJPos[5]]
descPos5 = [0.0] * 6
rtn, descPos5 = robot.GetForwardKin(jointPos5)
robot.MoveJ(joint_pos=jointPos5, desc_pos=descPos5, tool=0, user=0, vel=100, acc=100,␣
↪ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 5 rtn is {rtn}")
time.sleep(0.1)
jointPos6 = [curJPos[0], -150.0, 0.0, -90.0, 0.02, curJPos[5]]
descPos6 = [0.0] * 6
rtn, descPos6 = robot.GetForwardKin(jointPos6)
robot.MoveJ(joint_pos=jointPos6, desc_pos=descPos6, tool=0, user=0, vel=100, acc=100,␣
↪ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 6 rtn is {rtn}")
time.sleep(0.1)
jointPos7 = [curJPos[0], -180.0, 0.0, -90.0, 0.02, curJPos[5]]
descPos7 = [0.0] * 6
rtn, descPos7 = robot.GetForwardKin(jointPos7)
robot.MoveJ(joint_pos=jointPos7, desc_pos=descPos7, tool=0, user=0, vel=100, acc=100,␣
↪ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
```

(continues on next page)

(continued from previous page)

```
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 7 rtn is {rtn}")
time.sleep(0.1)
robot.MoveJ(joint_pos=jointPos6, desc_pos=descPos6, tool=0, user=0, vel=100, acc=100, ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 8 rtn is {rtn}")
time.sleep(0.1)
robot.MoveJ(joint_pos=jointPos5, desc_pos=descPos5, tool=0, user=0, vel=100, acc=100, ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 9 rtn is {rtn}")
time.sleep(0.1)
robot.MoveJ(joint_pos=jointPos4, desc_pos=descPos4, tool=0, user=0, vel=100, acc=100, ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 10 rtn is {rtn}")
time.sleep(0.1)
robot.MoveJ(joint_pos=jointPos3, desc_pos=descPos3, tool=0, user=0, vel=100, acc=100, ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 11 rtn is {rtn}")
time.sleep(0.1)
robot.MoveJ(joint_pos=jointPos2, desc_pos=descPos2, tool=0, user=0, vel=100, acc=100, ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.1)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 12 rtn is {rtn}")
time.sleep(0.1)
robot.MoveJ(joint_pos=jointPos1, desc_pos=descPos1, tool=0, user=0, vel=100, acc=100, ovl=100, exaxis_pos=epos, blendT=-1, offset_flag=0, offset_pos=offset_pos)
time.sleep(0.2)
rtn = robot.JointSensitivityCollect()
print(f"JointSensitivityCollect 13 rtn is {rtn}")
time.sleep(0.1)
calibResult = [0.0] * 6
linearity = [0.0] * 6
rtn,calibResult, linearity = robot.JointSensitivityCalibration()
print(f"JointSensitivityCalibration rtn is {rtn}")
rtn = robot.JointSensitivityEnable(0)
print(f"JointSensitivityEnable rtn is {rtn}")
print(f"jointSensor Calib result is {calibResult[0]},{calibResult[1]},{calibResult[2]},{calibResult[3]},{calibResult[4]},{calibResult[5]}")
print( f"jointSensor linearity is {linearity[0]},{linearity[1]},{linearity[2]},{linearity[3]},{linearity[4]},{linearity[5]}")
hysteresisError = [0.0] * 6
rtn,hysteresisError = robot.JointHysteresisError()
print(f"JointHysteresisError result is {hysteresisError[0]},{hysteresisError[1]},
```

(continues on next page)

(continued from previous page)

```
↪{hysteresisError[2]},{hysteresisError[3]},{hysteresisError[4]},{hysteresisError[5]}")
repeatability = [0.0] * 6
rtn,repeatability = robot.JointRepeatability()
print(f"JointRepeatability result is {repeatability[0]},{repeatability[1]},
↪{repeatability[2]},{repeatability[3]},{repeatability[4]},{repeatability[5]}")
M = [1.0] * 6
B = [1.0] * 6
K = [0.0] * 6
threshold = [1.0] * 6
setZeroFlag = 1
rtn = robot.SetAdmittanceParams(M, B, K, threshold, calibResult, setZeroFlag)
print(f"SetAdmittanceParams rtn is {rtn}")
robot.CloseRPC()
```

2.4.6.60 The number of error frames at eight slave ports of the robot is obtained

New in version python: SDK-v2.1.7

Prototype	`GetSlavePortErrCounter()`
Description	The number of error frames at eight slave ports of the robot is obtained
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `inRecvErr`Input the number of error frames received • `inCRCErr`Input the number of CRC error frames • `inTransmitErr`Input the number of forwarding error frames • `inLinkErr`Enter the number of link error frames • `outRecvErr`Outputs the number of frames received incorrectly • `outCRCErr`The number of CRC error frames is output • `outTransmitErr`Outputs the number of forwarding error frames • `outLinkErr`Output link error frames

2.4.6.61 Slave port error frame reset

New in version python: SDK-v2.1.7

Prototype	`JointSensitivityEnable(slaveID)`
Description	Slave port error frame reset
Mandatory parameters	• `slaveID`Slave station numbers 0 to 7
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.62 Get sample slave port error frame code

```python
from fairino import Robot
import time
import threading
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
inRecvErr = [0] * 8
inCRCErr = [0] * 8
inTransmitErr = [0] * 8
inLinkErr = [0] * 8
outRecvErr = [0] * 8
outCRCErr = [0] * 8
outTransmitErr = [0] * 8
outLinkErr = [0] * 8
rtn,inRecvErr, inCRCErr, inTransmitErr, inLinkErr, outRecvErr, outCRCErr, outTransmitErr, outLinkErr = robot.GetSlavePortErrCounter()
for i in range(8):
    if inRecvErr[i] != 0:
        print(f"inRecvErr {i} is {inRecvErr[i]}")
    if inCRCErr[i] != 0:
        print(f"inCRCErr {i} is {inCRCErr[i]}")
    if inTransmitErr[i] != 0:
        print(f"inTransmitErr {i} is {inTransmitErr[i]}")
    if inLinkErr[i] != 0:
        print(f"inLinkErr {i} is {inLinkErr[i]}")
    if outRecvErr[i] != 0:
        print(f"outRecvErr {i} is {outRecvErr[i]}")
    if outCRCErr[i] != 0:
        print(f"outCRCErr {i} is {outCRCErr[i]}")
    if outTransmitErr[i] != 0:
        print(f"outTransmitErr {i} is {outTransmitErr[i]}")
    if outLinkErr[i] != 0:
        print(f"outLinkErr {i} is {outLinkErr[i]}")
print("others has no err!")
for i in range(8):
    robot.SlavePortErrCounterClear(i)
robot.CloseRPC()
```

2.4.6.63 Set the feed-forward coefficient of each axis speed

New in version python: SDK-v2.1.7

Prototype	`SetVelFeedForwardRatio(radio)`
Description	Set the feed-forward coefficient of each axis speed
Mandatory parameters	• `radio`Each axis velocity feedforward coefficient
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.6.64 The velocity feedforward coefficients of each axis are obtained

New in version python: SDK-v2.1.7

Prototype	`GetVelFeedForwardRatio()`
Description	The velocity feedforward coefficients of each axis are obtained
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `radio`Each axis velocity feedforward coefficient

2.4.6.65 Robot velocity feedforward coefficient code example

```
from fairino import Robot
import time
import threading
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
setRadio = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
robot.SetVelFeedForwardRatio(setRadio)
getRadio = [0.0] * 6
rtn,getRadio = robot.GetVelFeedForwardRatio()
print(f"{getRadio[0]},{getRadio[1]},{getRadio[2]},{getRadio[3]},{getRadio[4]},{getRadio[5]}")
robot.CloseRPC()
```

2.4.6.66 Photoelectric Sensor TCP Calibration - Compute Tool RPY

Prototype	`TCPComputeRPY(Btool, Etool, sensor, radius, dz)`
Description	Photoelectric Sensor TCP Calibration - Compute Tool RPY
Required Parameters	• `Btool`: Robot Cartesian position • `Etool`: Current tool coordinate values • `sensor`: Current sensor coordinate values (not yet available) • `radius`: Circular motion radius in mm (not yet available) • `dz`: Movement distance along the negative Z-axis of the base coordinate system; when dz = 10000, the function directly returns tool RPY
Default Parameters	None
Return Value	• Error code. Success - 0, Failure - errcode

2.4.6.67 Photoelectric Sensor TCP Calibration - Compute Tool XYZ

Prototype	`TCPComputeXYZ(select, originDirection, pos1, pos2, pos3, pos4)`
Description	Photoelectric Sensor TCP Calibration - Compute Tool XYZ
Required Parameters	• `select`: 0-Compute tool TCP; 1-Compute sensor origin; 2-Compute sensor orientation; 3-Directly return tool TCP; 4-Record current workpiece coordinate system and tool coordinate system • `originDirection`: 0-X direction; 1-Y direction; 2-Z direction • `pos1`: Robot Cartesian position 1 • `pos2`: Robot Cartesian position 2 • `pos3`: Robot Cartesian position 3 • `pos4`: Robot Cartesian position 4
Default Parameters	None
Return Value	• Error code. Success - 0, Failure - errcode • Return value (returned on successful call) TCP Tool XYZ values

2.4.6.68 Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position

Prototype	`TCPRecordFlangePosStart()`
Description	Photoelectric Sensor TCP Calibration - Start Recording Flange Center Position
Required Parameters	None
Default Parameters	None
Return Value	• Error code. Success - 0, Failure - errcode

2.4.6.69 Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position

Prototype	`TCPRecordFlangePosEnd()`
Description	Photoelectric Sensor TCP Calibration - Stop Recording Flange Center Position
Required Parameters	None
Default Parameters	None
Return Value	• Error code. Success - 0, Failure - errcode

2.4.6.70 Photoelectric Sensor TCP Calibration - Get Tool Center Point Position

Prototype	`TCPGetRecordFlangePos()`
Description	Photoelectric Sensor TCP Calibration - Get Tool Center Point Position
Required Parameters	None
Default Parameters	None
Return Value	• Error code. Success - 0, Failure - errcode • Return value (returned on successful call) TCP Tool center point position (x, y, z)

2.4.6.71 Photoelectric Sensor TCP Calibration

Prototype	`PhotoelectricSensorTCPCalibration(luaPath, offsetX)`
Description	Photoelectric Sensor TCP Calibration
Required Parameters	• `luaPath`: Automatic calibration Lua program path: For QX version robots - '/fruser/FR_CalibrateTheToolTcp.lua'; For LA version robots - '/usr/local/etc/controller/lua/FR_CalibrateTheToolTcp.lua' • `offsetX`: Teaching point offset (x, y, z) in mm
Default Parameters	None
Return Value	• Error code. Success - 0, Failure - errcode • Return value (returned on successful call) TCP Tool XYZ values

2.4.6.72 Photoelectric Sensor TCP Calibration Code Example

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
offset = [10.0, 10.0, 3.0]
TCP = [0.0] * 6
rtn, TCP = robot.PhotoelectricSensorTCPCalibration("/fruser/FR_CalibrateTheToolTcp.lua",␣
↪offset)
print(f"PhotoelectricSensorTCPCalibration rtn is {rtn},{TCP[0]},{TCP[1]},{TCP[2]},
↪{TCP[3]},{TCP[4]},{TCP[5]}")
robot.CloseRPC()
return 0
```

2.4.7 Security Settings

2.4.7.1 Setting the collision level

Prototype	`SetAnticollision (mode,level,config)`
Description	Setting the collision level
Mandatory parameters	• `mode`: 0 - level, 1 - percentage; • `level=[j1,j2,j3,j4,j5,j6]`: collision threshold; • `config`: 0 - do not update configuration file, 1 - update configuration file
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.7.2 Setting the post-collision strategy

Prototype	`SetCollisionStrategy(strategy,safeTime,safeDistance,safeVel, safetyMargin)`
Description	Set post-crash strategy
Mandatory parameters	• `strategy`: 0 - report error and pause, 1 - keep running, 2 - error stop, 3 - heavy moment mode, 4 - shock response mode, 5 - impact rebound mode
Default parameters	• `safeTime`: safe stop time [1000-2000] ms, default: 1000 • `safeDistance`: safe stopping distance [1-150] mm, default: 100 • `safeVel`: safe stopping speed[50-250]mm/s, default: 250 • `safetyMargin[6]`: safety margin [1-10], default: [10,10,10,10,10,10,10]
Return Value	Error Code Success-0 Failure- errcode

2.4.7.3 The Custom collision detection threshold function starts to set the collision detection thresholds of the joint end and TCP end

New in version python: SDK-v2.1.0

Prototype	`CustomCollisionDetectionStart(flag, jointDetectionThreshould, tcpDetectionThreshould, block)`
Description	The Custom collision detection threshold function starts to set the collision detection thresholds of the joint end and TCP end
Mandatory parameters	• `flag`: 1- Only joint detection is turned on; 2- Only TCP detection is enabled. 3- Joint and TCP detection are enabled simultaneously • `jointDetectionThreshould`: Joint collision detection threshold j1-j6 • `tcpDetectionThreshould`: indicates TCP collision detection threshold, xyz-abc • `block`: 0- non-blocking; 1- block
Default parameters	NULL
Return Value	• errcode Success-0 Failure- errcode

2.4.7.4 The custom collision detection threshold function is disabled

New in version python: SDK-v2.1.0

Prototype	`CustomCollisionDetectionEnd()`
Description	The custom collision detection threshold function is disabled
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errcode Success-0 Failure- errcode

2.4.7.5 Robot collision level setting code example

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
mode = 0
config = 1
level1 = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
level2 = [50.0, 20.0, 30.0, 40.0, 50.0, 60.0]
rtn = robot.SetAnticollision(mode, level1, config)
print(f"SetAnticollision mode 0 rtn is {rtn}")
mode = 1
rtn = robot.SetAnticollision(mode, level2, config)
print(f"SetAnticollision mode 1 rtn is {rtn}")
p1Joint = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
p2Joint = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
p1Desc = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
p2Desc = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
exaxisPos = [0.0, 0.0, 0.0, 0.0]
```

(continues on next page)

(continued from previous page)

```
offdese = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
robot.MoveL(desc_pos=p2Desc, tool=0, user=0, vel=100, blendR=2)
robot.ResetAllError()
safety = [5, 5, 5, 5, 5, 5]
rtn = robot.SetCollisionStrategy(3, 1000, 150, 250, safety)
print(f"SetCollisionStrategy rtn is {rtn}")
jointDetectionThreshould = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
tcpDetectionThreshould = [60, 60, 60, 60, 60, 60]
rtn = robot.CustomCollisionDetectionStart(3, jointDetectionThreshould,␣
↪tcpDetectionThreshould, 0)
print(f"CustomCollisionDetectionStart rtn is {rtn}")
robot.MoveL(desc_pos=p1Desc, tool=0, user=0, vel=100)
robot.MoveL(desc_pos=p2Desc, tool=0, user=0, vel=100)
rtn = robot.CustomCollisionDetectionEnd()
print(f"CustomCollisionDetectionEnd rtn is {rtn}")
robot.CloseRPC()
```

2.4.7.6 Setting the positive limit

Prototype	`SetLimitPositive(p_limit)`
Description	Setting positive limits
Mandatory parameters	• `p_limit=[j1,j2,j3,j4,j5,j6]`: six joint positions
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.7.7 Setting the negative limit

Prototype	`SetLimitNegative(n_limit)`
Description	Setting negative limits
Mandatory parameters	• `n_limit=[j1,j2,j3,j4,j5,j6]`: six joint positions
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.7.8 Obtaining the soft limiting angle of a joint

Prototype	`GetJointSoftLimitDeg(flag=1)`
Description	Acquisition of joint soft limiting angle
Mandatory parameters	NULL
Default parameters	`flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `[j1min,j1max,j2min,j2max,j3min,j3max, j4min,j4max,j5min, j5max, j6min,j6max]`: Axis 1 to Axis 6, joints with negative and positive limits, in [mm]

2.4.7.9 Robot limit setting code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
plimit = [170.0, 80.0, 150.0, 80.0, 170.0, 160.0]
robot.SetLimitPositive(plimit)
nlimit = [-170.0, -260.0, -150.0, -260.0, -170.0, -160.0]
robot.SetLimitNegative(nlimit)
error,neg_deg = robot.GetJointSoftLimitDeg(0)
print(f"pos limit deg: {neg_deg[1]}, {neg_deg[3]}, {neg_deg[5]}, {neg_deg[7]}, {neg_deg[9]}, {neg_deg[11]}")
print(f"neg limit deg: {neg_deg[0]}, {neg_deg[2]}, {neg_deg[4]}, {neg_deg[6]}, {neg_deg[8]}, {neg_deg[10]}")
robot.CloseRPC()
```

2.4.7.10 Setting up a robot collision detection method

New in version python: SDK-v2.1.2

prototype	`SetCollisionDetectionMethod(method, thresholdMode)`
Description	Sets the robot collision detection method.
Mandatory parameters	• `method`: collision detection method: 0 - current mode; 1 - dual encoder; 2 - current and dual encoder on at the same time • `thresholdMode`: Collision level threshold method 0-Collision level fixed threshold mode 1- Customize collision detection thresholds
Default parameters	NULL
Return Value	• errcode Success-0 Failure- errcode

2.4.7.11 Set static undercollision detection to start off

New in version python: SDK-v2.0.5

Prototype	`SetStaticCollisionOnOff(status)`
Description	Set static undercollision detection to start off
Mandatory parameters	• `status`: 0 - off; 1 - on
Default parameters	NULL
Return Value	• errcode Success-0 Failure- errcode

2.4.7.12 Set up the robot collision detection method code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
rtn = robot.SetCollisionDetectionMethod(0,0)
rtn = robot.SetStaticCollisionOnOff(1)
print(f"SetStaticCollisionOnOff On rtn is {rtn}")
time.sleep(5)
rtn = robot.SetStaticCollisionOnOff(0)
print(f"SetStaticCollisionOnOff Off rtn is {rtn}")
robot.CloseRPC()
```

2.4.7.13 Joint torque and power detection

New in version python: SDK-v2.0.5

Prototype	`SetPowerLimit(status, power)`
Description	Joint torque and power detection
Mandatory parameters	• `status`: 0 - off; 1 - on • `power`Set the maximum power (W)
Default parameters	NULL
Return Value	• errcode Success-0 Failure- errcode

2.4.7.14 Sample code for joint torque power detection

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
robot.DragTeachSwitch(1)
robot.SetPowerLimit(1, 200)
error,torques = robot.GetJointTorques(1)
count = 100
robot.ServoJTStart()
while count > 0:
    error = robot.ServoJT(torques, 0.001)
    count -= 1
    time.sleep(0.001)  # 1ms delay
error = robot.ServoJTEnd()
robot.DragTeachSwitch(0)
robot.CloseRPC()
```

2.4.8 Status query

2.4.8.1 Get the current joint position (angle).

prototype	`GetActualJointPosDegree(flag=1)`
Description	Get the current position (angle) of the joint.
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking, default 1
Return Value	• errorcode Success-0 Failure- errcode • `joint_pos=[j1,j2,j3,j4,j5,j6]`: current joint position (angle)

2.4.8.2 Get the current joint position in radians.

Prototype	`GetActualJointPosRadian(flag=1)`
Description	Get the current position (in radians) of the joint.
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `joint_pos=[j1,j2,j3,j4,j5,j6]`: current joint position (in radians)

2.4.8.3 Get joint feedback speed -deg/s

Prototype	`GetActualJointSpeedsDegree(flag=1)`
Description	Get joint feedback speed -deg/s
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `speed=[j1,j2,j3,j4,j5,j6]`: joint feedback speed -deg/s

2.4.8.4 Obtain joint feedback acceleration-deg/s^2

Prototype	`GetActualJointAccDegree(flag=1)`
Description	Obtain joint feedback acceleration-deg/s^2
Mandatory parameters	NULL
Default parameters	• `flag` 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `acc=[j1,j2,j3,j4,j5,j6]`Joint feedback acceleration-deg/s^2

2.4.8.5 Get TCP command synthesis speed

Prototype	`GetTargetTCPCompositeSpeed(flag=1)`
Description	Get TCP command synthesis speed
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `[tcp_speed,ori_speed]`: tcp_speed - linear closing speed ori_speed - attitude closing speed

2.4.8.6 Getting TCP Feedback Hopping Speed

Prototype	`GetActualTCPCompositeSpeed(flag=1)`
Description	Get TCP feedback closing speed
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `[tcp_speed,ori_speed]`: tcp_speed - linear closing speed ori_speed - attitude closing speed

2.4.8.7 Get TCP command speed

Prototype	`GetTargetTCPSpeed(flag=1)`
Description	Get TCP command speed
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `speed=[x,y,z,rx,ry,rz]`: TCP command speed, mm/s

2.4.8.8 Getting TCP feedback speed

Prototype	`GetActualTCPSpeed(flag=1)`
Description	Get TCP feedback on speed
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `speed=[x,y,z,rx,ry,rz]`: TCP feedback speed

2.4.8.9 Get current tool position

Prototype	`GetActualTCPPose(flag=1)`
Description	Get current tool position
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `tcp_pose=[x,y,z,rx,ry,rz]`: current tool pose

2.4.8.10 Get the current tool coordinate system number

prototype	`GetActualTCPNum(flag=1)`
Description	Get the current tool coordinate system number
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `tool_id`: tool coordinate system number

2.4.8.11 Get the current workpiece coordinate system number

prototype	`GetActualWObjNum(flag=1)`
Description	Get the current workpiece coordinate system number
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `wobj_id`: the workpiece coordinate system number

2.4.8.12 Get the current end flange position

Prototype	`GetActualToolFlangePose(flag=1)`
Description	Get current end flange position
Mandatory parameters	NULL
Default parameters	• `flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `flange_pose=[x,y,z,rx,ry,rz]`: current end flange pose

2.4.8.13 Get current joint torque

Prototype	`GetJointTorques(flag=1)`
Description	Get the current joint torque
Mandatory parameters	NULL
Default parameters	`flag`: 0-blocking, 1-non-blocking Default 1
Return Value	• errorcode Success-0 Failure- errcode • `torques=[j1,j2,j3,j4,j5,j6]`: joint torques.

2.4.8.14 Get system time

Prototype	`GetSystemClock()`
Description	Get system time
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `t_ms`: system time in [ms]

2.4.8.15 Queries whether robot motion is complete

Prototype	`GetRobotMotionDone()`
Description	Query if robot movement is complete
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `state`: state of robot motion, 0 - unfinished, 1 - finished

2.4.8.16 Query the cache length of the robot motion queue

Prototype	`GetMotionQueueLength()`
Description	Query the cache length of the robot motion queue
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `len`Cache length

2.4.8.17 Obtain the emergency stop status of the robot

Prototype	`GetRobotEmergencyStopState()`
Description	Obtain the emergency stop status of the robot
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `state`Emergency stop status: 0- non-emergency stop, 1- emergency stop

2.4.8.18 Obtain the communication status between the SDK and the robot

Prototype	`GetSDKComState()`
Description	Obtain the communication status between the SDK and the robot
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `state`Communication status: 0- Normal communication, 1- Abnormal communication

2.4.8.19 Obtain the safety stop signal

Prototype	`GetSafetyStopState()`
Description	Obtain the safety stop signal
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `[si0_state,si1_state]`si0_state safety stop signal SI0, 0- invalid, 1- valid si1_state safety stop signal SI1, 0- invalid, 1- valid

2.4.8.20 Obtain the current temperature of the joint drive(℃)

Prototype	`GetJointDriverTemperature()`
Description	Obtain the current temperature of the joint drive(℃)
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `data=[t1,t2,t3,t4,t5,t6]`The current temperatures of each joint

2.4.8.21 Obtain the current torque of the joint drive(Nm)

Prototype	`GetJointDriverTorque()`
Description	Obtain the current torque of the joint drive(Nm)
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `data=[j1,j2,j3,j4,j5,j6]`Joint torque [fx,fy,fz,tx,ty,tz]

2.4.8.22 Obtain the status of the robot

Prototype	`GetRobotRealTimeState()`
Description	Obtain the status of the robot
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `robot_state_pkg`Robot state structure

2.4.8.23 Robot status query code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
error,[yangle, zangle] = robot.GetRobotInstallAngle()
print(f"yangle:{yangle},zangle:{zangle}")
error,j_deg = robot.GetActualJointPosDegree(0)
print(f"joint pos deg:{j_deg[0]},{j_deg[1]},{j_deg[2]},{j_deg[3]},{j_deg[4]},{j_deg[5]}")
error,jointSpeed = robot.GetActualJointSpeedsDegree(0)
print(f"joint speeds deg:{jointSpeed[0]},{jointSpeed[1]},{jointSpeed[2]},{jointSpeed[3]},
↪{jointSpeed[4]},{jointSpeed[5]}")
error,jointAcc = robot.GetActualJointAccDegree(0)
print(f"joint acc deg:{jointAcc[0]},{jointAcc[1]},{jointAcc[2]},{jointAcc[3]},
```

(continues on next page)

(continued from previous page)

```
↪{jointAcc[4]},{jointAcc[5]}")
error,[tcp_speed, ori_speed] = robot.GetTargetTCPCompositeSpeed(0)
print(f"GetTargetTCPCompositeSpeed tcp {tcp_speed}; ori {ori_speed}")
error,[tcp_speed, ori_speed] = robot.GetActualTCPCompositeSpeed(0)
print(f"GetActualTCPCompositeSpeed tcp {tcp_speed}; ori {ori_speed}")
error,targetSpeed = robot.GetTargetTCPSpeed(0)
print(f"GetTargetTCPSpeed {targetSpeed[0]},{targetSpeed[1]},{targetSpeed[2]},
↪{targetSpeed[3]},{targetSpeed[4]},{targetSpeed[5]}")
error,actualSpeed = robot.GetActualTCPSpeed(0)
print(f"GetActualTCPSpeed {actualSpeed[0]},{actualSpeed[1]},{actualSpeed[2]},
↪{actualSpeed[3]},{actualSpeed[4]},{actualSpeed[5]}")
error,tcp = robot.GetActualTCPPose(0)
print(f"tcp pose:{tcp[0]},{tcp[1]},{tcp[2]},{tcp[3]},{tcp[4]},{tcp[5]}")
error,flange = robot.GetActualToolFlangePose(0)
print(f"flange pose:{flange[0]},{flange[1]},{flange[2]},{flange[3]},{flange[4]},
↪{flange[5]}")
error,id = robot.GetActualTCPNum(0)
print(f"tcp num:{id}")
error,id = robot.GetActualWObjNum(0)
print(f"wobj num:{id}")
error,jtorque = robot.GetJointTorques(0)
print(f"torques:{jtorque[0]},{jtorque[1]},{jtorque[2]},{jtorque[3]},{jtorque[4]},
↪{jtorque[5]}")
error,t_ms = robot.GetSystemClock()
print(f"system clock:{t_ms}")
error,config = robot.GetRobotCurJointsConfig()
print(f"joint config:{config}")
error,motionDone = robot.GetRobotMotionDone()
print(f"GetRobotMotionDone:{motionDone}")
error,len = robot.GetMotionQueueLength()
print(f"GetMotionQueueLength:{len}")
error,emergState = robot.GetRobotEmergencyStopState()
print(f"GetRobotEmergencyStopState:{emergState}")
error,comstate = robot.GetSDKComState()
print(f"GetSDKComState:{comstate}")
error,[si0_state, si1_state] = robot.GetSafetyStopState()
print(f"GetSafetyStopState:{si0_state} {si1_state}")
error,temp = robot.GetJointDriverTemperature()
print(f"Temperature:{temp[0]},{temp[1]},{temp[2]},{temp[3]},{temp[4]},{temp[5]}")
error,torque = robot.GetJointDriverTorque()
print(f"torque:{torque[0]},{torque[1]},{torque[2]},{torque[3]},{torque[4]},{torque[5]}")
error,pkg = robot.GetRobotRealTimeState()
robot.CloseRPC()
```

2.4.8.24 Inverse kinematics solution

prototype	`GetInverseKin(type,desc_pos,config=-1)`
Description	Inverse kinematics, Cartesian position solving for joint positions
Mandatory parameters	• `type`: 0-absolute position (base coordinate system), 1-relative position (base coordinate system), 2-relative position (tool coordinate system) • `desc_pose`:[x,y,z,rx,ry,rz], tool position in [mm][°]
Default parameters	• `config`: joint configuration, [-1] - solved with reference to current joint position, [0~7] - solved based on joint configuration Default -1
Return Value	• errorcode Success-0 Failure- errcode • `joint_pos=[j1,j2,j3,j4,j5,j6]`: inverse kinematics solution, Cartesian positional solution for joint positions

2.4.8.25 Inverse Kinematics Solution - Specifying Reference Positions

prototype	`GetInverseKinRef(type,desc_pos,joint_pos_ref)`
Description	Inverse kinematics, tool position solving for joint positions, solving with reference to specified joint positions
Mandatory parameters	• `type`: 0-absolute position (base coordinate system), 1-relative position (base coordinate system), 2-relative position (tool coordinate system) • `desc_pos`: [x,y,z,rx,ry,rz] tool position in [mm][°] • `joint_pos_ref`: [j1,j2,j3,j4,j5,j6], joint reference position in [°]
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `joint_pos=[j1,j2,j3,j4,j5,j6]`: inverse kinematics solution, tool position solving for joint position

2.4.8.26 Inverse Kinematics Solution, Cartesian Space Includes Extended Axis Position

Prototype	`GetInverseKinExaxis(type, desc_pos, exaxis, tool, workPiece)`
Description	Inverse kinematics solution, Cartesian space includes extended axis position
Required Parameters	• `type`: 0-Absolute pose (base coordinate system), 1-Incremental pose (base coordinate system), 2-Incremental pose (tool coordinate system) • `desc_pos`: Cartesian pose • `exaxis`: Extended axis position • `tool`: Tool number • `workPiece`: Workpiece number
Default Parameters	None
Return Value	• Error code Success-0 Failure- errcode • `joint_pos`: Joint position

2.4.8.27 Inverse Kinematics Solution Including Extended Axis Position Code Example

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
desc = [99.957, -0.002, 29.994, -176.569, -6.757, -167.462]
exaxis = [100.0, 0.0, 0.0, 0.0]
jointPos = [0.0] * 6
offsetPos = [0.0] * 6
rtn,pkg = robot.GetRobotRealTimeState()
toolnum = pkg.tool
workPcsNum = pkg.user
rtn, jointPos = robot.GetInverseKinExaxis(0, desc, exaxis, toolnum, workPcsNum)
print(f"GetInverseKinExaxis joint is {jointPos[0]},{jointPos[1]},{jointPos[2]},
↪{jointPos[3]},{jointPos[4]},{jointPos[5]}")
robot.ExtAxisMove(exaxis, 100, -1)
robot.MoveJ(joint_pos=jointPos, desc_pos=desc, tool=toolnum, user=workPcsNum, vel=100.0,␣
↪acc=100.0, ovl=100.0, exaxis_pos=exaxis, blendT=-1, offset_flag=0, offset_
↪pos=offsetPos)
robot.CloseRPC()
return 0
```

2.4.8.28 Inverse kinematics solving-whether there is a solution

prototype	`GetInverseKinHasSolution(type,desc_pos,joint_pos_ref)`
Description	Inverse kinematics, tool position solving for joint position Is there a solution
Mandatory parameters	• `type`: 0-absolute position (base coordinate system), 1-relative position (base coordinate system), 2-relative position (tool coordinate system) • `desc_pos`: [x,y,z,rx,ry,rz] tool position in [mm][°] • `joint_pos_ref`: [j1,j2,j3,j4,j5,j6], joint reference position in [°]
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `result`: `True`-with solution, `False`-without solution

2.4.8.29 Positive kinematics solving

Prototype	`GetForwardKin(joint_pos)`
Description	Positive Kinematics, Joint Position Solving Tool Posture
Mandatory parameters	• `joint_pos`:[j1,j2,j3,j4,j5,j6]: joint position in [°]
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `desc_pos=[x,y,z,rx,ry,rz]`: positive kinematics solution, joint position solver tool position

2.4.8.30 Example code for robot forward and inverse kinematics calculation

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
j1 = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]
desc_pos1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
error, inverseRtn = robot.GetInverseKin(0, desc_pos=desc_pos1, config=-1)
print(f"dcs1 GetInverseKin rtn is {inverseRtn[0]}, {inverseRtn[1]}, {inverseRtn[2]}, "
      f"{inverseRtn[3]}, {inverseRtn[4]}, {inverseRtn[5]}")
error, inverseRtn = robot.GetInverseKinRef(0, desc_pos=desc_pos1, joint_pos_ref=j1)
print(f"dcs1 GetInverseKinRef rtn is {inverseRtn[0]}, {inverseRtn[1]}, {inverseRtn[2]}, "
      f"{inverseRtn[3]}, {inverseRtn[4]}, {inverseRtn[5]}")
error, hasResult = robot.GetInverseKinHasSolution(0, desc_pos=desc_pos1, joint_pos_
↪ref=j1)
print(f"dcs1 GetInverseKinRef result {hasResult}")
error, forwordResult = robot.GetForwardKin(j1)
print(f"jpos1 forwordResult rtn is {forwordResult[0]}, {forwordResult[1]},
↪{forwordResult[2]}, "
```

(continues on next page)

(continued from previous page)

```
16      f"{forwordResult[3]}, {forwordResult[4]}, {forwordResult[5]}")
17  robot.CloseRPC()
```

2.4.8.31 Query Robot Teaching Management Points Data

Prototype	`GetRobotTeachingPoint(name)`
Description	Query Robot Demonstration and Management point data
Mandatory parameters	`name`: name of point
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `[x,y,z,rx,ry,rz,j1,j2,j3,j4,j5,j6,TOOL,WOBJ,SPEED,ACC,E1,E2,E3,E4]`: point data

2.4.8.32 Get DH compensation parameters

New in version python: SDK-v2.0.1

Prototype	`GetDHCompensation()`
Description	Get DH compensation parameters
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `dhCompensation=[cmpstD1,cmpstA2,cmpstA3,cmpstD4,cmpstD5,cmpstD6]`: Robot DH Parameter Compensation Values (mm)

2.4.8.33 Obtain the SN code of the control box

New in version python: SDK-v2.1.1

Prototype	`GetRobotSN()`
Description	Obtain the SN code of the control box
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `SNCode`: SN of the control box

2.4.8.34 Query robot teaching management point data code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
name = "P1"
rtn, data = robot.GetRobotTeachingPoint(name)
print(f"{rtn} name is: {name}")
for i in range(20):
    print(f"data is: {data[i]}")
rtn,que_len = robot.GetMotionQueueLength()
print(f"GetMotionQueueLength rtn is: {rtn}, queue length is: {que_len}")
retval,dh = robot.GetDHCompensation()
print(f"retval is: {retval}")
print(f"dh is: {dh[0]} {dh[1]} {dh[2]} {dh[3]} {dh[4]} {dh[5]}")
error,sn = robot.GetRobotSN()
print(f"robot SN is {sn[0]}")
robot.CloseRPC()
```

2.4.8.35 Get the tool coordinate system according to the number

New in version python: SDK-v2.1.6

Prototype	`GetToolCoordWithID(id)`
Description	Get the tool coordinate system according to the number
Mandatory parameters	• `id`: Tool coordinate system number
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `coord`: Coordinate system value

2.4.8.36 The workpiece coordinate system is obtained according to the No

New in version python: SDK-v2.1.6

Prototype	`GetWObjCoordWithID(id)`
Description	The workpiece coordinate system is obtained according to the No
Mandatory parameters	• `id`: Workpiece coordinate system number
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `coord`: Coordinate system value

2.4.8.37 The external tool coordinate system is obtained according to the number

New in version python: SDK-v2.1.6

Prototype	`GetExToolCoordWithID(id)`
Description	The external tool coordinate system is obtained according to the number
Mandatory parameters	• `id`External tool coordinate system No
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `coord`: Coordinate system value

2.4.8.38 The extended axis coordinate system is obtained according to the No

New in version python: SDK-v2.1.6

Prototype	`GetExAxisCoordWithID(id)`
Description	The extended axis coordinate system is obtained according to the No
Mandatory parameters	• `id`Extended axis coordinate system No
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `coord`: Coordinate system value

2.4.8.39 Get the load mass and centroid according to the number

New in version python: SDK-v2.1.6

Prototype	`GetTargetPayloadWithID(id)`
Description	Get the load mass and centroid according to the number
Mandatory parameters	• `id`Extended axis coordinate system No
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `weight`: Quality of load • `cog`: Centroid of the load

2.4.8.40 Gets the current tool coordinate system

New in version python: SDK-v2.1.6

Prototype	`GetCurToolCoord()`
Description	Gets the current tool coordinate system
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `coord`: Coordinate system value

2.4.8.41 Gets the current workpiece coordinate system

New in version python: SDK-v2.1.6

Prototype	`GetCurWObjCoord()`
Description	Gets the current workpiece coordinate system
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `coord`: Coordinate system value

2.4.8.42 Gets the current external tool coordinate system

New in version python: SDK-v2.1.6

Prototype	`GetCurExToolCoord()`
Description	Gets the current external tool coordinate system
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `coord`: Coordinate system value

2.4.8.43 Gets the current extended axis coordinate system

New in version python: SDK-v2.1.6

	GetCurExAxisCoord()
	Gets the current extended axis coordinate system
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • coord: Coordinate system value

2.4.8.44 Get robot coordinate system and load code sample

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
id = 1
toolCoord = [0.0] * 6
extoolCoord = [0.0] * 6
wobjCoord = [0.0] * 6
exAxisCoord = [0.0] * 6
for i in range(100):
    print(f"ID:{id}")
    coordSet0 = [0.0] * 6
    coordSet = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
    etcp = [10.0, 20.0, 30.0, 40.0, 50.0, 60.0]
    etool = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
    cog = [1.0, 2.0, 3.0]
    if i % 2 == 0:
        robot.SetToolCoord(id, coordSet, 0, 0, 1, 0)
        time.sleep(0.1)
        robot.SetWObjCoord(id, coordSet, 0)
        time.sleep(0.1)
        robot.ExtAxisActiveECoordSys(id, 1, coordSet, 1)
        time.sleep(0.1)
        rtn = robot.SetExToolCoord(id, etcp, etool)
        time.sleep(0.1)
        rtn = robot.SetLoadWeight(id, 1.5)
        time.sleep(0.1)
        rtn = robot.SetLoadCoord(cog[0],cog[1],cog[2],id)
        time.sleep(0.1)
    else:
        robot.SetToolCoord(id, coordSet0, 0, 0, 1, 0)
        time.sleep(0.1)
        robot.SetWObjCoord(id, coordSet0, 0)
        time.sleep(0.1)
        robot.ExtAxisActiveECoordSys(id, 1, coordSet0, 1)
        time.sleep(0.1)
        rtn = robot.SetExToolCoord(id, coordSet0, coordSet0)
        time.sleep(0.1)
        rtn = robot.SetLoadWeight(id, 0)
        time.sleep(0.1)
```

(continues on next page)

(continued from previous page)

```
        rtn = robot.SetLoadCoord(coordSet0[0],coordSet0[1],coordSet0[2] , id)
        time.sleep(0.1)
    rtn, toolCoord = robot.GetCurToolCoord()
    print(f"GetToolCoord {toolCoord[0]},{toolCoord[1]},{toolCoord[2]},{toolCoord[3]},{toolCoord[4]},{toolCoord[5]}")
    rtn, wobjCoord = robot.GetCurWObjCoord()
    print(f"GetWObjCoord {wobjCoord[0]},{wobjCoord[1]},{wobjCoord[2]},{wobjCoord[3]},{wobjCoord[4]},{wobjCoord[5]}")
    rtn, extoolCoord = robot.GetCurExToolCoord()
    print(f"GetExToolCoord {extoolCoord[0]},{extoolCoord[1]},{extoolCoord[2]},{extoolCoord[3]},{extoolCoord[4]},{extoolCoord[5]}")
    rtn, exAxisCoord = robot.GetCurExAxisCoord()
    print(f"GetExAxisCoord {exAxisCoord[0]},{exAxisCoord[1]},{exAxisCoord[2]},{exAxisCoord[3]},{exAxisCoord[4]},{exAxisCoord[5]}")
    weight = 0.0
    getCog = [0.0] * 3
    rtn, weight = robot.GetTargetPayload(0)
    rtn, getCog = robot.GetTargetPayloadCog(0)
    print(f"GetTargetPayload {weight},{getCog[0]},{getCog[1]},{getCog[2]}")

    rtn, toolCoord = robot.GetToolCoordWithID(id)
    print(f"GetToolCoordWithID {id},{toolCoord[0]},{toolCoord[1]},{toolCoord[2]},{toolCoord[3]},{toolCoord[4]},{toolCoord[5]}")
    rtn, wobjCoord = robot.GetWObjCoordWithID(id)
    print(f"GetWObjCoordWithID {id},{wobjCoord[0]},{wobjCoord[1]},{wobjCoord[2]},{wobjCoord[3]},{wobjCoord[4]},{wobjCoord[5]}")
    rtn, extoolCoord = robot.GetExToolCoordWithID(id)
    print(f"GetExToolCoordWithID {id},{extoolCoord[0]},{extoolCoord[1]},{extoolCoord[2]},{extoolCoord[3]},{extoolCoord[4]},{extoolCoord[5]}")
    rtn, exAxisCoord = robot.GetExAxisCoordWithID(id)
    print(f"GetExAxisCoordWithID {id},{exAxisCoord[0]},{exAxisCoord[1]},{exAxisCoord[2]},{exAxisCoord[3]},{exAxisCoord[4]},{exAxisCoord[5]}")
    weight = 0.0
    getCog = [0.0] * 3
    rtn, weight, getCog = robot.GetTargetPayloadWithID(id)
    print(f"GetTargetPayloadWithID {id},{weight},{getCog[0]},{getCog[1]},{getCog[2]}")
    time.sleep(0.5)
    print(f"times {i}")
```

2.4.9 Trajectory recurrence

2.4.9.1 Setting Track Recording Parameters

prototype	`SetTPDParam(name, period_ms, type=1,di_choose=0, do_choose=0)`
Description	Setting parameters for track logging
Mandatory parameters	• `name`: track name; • `period_ms`: sampling period, fixed value, 2ms or 4ms or 8ms;
Default parameters	• `type`: data type, 1-joint position; • `di_choose`: DI choose, bit0~bit7 corresponds to control box DI0~DI7, bit8~bit9 corresponds to end DI0~DI1, 0-no choose, 1-choose Default 0. • `do_choose`: DO choose, bit0~bit7 corresponds to control box DO0~DO7, bit8~bit9 corresponds to end DO0~DO1, 0-no choose, 1-choose Default 0
Return Value	Error Code Success-0 Failure- errcode

2.4.9.2 Start Track Recording

Prototype	`SetTPDStart(name, period_ms, type=1,di_choose=0, do_choose=0)`
Description	Start Track Record
Mandatory parameters	• `name`: track name; • `period_ms`: sampling period, fixed value, 2ms or 4ms or 8ms;
Default parameters	• `type`: number datatype, 1-joint position default 1; • `di_choose`: DI choose, bit0~bit7 corresponds to control box DI0~DI7, bit8~bit9 corresponds to end DI0~DI1, 0-no choose, 1-choose Default 0. • `do_choose`: DO choose, bit0~bit7 corresponds to control box DO0~DO7, bit8~bit9 corresponds to end DO0~DO1, 0-no choose, 1-choose Default 0
Return Value	Error Code Success-0 Failure- errcode

2.4.9.3 Stop Track Recording

prototype	`SetWebTPDStop()`
Description	Stop Track Record
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.4 Deleting track records

Prototype	`SetTPDDelete(name)`
Description	Delete Track Record
Mandatory parameters	• `name`: track name
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.5 code example

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
type = 1
name = "tpd2025"
period_ms = 4
di_choose = 0
do_choose = 0
robot.SetTPDParam(name, period_ms)
robot.Mode(1)
time.sleep(1)
robot.DragTeachSwitch(1)
robot.SetTPDStart(name, period_ms)
print("SetTPDStart")
time.sleep(10)
robot.SetWebTPDStop()
robot.DragTeachSwitch(0)
robot.CloseRPC()
```

2.4.9.6 Trajectory preloading

Prototype	`LoadTPD(name)`
Description	Trajectory Preloading
Mandatory parameters	• `name`: track name
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.7 Trajectory Reproduction

Prototype	`MoveTPD(name,blend,ovl)`
Description	Trajectory Reproduction
Mandatory parameters	• `name`: track name • `blend`: smooth or not, 0 - not smooth, 1 - smooth • `ovl`: velocity scaling factor, range [0 to 100]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.8 Get the starting position of the trajectory

Prototype	`GetTPDStartPose(name)`
Description	Get trajectory start position
Mandatory parameters	• `name`: track name
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `desc_pose=[x,y,z,rx,ry,rz]`: trajectory start position

2.4.9.9 Example of robot TPD trajectory recording code

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
type = 1
name = "tpd2025"
period_ms = 4
di_choose = 0
do_choose = 0
ovl = 100.0
blend = 0
rtn = robot.LoadTPD(name)
print(f"LoadTPD rtn is: {rtn}")
error,start_pose = robot.GetTPDStartPose(name)
print(f"start pose, xyz is: {start_pose[0]},{start_pose[1]},{start_pose[2]}. "
      f"rpy is: {start_pose[3]},{start_pose[4]},{start_pose[5]}")
robot.MoveCart(start_pose, 0, 0, 100, 100)
time.sleep(1)
rtn = robot.MoveTPD(name, blend, ovl)
print(f"MoveTPD rtn is: {rtn}")
time.sleep(5)
```

(continues on next page)

(continued from previous page)

```
robot.SetTPDDelete(name)
robot.CloseRPC()
```

2.4.9.10 Trajectory preprocessing

prototype	`LoadTrajectoryJ(name,ovl,opt=1)`
description	trajectory preprocessing
Mandatory parameters	• `name`: track name, e.g., /fruser/traj/trajHelix_aima_1.txt. • `ovl`: percentage of speed scaling, range [0~100];
Default parameter	• `opt`: 1-control point, default 1
Return Value	Error Code Success-0 Failure- errcode

2.4.9.11 Trajectory Reproduction

Prototype	`MoveTrajectoryJ()`
Description	Trajectory Reproduction
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.12 Getting the starting position of the trajectory

Prototype	`GetTrajectoryStartPose(name)`
Description	Get trajectory starting position
Mandatory parameters	`name`: track name
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `desc_pose=[x,y,z,rx,ry,rz]`: trajectory start position

2.4.9.13 Get track point number

prototype	`GetTrajectoryPointNum()`
Description	Get track point number
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `pnum`: track point number

2.4.9.14 Setting the speed of the trajectory in operation

Prototype	`SetTrajectoryJSpeed(ovl)`
Description	Sets the speed of the trajectory as it runs.
Mandatory parameter	`ovl`: speed scaling percentage, range [0~100]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.15 Setting the force and torque during trajectory operation

Prototype	`SetTrajectoryJForceTorque(ft)`
Description	Setting the force and torque in the trajectory run
Mandatory parameters	`ft=[fx,fy,fz,tx,ty,tz]`: units N and Nm
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.16 Setting the force along the x-direction in the trajectory run

Prototype	`SetTrajectoryJForceFx(fx)`
Description	Set the force along the x-direction in the trajectory run
Mandatory parameter	`ft`: force in x direction, in N
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.17 Setting the force along the y-direction in the trajectory run

Prototype	`SetTrajectoryJForceFx(fy)`
Description	Set the force along the y-direction in the trajectory run
Mandatory parameter	`fy`: force along y direction in N
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.18 Setting the force along the z-direction in a trajectory run

Prototype	`SetTrajectoryJForceFx(fz)`
Description	Set the force along the z-direction in the trajectory run
Mandatory parameter	`fz`: force along the z-direction, in N
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.19 Setting the torque around the x-axis in a trajectory run

Prototype	`SetTrajectoryJTorqueTx(tx)`
Description	Set the torque around the x-axis for the trajectory run
Mandatory parameter	`tx`: torque around x-axis in Nm
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.20 Setting the torque around the y-axis in trajectory operation

prototype	`SetTrajectoryJTorqueTx(ty)`
Description	Set the torque around the y-axis for the trajectory run
Mandatory parameter	`ty`: torque around y-axis in Nm
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.21 Setting the torque around the z-axis in trajectory operation

Prototype	`SetTrajectoryJTorqueTx(tz)`
Description	Sets the torque around the z-axis for the trajectory run
Mandatory parameter	`tz`: torque around z-axis in Nm
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.22 Upload trace J file

New in version python: SDK-v2.0.7

Prototype	`TrajectoryJUpLoad(filePath)`
Description	Upload trace J file
Mandatory parameter	• `filePath`:Full path name of the uploaded trajectory fileC://test/testJ.txt
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.23 Delete the track J file

New in version python: SDK-v2.0.7

Prototype	`TrajectoryJDelete(filePath)`
Description	Delete the track J file
Mandatory parameter	• `filePath`:Removes the full pathname of the trace file,C://test/testJ.txt
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.24 Robot trajectory J file reproduction code example

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
rtn = robot.TrajectoryJUpLoad("D://zUP/traj.txt")
print(f"Upload TrajectoryJ A {rtn}")
traj_file_name = "/fruser/traj/traj.txt"
rtn = robot.LoadTrajectoryJ(traj_file_name, 100, 1)
print(f"LoadTrajectoryJ {traj_file_name}, rtn is: {rtn}")
rtn,traj_start_pose = robot.GetTrajectoryStartPose(traj_file_name)
print(f"GetTrajectoryStartPose is: {rtn}")
print(f"desc_pos:{traj_start_pose[0]},{traj_start_pose[1]},{traj_start_pose[2]},"
      f"{traj_start_pose[3]},{traj_start_pose[4]},{traj_start_pose[5]}")
time.sleep(1)
robot.SetSpeed(50)
robot.MoveCart(traj_start_pose, 0, 0, 50, 100, 100)
rtn,traj_num = robot.GetTrajectoryPointNum()
print(f"GetTrajectoryStartPose rtn is: {rtn}, traj num is: {traj_num}")
rtn = robot.SetTrajectoryJSpeed(50.0)
print(f"SetTrajectoryJSpeed is: {rtn}")
traj_force = [0.0,0.0,0.0,0.0,0.0,0.0]
traj_force[0] = 10  # fx = 10
rtn = robot.SetTrajectoryJForceTorque(traj_force)
print(f"SetTrajectoryJForceTorque rtn is: {rtn}")
rtn = robot.SetTrajectoryJForceFx(10.0)
print(f"SetTrajectoryJForceFx rtn is: {rtn}")
rtn = robot.SetTrajectoryJForceFy(0.0)
print(f"SetTrajectoryJForceFy rtn is: {rtn}")
rtn = robot.SetTrajectoryJForceFz(0.0)
print(f"SetTrajectoryJForceFz rtn is: {rtn}")
rtn = robot.SetTrajectoryJTorqueTx(10.0)
print(f"SetTrajectoryJTorqueTx rtn is: {rtn}")
rtn = robot.SetTrajectoryJTorqueTy(10.0)
print(f"SetTrajectoryJTorqueTy rtn is: {rtn}")
rtn = robot.SetTrajectoryJTorqueTz(10.0)
print(f"SetTrajectoryJTorqueTz rtn is: {rtn}")
rtn = robot.MoveTrajectoryJ()
print(f"MoveTrajectoryJ rtn is: {rtn}")
robot.CloseRPC()
```

2.4.9.25 Trajectory preprocessing(Trajectory foresight)

New in version python: SDK-v2.1.4

Prototype	`LoadTrajectoryLA(name, mode, errorLim, type, precision, vamx, amax, jmax, flag)`
Description	Trajectory preprocessing(Trajectory foresight)
Mandatory parameter	• `name`:track file name • `mode`: sampling mode, 0- No sampling; 1- equal data interval sampling; 2- Equal error limit sampling • `errorLim`: error limitation, effective using straight line fitting • `type`: smooth mode, 0-Bessel smooth • `precision`: smoothing accuracy, effective when using Bessel smoothing • `vamx`: set maximum speed, mm/s • `amax`: set maximum acceleration, mm/s2 • `jmax`: maximum acceleration, mm/s3 • `flag`: Uniform speed forward opening switch 0- No opening; 1- Start
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.26 Trajectory reproduction(Trajectory foresight)

New in version python: SDK-v2.1.0

Prototype	`MoveTrajectoryLA()`
Description	Trajectory reproduction(Trajectory foresight)
Mandatory parameter	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.9.27 Code example for trajectory reproduction

```
from fairino import Robot
import time
# A connection is established with the robot controller and a robot object is returned if the connection is successful
robot = Robot.RPC('192.168.58.2')
rtn = robot.TrajectoryJUpLoad("D://zUP/traj.txt")
print(f"Upload TrajectoryJ A {rtn}")
traj_file_name = "/fruser/traj/traj.txt"
rtn = robot.LoadTrajectoryLA(traj_file_name, 1, 2, 0, 2, 50, 200, 1000, 0)
print(f"LoadTrajectoryLA {traj_file_name}, rtn is: {rtn}")
rtn, traj_start_pose = robot.GetTrajectoryStartPose(traj_file_name)
print(f"GetTrajectoryStartPose is: {rtn}")
print(f"desc_pos: {traj_start_pose[0]},{traj_start_pose[1]},{traj_start_pose[2]},{traj_start_pose[3]},{traj_start_pose[4]},{traj_start_pose[5]}")
time.sleep(1)
robot.SetSpeed(50)
```

(continues on next page)

(continued from previous page)

```
robot.MoveCart(traj_start_pose, 0, 0, 100, 100, 100)
rtn = robot.MoveTrajectoryLA()
print(f"MoveTrajectoryLA rtn is: {rtn}")
robot.CloseRPC()
```

2.4.10 WebAPP program use

2.4.10.1 Setting the default job program to load automatically on boot

Prototype	`LoadDefaultProgConfig(flag,program_name)`
Description	Sets the default job program to be automatically loaded on boot
Required parameters	• `flag`: 1-automatically load the default program on power-up, 0-don't automatically load the default program • `program_name`: the name of the job program and its path, e.g. "/fruser/movej.lua", where /fruser/ is the fixed path for QX, and /usr/local/etc/controller/lua/ is the fixed path for LA
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.10.2 Load the specified job program

Prototype	`ProgramLoad(program_name)`
Description	Load the specified job program.
Mandatory parameters	• `program_name`: name of the job program and path, e.g. "/fruser/movej.lua", where /fruser/ is the fixed path for QX, and /usr/local/etc/controller/lua/ is the fixed path for LA
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.10.3 Get the name of the loaded job program

Prototype	`GetLoadedProgram()`
Description	Get the name of the loaded job program
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `program_name`: the name of the loaded operating program.

2.4.10.4 Get the line number of the current robot job program

Prototype	`GetCurrentLine()`
Description	Get the execution line number of the current robot job program
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `line_num`: the line number of the current robot job program execution

2.4.10.5 Run the currently loaded job program

Prototype	`ProgramRun()`
Description	Run the currently loaded job program
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.10.6 Suspend the currently running job program

Prototype	`ProgramPause()`
Description	Suspend the currently running job program.
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.10.7 Resuming a currently suspended program

Prototype	`ProgramResume()`
Description	Resume the currently suspended job program
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.10.8 Terminate the currently running job program

Prototype	`ProgramStop()`
Description	Terminate the currently running job program.
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.10.9 Obtaining robot job program execution status

Prototype	`GetProgramState()`
Description	Get robot job program execution status
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `state`: state of execution of the robot's operating program, 1 - program stopped or no program running, 2 - program running, 3 - program suspended

2.4.10.10 Robot LUA program operation code example

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
program_name = "/fruser/test0610.lua"
loaded_name = ""
state = 0
line = 0
robot.Mode(0)
robot.LoadDefaultProgConfig(0, program_name)
robot.ProgramLoad(program_name)
robot.ProgramRun()
time.sleep(1)
robot.ProgramPause()
error,state = robot.GetProgramState()
print(f"program state:{state}")
error,line = robot.GetCurrentLine()
print(f"current line:{line}")
error,loaded_name = robot.GetLoadedProgram()
print(f"program name:{loaded_name}")
time.sleep(1)
robot.ProgramResume()
time.sleep(1)
robot.ProgramStop()
time.sleep(1)
robot.CloseRPC()
```

2.4.10.11 Download Lua files

New in version python: SDK-v2.0.2

Prototype	`LuaDownLoad(fileName, savePath)`
Description	Download Lua File
Mandatory parameter	• `fileName`: the name of the lua file to be downloaded, e.g. "test.lua"
• `savePath`: the local path to save the file	e.g. "D://Down/""
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.10.12 Deleting Lua files

New in version python: SDK-v2.0.2

Prototype	`LuaDelete(fileName)`
Description	Deleting Lua files.
Required Parameters	• `fileName`: the name of the lua file to be deleted, "test.lua"
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.10.13 Get the names of all current lua files

New in version python: SDK-v2.0.2

Prototype	`GetLuaList()`
Description	Get the names of all current lua files
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `lua_num`: number of lua files • `luaNames`: list of lua file names

2.4.10.14 Uploading Lua files

New in version python: SDK-v2.0.2

Prototype	`LuaUpload(filePath)`
Description	Uploading a Lua file
Mandatory parameter	• `filePath`: full path name of the uploaded file e.g. D://test/test.lua
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • errorStr(lua file exists error returned)

2.4.10.15 Robot LUA file upload and download code examples

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
rtn,lua_num,luaNames = robot.GetLuaList()
print(f"res is:{rtn}")
print(f"size is:{lua_num}")
for name in luaNames:
    print(name)
rtn = robot.LuaDownLoad("test0610.lua", "D://zDOWN/")
print(f"LuaDownLoad rtn is:{rtn}")
rtn = robot.LuaUpload("D://zDOWN/test0610.lua")
print(f"LuaUpload rtn is:{rtn}")
rtn = robot.LuaDelete("test0610.lua")
print(f"LuaDelete rtn is:{rtn}")
robot.CloseRPC()
```

2.4.11 Peripherals

2.4.11.1 Configuration of jaws

Prototype	`SetGripperConfig(company,device,softversion=0,bus=0)`
Description	Configuration jaws
Required Parameters	• `company`: gripper claw manufacturer, 1-Robotiq, 2-Huiling, 3-Tianji, 4-Dahuan, 5-Knowledge; • `device`: device number, Robotiq (0-2F-85 series), Huiling (0-NK series, 1-Z-EFG-100), Tianji (0-TEG-110), Dahuan (0-PGI-140), Zhixing (0-CTPM2F20)
Default parameters	• `softversion`: software version number, not used for now, default is 0; • `bus`: device mount end bus location, not used yet, default is 0;
Return Value	Error Code Success-0 Failure- errcode

2.4.11.2 Get Jaw Configuration

Prototype	`GetGripperConfig()`
Description	Get Jaw Configuration
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `[number,company,device,softversion]`: number, jaw number; company, jaw manufacturer, 1-Robotiq, 2-Huiling, 3-Tianji, 4-Dahuan, 5-Zhixing ;device, device number, Robotiq (0-2F-85 series), Huiling (0 -NK series,1-Z-EFG-100), Tianji(0-TEG-110), Dahuan(0-PGI-140), Zhixing(0-CTPM2F20) ;softvesion, software version number, not in use for the time being, default is 0.

2.4.11.3 Activate jaws

Prototype	`ActGripper(index,action)`
Description	Activate the jaws.
Mandatory parameter	• `index`: jaw number; • `action`: 0-reset, 1-activate
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.4 Control jaws

Prototype	`MoveGripper(index,pos,vel,force,maxtime,block,type,rotNum, rotVel,rotTorque)`
Description	Control jaws
Mandatory parameter	• `index`: jaw number; • `pos`: percentage of position, range [0~100]; • `vel`: percentage of speed, in the range [0 to 100]; `vel`: percentage of speed, in the range [0 to 100]. • `force`: percentage of torque, range [0 to 100]; • `maxtime`: maximum wait time, range [0~30000], unit [ms]; • `block`: 0-blocking, 1-non-blocking; • `type`: type of jaws, 0-parallel jaws; 1-rotary jaws; • `rotNum`:rotNum The number of rotations; • `rotVel`: Percentage of rotational velocity [0-100]; • `rotTorque`: percentage of rotational torque [0-100].
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.5 Getting the jaw movement status

Prototype	`GetGripperMotionDone()`
Description	Get the state of the jaw motion
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `[fault,status]`: status of gripper movement, fault:0-no error, 1-with error; status:0-movement not completed, 1-movement completed

2.4.11.6 Obtain the activated status of the gripper

New in version python: SDK-v2.1.2

Prototype	`GetGripperActivateStatus()`
Description	Obtain the activated status of the gripper
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `gripper_active`bit0 to bit15 correspond to the claw numbers 0 to 15. bit=0 indicates not activated, and bit=1 indicates activated

2.4.11.7 Obtain the position of the gripper

New in version python: SDK-v2.1.2

Prototype	`GetGripperCurPosition()`
Description	Obtain the position of the gripper
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `position`Position percentage, ranging from 0 to 100%

2.4.11.8 Obtain the gripper speed

New in version python: SDK-v2.1.2

Prototype	`GetGripperCurSpeed()`
Description	Obtain the gripper speed
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `speed`Speed percentage, range 0 to 100%

2.4.11.9 Obtain the gripper current

New in version python: SDK-v2.1.2

Prototype	`GetGripperCurCurrent()`
Description	Obtain the gripper current
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `current`Current percentage, range 0 to 100%

2.4.11.10 Obtain the gripper voltage

New in version python: SDK-v2.1.2

Prototype	`GetGripperVoltage()`
Description	Obtain the gripper voltage
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `voltage`Voltage, unit: 0.1V

2.4.11.11 Obtain the temperature of the gripper

New in version python: SDK-v2.1.2

Prototype	`GetGripperTemp()`
Description	Obtain the temperature of the gripper
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `temp`Temperature, unit: 0.1V

2.4.11.12 Calculate pre-capture point-visual

Prototype	`ComputePrePick(desc_pos, zlength, zangle)`
Description	Calculate pre-capture point-visual
Mandatory parameter	• `desc_pos`: clip grab point Cartesian position. • `zlength`: z-axis offset. • `zangle`: rotational offset around the z-axis
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `pre_pos`: pre-grip point Cartesian position

2.4.11.13 Calculate retreat point-visual

Prototype	`ComputePostPick(desc_pos, zlength, zangle)`
Description	Computing Retreat Point-Vision
Mandatory parameters	• `desc_pos`: grab point Cartesian position; • `zlength`: z-axis offset. • `zangle`: rotational offset around the z-axis
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `post_pos`: retreat point Cartesian poses

2.4.11.14 Robot claw operation code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
company = 4
device = 0
softversion = 0
bus = 2
index = 2
act = 0
max_time = 30000
block = 0
status = 0
fault = 0
active_status = 0
current_pos = 0
current = 0
voltage = 0
temp = 0
speed = 0
robot.SetGripperConfig(company, device, softversion, bus)
time.sleep(1)
error,[company, device, softversion, bus] = robot.GetGripperConfig()
print(f"gripper config:{company},{device},{softversion},{bus}")
robot.ActGripper(index, act)
time.sleep(1)
act = 1
robot.ActGripper(index, act)
time.sleep(1)
error = robot.MoveGripper(index, 90, 50, 50, max_time, block, 0, 0, 0, 0)
print(f"MoveGripper retval is:{error}")
time.sleep(1)
error = robot.MoveGripper(index, 30, 50, 0, max_time, block, 0, 0, 0, 0)
print(f"MoveGripper retval is:{error}")
error, [fault, status] = robot.GetGripperMotionDone()
print(f"motion status:{fault},{status}")
error, [fault, active_status] = robot.GetGripperActivateStatus()
print(f"gripper active fault is:{fault},status is:{active_status}")
error, [fault, current_pos] = robot.GetGripperCurPosition()
print(f"fault is:{fault},current position is:{current_pos}")
error, [fault, current] = robot.GetGripperCurCurrent()
print(f"fault is:{fault},current current is:{current}")
error, [fault, voltage] = robot.GetGripperVoltage()
print(f"fault is:{fault},current voltage is:{voltage}")
error, [fault, temp] = robot.GetGripperTemp()
print(f"fault is:{fault},current temperature is:{temp}")
error, [fault, speed] = robot.GetGripperCurSpeed()
print(f"fault is:{fault},current speed is:{speed}")
retval = 0
prepick_pose = [0.0]*6
postpick_pose = [0.0]*6
```

(continues on next page)

(continued from previous page)

```
p1Desc = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
p2Desc = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
retval, prepick_pose = robot.ComputePrePick(p1Desc, 10, 0)
print(f"ComputePrePick retval is:{retval}")
print(f"xyz is:{prepick_pose[0]},{prepick_pose[1]},{prepick_pose[2]};rpy is:{prepick_
↪pose[3]},{prepick_pose[4]},{prepick_pose[5]}")
retval, postpick_pose = robot.ComputePostPick(p2Desc, -10, 0)
print(f"ComputePostPick retval is:{retval}")
print(f"xyz is:{postpick_pose[0]},{postpick_pose[1]},{postpick_pose[2]};rpy is:{postpick_
↪pose[3]},{postpick_pose[4]},{postpick_pose[5]}")
robot.CloseRPC()
```

2.4.11.15 Get the number of rotation turns of the rotary gripper

New in version python: SDK-v2.0.7

Prototype	`GetGripperRotNum()`
Description	Get the number of rotation turns of the rotary gripper
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `num`Number of turns

2.4.11.16 Gets the percentage of rotation speed of the rotating gripper

New in version python: SDK-v2.0.7

Prototype	`GetGripperRotSpeed()`
Description	Gets the percentage of rotation speed of the rotating gripper
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `speed`Percent rotation speed

2.4.11.17 Obtains the percentage of rotating torque of the rotating gripper

New in version python: SDK-v2.0.7

Prototype	`GetGripperRotTorque()`
Description	Obtains the percentage of rotating torque of the rotating gripper
Mandatory parameter	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `fault`0-correct1-error • `torque`Percent torque of rotation

2.4.11.18 Get the rotary gripper status code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
fault = 0
rotNum = 0.0
rotSpeed = 0
rotTorque = 0
error,fault, rotNum = robot.GetGripperRotNum()
error,fault, rotSpeed = robot.GetGripperRotSpeed()
error,fault, rotTorque = robot.GetGripperRotTorque()
print(f"gripper rot num:{rotNum},gripper rotSpeed:{rotSpeed},gripper rotTorque:{rotTorque}")
robot.CloseRPC()
```

2.4.11.19 Drive belt start and stop

Prototype	`ConveyorStartEnd(status)`
Description	Drive belt start, stop
Mandatory parameters	• `status`: status of the drive belt, 1-start, 0-stop
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.20 Record IO detection points

Prototype	`ConveyorPointIORecord()`
Description	Record IO detection points
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.21 Record point A

Prototype	`ConveyorPointARecord()`
Description	Record point A.
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.22 Recording reference points

Prototype	`ConveyorRefPointRecord()`
Description	Record reference point
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.23 Record point B

Prototype	`ConveyorPointBRecord()`
Description	Record point B.
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.24 Conveyorized workpiece IO inspection

Prototype	`ConveyorIODetect(max_t)`
Description	Conveyorized workpiece IO detection
Mandatory parameter	• `max_t`: Maximum detection time in ms
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.25 Get the current position of the object

Prototype	`ConveyorGetTrackData(mode)`
Description	Get the current position of the object.
Mandatory Parameters	• `mode`: 1-Tracking Capture 2-Tracking Motion 3-TPD Tracking
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.26 Drive belt tracking started

Prototype	`ConveyorTrackStart(status)`
Description	Drive belt tracking started
Mandatory parameters	• `status`: status, 1-start, 0-stop
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.27 Belt tracking stop

Prototype	`ConveyorTrackEnd()`
Description	Drive belt tracking stop
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.28 Drive Belt Parameter Configuration

Prototype	`ConveyorSetParam(param, followType, startDis, endDis)`
Description	Configuration of drive belt parameters
Mandatory parameters	• **`param=[encChannel,resolution,lead,wpAxis,vision,speedRadio]`:** – `encChannel`: encoder channels 1-2 – `resolution`: encoder resolution Number of pulses per encoder revolution – `lead`: Mechanical transmission ratio Distance traveled by the conveyor belt in one revolution of the encoder – `wpAxis`: Workpiece coordinate system number Select the coordinate system number of the workpiece for the tracking motion function, and set the tracking gripping and TPD tracking to 0. – `vision`: whether or not to match vision 0 - no 1 - match, – `speedRadio`: speed ratio For conveyor tracking gripping speed range (1-100) Tracking motion, TPD tracking set to 1 • `followType`: Track movement type, 0- track movement; 1- Follow-up campaign
Default parameters	• `startDis`: Tracking grasp needs to be set, tracking start distance, -1: automatic calculation (automatic tracking after the workpiece reaches the robot), the unit is mm, the default value is 0 • `endDis`: The tracking capture needs to be set. The tracking termination distance, in mm, is 100 by default
Return Value	Error Code Success-0 Failure- errcode

2.4.11.29 Belt Grip Point Compensation

Prototype	`ConveyorCatchPointComp(cmp)`
Description	Drive belt grip point compensation
Mandatory parameters	• `cmp`: Compensate for position [x,y,z]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.30 linear motion

prototype	`ConveyorTrackMoveL(name,tool,wobj,vel=20,acc=100,ovl=100, blendR=-1.0)`
Description	Linear motion
Mandatory parameters	• `name`: cvrCatchPoint or cvrRaisePoint • `tool`: tool number • `wobj`: workpiece number
Default Parameters	• `vel`: speed default 20 • `acc`: acceleration default 100 • `ovl`: velocity scaling factor default 100 • `blendR`: [-1.0]-motion in place (blocking), [0~1000]-smoothing radius (non-blocking) in [mm] default -1.0
Return Value	Error Code Success-0 Failure- errcode

2.4.11.31 Conveyor communication input detection

New in version python: SDK-v2.1.1

Prototype	`ConveyorComDetect(timeout)`
Description	Conveyor communication input detection
Mandatory parameters	• `timeout`: Wait timeout duration ms
Default Parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.32 Conveyor communication input detection triggered

New in version python: SDK-v2.1.1

Prototype	`ConveyorComDetectTrigger()`
Description	Conveyor communication input detection triggered
Mandatory parameters	NULL
Default Parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.33 Robot conveyor operation code example

```python
from fairino import Robot
import time
import threading
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
retval = robot.ConveyorStartEnd(1)
print(f"ConveyorStartEnd retval is:{retval}")
retval = robot.ConveyorPointIORecord()
print(f"ConveyorPointIORecord retval is:{retval}")
retval = robot.ConveyorPointARecord()
print(f"ConveyorPointARecord retval is:{retval}")
retval = robot.ConveyorRefPointRecord()
print(f"ConveyorRefPointRecord retval is:{retval}")
retval = robot.ConveyorPointBRecord()
print(f"ConveyorPointBRecord retval is:{retval}")
retval = robot.ConveyorStartEnd(0)
print(f"ConveyorStartEnd retval is:{retval}")
param = [1.0, 10000.0, 200.0, 0.0, 0.0, 20.0]
retval = robot.ConveyorSetParam(param,0)
print(f"ConveyorSetParam retval is:{retval}")
cmp = [0.0, 0.0, 0.0]
retval = robot.ConveyorCatchPointComp(cmp)
print(f"ConveyorCatchPointComp retval is:{retval}")
index = 1
max_time = 30000
block = 0
retval = 0
p1Desc = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
p2Desc = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
retval = robot.MoveCart(p1Desc, 1, 0, 100.0)
print(f"MoveCart retval is:{retval}")
retval = robot.WaitMs(1)
print(f"WaitMs retval is:{retval}")
retval = robot.ConveyorIODetect(10000)
print(f"ConveyorIODetect retval is:{retval}")
retval = robot.ConveyorGetTrackData(1)
print(f"ConveyorGetTrackData retval is:{retval}")
retval = robot.ConveyorTrackStart(1)
print(f"ConveyorTrackStart retval is:{retval}")
retval = robot.ConveyorTrackMoveL("cvrCatchPoint", 1, 0, 100)
print(f"TrackMoveL retval is:{retval}")
retval = robot.MoveGripper(index, 51, 40, 30, max_time, block, 0, 0, 0, 0)
print(f"MoveGripper retval is:{retval}")
retval = robot.ConveyorTrackMoveL("cvrRaisePoint", 1, 0, 100)
print(f"TrackMoveL retval is:{retval}")
retval = robot.ConveyorTrackEnd()
print(f"ConveyorTrackEnd retval is:{retval}")
robot.MoveCart(p2Desc, 1, 0, 100.0, 100.0)
retval = robot.MoveGripper(index, 100, 40, 10, max_time, block, 0, 0, 0, 0)
print(f"MoveGripper retval is:{retval}")
```

(continues on next page)

(continued from previous page)

```
robot.CloseRPC()
```

2.4.11.34 End Sensor Configuration

New in version python: SDK-v2.0.5

Prototype	`AxleSensorConfig(idCompany, idDevice, idSoftware, idBus)`
Description	End Sensor Configuration
Mandatory parameters	• `idCompany`: manufacturer, 18-Junkong; 25-Huide • `idDevice`: type, 0-JUNKONG/RYR6T.V1.0 • `idSoftware`: software version, 0-J1.0/HuiDe1.0 (not yet available) • `idBus`: mount location, 1-end port 1; 2-end port 2… ..8-end port 8 (not open yet)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.35 Get End Sensor Configuration

New in version python: SDK-v2.0.5

Prototype	`AxleSensorConfigGet()`
Description	Get end sensor configuration
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `idCompany`: manufacturer, 18-Junkong; 25-Huide • `idDevice`: type, 0-JUNKONG/RYR6T.V1.0

2.4.11.36 End sensor activation

New in version python: SDK-v2.0.5

Prototype	`AxleSensorActivate(actFlag)`
Description	End sensor activation
Mandatory parameters	`actFlag`: 0-reset; 1-activate
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `coord`: coordinate system values [x,y,z,rx,ry,rz]

2.4.11.37 End Sensor Register Write

New in version python: SDK-v2.0.5

Prototype	`AxleSensorRegWrite(devAddr, regHAddr, regLAddr, regNum, data1, data2, isNoBlock)`
Description	End Sensor Register Write
Mandatory parameters	• `devAddr`: device address number 0-255 • `regHAddr`: register address high 8 bits • `regLAddr`: register address lower 8 bits • `regNum`: number of registers 0-255 • `data1`: write to register value 1 • `data2`: write register value 2 • `isNoBlock`: whether blocking 0 - blocking; 1 - non-blocking
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.38 End sensor code example

```
from fairino import Robot
import time
import threading
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
robot.AxleSensorConfig(18, 0, 0, 1)
error, company, type = robot.AxleSensorConfigGet()
print(f"company is:{company},type is:{type}")
rtn = robot.AxleSensorActivate(1)
print(f"AxleSensorActivate rtn is:{rtn}")
time.sleep(1)
rtn = robot.AxleSensorRegWrite(1, 4, 6, 1, 0, 0, 0)
print(f"AxleSensorRegWrite rtn is:{rtn}")
robot.CloseRPC()
```

2.4.11.39 Obtaining Robot Peripheral Protocols

New in version python: SDK-v2.0.3

Prototype	`GetExDevProtocol()`
Description	Get robot peripheral protocol
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode. • `protocol`: Robot peripheral protocol number 4096-Extended Axis Control Card; 4097-ModbusSlave; 4098-ModbusMaster

2.4.11.40 Setting up robot peripheral protocols

New in version python: SDK-v2.0.3

Prototype	`SetExDevProtocol(protocol)`
Description	Setting the robot peripheral protocol
Mandatory parameters	• `protocol`: robot peripheral protocol number 4096 - Extended Axis Control Card; 4097 - ModbusSlave; 4098 - ModbusMaster
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.41 Example of setup robot peripheral protocol code

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
protocol = 4096
rtn = robot.SetExDevProtocol(protocol)
print(f"SetExDevProtocol rtn:{rtn}")
rtn, protocol = robot.GetExDevProtocol()
print(f"GetExDevProtocol rtn:{rtn},protocol is:{protocol}")
robot.CloseRPC()
```

2.4.11.42 Getting end communication parameters

New in version python: SDK-v2.0.5

Prototype	`GetAxleCommunicationParam()`
Description	Get end communication parameters
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `baudRate`: baud rate: support 1-9600, 2-14400, 3-19200, 4-38400, 5-56000, 6-67600, 7-115200, 8-128000 • `dataBit`: data bit: data bit support (8,9), currently commonly used 8 • `stopBit`: stop bit: 1-1, 2-0.5, 3-2, 4-1.5, currently 1 is commonly used. • `verify`: check digit: 0-None, 1-Odd, 2-Even, currently 0. • `timeout`: timeout time: 1~1000ms, this value needs to be combined with the peripheral with the setting of reasonable time parameters • `timeoutTimes`: timeout times: 1~10, mainly for timeout retransmission, reduce occasional exceptions to improve user experience • `period`: periodic instruction time interval:1~1000ms, mainly used for the time interval between each issuance of periodic instructions

2.4.11.43 Setting the end communication parameters

New in version python: SDK-v2.0.5

Prototype	`SetAxleCommunicationParam(baudRate, dataBit, stopBit, verify, timeout, timeoutTimes, period)`
description	set end communication parameters
Required Parameters	• `baudRate`: baud rate: supports 1-9600, 2-14400, 3-19200, 4-38400, 5-56000, 6-67600, 7-115200, 8-128000 • `dataBit`: data bit: data bit support (8,9), currently commonly used 8 • `stopBit`: stop bit: 1-1, 2-0.5, 3-2, 4-1.5, currently 1 is commonly used. • `verify`: check digit: 0-None, 1-Odd, 2-Even, currently 0. • `timeout`: timeout time: 1~1000ms, this value needs to be combined with the peripheral with the setting of reasonable time parameters • `timeoutTimes`: timeout times: 1~10, mainly for timeout retransmission, reduce occasional exceptions to improve user experience • `period`: periodic instruction time interval:1~1000ms, mainly used for the time interval between each issuance of periodic instructions
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.44 Setting the end file transfer type

New in version python: SDK-v2.0.5

Prototype	`SetAxleFileType(type)`
Description	Set the end file transfer type
Mandatory parameters	• `type`: 1-MCU upgrade file, 2-LUA file
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.45 Setting Enable End LUA Execution

New in version python: SDK-v2.0.5

Prototype	`SetAxleLuaEnable(enable)`
Description	Set Enable End LUA Enforcement
Mandatory parameters	• `enable`: 0 - not enabled; 1 - enabled
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.46 End LUA file exception error recovery

New in version python: SDK-v2.0.5

Prototype	`SetRecoverAxleLuaErr(enable)`
Description	End LUA File Exception Error Recovery
Mandatory parameters	• `status`: 0 - no recovery; 1 - recovery
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.47 Get end LUA execution enable status

New in version python: SDK-v2.0.5

Prototype	`GetAxleLuaEnableStatus()`
description	Get end LUA execution enable state
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `enable`: 0-don't enable; 1-enable

2.4.11.48 Setting the end LUA end device enable type

New in version python: SDK-v2.0.5

prototype	`SetAxleLuaEnableDeviceType(forceSensorEnable, gripperEnable, IOEnable)`
Description	Set end LUA end device enable type
Mandatory parameters	• `forceSensorEnable`: force sensor enable status, 0 - not enabled; 1 - enabled • `gripperEnable`: gripper enable status, 0 - not enabled; 1 - enabled • `IOEnable`: IO device enable status, 0-not enabled; 1-enabled
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.49 Get End LUA End Device Enablement Type

New in version python: SDK-v2.0.5

Prototype	`GetAxleLuaEnableDeviceType()`
Description	Get End LUA End Device Enablement Type
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `forceSensorEnable`: force sensor enable status, 0 - not enabled; 1 - enabled • `gripperEnable`: gripper enable status, 0 - not enabled; 1 - enabled • `IOEnable`: IO device enable status, 0-not enabled; 1-enabled

2.4.11.50 Get the currently configured end device

New in version python: SDK-v2.0.5

Prototype	`GetAxleLuaEnableDevice()`
Description	Get the currently configured end device
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `forceSensorEnable[8]`: force sensor enable state, 0 - not enabled; 1 - enabled • `gripperEnable[8]`: gripper enable status, 0 - not enabled; 1 - enabled • `IOEnable[8]`: IO device enable status, 0-not enabled; 1-enabled

2.4.11.51 Setting to enable the jaw movement control function

New in version python: SDK-v2.0.5

Prototype	`SetAxleLuaGripperFunc(id, func)`
Description	Set to enable the jaw motion control function
Mandatory parameters	• `id`: gripper device number • `func`: 0-jaw enable; 1-jaw initialization; 2-position setting; 3-speed setting; 4-torque setting; 6-reading the jaw status; 7-reading the initialization status; 8-reading the fault code; 9-reading the position; 10-reading the speed; 11-reading the torque,12-15 reserved
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.52 Getting to Enable Jaw Motion Control

New in version python: SDK-v2.0.5

Prototype	`GetAxleLuaGripperFunc(id)`
Description	Get the Enable Jaw Motion Control function
Mandatory parameter	• `id`: gripper device number
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `func`: 0-jaw enable; 1-jaw initialization; 2-position setting; 3-speed setting; 4-torque setting; 6-reading the jaw status; 7-reading the initialization status; 8-reading the fault code; 9-reading the position; 10-reading the speed; 11-reading the torque,12-15 reserved

2.4.11.53 The Ethercat slave file is written by the robot

New in version python: SDK-v2.0.5

Prototype	`SlaveFileWrite(type,slaveID,fileName)`
Description	The Ethercat slave file is written by the robot
Mandatory parameter	• `type`Slave file type, 1- Upgrade slave file; 2- Upgrade the slave station configuration file • `slaveID`From the station number • `fileName`Upload the file name
Default parameters	NULL
Return Value	errorcode Success-0 Failure- errcode

2.4.11.54 Upload the end Lua open protocol file

New in version python: SDK-v2.0.5

Prototype	`AxleLuaUpload(filePath)`
Description	Upload the end Lua open protocol file
Mandatory parameter	• `filePath`Local lua file path name …/AXLE_LUA_End_DaHuan.lua
Default parameters	NULL
Return Value	errorcode Success-0 Failure- errcode

2.4.11.55 The robot Ethercat enters boot mode from the station

New in version python: SDK-v2.0.5

Prototype	SetSysServoBootMode(filePath)
Description	The robot Ethercat enters boot mode from the station
Mandatory parameter	NULL
Default parameters	NULL
Return Value	errorcode Success-0 Failure- errcode

2.4.11.56 Example of LUA file manipulation code at the end of the robot

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.AxleLuaUpload("D://zUP/AXLE_LUA_End_DaHuan.lua")
param = [7, 8, 1, 0, 5, 3, 1]  # AxleComParam
robot.SetAxleCommunicationParam(7, 8, 1, 0, 5, 3, 1)
error,getParam0,getParam1,getParam2,getParam3,getParam4,getParam5,getParam6 = robot.
↪GetAxleCommunicationParam()
print(f"GetAxleCommunicationParam param is:{getParam0} {getParam1} {getParam2}
↪{getParam3} {getParam4} {getParam5} {getParam6}")
robot.SetAxleLuaEnable(1)
error,luaEnableStatus = robot.GetAxleLuaEnableStatus()
robot.SetAxleLuaEnableDeviceType(0, 1, 0)
error,forceEnable, gripperEnable, ioEnable = robot.GetAxleLuaEnableDeviceType()
print(f"GetAxleLuaEnableDeviceType param is:{forceEnable} {gripperEnable} {ioEnable}")
func = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
robot.SetAxleLuaGripperFunc(1, func)
error,getFunc = robot.GetAxleLuaGripperFunc(1)
error,getforceEnable, getgripperEnable, getioEnable = robot.GetAxleLuaEnableDevice()
print("\ngetforceEnable status:", end=" ")
for i in range(8):
    print(f"{getforceEnable[i]},", end="")
print("\ngetgripperEnable status:", end=" ")
for i in range(8):
    print(f"{getgripperEnable[i]},", end="")
print("\ngetioEnable status:", end=" ")
for i in range(8):
    print(f"{getioEnable[i]},", end="")
print()
robot.ActGripper(1, 0)
time.sleep(2)
robot.ActGripper(1, 1)
time.sleep(2)
robot.MoveGripper(1, 90, 10, 100, 50000, 0, 0, 0, 0, 0)
while True:
    error,pkg = robot.GetRobotRealTimeState()
    print(f"gripper pos is:{pkg.gripper_position}")
    time.sleep(0.1)
```

(continues on next page)

(continued from previous page)

```
robot.CloseRPC()
```

2.4.11.57 Obtain the status of the SmartTool button

New in version python: SDK-v2.1.2

Prototype	`GetSmarttoolBtnState()`
Description	Obtain the status of the SmartTool button
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `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

2.4.11.58 SmartTool button code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
while True:
    error,state = robot.GetSmarttoolBtnState()
    print(f"{state:016b}")
    time.sleep(0.1)
```

2.4.11.59 Set the load detection before drag is started

New in version python: SDK-v2.1.6

Prototype	`SetTorqueDetectionSwitch(flag)`
Description	Set the load detection before drag is started
Mandatory parameters	• `flag`0- closed; 1- On
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.60 Laser peripheral open and close function

New in version python: SDK-v2.1.6

Prototype	`LaserTrackingLaserOnOff(OnOff, weldId)`
Description	Laser peripheral open and close function
Mandatory parameters	• `OnOff`0- closed; 1- On
Default parameters	• `weldId`The default weld ID is 0
Return Value	Error Code Success-0 Failure- errcode

2.4.11.61 Laser tracking start-end function

New in version python: SDK-v2.1.6

Prototype	`LaserTrackingTrackOnOff(OnOff, coordId)`
Description	Laser tracking start-end function
Mandatory parameters	• `OnOff`0- closed; 1- On • `coordId`Laser peripheral tool coordinate system No
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.62 Laser positioning - Fixed direction

New in version python: SDK-v2.1.6

Prototype	`LaserTrackingSearchStart_xyz(direction, vel, distance, timeout, posSensorNum)`
Description	Laser positioning - Fixed direction
Mandatory parameters	• `direction`0-x+ 1-x- 2-y+ 3-y- 4-z+ 5-z- • `vel`Unit of speed % • `distance`The maximum positioning distance unit is mm • `timeout`The unit of seek timeout time is ms • `posSensorNum`The coordinate number of the tool calibrated by laser
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.63 Laser positioning - in any direction

New in version python: SDK-v2.1.6

Prototype	`LaserTrackingSearchStart_point(directionPoint, vel, distance, timeout, posSensorNum)`
Description	Laser positioning - in any direction
Mandatory parameters	• `directionPoint`To the left of the xyz of the input point for positioning,[x,y,z] • `vel`Unit of speed % • `distance`The maximum positioning distance unit is mm • `timeout`The unit of seek timeout time is ms • `posSensorNum`The coordinate number of the tool calibrated by laser
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.64 Laser IP configuration

New in version python: SDK-v2.1.6

Prototype	`LaserTrackingSensorConfig(ip, port)`
Description	Laser IP configuration
Mandatory parameters	• `ip`The ip address of the laser peripheral • `port`The port number of the laser peripheral
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.65 Configuration of sampling period for laser peripherals

New in version python: SDK-v2.1.6

Prototype	`LaserTrackingSensorSamplePeriod(period)`
Description	Configuration of sampling period for laser peripherals
Mandatory parameters	• `period`The unit of sampling period for laser peripherals is ms
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.66 Laser peripheral driver loading

New in version python: SDK-v2.1.6

Prototype	`LoadPosSensorDriver(type)`
Description	Laser peripheral driver loading
Mandatory parameters	• `type`Protocol type of the laser device driver: 101-Ruineng 102-Chuangxiang 103-Quanshi 104-Tongzhou 105-Aotai
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.67 Laser peripheral driver unloading

New in version python: SDK-v2.1.6

Prototype	`UnLoadPosSensorDriver()`
Description	Laser peripheral driver unloading
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.68 Laser weld seam trajectory recording

New in version python: SDK-v2.1.6

Prototype	`LaserSensorRecord1(status, delayTime)`
Description	Laser weld seam trajectory recording
Mandatory parameters	• `status`0- Stop recording 1- Real-time tracking 2- Start recording • `delayTime`The delay time unit is ms
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.69 Laser weld seam trajectory reproduction

New in version python: SDK-v2.1.6

Prototype	`LaserSensorReplay(delayTime, speed)`
Description	Laser weld seam trajectory reproduction
Mandatory parameters	• `delayTime`The delay time unit is ms • `speed`Unit of speed %
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.70 Laser tracking reproduction

New in version python: SDK-v2.1.6

Prototype	`MoveLTR()`
Description	Laser tracking reproduction
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.71 Laser weld seam trajectory reproduction

New in version python: SDK-v2.1.6

Prototype	`LaserSensorRecordandReplay(delayMode, delayTime, delayDisExAxisNum, delayDis, sensitivePara, int trackMode, int triggerMode, double runTime, speed)`
Description	Laser Weld Seam Trajectory Replay
Required Parameters	• `delayMode`: Mode 0-Delay Time 1-Delay Distance • `delayTime`: Delay time in milliseconds (ms) • `delayDisExAxisNum`: Extended Axis Number • `delayDis`: Delay distance in millimeters (mm) • `sensitivePara`: Compensation Sensitivity Coefficient • `trackMode`: Fixed-point Tracking Type. 0-Extended Axis Asynchronous Motion; 1-Robot • `triggerMode`: Fixed-point Tracking Trigger Method. 0-Tracking Duration; 1-IO • `runTime`: Robot Fixed-point Tracking Duration in seconds (s) • `speed`: Speed in percentage (%)
Default Parameters	None
Return Value	Error Code. Success - 0, Failure - errcode

2.4.11.72 Movement to the starting point of weld record

New in version python: SDK-v2.1.6

Prototype	`MoveToLaserRecordStart(moveType, ovl)`
Description	Movement to the starting point of weld record
Mandatory parameters	• `moveType`0-PTP 1-LIN • `ovl`Speed unit %
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.73 Movement to the end of the weld record

New in version python: SDK-v2.1.6

Prototype	`MoveToLaserRecordEnd(moveType, ovl)`
Description	Movement to the end of the weld record
Mandatory parameters	• `moveType`0-PTP 1-LIN • `ovl`Speed unit %
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.74 Move to the laser sensor to find the site

New in version python: SDK-v2.1.6

Prototype	`MoveToLaserSeamPos(moveFlag, ovl, dataFlag, plateType, trackOffectType, offset)`
Description	Move to the laser sensor to find the site
Mandatory parameters	• `moveFlag`Motion type: 0-PTP; 1-LIN • `ovl`Speed scaling factor, 0-100 • `dataFlag`Weld cache data selection: 0-execution planning data; 1- Perform logging data • `plateType`Plate type: 0-corrugated plate; 1- Corrugated board; 2- Fence board; 3- oil barrel; 4- Corrugated shell steel • `trackOffectType`Laser sensor offset type: 0-no offset; 1-base coordinate shift; 2- Tool coordinate offset; 3-Laser sensor raw data offset • `offset`Offset value
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.11.75 Obtain the coordinate information of the laser sensor location

New in version python: SDK-v2.1.6

Prototype	`GetLaserSeamPos(trackOffectType, offset)`
Description	Obtain the coordinate information of the laser sensor location
Mandatory parameters	• `trackOffectType`Laser sensor offset type: 0-no offset; 1-base coordinate shift; 2- Tool coordinate offset; 3-Laser sensor raw data offset • `offset`Offset value
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `jPos`Joint position[°] • `descPos`Cartesian position[mm] • `tool`Tool coordinate system • `user`Workpiece coordinate system • `exaxis`Extension axis position[mm]

2.4.11.76 Example of laser peripheral sensor parameter configuration and debugging code

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.LaserTrackingSensorConfig("192.168.58.20", 5020)
robot.LaserTrackingSensorSamplePeriod(20)
robot.LoadPosSensorDriver(101)
robot.LaserTrackingLaserOnOff(0, 0)
time.sleep(3)
robot.LaserTrackingLaserOnOff(1, 0)
robot.CloseRPC()
```

2.4.11.77 Code example of laser trajectory scanning and trajectory reproduction

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.OpenLuaUpload("D://zUP/CtrlDev_laser_ruiniu-0117.lua")
time.sleep(2)
robot.SetCtrlOpenLUAName(0, "CtrlDev_laser_ruiniu-0117.lua")
robot.UnloadCtrlOpenLUA(0)
robot.LoadCtrlOpenLUA(0)
time.sleep(8)
i = 0
while i<10:
    startjointPos = [56.205, -117.951, 141.872, -118.149, -94.217, -122.176]
    startdescPose = [-97.552, -282.855, 26.675, 174.182, -1.338, -91.707]
    exaxisPos = [0.0] * 4
    offdese = [0.0] * 6
    robot.MoveL(desc_pos=startdescPose,tool= 1,user= 0,vel= 100,acc= 100,ovl= 100,
```

(continues on next page)

(continued from previous page)

```
↪blendR= -1,exaxis_pos= exaxisPos,search= 0,offset_flag= 0, offset_pos= offdese,
↪overSpeedStrategy= 1,speedPercent= 1)
    robot.LaserSensorRecord1(2, 10)
    endjointPos = [68.809, -87.100, 121.120, -127.233, -95.038, -109.555]
    enddescPose = [-103.555, -464.234, 13.076, 174.179, -1.344, -91.709]
    robot.MoveL(desc_pos=enddescPose,tool= 1,user= 0,vel= 50,acc= 100,ovl= 100,blendR= -
↪1,exaxis_pos= exaxisPos,search= 0,offset_flag= 0, offset_pos= offdese,
↪overSpeedStrategy= 1,speedPercent= 1)
    robot.LaserSensorRecord1(0, 10)
    robot.MoveToLaserRecordStart(1, 30)
    robot.LaserSensorReplay(10, 100)
    robot.MoveLTR()
    robot.LaserSensorRecord1(0, 10)
    i = i+1
robot.CloseRPC()
```

2.4.11.78 Code examples for laser locating and real-time tracking

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.OpenLuaUpload("D://zUP/CtrlDev_laser_ruiniu-0117.lua")
time.sleep(2)
robot.SetCtrlOpenLUAName(0, "CtrlDev_laser_ruiniu-0117.lua")
robot.UnloadCtrlOpenLUA(0)
robot.LoadCtrlOpenLUA(0)
time.sleep(8)
time.sleep(8)
i = 0
while i < 10:
    startjointPos = [56.205, -117.951, 141.872, -118.149, -94.217, -122.176]
    startdescPose = [-97.552, -282.855, 26.675, 174.182, -1.338, -91.707]
    exaxisPos = [0.0] * 4
    offdese = [0.0] * 6
    directionPoint = [0.0] * 3
    robot.MoveL(desc_pos=startdescPose,tool= 1,user= 0,vel= 100,acc= 100,ovl= 100,
↪blendR= -1,exaxis_pos= exaxisPos,search= 0,offset_flag= 0, offset_pos= offdese,
↪overSpeedStrategy= 1,speedPercent= 1)
    robot.LaserTrackingSearchStart_xyz(3, 100, 300, 1000, 3)
    robot.LaserTrackingSearchStop()
    robot.MoveToLaserSeamPos(1, 30, 0, 0, 0, offdese)
    robot.LaserTrackingTrackOnOff(1, 3)
    endjointPos = [68.809, -87.100, 121.120, -127.233, -95.038, -109.555]
    enddescPose = [-103.555, -464.234, 13.076, 174.179, -1.344, -91.709]
    robot.MoveL(desc_pos=enddescPose,tool= 1,user= 0,vel= 20,acc= 100,ovl= 100,blendR= -
↪1,exaxis_pos= exaxisPos,search= 0,offset_flag= 0, offset_pos= offdese,
↪overSpeedStrategy= 1,speedPercent= 1)
    robot.LaserTrackingTrackOnOff(0, 3)
    i = i + 1
```

(continues on next page)

(continued from previous page)

```
    print(i)
robot.CloseRPC()
```

2.4.11.79 Code example of the extended axis synchronized with the robot for laser tracking

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
startexaxisPos = [0.0, 0.0, 0.0, 0.0]
seamexaxisPos = [-10.0, 0.0, 0.0, 0.0]
endexaxisPos = [-30.0, 0.0, 0.0, 0.0]
offdese = [0.0] * 6
seamjointPos = [0.0] * 6
seamdescPose = [0.0] * 6
i=0
while i < 10:
    startjointPos = [58.337, -119.628, 146.037, -116.358, -92.224, -117.654]
    startdescPose = [-53.375, -255.363, 0.919, 178.054, 1.077, -94.026]
    robot.ExtAxisSyncMoveJ(joint_pos=startjointPos, tool=1,user= 0,vel= 100,acc= 100,ovl=100,exaxis_pos= startexaxisPos,blendT= -1,offset_flag= 0,offset_pos= offdese)
    ret = robot.LaserTrackingSearchStart_xyz(3, 100, 300, 1000, 2)
    robot.LaserTrackingSearchStop()
    tool = 0
    user = 0
    rnte, seamjointPos, seamdescPose, tool, user, startexaxisPos = robot.GetLaserSeamPos(0, offdese)
    print(f"{seamjointPos[0]},{seamjointPos[1]},{seamjointPos[2]},{seamjointPos[3]},{seamjointPos[4]},{seamjointPos[5]},{seamdescPose[0]},{seamdescPose[1]},{seamdescPose[2]},{seamdescPose[3]},{seamdescPose[4]},{seamdescPose[5]}")
    if ret == 0:
        robot.ExtAxisSyncMoveJ(joint_pos=seamjointPos, tool=1,user= 0,vel= 100,acc= 100,ovl=100,exaxis_pos= seamexaxisPos,blendT= -1,offset_flag= 0,offset_pos= offdese)
        robot.LaserTrackingTrackOnOff(1, 2)
        endjointPos = [70.580, -90.918, 126.593, -125.154, -92.162, -105.403]
        enddescPose = [-53.375, -419.020, 0.920, 178.054, 1.076, -94.026]
        robot.ExtAxisSyncMoveL(desc_pos=enddescPose, tool=1,user= 0,vel= 20,acc= 100,ovl=100,blendR= -1,exaxis_pos= endexaxisPos,offset_pos= offdese)
        robot.LaserTrackingTrackOnOff(0, 2)
    i = i+1
    print(i)
robot.CloseRPC()
```

2.4.12 Force Control

2.4.12.1 Force Sensor Configuration

Prototype	`FT_SetConfig(company,device,softversion=0,bus=0)`
Description	Force Sensor Configuration
Required Parameters	• `company`: Sensor Manufacturer, 17 - Kunwei Technology, 19 - Aerospace 11th Academy, 20 - ATI Sensors, 21 - Zhongke MiDot, 22 - Weihang Minxin23-NBIT24-XJC26-NSR; • `device`: device number, Kunwei (0-KWR75B), Aisino Eleventh Academy (0-MCS6A-200-4), ATI (0-AXIA80-M8), Zhongke MiDot (0-MST2010), Weihang Minxin (0-WHC6L-YB-10A)NBIT(0-XLH93003ACS)XJC(0-XJC-6F-D82)NSR(0-NSR-FTSensorA);
Default parameters	• `softversion`: software version number, not used for now, default is 0; • `bus`: device mount end bus location, not used yet, default is 0;
Return Value	Error Code Success-0 Failure- errcode

2.4.12.2 Get Force Sensor Configuration

Prototype	`FT_GetConfig()`
Description	Get Force Sensor Configuration
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `[number,company,device,softversion,bus]`number ;company 17-19-20-ATI 21-22-;device (0- KWR75B), Aisino Eleven (0-MCS6A-200-4), ATI (0-AXIA80-M8), Zhongke MiDot (0-MST2010), Weihang Minxin (0-WHC6L-YB10A); softvesion software version number, not used for the time being, the default is 0.

2.4.12.3 Force sensor activation

Prototype	`FT_Activate(state)`
Description	Force sensor activation
Mandatory parameters	• `state`: 0-reset, 1-activate
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.4 Force Sensor Zeroing

Prototype	`FT_SetZero(state)`
Description	Force Transducer Zeroing
Mandatory parameters	• `state`: 0-removal of zeros, 1-zero correction
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.5 Setting the force transducer reference coordinate system

Changed in version python: SDK-v2.0.5

prototype	`FT_SetRCS(ref,coord=[0,0,0,0,0,0,0])`
Description	Setting the force transducer reference coordinate system
Mandatory parameters	• `ref`: 0 - tool coordinate system, 1 - base coordinate system
Default parameters	• `coord`: [x,y,z,rx,ry,rz] customized coordinate system values, default [0,0,0,0,0,0,0]
Return Value	Error Code Success-0 Failure- errcode

2.4.12.6 Setting the load weight under the force transducer

New in version python: SDK-v2.0.5

Prototype	`SetForceSensorPayload(weight)`
Description	Set the load weight under the force transducer
Mandatory parameters	• `weight`: load weight kg
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.7 Setting the load center of mass under the force transducer

New in version python: SDK-v2.0.5

Prototype	`SetForceSensorPayloadCog(x,y,z)`
Description	Setting the center of mass of the load under the force transducer
Mandatory parameters	• `x`: load center of mass x mm • `y`: load center of mass y mm • `z`: load center of mass z mm
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.8 Getting the load weight under the force transducer

New in version python: SDK-v2.0.5

Prototype	`GetForceSensorPayload()`
Description	Get the weight of the load under the force transducer.
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `weight`: loaded weight kg

2.4.12.9 Obtaining the center of mass of the load under the force transducer

New in version python: SDK-v2.0.5

Prototype	`GetForceSensorPayloadCog()`
Description	Getting the center of mass of the load under the force transducer
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `x`: load center of mass x mm • `y`: load center of mass y mm • `z`: load center of mass z mm

2.4.12.10 Automatic zeroing of force sensors

New in version python: SDK-v2.0.5

Prototype	`ForceSensorAutoComputeLoad()`
Description	Automatic zeroing of force sensors
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `weight`: sensor mass kg • `pos=[x,y,z]`: sensor center of mass mm

2.4.12.11 Obtaining force/torque data in the reference coordinate system

Prototype	`FT_GetForceTorqueRCS()`
Description	Get force/torque data in reference coordinate system
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `data=[fx,fy,fz,tx,ty,tz]`: force/torque data in the reference coordinate system.

2.4.12.12 Obtaining Force Sensor Raw Force/Torque Data

Prototype	`FT_GetForceTorqueOrigin()`
Description	Get force sensor raw force/torque data
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `data=[fx,fy,fz,tx,ty,tz]`: Force sensor raw force/torque data

2.4.12.13 Force sensor configuration and automatic zero correction code example

```
import frrpc
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = frrpc.RPC('192.168.58.2')
company = 24
device = 0
softversion = 0
bus = 1
index = 1
robot.FT_SetConfig(company, device, softversion, bus)
time.sleep(1)
error,[company, device, softversion, bus] = robot.FT_GetConfig()
print(f"FT config:{company},{device},{softversion},{bus}")
time.sleep(1)
robot.FT_Activate(0)
time.sleep(1)
robot.FT_Activate(1)
time.sleep(1)
time.sleep(1)
robot.FT_SetZero(0)
time.sleep(1)
error,ft = robot.FT_GetForceTorqueOrigin()
print(f"ft origin:{ft[0]},{ft[1]},{ft[2]},{ft[3]},{ft[4]},{ft[5]}")
robot.FT_SetZero(1)
```

(continues on next page)

(continued from previous page)

```
time.sleep(1)
ftCoord = [0, 0, 0, 0, 0, 0]
robot.FT_SetRCS(0, ftCoord)
robot.SetForceSensorPayload(0.824)
robot.SetForceSensorPayloadCog(0.778, 2.554, 48.765)
error,weight = robot.GetForceSensorPayload()
error,x, y, z = robot.GetForceSensorPayloadCog()
print(f"the FT load is  {weight}, {x} {y} {z}")
robot.SetForceSensorPayload(0)
robot.SetForceSensorPayloadCog(0, 0, 0)
error,computeWeight, tran = robot.ForceSensorAutoComputeLoad()
print(f"the result is weight {computeWeight} pos is {tran[0]} {tran[1]} {tran[2]}")
robot.CloseRPC()
```

2.4.12.14 Load weight identification records

Prototype	`FT_PdIdenRecord(tool_id)`
Description	Load weight identification record
Mandatory parameters	• `tool_id`: sensor coordinate system number, range [0~14]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.15 Load weight identification calculation

Prototype	`FT_PdIdenCompute()`
Description	Load weight identification calculation
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `weight`: weight of the load in kg

2.4.12.16 Load center of mass identification records

prototype	`FT_PdCogIdenRecord(tool_id,index)`
Description	Load center of mass identification record
Mandatory parameter	• `tool_id`: sensor coordinate system number, range [0~14]. • `index`: point number [1 to 3]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.17 Load center of mass identification calculation

Prototype	`FT_PdCogIdenCompute()`
Description	Load center of mass identification calculation
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `cog=[cogx,cogy,cogz]`: center of mass of load in mm

2.4.12.18 Force sensor load identification code example

```
import frrpc
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = frrpc.RPC('192.168.58.2')
company = 24
device = 0
softversion = 0
bus = 1
index = 1
robot.FT_SetConfig(company, device, softversion, bus)
time.sleep(1)
error,[company, device, softversion, bus] = robot.FT_GetConfig()
print(f"FT config:{company},{device},{softversion},{bus}")
time.sleep(1)
robot.FT_Activate(0)
time.sleep(1)
robot.FT_Activate(1)
time.sleep(1)
robot.FT_SetZero(0)
time.sleep(1)
error,ft = robot.FT_GetForceTorqueOrigin()
print(f"ft origin:{ft[0]},{ft[1]},{ft[2]},{ft[3]},{ft[4]},{ft[5]}")
robot.FT_SetZero(1)
time.sleep(1)
tcoord = [0, 0, 35.0, 0, 0, 0]
robot.SetToolCoord(10, tcoord, 1, 0, 0, 0)
robot.FT_PdIdenRecord(10)
time.sleep(1)
error,weight = robot.FT_PdIdenCompute()
print(f"payload weight:{weight}")
desc_p1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
desc_p2 = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
desc_p3 = [-327.622, 402.230, 320.402, -178.067, 2.127, -46.207]
robot.MoveCart(desc_p1, 0, 0, 100.0)
time.sleep(1)
robot.FT_PdCogIdenRecord(10, 1)
robot.MoveCart(desc_p2, 0, 0, 100.0)
time.sleep(1)
```

(continues on next page)

(continued from previous page)

```
robot.FT_PdCogIdenRecord(10, 2)
robot.MoveCart(desc_p3, 0, 0, 100.0)
time.sleep(1)
robot.FT_PdCogIdenRecord(10, 3)
time.sleep(1)
error,cog = robot.FT_PdCogIdenCompute()
print(f"FT_PdCogIdenCompute return {error}")
print(f"cog:{cog[0]},{cog[1]},{cog[2]}")
robot.CloseRPC()
```

2.4.12.19 Collision Guard

Prototype	`FT_Guard(flag,sensor_num,select,force_torque,max_threshold, min_threshold)`
description	collision guarding
Required Parameters	• `flag`: 0 - turn off collision guarding, 1 - turn on collision guarding; • `sensor_num`: force sensor number; • `select`: whether or not the six degrees of freedom detect collisions [fx,fy,fz,mx,my,mz], 0 - not valid, 1 - valid; • `force_torque`: collision detection force/torque in N or Nm; • `max_threshold`: maximum threshold; • `min_threshold`: minimum threshold; • Force/torque detection range: (force_torque-min_threshold,force_torque+max_threshold)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.20 Collision guard code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
company = 24
device = 0
softversion = 0
bus = 1
index = 1
robot.FT_SetConfig(company, device, softversion, bus)
time.sleep(1)
error,[company, device, softversion, bus] = robot.FT_GetConfig()
print(f"FT config:{company},{device},{softversion},{bus}")
time.sleep(1)
robot.FT_Activate(0)
time.sleep(1)
robot.FT_Activate(1)
time.sleep(1)
```

(continues on next page)

(continued from previous page)

```
robot.FT_SetZero(0)
time.sleep(1)
sensor_id = 1
select = [1, 1, 1, 1, 1, 1]
max_threshold = [10.0, 10.0, 10.0, 10.0, 10.0, 10.0]
min_threshold = [5.0, 5.0, 5.0, 5.0, 5.0, 5.0]
ft = None
desc_p1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
desc_p2 = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
desc_p3 = [-327.622, 402.230, 320.402, -178.067, 2.127, -46.207]
error = robot.FT_Guard(1, sensor_id, select,[0.0,0.0,0.0,0.0,0.0,0.0], max_threshold,␣
↪min_threshold)
robot.MoveCart(desc_p1, 0, 0, 100.0)
robot.MoveCart(desc_p2, 0, 0, 100.0)
robot.MoveCart(desc_p3, 0, 0, 100.0)
robot.FT_Guard(0, sensor_id, select,[0.0,0.0,0.0,0.0,0.0,0.0], max_threshold, min_
↪threshold)
robot.CloseRPC()
```

2.4.12.21 constant force control

prototype	`FT_Control(flag, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis, max_ang, M=None, B=None, threshold=[0.2, 0.2], adjustCoeff=[1.0,1.0], polishRadio=0, filter_Sign=0, posAdapt_sign=0, isNoBlock=0)`
Description	Constant force control
Mandatory parameter	• `flag`: constant force control on flag, 0-off, 1-on; • `sensor_id`: force sensor number; • `select`: whether the six degrees of freedom are detected [fx,fy,fz,mx,my,mz], 0 - not valid, 1 - valid; • `ft`: detection of force/torque in N or Nm; • `ft_pid`: [f_p,f_i,f_d,m_p,m_i,m_d], force PID parameter, moment PID parameter; • `adj_sign`: adaptive start/stop state, 0-off, 1-on; • `ILC_sign`: ILC control start/stop status, 0-stop, 1-training, 2-practical; • `max_dis`: maximum adjustment distance; • `max_ang`: maximum angle of adjustment;
Default parameters	• `M`Quality parameters • `B`Damping parameter • `threshold`Activation threshold for rx, ry [0-10], default 0.2 • `adjustCoeff`Torque adjustment coefficient for rx, ry [0-1], default 1 • `polishRadio`Grinding disc radius, unit: mm • `filter_Sign`Filter on indicator 0- off; 1- On, default 0- off • `posAdapt_sign`The posture conforms to the opening mark 0-off. 1- On, default 0- off • `isNoBlock`Block flag, 0- block; 1- Non-blocking default 0- blocking
Return Value	Error Code Success-0 Failure- errcode

2.4.12.22 A code example of constant force control with damping

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
sensor_id = 10
select = [0, 0, 1, 0, 0, 0]
ft_pid = [0.0008, 0.0, 0.0, 0.0, 0.0, 0.0]
adj_sign = 0
ILC_sign = 0
max_dis = 1000.0
max_ang = 20.0
ft = [0.0] * 6
epos = [0.0] * 4
j1 = [80.765, -98.795, 106.548, -97.734, -89.999, 94.842]
j2 = [43.067, -84.429, 92.620, -98.175, -90.011, 57.144]
desc_p1 = [5.009, -547.463, 262.053, -179.999, -0.019, 75.923]
desc_p2 = [-347.966, -547.463, 262.048, -180.000, -0.019, 75.923]
offset_pos = [0.0] * 6
M = [2.0, 2.0]
B = [15.0, 15.0]
threshold = [1.0, 1.0]
adjustCoeff = [1.0, 0.8]
polishRadio = 0.0
filter_Sign = 0
posAdapt_sign = 1
isNoBlock = 0
ft[2] = -10.0
while True:
    rtn = robot.FT_Control(1, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis,
↪ max_ang, M, B, threshold,adjustCoeff, 0, 0, 1, 0)
    print(f"FT_Control start rtn is {rtn}")
    rtn = robot.MoveL(desc_pos=desc_p1, tool=1, user=0, vel=100, acc=100, ovl=100,␣
↪blendR=-1, blendMode = 0, exaxis_pos=epos, search=0, offset_flag=0, offset_pos=offset_
↪pos)
    rtn = robot.MoveL(desc_pos=desc_p2, tool=1, user=0, vel=100, acc=100, ovl=100,␣
↪blendR=-1, blendMode = 0, exaxis_pos=epos, search=0, offset_flag=0, offset_pos=offset_
↪pos)
    rtn = robot.FT_Control(0, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis,
↪ max_ang, M, B, threshold,adjustCoeff, 0, 0, 1, 0)
    print(f"FT_Control end rtn is {rtn}")
robot.CloseRPC()
return 0
```

2.4.12.23 Helix Exploration

Prototype	`FT_SpiralSearch(rcs, ft, dr=0.7, max_t_ms=60000, max_vel=5)`
Description	Helix Explorations
Mandatory parameters	• `rcs`: reference coordinate system, 0 - tool coordinate system, 1 - base coordinate system • `ft`: force or moment threshold (0 to 100) in N or Nm;
Default parameters	• `dr`: radius feed per revolution in mm Default 0.7. • `max_t_ms`: maximum time to explore, in ms default 60000. • `max_vel`: maximum value of linear velocity in mm/s Default 5
Return Value	Error Code Success-0 Failure- errcode

2.4.12.24 Rotational Insertion

Prototype	`FT_RotInsertion(rcs, ft, orn, angVelRot=3, angleMax=45, angAccmax=0, rotorn=1, strategy=0)`
Description	Rotational Insertion
Required Parameters	• `rcs`: Reference coordinate system, 0 - Tool coordinate system, 1 - Base coordinate system; • `ft`: Force or torque threshold (0~100), unit N or Nm; • `orn`: Force/torque direction, 1 - Along Z-axis direction, 2 - Around Z-axis direction;
Default Parameters	• `angVelRot`: Rotational angular velocity, unit °/s, default 3; • `angleMax`: Maximum rotation angle, unit °, default 45; • `angAccmax`: Maximum rotational acceleration, unit °/s^2, currently unused, default 0; • `rotorn`: Rotation direction, 1 - Clockwise, 2 - Counterclockwise, default 1; • `strategy`: Processing strategy for undetected force/torque, 0 - Error; 1 - Warning, continue motion
Return Value	Error code. Success - 0, Failure - errcode

2.4.12.25 Code Example for Spiral Search, Linear Insertion, and Other Commands

```
from fairino import Robot
import time
robot = Robot.RPC('192.168.58.2')
j1=[-11.904,-99.669,117.473,-108.616,-91.726,74.256]
j2=[-45.615,-106.172,124.296,-107.151,-91.282,74.255]
j3=[-29.777,-84.536,109.275,-114.075,-86.655,74.257]
j4=[-31.154,-95.317,94.276,-88.079,-89.740,74.256]
desc_pos1=[-419.524,-13.000,351.569,-178.118,0.314,3.833]
desc_pos2=[-321.222,185.189,335.520,-179.030,-1.284,-29.869]
```

(continues on next page)

(continued from previous page)

```
desc_pos3=[-487.434,154.362,308.576,176.600,0.268,-14.061]
desc_pos4=[-443.165,147.881,480.951,179.511,-0.775,-15.409]
offset_pos=[0.0]*6
epos=[0.0]*4
tool=0
user=0
vel=100.0
acc=100.0
ovl=100.0
oacc=100.0
blendT=0.0
blendR=0.0
flag=0
search=0
blendMode=0
velAccMode=0
robot.SetSpeed(20)
rtn=robot.MoveJ(joint_pos=j1,tool=tool,user=user,vel=vel,acc=acc,ovl=ovl,exaxis_pos=epos,
↪blendT=blendT,offset_flag=flag,offset_pos=offset_pos)
print(f"movejerrcode:{rtn}")
rtn=robot.MoveL(desc_pos=desc_pos2,tool=tool,user=user,vel=vel,acc=acc,ovl=ovl,
↪blendR=blendR,blendMode=blendMode,exaxis_pos=epos,search=search,offset_flag=flag,
↪offset_pos=offset_pos,oacc=oacc,velAccParamMode=velAccMode)
print(f"movelerrcode:{rtn}")
rtn=robot.MoveC(desc_pos_p=desc_pos3,tool_p=tool,user_p=user,vel_p=vel,acc_p=acc,exaxis_
↪pos_p=epos,offset_flag_p=flag,offset_pos_p=offset_pos,desc_pos_t=desc_pos4,tool_t=tool,
↪user_t=user,vel_t=vel,acc_t=acc,exaxis_pos_t=epos,offset_flag_t=flag,offset_pos_
↪t=offset_pos,ovl=ovl,blendR=blendR,oacc=oacc,velAccParamMode=velAccMode)
print(f"movecerrcode:{rtn}")
rtn=robot.MoveJ(joint_pos=j2,tool=tool,user=user,vel=vel,acc=acc,ovl=ovl,exaxis_pos=epos,
↪blendT=blendT,offset_flag=flag,offset_pos=offset_pos)
print(f"movejerrcode:{rtn}")
rtn=robot.Circle(desc_pos_p=desc_pos3,tool_p=tool,user_p=user,vel_p=vel,acc_p=acc,exaxis_
↪pos_p=epos,desc_pos_t=desc_pos1,tool_t=tool,user_t=user,vel_t=vel,acc_t=acc,exaxis_pos_
↪t=epos,ovl=ovl,offset_flag=flag,offset_pos=offset_pos,oacc=oacc,blendR=-1,
↪velAccParamMode=velAccMode)
print(f"circleerrcode:{rtn}")
rtn=robot.MoveCart(desc_pos=desc_pos4,tool=tool,user=user,vel=vel,acc=acc,ovl=ovl,
↪blendT=blendT,config=-1)
print(f"MoveCarterrcode:{rtn}")
rtn=robot.MoveJ(joint_pos=j1,tool=tool,user=user,vel=vel,acc=acc,ovl=ovl,exaxis_pos=epos,
↪blendT=blendT,offset_flag=flag,offset_pos=offset_pos)
print(f"movejerrcode:{rtn}")
rtn=robot.MoveL(desc_pos=desc_pos2,tool=tool,user=user,vel=vel,acc=acc,ovl=ovl,
↪blendR=blendR,blendMode=blendMode,exaxis_pos=epos,search=search,offset_flag=flag,
↪offset_pos=offset_pos,config=-1,velAccParamMode=velAccMode)
print(f"movelerrcode:{rtn}")
rtn=robot.MoveC(desc_pos_p=desc_pos3,tool_p=tool,user_p=user,vel_p=vel,acc_p=acc,exaxis_
↪pos_p=epos,offset_flag_p=flag,offset_pos_p=offset_pos,desc_pos_t=desc_pos4,tool_t=tool,
↪user_t=user,vel_t=vel,acc_t=acc,exaxis_pos_t=epos,offset_flag_t=flag,offset_pos_
↪t=offset_pos,ovl=ovl,blendR=blendR,config=-1,velAccParamMode=velAccMode)
print(f"movecerrcode:{rtn}")
```

(continues on next page)

(continued from previous page)

```
rtn=robot.MoveJ(joint_pos=j2,tool=tool,user=user,vel=vel,acc=acc,ovl=ovl,exaxis_pos=epos,
↪blendT=blendT,offset_flag=flag,offset_pos=offset_pos)
print(f"movejerrcode:{rtn}")
rtn=robot.Circle(desc_pos_p=desc_pos3,tool_p=tool,user_p=user,vel_p=vel,acc_p=acc,exaxis_
↪pos_p=epos,desc_pos_t=desc_pos1,tool_t=tool,user_t=user,vel_t=vel,acc_t=acc,exaxis_pos_
↪t=epos,ovl=ovl,offset_flag=flag,offset_pos=offset_pos,oacc=oacc,blendR=-1,
↪velAccParamMode=velAccMode)
print(f"circleerrcode:{rtn}")
robot.CloseRPC()
return 0
```

2.4.12.26 Linear insertion

prototype	`FT_LinInsertion(rcs, ft, disMax, linorn, lin_v=1.0, lin_a=1.0)`
Description	Linear Insertion
Mandatory parameters	• `rcs`: reference coordinate system, 0 - tool coordinate system, 1 - base coordinate system; • `ft`: force or moment threshold (0 to 100) in N or Nm. • `disMax`: maximum insertion distance in mm. • `linorn`: insertion direction: 0-negative direction, 1-positive direction
Default parameters	• `lin_v`: linear velocity in mm/s Default 1. • `lin_a`: linear acceleration in mm/s^2, not used yet Default 1
Return Value	Error Code Success-0 Failure- errcode

2.4.12.27 Examples of instruction code for spiral exploration, straight line insertion, etc

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
company = 24
device = 0
softversion = 0
bus = 1
index = 1
robot.FT_SetConfig(company, device, softversion, bus)
time.sleep(1)
error,[company, device, softversion, bus] = robot.FT_GetConfig()
print(f"FT config:{company},{device},{softversion},{bus}")
time.sleep(1)
robot.FT_Activate(0)
time.sleep(1)
robot.FT_Activate(1)
time.sleep(1)
robot.FT_SetZero(0)
```

(continues on next page)

(continued from previous page)

```
time.sleep(1)
status = 1
sensor_num = 1
gain = [0.0001, 0.0, 0.0, 0.0, 0.0, 0.0]
adj_sign = 0
ILC_sign = 0
max_dis = 100.0
max_ang = 5.0
ft = [0.0,0.0,-10.0,0.0,0.0,0.0]
rcs = 0
dr = 0.7
fFinish = 1.0
t = 60000.0
vmax = 3.0
force_goal = 20.0
lin_v = 0.0
lin_a = 0.0
disMax = 100.0
linorn = 1
angVelRot = 2.0
forceInsertion = 1.0
angleMax = 45
orn = 1
angAccmax = 0.0
rotorn = 1
select1 = [0, 0, 1, 1, 1, 0]
robot.FT_Control(status, sensor_num, select1, ft, gain, adj_sign, ILC_sign, max_dis, max_
↪ang, 0, 0, 0)
rtn = robot.FT_SpiralSearch(rcs, dr, fFinish, t, vmax)
print(f"FT_SpiralSearch rtn is {rtn}")
robot.FT_Control(0, sensor_num, select1, ft, gain, adj_sign, ILC_sign, max_dis, max_ang,␣
↪0, 0, 0)
select2 = [1, 1, 1, 0, 0, 0]
gain[0] = 0.00005
ft[2] = -30.0
robot.FT_Control(1, sensor_num, select2, ft, gain, adj_sign, ILC_sign, max_dis, max_ang,␣
↪0, 0, 0)
rtn = robot.FT_LinInsertion(rcs, force_goal, lin_v, lin_a, disMax, linorn)
print(f"FT_LinInsertion rtn is {rtn}")
robot.FT_Control(0, sensor_num, select2, ft, gain, adj_sign, ILC_sign, max_dis, max_ang,␣
↪0, 0, 0)
select3 = [0, 0, 1, 1, 1, 0]
ft[2] = -10.0
gain[0] = 0.0001
robot.FT_Control(1, sensor_num, select3, ft, gain, adj_sign, ILC_sign, max_dis, max_ang,␣
↪0, 0, 0)
rtn = robot.FT_RotInsertion(rcs, angVelRot, forceInsertion, angleMax, orn, angAccmax,␣
↪rotorn)
print(f"FT_RotInsertion rtn is {rtn}")
robot.FT_Control(0, sensor_num, select3, ft, gain, adj_sign, ILC_sign, max_dis, max_ang,␣
↪0, 0, 0)
select4 = [1, 1, 1, 0, 0, 0]
```

(continues on next page)

(continued from previous page)

```
ft[2] = -30.0
robot.FT_Control(1, sensor_num, select4, ft, gain, adj_sign, ILC_sign, max_dis, max_ang,␣
↪0, 0, 0)
rtn = robot.FT_LinInsertion(rcs, force_goal, lin_v, lin_a, disMax, linorn)
print(f"FT_LinInsertion rtn is {rtn}")
robot.FT_Control(0, sensor_num, select4, ft, gain, adj_sign, ILC_sign, max_dis, max_ang,␣
↪0, 0, 0)
robot.CloseRPC()
```

2.4.12.28 Surface positioning

prototype	`FT_FindSurface (rcs, dir, axis, disMax, ft, lin_v=3.0, lin_a=0.0)`
Description	Surface Positioning
Mandatory parameters	• `rcs`: reference coordinate system, 0 - tool coordinate system, 1 - base coordinate system; • `dir`: direction of movement, 1-positive, 2-negative; • `axis`: moving axes, 1-x, 2-y, 3-z; • `disMax`: large exploration distance in mm. • `ft`: action termination force threshold in N;
Default Parameters	• `lin_v`: explore linear velocity in mm/s default 3. • `lin_a`: explore linear acceleration in mm/s^2 default 0;
Return Value	Error Code Success-0 Failure- errcode

2.4.12.29 Calculation of the center plane position begins

Prototype	`FT_CalCenterStart()`
Description	Calculation of mid-plane position begins
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.30 Calculate end of mid-plane position

Prototype	`FT_CalCenterEnd()`
Description	End of calculation of mid-plane position
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode - `pos=[x,y,z,rx,ry,rz]`: mid-plane position

2.4.12.31 Sample code for surface localization

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
company = 24
device = 0
softversion = 0
bus = 1
index = 1
robot.FT_SetConfig(company, device, softversion, bus)
time.sleep(1)
error,[company, device, softversion, bus] = robot.FT_GetConfig()
print(f"FT config:{company},{device},{softversion},{bus}")
time.sleep(1)
robot.FT_Activate(0)
time.sleep(1)
robot.FT_Activate(1)
time.sleep(1)
robot.FT_SetZero(0)
time.sleep(1)
rcs = 0
dir = 1
axis = 1
lin_v = 3.0
lin_a = 0.0
maxdis = 50.0
ft_goal = 2.0
desc_pos = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]
xcenter = [0, 0, 0, 0, 0, 0]
ycenter = [0, 0, 0, 0, 0, 0]
ft = [-2.0, 0.0, 0.0, 0.0, 0.0, 0.0]
robot.MoveCart(desc_pos, 9, 0, 100.0)
robot.FT_CalCenterStart()
robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal)
robot.MoveCart(desc_pos, 9, 0)
robot.WaitMs(1000)
dir = 2
robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal)
error,xcenter = robot.FT_CalCenterEnd()
print(f"xcenter:{xcenter[0]},{xcenter[1]},{xcenter[2]},{xcenter[3]},{xcenter[4]},{xcenter[5]}")
robot.MoveCart(xcenter, 9, 0, 60.0)
robot.FT_CalCenterStart()
dir = 1
axis = 2
lin_v = 6.0
maxdis = 150.0
robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal)
robot.MoveCart(desc_pos, 9, 0, 100.0)
robot.WaitMs(1000)
```

(continues on next page)

(continued from previous page)

```
dir = 2
robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal)
error,ycenter = robot.FT_CalCenterEnd()
print(f"ycenter:{ycenter[0]},{ycenter[1]},{ycenter[2]},{ycenter[3]},{ycenter[4]},
↪{ycenter[5]}")
robot.MoveCart(ycenter, 9, 0, 60.0)
robot.CloseRPC()
```

2.4.12.32 Soft control on

Prototype	`FT_ComplianceStart(p, force)`
Description	Softening control on
Mandatory parameters	• p: Position adjustment factor or softness factor • `force`: soft opening force threshold in N
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.33 Soft control off

Prototype	`FT_ComplianceStop()`
Description	Softening control off
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.34 Soft control code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
company = 24
device = 0
softversion = 0
bus = 1
index = 1
robot.FT_SetConfig(company, device, softversion, bus)
time.sleep(1)
error,[company, device, softversion, bus] = robot.FT_GetConfig()
print(f"FT config:{company},{device},{softversion},{bus}")
time.sleep(1)
robot.FT_Activate(0)
time.sleep(1)
```

(continues on next page)

(continued from previous page)

```
robot.FT_Activate(1)
time.sleep(1)
robot.FT_SetZero(0)
time.sleep(1)
flag = 1
sensor_id = 1
select = [1, 1, 1, 0, 0, 0]
ft_pid = [0.0005, 0.0, 0.0, 0.0, 0.0, 0.0]
adj_sign = 0
ILC_sign = 0
max_dis = 100.0
max_ang = 0.0
ft = [-10.0, -10.0, -10.0, 0.0, 0.0, 0.0]
offset_pos = [0.0,0.0,0.0,0.0,0.0,0.0]  #  DescPose(0, 0, 0, 0, 0, 0)
epos = [0.0,0.0,0.0,0.0]  #  ExaxisPos(0, 0, 0, 0)
j1 = [-11.904, -99.669, 117.473, -108.616, -91.726, 74.256]  # JointPos
j2 = [-45.615, -106.172, 124.296, -107.151, -91.282, 74.255]
desc_p1 = [-419.524, -13.000, 351.569, -178.118, 0.314, 3.833]  # DescPose
desc_p2 = [-321.222, 185.189, 335.520, -179.030, -1.284, -29.869]
robot.FT_Control(flag, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis, max_
↪ang, 0, 0, 0)
p = 0.00005
force = 30.0
rtn = robot.FT_ComplianceStart(p, force)
print(f"FT_ComplianceStart rtn is {rtn}")
count = 3
while count > 0:
    robot.MoveL(desc_pos=desc_p1,tool= 0,user= 0,vel= 100.0)
    robot.MoveL(desc_pos=desc_p2,tool= 0,user= 0,vel= 100.0)
    count -= 1
robot.FT_ComplianceStop()
print(f"FT_ComplianceStop rtn is {rtn}")
flag = 0
robot.FT_Control(flag, sensor_id, select, ft, ft_pid, adj_sign, ILC_sign, max_dis, max_
↪ang, 0, 0, 0)
robot.CloseRPC()
```

2.4.12.35 Load recognition filter initialization

New in version python: SDK-v2.0.1

Prototype	`LoadIdentifyDynFilterInit()`
Description	Load Recognition Filter Initialization
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.36 Initialization of load recognition variables

New in version python: SDK-v2.0.1

Prototype	`LoadIdentifyDynVarInit()`
Description	Load Identification Variable Initialization
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.37 Load Recognition Main Program

New in version python: SDK-v2.0.1

Prototype	`LoadIdentifyMain(joint_torque, joint_pos, t)`
Description	Load Recognition Master Program
Mandatory parameters	• `joint_torque`: Joint torque j1-j6; • `joint_pos`: joint position j1-j6 • `t`: sampling period
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.38 Getting Load Recognition Results

New in version python: SDK-v2.0.1

Prototype	`LoadIdentifyGetResult(gain)`
Description	Get load identification results
Mandatory parameters	• `gain`: gravity term coefficient double[6], centrifugal term coefficient double[6]
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `weight`: weight of the load • `cog=[x,y,z]`: load center of mass coordinates

2.4.12.39 Robot load identification code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
retval = robot.LoadIdentifyDynFilterInit()
print(f"LoadIdentifyDynFilterInit retval is: {retval}")
```

(continues on next page)

(continued from previous page)

```
retval = robot.LoadIdentifyDynVarInit()
print(f"LoadIdentifyDynVarInit retval is: {retval}")
error, posJ = robot.GetActualJointPosDegree(0)
posJ[1] += 10  # Modify joint 2
error, joint_toq = robot.GetJointTorques(0)
joint_toq[1] += 2  # Modify torque on joint 2
tmpTorque = joint_toq.copy()
retval = robot.LoadIdentifyMain(tmpTorque, posJ, 1)
print(f"LoadIdentifyMain retval is: {retval}")
gain = [0, 0.05, 0, 0, 0, 0, 0, 0.02, 0, 0, 0, 0]
weight = [0.0]
load_pos = [0.0, 0.0, 0.0]
retval, weight, load_pos = robot.LoadIdentifyGetResult(gain)
print(f"LoadIdentifyGetResult retval is: {retval} ; weight is {weight}  cog is {load_
↪pos[0]} {load_pos[1]} {load_pos[2]}")
robot.CloseRPC()
```

2.4.12.40 Force Sensor Assisted Drag

New in version python: SDK-v2.1.3

Prototype	`EndForceDragControl(status, asaptiveFlag, interfereDragFlag, ingularityConstraintsFlag, M, B, K, F, Fmax, Vmax, forceCollisionFlag=0)`
Description	Force Sensor Assisted Drag
Mandatory parameters	• `status`: control status, 0 - off; 1 - on • `asaptiveFlag`: adaptive on flag, 0 - off; 1 - on • `interfereDragFlag`: interference area drag flag, 0 - off; 1 - on • `ingularityConstraintsFlag`: singularity strategy, 0- evade; 1- Crossing • `forceCollisionFlag`: Robot collision detection mark during assisted dragging 0- Close 1- Enable • `M=[m1,m2,m3,m4,m5,m6]`: inertia factor • `B=[b1,b2,b3,b4,b5,b6]`: damping factor • `K=[k1,k2,k3,k4,k5,k6]`: stiffness factor • `F=[f1,f2,f3,f4,f5,f6]`: drag six-dimensional force thresholds • `Fmax`: Maximum towing force limitation • `Vmax`: maximum joint speed limit
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.41 Get force sensor drag switch status

New in version python: SDK-v2.0.5

Prototype	`GetForceAndTorqueDragState()`
Description	Get force sensor drag switch status
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `dragState`: force sensor assisted drag control state, 0 - off; 1 - on • `sixDimensionalDragState`: six-dimensional force-assisted drag control state, 0-off; 1-on

2.4.12.42 The force sensor turns on automatically after the error is cleared.

New in version python: SDK-v2.0.5

Prototype	`SetForceSensorDragAutoFlag(status)`
Description	The force sensor turns on automatically after an error is cleared.
Mandatory parameters	• `status`: control status, 0 - off; 1 - on
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.43 Force sensor assisted drag code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
robot.SetForceSensorDragAutoFlag(1)
M = [15.0, 15.0, 15.0, 0.5, 0.5, 0.1]
B = [150.0, 150.0, 150.0, 5.0, 5.0, 1.0]
K = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
F = [10.0, 10.0, 10.0, 1.0, 1.0, 1.0]
robot.EndForceDragControl(1, 0, 0, 0, M, B, K, F, 50, 100)
time.sleep(5)
drag_state = 0
six_dimensional_drag_state = 0
error,drag_state, six_dimensional_drag_state = robot.GetForceAndTorqueDragState()
print(f"the drag state is {drag_state} {six_dimensional_drag_state}")
robot.EndForceDragControl(0, 0, 0, 0, M, B, K, F, 50, 100)
robot.CloseRPC()
```

2.4.12.44 Setting up hybrid drag switches and parameters for six-dimensional force and joint impedance

New in version python: SDK-v2.0.5

Prototype	`ForceAndJointImpedanceStartStop(status, impedanceFlag, lamdeDain, KGain, BGain, dragMaxTcpVel, dragMaxTcpOriVel)`
Description	Setting up the six-dimensional force and joint impedance hybrid drag switch and parameters
Mandatory parameters	• `status`: control status, 0 - off; 1 - on • `impedanceFlag`: impedance on flag, 0 - off; 1 - on • `lamdeDain`: [D1, D2, D3, D4, D5, D6] drag gain • `KGain`: [K1,K2,K3,K4,K5,K6] Stiffness Gain • `BGain`: [B1,B2,B3,B4,B5,B] damping gain • `dragMaxTcpVel`: maximum line speed limit at the end of a drag • `dragMaxTcpOriVel`: drag end maximum angular velocity limit
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.45 Six dimensional force and joint impedance mixed drag code example

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
robot.DragTeachSwitch(1)
lamde_dain = [3.0, 2.0, 2.0, 2.0, 2.0, 3.0]
k_gain = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
b_gain = [150.0, 150.0, 150.0, 5.0, 5.0, 1.0]
rtn = robot.ForceAndJointImpedanceStartStop(1, 0, lamde_dain, k_gain, b_gain, 1000, 180)
print(f"ForceAndJointImpedanceStartStop rtn is {rtn}")
time.sleep(5)
robot.DragTeachSwitch(0)
rtn = robot.ForceAndJointImpedanceStartStop(0, 0, lamde_dain, k_gain, b_gain, 1000, 180)
print(f"ForceAndJointImpedanceStartStop rtn is {rtn}")
robot.CloseRPC()
```

2.4.12.46 Impedance start and stop control

New in version python: SDK-v2.1.6

Prototype	`ImpedanceControlStartStop(status, workSpace, forceThreshold, m, b, k, maxV, maxVA, maxW, maxWA)`
Description	Impedance start and stop control
Mandatory parameters	• `status`0- closed; 1- On • `workSpace`0-joint space; 1-Dekal space • `forceThreshold`Trigger force threshold (N) • mQuality parameter • bParameter of damping • kParameter of stiffness • `maxV`Maximum linear velocity (mm/s) • `maxVA`Maximum linear acceleration (mm/s2) • `maxW`Maximum angular velocity (°/s) • `maxWA`Maximum angular acceleration (°/s2)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.12.47 Example code for impedance start and stop control

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
j1 = [102.622, -135.990, 120.769, -73.950, -90.848, 35.507]
j2 = [93.674, -80.062, 82.947, -92.199, -90.967, 26.559]
desc_pos1 = [136.552, -149.799, 449.532, 179.817, -1.172, 157.123]
desc_pos2 = [136.540, -561.048, 449.542, 179.819, -1.172, 157.122]
offset_pos = [0.0] * 6
epos = [0.0] * 4
tool = 0
user = 0
vel = 100.0
acc = 200.0
ovl = 100.0
blendT = -1.0
blendR = -1.0
flag = 0
search = 0
robot.SetSpeed(20)
company = 17
device = 0
softversion = 0
bus = 1
robot.FT_SetConfig(company, device, softversion, bus)
time.sleep(1)
rnt,[company, device, softversion, bus] = robot.FT_GetConfig()
print(f"FT config:{company},{device},{softversion},{bus}")
```

(continues on next page)

(continued from previous page)

```
time.sleep(1)
robot.FT_Activate(0)
time.sleep(1)
robot.FT_Activate(1)
time.sleep(1)
time.sleep(1)
robot.FT_SetZero(0)
time.sleep(1)
robot.FT_SetZero(1)
time.sleep(1)
forceThreshold = [30.0, 30.0, 30.0, 5.0, 5.0, 5.0]
m = [0.1, 0.1, 0.1, 0.02, 0.02, 0.02]
b = [1.0, 1.0, 1.0, 0.08, 0.08, 0.08]
k = [0.0] * 6
rtn = robot.ImpedanceControlStartStop(1, 1, forceThreshold, m, b, k, 1000, 500, 100, 100)
print(f"ImpedanceControlStartStop errcode:{rtn}")
rtn = robot.MoveL(desc_pos=desc_pos1,tool= tool,user= user,vel= vel,speedPercent=1)
rtn = robot.MoveL(desc_pos=desc_pos2,tool= tool,user= user,vel= vel,speedPercent=1)
rtn = robot.MoveL(desc_pos=desc_pos1,tool= tool,user= user,vel= vel,speedPercent=1)
rtn = robot.MoveL(desc_pos=desc_pos2,tool= tool,user= user,vel= vel,speedPercent=1)
rtn = robot.MoveL(desc_pos=desc_pos1,tool= tool,user= user,vel= vel,speedPercent=1)
rtn = robot.MoveL(desc_pos=desc_pos2,tool= tool,user= user,vel= vel,speedPercent=1)
print(f"movel errcode:{rtn}")
robot.ImpedanceControlStartStop(0, 1, forceThreshold, m, b, k, 1000, 500, 100, 100)
robot.CloseRPC()
```

2.4.12.48 Enable Torque Compensation Function and Compensation Coefficients

Prototype	`SerCoderCompenParams(status, torqueCoeff)`
Description	Enable Torque Compensation Function and Compensation Coefficients
Required Parameters	• `status`Switch, 0-Disable; 1-Enable • `torqueCoeff`J1-J6 Torque compensation coefficients [0-1]
Default Parameters	None
Return Value	Error code 0-Success Error- errcode

2.4.13 Extended Axis

2.4.13.1 Setting the 485 Extended Axis Parameters

New in version python: SDK-v2.0.3

Prototype	`AuxServoSetParam(servoId,servoCompany,servoModel, servoSoftVersion, servoResolution,axisMechTransRatio)`
Description	Setting the 485 extended axis parameters
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponds to slave ID; • `servoCompany`: servo drive manufacturer, 1 - Dynatec; • `servoModel`: Servo drive model, 1-FD100-750C; • `servoSoftVersion`: servo drive software version, 1-V1.0; • `servoResolution`: encoder resolution; • `axisMechTransRatio`: mechanical transmission ratio;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.2 Getting 485 Expansion Axis Configuration Parameters

New in version python: SDK-v2.0.3

Prototype	`AuxServoGetParam(servoId)`
Description	Get 485 extended axis configuration parameters
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponding to slave ID;
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode. • `servoCompany`: servo drive manufacturer, 1 - Dynatec; • `servoModel`: servo drive model, 1-FD100-750C; • `servoSoftVersion`: servo drive software version, 1-V1.0; • `servoResolution`: encoder resolution; • `axisMechTransRatio`: mechanical transmission ratio;

2.4.13.3 Setting the 485 expansion axis enable/disable

New in version python: SDK-v2.0.3

Prototype	`AuxServoEnable(servoId,status)`
Description	Setting the 485 extension axis to enable/de-enable
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponds to slave ID; • `status`: enabling status, 0-de-enabling, 1-enabling;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.4 Setting the 485 Extended Axis Control Mode

New in version python: SDK-v2.0.3

Prototype	`AuxServoSetControlMode(servoId,mode)`
Description	Setting the 485 Extended Axis Control Mode
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponds to slave ID; • `mode`: control mode, 0-position mode, 1-velocity mode;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.5 Setting the 485 extended axis target position (position mode)

New in version python: SDK-v2.0.3

Prototype	`AuxServoSetTargetPos(servoId,pos,speed)`
Description	Setting the 485 extended axis target position (position mode)
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponds to slave ID; • `pos`: target position, mm or °; • `speed`: target speed, mm/s or °/s;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.6 Setting the 485 extended axis target torque (torque mode)-not yet available

New in version python: SDK-v2.0.3

prototype	`AuxServoSetTargetTorque(servoId,torque)`
Description	Setting the 485 extended axis target torque (torque mode)
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponds to slave ID; • `torque`: target moment, Nm;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.7 Setting the 485 extended axis back to zero

New in version python: SDK-v2.0.3

Prototype	`AuxServoHoming(servoId,mode,searchVel,latchVel)`
Description	Setting the 485 extension axis back to zero
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponds to slave ID; • `mode`: return to zero mode, 1 - return to zero at current position; 2 - return to zero at negative limit; 3 - return to zero at positive limit. • `searchVel`: return-to-zero velocity, mm/s or °/s. • `latchVel`: hoop speed, mm/s or °/s;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.8 Clearing 485 Expansion Axis Error Messages

New in version python: SDK-v2.0.3

Prototype	`AuxServoClearError(servoId)`
Description	Clearing 485 Extended Axis Error Message
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponding to slave ID;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.9 Get 485 extended axis servo status

New in version python: SDK-v2.0.3

prototype	`AuxServoGetStatus(servoId)`
Description	Get 485 extended axis servo status
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponding to slave ID;
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode. • `servoErrCode`: servo drive error code • `servoState`: servo drive state bit0:0-not enabled; 1-enabled; bit1:0-not in motion; 1-being in motion; bit2 0-positive limit not triggered; 1-positive limit triggered; bit3 0-negative limit not triggered; 1-negative limit triggered; bit4 0-positioning not completed; 1-positioning complete; bit5: 0-not zero return; 1-zero return complete; • `servoPos`: servo current position mm or °; • `servoSpeed`: servo current speed mm/s or °/s; • `servoTorque`: servo current torque Nm;

2.4.13.10 Setting the 485 extended axis target speed (velocity mode)

New in version python: SDK-v2.0.3

Prototype	`AuxServoSetTargetSpeed(servoId,speed)`
Description	Setting the 485 extended axis target speed (velocity mode)
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponds to slave ID; • `speed`: target speed, mm/s or °/s;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.11 Setting the 485 extended axis data axis number in the status feedback

New in version python: SDK-v2.0.3

Prototype	`AuxServosetStatusID(servoId)`
Description	Sets the 485 extended axis data axis number in status feedback
Mandatory parameters	• `servoId`: servo drive ID, range [1-15], corresponding to slave ID;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.12 Setting the 485 Extended Axis Motion Acceleration and Deceleration Speed

New in version python: SDK-v2.0.5

Prototype	`AuxServoSetAcc(acc, dec)`
Description	Sets the 485 extended axis motion acceleration and deceleration speeds.
Mandatory parameters	• `acc`: 485 extended axis motion acceleration • `dec`: 485 extended axis motion deceleration
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.13 Setting the 485 extended axis emergency stop acceleration and deceleration speeds

New in version python: SDK-v2.0.5

Prototype	`AuxServoSetEmergencyStopAcc(acc, dec)`
Description	Setting the 485 extended axis emergency stop acceleration and deceleration speed
Mandatory parameters	• `acc`: 485 extended axis emergency stop acceleration • `dec`: 485 extended axis emergency stop deceleration
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.14 Get 485 Extended Axis Motion Acceleration and Deceleration

New in version python: SDK-v2.0.5

Prototype	`AuxServoGetAcc()`
Description	Get 485 extended axis motion plus or minus velocity
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • acc: 485 extended axis motion acceleration • dec: 485 extended axis motion deceleration

2.4.13.15 Get 485 extended axis emergency stop acceleration and deceleration speeds

New in version python: SDK-v2.0.5

Prototype	`AuxServoGetEmergencyStopAcc()`
Description	Get 485 extended axis emergency stop acceleration and deceleration speed
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • acc: 485 extended axis emergency stop acceleration • dec: 485 extended axis emergency stop deceleration

2.4.13.16 Extended axis control code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
retval = robot.AuxServoSetParam(1, 1, 1, 1, 131072, 15.45)
print(f"AuxServoSetParam is: {retval}")
servoCompany = 0
servoModel = 0
servoSoftVersion = 0
servoResolution = 0
axisMechTransRatio = 0.0
retval, servoCompany, servoModel, servoSoftVersion, servoResolution, axisMechTransRatio = robot.AuxServoGetParam(1)
print(f"servoCompany {servoCompany}\n"
      f"servoModel {servoModel}\n"
      f"servoSoftVersion {servoSoftVersion}\n"
      f"servoResolution {servoResolution}\n"
      f"axisMechTransRatio {axisMechTransRatio}\n")
retval = robot.AuxServoSetParam(1, 10, 11, 12, 13, 14)
```

(continues on next page)

(continued from previous page)

```
print(f"AuxServoSetParam is: {retval}")
retval, servoCompany, servoModel, servoSoftVersion, servoResolution, axisMechTransRatio␣
↪= robot.AuxServoGetParam(1)
print(f"servoCompany {servoCompany}\n"
      f"servoModel {servoModel}\n"
      f"servoSoftVersion {servoSoftVersion}\n"
      f"servoResolution {servoResolution}\n"
      f"axisMechTransRatio {axisMechTransRatio}\n")
retval = robot.AuxServoSetParam(1, 1, 1, 1, 131072, 36)
print(f"AuxServoSetParam is: {retval}")
time.sleep(3)
robot.AuxServoSetAcc(3000, 3000)
robot.AuxServoSetEmergencyStopAcc(5000, 5000)
time.sleep(1)
emagacc = 0.0
emagdec = 0.0
acc = 0.0
dec = 0.0
error,emagacc, emagdec = robot.AuxServoGetEmergencyStopAcc()
print(f"emergency acc is {emagacc}  dec is {emagdec}")
error,acc, dec = robot.AuxServoGetAcc()
print(f"acc is {acc}  dec is {dec}")
robot.AuxServoSetControlMode(1, 0)
time.sleep(2)
retval = robot.AuxServoEnable(1, 0)
print(f"AuxServoEnable disenable {retval}")
time.sleep(1)
servoErrCode = 0
servoState = 0
servoPos = 0.0
servoSpeed = 0.0
servoTorque = 0.0
retval, servoErrCode, servoState, servoPos, servoSpeed, servoTorque = robot.
↪AuxServoGetStatus(1)
print(f"AuxServoGetStatus servoState {servoState}")
time.sleep(1)
retval = robot.AuxServoEnable(1, 1)
print(f"AuxServoEnable enable {retval}")
time.sleep(1)
retval, servoErrCode, servoState, servoPos, servoSpeed, servoTorque = robot.
↪AuxServoGetStatus(1)
print(f"AuxServoGetStatus servoState {servoState}")
time.sleep(1)
retval = robot.AuxServoHoming(1, 1, 5, 1,100)
print(f"AuxServoHoming {retval}")
time.sleep(3)
retval = robot.AuxServoSetTargetPos(1, 200, 30,100)
print(f"AuxServoSetTargetPos {retval}")
time.sleep(1)
retval, servoErrCode, servoState, servoPos, servoSpeed, servoTorque = robot.
↪AuxServoGetStatus(1)
print(f"AuxServoGetStatus servoSpeed {servoSpeed}")
```

(continues on next page)

(continued from previous page)

```
time.sleep(8)
robot.AuxServoSetControlMode(1, 1)
time.sleep(2)
robot.AuxServoEnable(1, 0)
time.sleep(1)
robot.AuxServoEnable(1, 1)
time.sleep(1)
robot.AuxServoSetTargetSpeed(1, 100, 80)
time.sleep(5)
robot.AuxServoSetTargetSpeed(1, 0, 80)
robot.CloseRPC()
```

2.4.13.17 Parameter configuration for UDP extended axis communication

New in version python: SDK-v2.1.2

prototype	`ExtDevSetUDPComParam(ip, port, period, lossPkgTime, lossPkgNum, disconnectTime, reconnectEnable, reconnectPeriod, reconnectNum, selfConnect)`
Description	UDP Extended Axis Communication Parameter Configuration
Mandatory parameters	• `ip`: PLC IP address; • `port`: port number; • `period`: communication period (ms, not open yet); • `lossPkgTime`: packet loss detection time (ms); • `lossPkgNum`: number of packets lost; • `disconnectTime`: the length of the communication disconnect confirmation; • `reconnectEnable`: communication disconnection auto reconnect enable 0-not enable 1-enable; • `reconnectPeriod`: reconnect period interval (ms); • `reconnectNum`: number of reconnections • `selfConnect`: Whether the connection is automatically established upon power-off and restart; 0- No connection established; 1- Establish a connection
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.18 Get UDP extended axis communication parameters

New in version python: SDK-v2.0.4

Prototype	`ExtDevGetUDPComParam()`
Description	Get UDP extended axis communication parameters
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode; • `ip`: PLC IP address; • `port`: port number; • `period`: communication period (ms, not open yet); • `lossPkgTime`: packet loss detection time (ms); • `lossPkgNum`: number of packets lost; • `disconnectTime`: the length of the communication disconnect confirmation; • `reconnectEnable`: communication disconnection auto reconnect enable 0-not enable 1-enable; • `reconnectPeriod`: reconnect period interval (ms); • `reconnectNum`: number of reconnections

2.4.13.19 Load UDP communication

New in version python: SDK-v2.0.4

Prototype	`ExtDevLoadUDPDriver()`
Description	Load UDP communication
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.20 Offloading UDP communication

New in version python: SDK-v2.0.4

Prototype	`ExtDevUnloadUDPDriver()`
Description	Offloading UDP communication
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.21 UDP Extended Axis Communication Recovery after Abnormal Disconnection

New in version python: SDK-v2.0.4

prototype	`ExtDevUDPClientComReset()`
Description	UDP Extended Axis Communication Abnormally Disconnected After Restoring Connection
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.22 UDP extension axis communication is closed after abnormal disconnection.

New in version python: SDK-v2.0.4

Prototype	`ExtDevUDPClientComClose()`
Description	UDP Extended Axis Communication Abnormal Disconnect Closes Communication
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.23 UDP Extended Axis Parameter Configuration

New in version python: SDK-v2.0.4

Prototype	`ExtAxisParamConfig(axisId, axisType, axisDirection, axisMax, axisMin, axisVel, axisAcc,axisLead, encResolution, axisOffect. axisCompany, axisModel, axisEncType)`
Description	UDP Extended Axis Parameter Configuration
Mandatory parameters	• `axisId`: axis number [1-4]; • `axisType`: extended axis type 0 - translation; 1 - rotation; • `axisDirection`: extended axis direction 0 - forward; 1 - reverse; • `axisMax`: Maximum position of the extended axis in mm; • `axisMin`: Extended axis minimum position in mm; • `axisVel`: speed mm/s; • `axisAcc`: acceleration mm/s2; • `axisLead`: lead in mm; • `encResolution`: encoder resolution; • `axisOffect`: Extended axis offset from the start of the weld; • `axisCompany`: drive manufacturers 1-Hochuan; 2-HuiChuan; 3-Panasonic; • `axisModel`: Drive Model 1-Hochuan-SV-XD3EA040L-E, 2-Hochuan-SV-X2EA150A-A, 1-HuiChuan-SV620PT5R4I, 1-Panasonic-MADLN15SG, 2-Panasonic-MSDLN25SG, 3-Panasonic-MCDLN35SG; • `axisEncType`: encoder type 0-incremental; 1-absolute;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.24 Setting the extended robot position relative to the extended axis

New in version python: SDK-v2.0.4

Prototype	`SetRobotPosToAxis(installType)`
Description	Set the position of the extended robot relative to the extended axis
Mandatory parameters	• `installType`: 0 - robot mounted on external axis, 1 - robot mounted outside external axis;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.25 Setting the extended axis system DH parameter configuration

New in version python: SDK-v2.0.4

Prototype	`SetAxisDHParaConfig(axisConfig,axisDHd1,axisDHd2,axisDHd3, axisDHd4,axisDHa1,axisDHa2,axisDHa3,axisDHa4)`
Description	Sets the extended axis system DH parameter configuration
Mandatory parameters	• `axisConfig`: external axis configuration, 0 - single degree of freedom linear slide, 1 - two degree of freedom L-type indexer, 2 - three degree of freedom, 3 - four degree of freedom, 4 - single degree of freedom indexer; • `axisDHd1`: external axis DH parameter d1 mm; • `axisDHd2`: external axis DH parameter d2 mm; • `axisDHd3`: external axis DH parameter d3 mm; • `axisDHd4`: external axis DH parameter d4 mm; • `axisDHa1`: external axis DH parameter a1 mm; • `axisDHa2`: external axis DH parameter a2 mm; • `axisDHa3`: external axis DH parameter a3 mm; • `axisDHa4`: external axis DH parameter a4 mm;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.26 UDP Extended Axis Enable

New in version python: SDK-v2.0.4

Prototype	`ExtAxisServoOn(axisID, status)`
description	UDP Extended Axis Enable
Mandatory parameters	• `axisID`: axis number [1-4]; • `status`: 0-de-enable; 1-enable;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.27 UDP Extended Axis Zero Return

New in version python: SDK-v2.0.4

Prototype	`ExtAxisSetHoming(axisID, mode, searchVel, latchVel)`
Description	UDP extended axis back to zero
Mandatory parameters	• `axisID`: axis number [1-4]; • `mode`: return to zero mode 0 current position return to zero, 1 negative limit return to zero, 2 - positive limit return to zero; • `searchVel`: search velocity (mm/s); • `latchVel`: zeroing hoop speed (mm/s);
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.28 UDP Extended Axis Tap Start

New in version python: SDK-v2.0.4

Prototype	`ExtAxisStartJog( axisID, direction, vel, acc, maxDistance)`
Description	UDP Extended Axis Tap Start
Mandatory parameters	• `axisID`: axis number [1-4]; • `direction`: direction of rotation 0 - reverse; 1 - forward; • `vel`: velocity (mm/s); • `acc`: acceleration (mm/s); • `maxDistance`: maximum pointing distance;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.29 UDP Extended Axis Tap Stop

New in version python: SDK-v2.0.4

Prototype	`ExtAxisStopJog(axisID)`
Description	UDP Extended Axis Tap Stop
Mandatory parameters	• `axisID`: axis number [1-4];
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.30 Example of UDP extension axis configuration and tapping code

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
rtn = robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100, 5, 1)
print(f"ExtDevSetUDPComParam rtn is {rtn}")
ip = ""
port = 0
period = 0
lossPkgTime = 0
lossPkgNum = 0
disconnectTime = 0
reconnectEnable = 0
reconnectPeriod = 0
reconnectNum = 0
rtn,[ip, port, period, lossPkgTime, lossPkgNum,disconnectTime, reconnectEnable, reconnectPeriod, reconnectNum] = robot.ExtDevGetUDPComParam()
param_str = (f"\nip {ip}\nport {port}\nperiod {period}\nlossPkgTime {lossPkgTime}"
             f"\nlossPkgNum {lossPkgNum}\ndisConntime {disconnectTime}"
             f"\nreconnecable {reconnectEnable}\nreconnperiod {reconnectPeriod}"
             f"\nreconnnun {reconnectNum}")
print(f"ExtDevGetUDPComParam rtn is {rtn}{param_str}")
robot.ExtDevLoadUDPDriver()
rtn = robot.ExtAxisServoOn(1, 1)
print(f"ExtAxisServoOn axis id 1 rtn is {rtn}")
rtn = robot.ExtAxisServoOn(2, 1)
print(f"ExtAxisServoOn axis id 2 rtn is {rtn}")
time.sleep(2)
robot.ExtAxisSetHoming(1, 0, 10, 2)
time.sleep(2)
rtn = robot.ExtAxisSetHoming(2, 0, 10, 2)
print(f"ExtAxisSetHoming rtn is {rtn}")
time.sleep(4)
rtn = robot.SetRobotPosToAxis(1)
print(f"SetRobotPosToAxis rtn is {rtn}")
rtn = robot.SetAxisDHParaConfig(10, 20, 0, 0, 0, 0, 0, 0, 0)
print(f"SetAxisDHParaConfig rtn is {rtn}")
rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905, 262144, 200, 1, 0, 0)
print(f"ExtAxisParamConfig axis 1 rtn is {rtn}")
rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444, 262144, 200, 1, 0, 0)
print(f"ExtAxisParamConfig axis 2 rtn is {rtn}")
time.sleep(3)
robot.ExtAxisStartJog(1, 0, 10, 10, 30)
time.sleep(1)
robot.ExtAxisStopJog(1)
time.sleep(3)
robot.ExtAxisServoOn(1, 0)
time.sleep(3)
```

(continues on next page)

(continued from previous page)

```
robot.ExtAxisStartJog(2, 0, 10, 10, 30)
time.sleep(1)
robot.ExtAxisStopJog(2)
time.sleep(3)
robot.ExtAxisServoOn(2, 0)
robot.ExtDevUnloadUDPDriver()
robot.CloseRPC()
```

2.4.13.31 Setting the reference point of the extended axis coordinate system - four-point method

New in version python: SDK-v2.0.4

Prototype	`ExtAxisSetRefPoint(pointNum)`
Description	Setting the reference point of the extended axis coordinate system - four-point method
Mandatory parameters	• `pointNum`: point number [1-4];
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.32 Calculating the Extended Axis Coordinate System - Four Point Method

New in version python: SDK-v2.0.4

Prototype	`ExtAxisComputeECoordSys()`
Description	Calculating Extended Axis Coordinate Systems - Four Point Method
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode. • `coord`: extended axis coordinate system values [x,y,z,rx,ry,rz];

2.4.13.33 Reference Point Setting for the Shifter Coordinate System - Four-Point Method

New in version python: SDK-v2.0.4

Prototype	`PositionorSetRefPoint(pointNum)`
Description	Reference Point Setting for the Variable Position Machine Coordinate System - Four Point Method
Mandatory parameters	• `pointNum`: point number [1-4];
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.34 Shifter Coordinate System Calculation - Four Point Method

New in version python: SDK-v2.0.4

Prototype	`PositionorComputeECoordSys()`
Description	Translator Coordinate System Calculation - Four Point Method
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode. • `coord`: the value of the coordinate system of the translocator [x,y,z,rx,ry,rz];

2.4.13.35 Setting of the calibration reference point in the position in the coordinate system of the end of the translator

New in version python: SDK-v2.0.4

prototype	`SetRefPointInExAxisEnd(pos)`
Description	Set the calibration reference point to be positioned in the coordinate system of the end of the variator
Mandatory parameters	• `pos`: bit position values [x,y,z,rx,ry,rz];
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.36 Applying the Extended Axis Coordinate System

New in version python: SDK-v2.0.4

Prototype	`ExtAxisActiveECoordSys(applyAxisId,axisCoordNum,coord,calibFlag)`
Description	Apply extended axis coordinate system
Mandatory parameters	• `applyAxisId`:Extended Axis Numbering bit0-bit3 corresponds to extended axis numbering 1-4, e.g., if you apply extended axes 1 and 3, it would be 0b 0000 0101, or 5; • `axisCoordNum`: extended axis coordinate system number; • `coord`: coordinate system value [x,y,z,rx,ry,rz]; • `calibFlag`: calibration flag 0 - no, 1 - yes;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.37 Obtain the extended axis coordinate system

New in version python: SDK-v2.1.2

Prototype	`ExtAxisGetCoord()`
Description	Obtain the extended axis coordinate system
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • coordExtended axis coordinate system

2.4.13.38 Extended axis coordinate system calibration code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
rtn = robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 200, 1, 100, 5, 1)
print(f"ExtDevSetUDPComParam rtn is {rtn}")
rtn,udp_params = robot.ExtDevGetUDPComParam()
ip, port, period, lossPkgTime, lossPkgNum, disconnectTime, reconnectEnable, reconnectPeriod, reconnectNum = udp_params
patam = (
    f"\nip {ip}\nport {port}\nperiod {period}\nlossPkgTime {lossPkgTime}\n"
    f"lossPkgNum {lossPkgNum}\ndisConntime {disconnectTime}\nreconnecable {reconnectEnable}\n"
    f"reconnperiod {reconnectPeriod}\nreconnnun {reconnectNum}"
)
print(f"ExtDevGetUDPComParam rtn is {rtn}{patam}")
robot.ExtDevLoadUDPDriver()
rtn = robot.ExtAxisServoOn(1, 1)
print(f"ExtAxisServoOn axis id 1 rtn is {rtn}")
rtn = robot.ExtAxisServoOn(2, 1)
print(f"ExtAxisServoOn axis id 2 rtn is {rtn}")
time.sleep(2)
robot.ExtAxisSetHoming(1, 0, 10, 2)
time.sleep(2)
rtn = robot.ExtAxisSetHoming(2, 0, 10, 2)
print(f"ExtAxisSetHoming rtn is {rtn}")
time.sleep(4)
rtn = robot.SetRobotPosToAxis(1)
print(f"SetRobotPosToAxis rtn is {rtn}")
rtn = robot.SetAxisDHParaConfig(1, 128.5, 206.4, 0, 0, 0, 0, 0, 0)
print(f"SetAxisDHParaConfig rtn is {rtn}")
rtn = robot.ExtAxisParamConfig(1, 1, 1, 1000, -1000, 1000, 1000, 1.905, 262144, 200, 1, 0, 0)
print(f"ExtAxisParamConfig axis 1 rtn is {rtn}")
rtn = robot.ExtAxisParamConfig(2, 1, 1, 1000, -1000, 1000, 1000, 4.444, 262144, 200, 1, 0, 0)
```

(continues on next page)

(continued from previous page)

```
print(f"ExtAxisParamConfig axis 2 rtn is {rtn}")
toolCoord = [0, 0, 210, 0, 0, 0]
robot.SetToolCoord(1, toolCoord, 0, 0, 1, 0)
jSafe = [115.193, -96.149, 92.489, -87.068, -89.15, -83.488]
j1 = [117.559, -92.624, 100.329, -96.909, -94.057, -83.488]
j2 = [112.239, -90.096, 99.282, -95.909, -89.824, -83.488]
j3 = [110.839, -83.473, 93.166, -89.22, -90.499, -83.487]
j4 = [107.935, -83.572, 95.424, -92.873, -87.933, -83.488]
descSafe = [0.0,0.0,0.0,0.0,0.0,0.0,0.0]
desc1 = [0.0,0.0,0.0,0.0,0.0,0.0,0.0]
desc2 = [0.0,0.0,0.0,0.0,0.0,0.0,0.0]
desc3 = [0.0,0.0,0.0,0.0,0.0,0.0,0.0]
desc4 = [0.0,0.0,0.0,0.0,0.0,0.0,0.0]
exaxisPos = [0.0,0.0,0.0,0.0]
offdese = [0.0,0.0,0.0,0.0,0.0,0.0,0.0]
error, descSafe = robot.GetForwardKin(jSafe)
robot.MoveJ(joint_pos=jSafe,tool= 1,user= 0,vel= 100)
time.sleep(2)
error, desc1 = robot.GetForwardKin(j1)
robot.MoveJ(joint_pos=j1,tool= 1,user= 0,vel= 100)
time.sleep(2)
actualTCPPos = [0.0,0.0,0.0,0.0,0.0,0.0,0.0]
error, actualTCPPos = robot.GetActualTCPPose(0)
robot.SetRefPointInExAxisEnd(actualTCPPos)
rtn = robot.PositionorSetRefPoint(1)
print(f"PositionorSetRefPoint 1 rtn is {rtn}")
time.sleep(2)
robot.MoveJ(joint_pos=jSafe,tool= 1,user= 0,vel= 100)
robot.ExtAxisStartJog(1, 0, 50, 50, 10)
time.sleep(1)
robot.ExtAxisStartJog(2, 0, 50, 50, 10)
time.sleep(1)
error, desc2 = robot.GetForwardKin(j2)
rtn = robot.MoveJ(joint_pos=j2,tool= 1,user= 0,vel= 100)
rtn = robot.PositionorSetRefPoint(2)
print(f"PositionorSetRefPoint 2 rtn is {rtn}")
time.sleep(2)
robot.MoveJ(joint_pos=jSafe,tool= 1,user= 0,vel= 100)
robot.ExtAxisStartJog(1, 0, 50, 50, 10)
time.sleep(1)
robot.ExtAxisStartJog(2, 0, 50, 50, 10)
time.sleep(1)
error, desc3 = robot.GetForwardKin(j3)
robot.MoveJ(joint_pos=j3,tool= 1,user= 0,vel= 100)
rtn = robot.PositionorSetRefPoint(3)
print(f"PositionorSetRefPoint 3 rtn is {rtn}")
time.sleep(2)
robot.MoveJ(joint_pos=jSafe,tool= 1,user= 0,vel= 100)
robot.ExtAxisStartJog(1, 0, 50, 50, 10)
time.sleep(1)
robot.ExtAxisStartJog(2, 0, 50, 50, 10)
time.sleep(1)
```

(continues on next page)

(continued from previous page)

```
error, desc4 = robot.GetForwardKin(j4)
robot.MoveJ(joint_pos=j4,tool= 1,user= 0,vel= 100)
rtn = robot.PositionorSetRefPoint(4)
print(f"PositionorSetRefPoint 4 rtn is {rtn}")
time.sleep(2)
axisCoord = [0.0,0.0,0.0,0.0,0.0,0.0]
error,axisCoord = robot.PositionorComputeECoordSys()
robot.MoveJ(joint_pos=jSafe,tool= 1,user= 0,vel= 100)
print(f"PositionorComputeECoordSys rtn is {axisCoord[0]} {axisCoord[1]} {axisCoord[2]}
↪{axisCoord[3]} {axisCoord[4]} {axisCoord[5]}")
rtn = robot.ExtAxisActiveECoordSys(3, 1, axisCoord, 1)
print(f"ExtAxisActiveECoordSys rtn is {rtn}")
robot.CloseRPC()
```

2.4.13.39 UDP Extended Axis Motion

New in version python: SDK-v2.1.4

Prototype	`ExtAxisMove(pos,ovl,blend=-1)`
Description	UDP Extended Axis Motion
Mandatory parameters	• `pos=[exaxis[0],exaxis[1],exaxis[2],exaxis[3]]`: target position Axis 1 position to Axis 4 position; • `ovl`: percentage of speed
Default parameters	• `blend`Smoothing parameter (mm or ms), -1, waiting for the motion to complete, Default -1
Return Value	Error Code Success-0 Failure- errcode

2.4.13.40 UDP Extended axis motion code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
axisPos = [20,0,0,0]
robot.ExtAxisMove(axisPos, 50, -1)
robot.CloseRPC()
return 0
```

2.4.13.41 UDP extension axes synchronized with robot joint motion

New in version python: SDK-v2.0.4

Prototype	`ExtAxisSyncMoveJ(joint_pos,tool,user,exaxis_pos, desc_pos=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0], vel=20.0, acc=0.0, ovl= 100.0, blendT=-1.0, offset_flag=0, offset_pos=[ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])`
Description	UDP extension axis synchronized motion with robot joint motion
Mandatory parameters	• `joint_pos`: target joint position in [°]; • `desc_pos`: Cartesian position of the target in [mm][°]; • `tool`: tool number, [0 to 14] • `user`: artifact number, [0~14] • `exaxis_pos`: external axis 1 position ~ external axis 4 positions
Default Parameters	• `desc_pos`: target Cartesian position in [mm][°] Default initial value [0.0,0.0,0.0,0.0,0.0,0.0,0.0], default value calls positive kinematics to solve for the return value. • `vel`: percentage of speed, [0~100] default 20.0; • `acc`: percentage of acceleration, [0~100] not open yet, default 0.0; • `ovl`: velocity scaling factor, [0~100] default 100.0 ; • `blendT`: [-1.0]-motion in place (blocking), [0~500.0]-smoothing time (non-blocking) in [ms] default -1.0; • `offset_flag`: [0] - no offset, [1] - offset in workpiece/base coordinate system, [2] - offset in tool coordinate system Default 0; • `offset_pos`: position offset in [mm][°] default [0.0,0.0,0.0,0.0,0.0,0.0] ;
Return Value	Error Code Success-0 Failure- errcode;

2.4.13.42 UDP extension axes synchronized with robot joint motion code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
# Set the UDP communication parameters and load
robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10)
robot.ExtDevLoadUDPDriver()
# Set the parameters of the extended axis
robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0)
robot.SetRobotPosToAxis(1)
robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0)
# Expand the axis to enable and return to zero
robot.ExtAxisServoOn(1, 0)
robot.ExtAxisSetHoming(1, 0, 20, 3)
# Extended axis coordinate system calibration
pos = []  # Please fill in the specific coordinates
robot.SetRefPointInExAxisEnd(pos)
robot.PositionorSetRefPoint(1)  # This operation should be repeated 4 times (using 4 dots)
```

(continues on next page)

(continued from previous page)

```
error,coord = robot.PositionorComputeECoordSys()
robot.ExtAxisActiveECoordSys(1, 1, coord, 1)
# Synchronize the starting and ending points of the movement
startdescPose = []  # Please fill in the specific coordinates
startjointPos = []  # Please fill in the specific coordinates
startexaxisPos = []  # Please fill in the specific coordinates
enddescPose = []  # Please fill in the specific coordinates
endjointPos = []  # Please fill in the specific coordinates
endexaxisPos = []  # Please fill in the specific coordinates
# Move to the starting point
robot.ExtAxisMove(startexaxisPos, 20, -1)
offdese = [0, 0, 0, 0, 0, 0]
robot.MoveJ(joint_pos=startjointPos,tool= 1,user= 1,vel= 100,acc= 100,ovl= 100,exaxis_
↪pos= startexaxisPos,blendT= 0,offset_flag= 0,offset_pos= offdese)
robot.ExtAxisSyncMoveJ(endjointPos, enddescPose, 1, 1, endexaxisPos, 100, 100, 100, -1,␣
↪0, offdese)
robot.CloseRPC()
```

2.4.13.43 UDP extension axes synchronized with robot linear motion

New in version python: SDK-v2.1.5

Prototype	`ExtAxisSyncMoveL(desc_pos, tool, user, exaxis_pos, joint_pos = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0], vel=20.0, acc=0.0, ovl=100.0, blendR=-1.0, search=0, offset_flag= 0, offset_pos=[0.0, 0.0, 0. 0, 0.0, 0.0, 0.0, 0.0],config=-1)`
Description	UDP extension axis synchronized motion with robot linear motion
Mandatory parameters	• `desc_pos`: target Cartesian position in [mm][°]; • `tool`: tool number, [0 to 14]; • `user`: artifact number, [0 to 14]; • `exaxis_pos`: external axis 1 position ~ external axis 4 positions;
Default Parameters	• `joint_pos`: target joint position in [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0], default value calls inverse kinematics to solve for the return value. • `vel`: percentage of speed, [0~100] default 20.0; • `acc`: percentage of acceleration, [0~100] not open yet, default 0.0; • `ovl`: velocity scaling factor, [0~100] default 100.0; • `blendR`: [-1.0]-motion in place (blocking), [0~500.0]-smoothing time (non-blocking) in [ms] default -1.0; • `search`: [0] - no wire search, [1] - wire search; • `offset_flag`: [0] - no offset, [1] - offset in workpiece/base coordinate system, [2] - offset in tool coordinate system Default 0; • `offset_pos`: position offset in [mm][°] default [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0] ; • `config`: Reverse the joint space configuration, [-1]- calculate based on the current joint position, [0~7]- solve based on the specific joint space configuration, default -1
Return Value	Error Code Success-0 Failure- errcode;

2.4.13.44 UDP extension axes synchronized with robot linear motion code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
# Set the UDP communication parameters and load
robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10)
robot.ExtDevLoadUDPDriver()
# Set the parameters of the extended axis
robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0)
robot.SetRobotPosToAxis(1)
robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0)
# Expand the axis to enable and return to zero
robot.ExtAxisServoOn(1, 0)
robot.ExtAxisSetHoming(1, 0, 20, 3)
# Extended axis coordinate system calibration
pos = []  # Please fill in the coordinates of the marking points
robot.SetRefPointInExAxisEnd(pos)
robot.PositionorSetRefPoint(1)  # It needs to be called four times for calibration
error,coord = robot.PositionorComputeECoordSys()
robot.ExtAxisActiveECoordSys(1, 1, coord, 1)
# Synchronize the starting and ending points of the movement
startdescPose = []  # Please fill in the coordinates of the marking points
startjointPos = []  # Please fill in the coordinates of the marking points
startexaxisPos = []  # Please fill in the coordinates of the marking points
enddescPose = []  # Please fill in the coordinates of the marking points
endjointPos = []  # Please fill in the coordinates of the marking points
endexaxisPos = []  # Please fill in the coordinates of the marking points
# Move to the starting point
robot.ExtAxisMove(startexaxisPos, 20, -1)
offdese = [0, 0, 0, 0, 0, 0]
robot.MoveJ(joint_pos=startjointPos, tool= 1,user= 1,vel= 100,acc= 100,ovl= 100,exaxis_pos= startexaxisPos,blendT= 0)
# Perform synchronous linear motion
robot.ExtAxisSyncMoveL(endjointPos, enddescPose, 1, 1, endexaxisPos, 100, 100, 100, 0, 0, offdese)
robot.CloseRPC()
```

2.4.13.45 UDP extension axes synchronized with robot circular motion

New in version python: SDK-v2.1.5

Prototype	`ExtAxisSyncMoveC(desc_pos_p, tool_p, user_p,exaxis_pos_p, desc_pos_t, tool_t, user_t,exaxis_pos_t,joint_pos_p=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0], joint_pos_t=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0],vel_p =20.0, acc_p=100.0, offset_flag_p=0, offset_pos_p =[0.0, 0.0, 0.0, 0.0, 0.0, 0.0], vel_t=20.0, acc_t=100.0, offset_flag_t=0, offset_pos_t=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0], ovl=100.0, blendR=-1.0, config=-1)`
Description	UDP extension axis synchronized with robot circular motion
Mandatory parameters	• `desc_pos_p`: path point Cartesian position in [mm][°]; • `tool_p`: pathpoint tool number, [0~14]; • `user_p`: pathpoint artifact number, [0~14]; • `exaxis_pos_p`: path point external axis 1 position ~ external axis 4 position Default [0.0,0.0,0.0,0.0]; • `desc_pos_t`: Cartesian position of the target point in [mm][°]; • `tool_t`: tool number, [0~14]; • `user_t`: artifact number, [0~14]; • `exaxis_pos_t`: target point external axis 1 position ~ external axis 4 position default [0.0,0.0,0.0,0.0];
Default Parameters	• `joint_pos_p`: target joint position in [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0,0.0], default value calls inverse kinematics to solve for the return value. • `joint_pos_t`: target joint position in [°] Default initial value is [0.0,0.0,0.0,0.0,0.0,0.0,0.0], default value calls inverse kinematics to solve for the return value. • `vel_p`: path point velocity percentage, [0~100] default 20.0; • `acc_p`: path point acceleration percentage, [0~100] not open yet, default 0.0; • `offset_flag_p`: whether the path point is offset [0]-no offset, [1]-offset in workpiece/base coordinate system, [2]-offset in tool coordinate system Default 0; • `offset_pos_p`: path point position offset in [mm][°] Default [0.0,0.0,0.0,0.0,0.0,0.0]; • `vel_t`: Target point velocity percentage, [0~100] default 20.0; • `acc_t`: target point acceleration percentage, [0~100] Not open yet Default 0.0; • `offset_flag_t`: whether the target point is offset or not [0]-no offset, [1]-offset in workpiece/base coordinate system, [2]-offset in tool coordinate system Default 0; • `offset_pos_t`: target point attitude offset in [mm][°] Default [0.0,0.0,0.0,0.0,0.0,0.0]; • `ovl`: velocity scaling factor, [0~100] default 100.0; • `blendR`: [-1.0] - motion in place (blocking), [0~1000] - smoothing radius (non-blocking) in [mm] default -1.0; • `config`: Reverse the joint space configuration, [-1]- calculate based on the current joint position, [0~7]- solve based on the specific joint space configuration, default -1
Return Value	Error Code Success-0 Failure- errcode;

2.4.13.46 UDP extension axes synchronized with robot circular motion code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
# Set the UDP communication parameters and load
robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 100, 3, 100, 1, 100, 10)
robot.ExtDevLoadUDPDriver()
# Set the parameters of the extended axis
robot.SetAxisDHParaConfig(4, 200, 200, 0, 0, 0, 0, 0, 0)
robot.SetRobotPosToAxis(1)
robot.ExtAxisParamConfig(1, 0, 1, 100, -100, 10, 10, 12, 131072, 0, 1, 0, 0)
# Expand the axis to enable and return to zero
robot.ExtAxisServoOn(1, 0)
robot.ExtAxisSetHoming(1, 0, 20, 3)
# Extended axis coordinate system calibration
pos = []  # Enter the coordinates of the reference point
robot.SetRefPointInExAxisEnd(pos)
robot.PositionorSetRefPoint(1)  # Call four times to complete the calibration
coord = []
error,coord = robot.PositionorComputeECoordSys()
robot.ExtAxisActiveECoordSys(1, 1, coord, 1)
# Synchronous arc starting point, middle point and end point
startdescPose = []# Input coordinates
startjointPos = []# Input coordinates
startexaxisPos =[]  # Enter the coordinates of the extended axis
middescPose = []# Input the midpoint
midjointPos = []
midexaxisPos =[]
enddescPose = []
endjointPos = []
endexaxisPos =[]
# Move to the starting point
robot.ExtAxisMove(startexaxisPos, 20, -1)
offdese = [0, 0, 0, 0, 0, 0]
robot.MoveJ(joint_pos=startjointPos,tool= 1,user= 1,vel= 100,acc= 100,ovl= 100,exaxis_pos= startexaxisPos,blendT= 0,offset_flag= 0,offset_pos= offdese)
# Start synchronous circular arc movement
robot.ExtAxisSyncMoveC(midjointPos,middescPose,1,1,midexaxisPos,
                       endjointPos,enddescPose,1,1,endexaxisPos,
                       100,100,0,offdese,
                       100,100,0,offdese,
                       100,0)
robot.CloseRPC()
```

2.4.13.47 Setting the Extended DO

New in version python: SDK-v2.0.4

Prototype	`SetAuxDO(DONum,bOpen,smooth,block)`
Description	Setting the Extended DO
Mandatory parameters	• `DONum`: DO number; • `bOpen`: switch True-Open, False-Off; • `smooth`: smooth or not True - yes, False - no; • `block`: whether to block True - Yes, False - No;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.48 Setting up Extended AO

New in version python: SDK-v2.0.4

Prototype	`SetAuxAO(AONum,value,block)`
Description	Setting the Extended AO
Mandatory parameters	• `AONum`: AO number; • `value`: analog value [0-4095]; • `block`: whether to block True - Yes, False - No;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.49 Setting the Extended DI Input Filter Time

New in version python: SDK-v2.0.4

Prototype	`SetAuxDIFilterTime(filterTime)`
Description	Setting the Extended DI Input Filter Time
Mandatory parameters	• `filterTime`: filter time (ms);
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.50 Setting the Extended AI Input Filter Time

New in version python: SDK-v2.0.4

Prototype	`SetAuxAIFilterTime(AINum,filterTime)`
Description	Set the extended AI input filter time
Mandatory parameters	• `AINum`: AI number; • `filterTime`: filter time (ms);
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.51 Waiting for extended DI input

New in version python: SDK-v2.0.4

Prototype	`WaitAuxDI(DINum,bOpen,time,errorAlarm)`
Description	Waiting for extended DI input
Mandatory parameters	• `DINum`: DI number; • `bOpen`: switch True-Open, False-Off; • `time`: maximum waiting time (ms); • `errorAlarm`: whether to continue the campaign True-Yes,False-No
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.52 Waiting for extended AI input

New in version python: SDK-v2.0.4

Prototype	`WaitAuxAI(,AINum,sign,value,time,errorAlarm)`
Description	Waiting for extended AI input
Mandatory parameters	• `AINum`: AI number; • `sign`: 0 - greater than; 1 - less than; • `value`: AI value; • `time`: maximum waiting time (ms); • `errorAlarm`: whether to continue the campaign True-Yes,False-No
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.53 Get Extended DI Value

New in version python: SDK-v2.0.4

Prototype	`GetAuxDI(DINum,isNoBlock)`
Description	Get Extended DI Value
Mandatory parameters	• `DINum`: DI number; • `isNoBlock`: whether to block True-blocking false-non-blocking;
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode; • `isOpen`: 0 - off; 1 - on;

2.4.13.54 Get Extended AI Value

New in version python: SDK-v2.0.4

Prototype	`GetAuxAI(AINum,isNoBlock)`
Description	Get Extended AI Value
Mandatory parameters	• `AINum`: AI number; • `isNoBlock`: whether to block True-blocking False-non-blocking
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode; • `value`: input value;

2.4.13.55 Extended IO code examples

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
for i in range(128):
    robot.SetAuxDO(i, True, False, True)
    time.sleep(0.1)
for i in range(128):
    robot.SetAuxDO(i, False, False, True)
    time.sleep(0.1)
for i in range(409):
    value1 = i * 10
    value2 = 4095 - i * 10
    robot.SetAuxAO(0, value1, True)
    robot.SetAuxAO(1, value2, True)
    robot.SetAuxAO(2, value1, True)
```

(continues on next page)

(continued from previous page)

```
    robot.SetAuxAO(3, value2, True)
    time.sleep(0.01)
robot.SetAuxDIFilterTime(10)
robot.SetAuxAIFilterTime(0, 10)
for i in range(20):
    curValue = False
    error, curValue = robot.GetAuxDI(i, False)  #
    print(f"DI{i}   {curValue}")
curValue = -1
for i in range(4):
    error, curValue = robot.GetAuxAI(i, True)  #
    print(f"AI{i}   {curValue}")
robot.WaitAuxDI(1, False, 1000, False)
robot.WaitAuxAI(1, 1, 132, 1000, False)
robot.CloseRPC()
```

2.4.13.56 Removable Device Enable

New in version python: SDK-v2.0.5

Prototype	`TractorEnable(enable)`
description	removable device enable
Mandatory parameters	• `enable`: enable state, 0-de-enable, 1-enable
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.57 Zeroing of removable units

New in version python: SDK-v2.0.5

Prototype	`TractorHoming()`
Description	Removable unit back to zero
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.58 Movable unit linear motion

New in version python: SDK-v2.0.5

Prototype	`TractorMoveL(distance, vel)`
Description	Movable device linear motion
Mandatory parameters	• `distance`: distance of linear movement (mm) • `vel`: percentage of linear motion speed (0-100)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.59 Movable unit circular motion

New in version python: SDK-v2.0.5

prototype	`TractorMoveC(radio, angle, vel)`
Description	Movable device circular motion
Mandatory parameters	• `radio`: radius of circular motion (mm) • `angle`: angle of circular motion (°) • `vel`: Percentage of speed of circular motion (0-100)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.60 Stopping motion of movable devices

New in version python: SDK-v2.0.5

Prototype	`ProgramStop()`
Description	Movable unit stops moving
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.61 Portable device code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.ExtDevSetUDPComParam("192.168.58.2", 2021, 2, 50, 5, 50, 1, 50, 10, 1)
robot.ExtDevLoadUDPDriver()
rtn = robot.ExtAxisServoOn(1, 1)
rtn = robot.ExtAxisServoOn(2, 1)
time.sleep(2)
robot.ExtAxisSetHoming(1, 0, 10, 2)
time.sleep(2)
rtn = robot.ExtAxisSetHoming(2, 0, 10, 2)
time.sleep(4)
```

(continues on next page)

(continued from previous page)

```
robot.ExtAxisParamConfig(1, 0, 0, 50000, -50000, 1000, 1000, 6.280, 16384, 200, 0, 0, 0)
robot.ExtAxisParamConfig(2, 0, 0, 50000, -50000, 1000, 1000, 6.280, 16384, 200, 0, 0, 0)
robot.SetAxisDHParaConfig(5, 0, 0, 0, 0, 0, 0, 0, 0)
robot.TractorEnable(False)
time.sleep(2)
robot.TractorEnable(True)
time.sleep(2)
robot.TractorHoming()
time.sleep(2)
robot.TractorMoveL(100, 2)
time.sleep(5)
robot.TractorStop()
robot.TractorMoveL(-100, 20)
time.sleep(5)
robot.TractorMoveC(300, 90, 20)
time.sleep(10)
robot.TractorMoveC(300, -90, 20)
time.sleep(1)
robot.CloseRPC()
```

2.4.13.62 Laser sensor recording points

New in version python: SDK-v2.1.4

Prototype	`LaserRecordPoint(coordID)`
Description	Laser sensor recording points
Mandatory par	• `coordID`Laser sensor coordinate system
Default param	NULL
Return Value	• Error Code Success-0 Failure- errcode • `joint`The laser sensor identifies the position of the point joint • `desc`The laser sensor identifies the Cartesian position of the point • `exaxis`The laser sensor identifies the extended axis position of the point

2.4.13.63 Sample code for laser sensor recording points

New in version python: SDK-v2.1.4

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
direction_point = [0, 0, 0]
rtn = robot.LaserTrackingSearchStart(2, direction_point, 10, 100, 10000, 2)
print(f"LaserTrackingSearchStart rtn is {rtn}")
robot.LaserTrackingSearchStop()
coord_id = 2
```

(continues on next page)

(continued from previous page)

```
rtn, joint, desc, exaxis = robot.LaserRecordPoint(coord_id)
print(f"rtn is {rtn}")
print(f"desc_pos:{desc[0]},{desc[1]},{desc[2]},"
      f"{desc[3]},{desc[4]},{desc[5]}")
print(f"joint_pos:{joint[0]},{joint[1]},{joint[2]},{joint[3]},{joint[4]},{joint[5]}")
print(f"exaxis pos is {exaxis[0]} {exaxis[1]} {exaxis[2]} {exaxis[3]}")
off = [0] * 6
robot.MoveJ(joint,tool=1,user=0,vel=100,acc=100,ovl=50,exaxis_pos=exaxis,blendT=-1,
↪offset_flag=0,offset_pos=off)
robot.CloseRPC()
```

2.4.13.64 Set the synchronous movement strategy of the extended axis and the robot

New in version python: SDK-v2.1.5

Prototype	`SetExAxisRobotPlan(strategy)`
Description	Set the synchronous movement strategy of the extended axis and the robot
Mandatory par	• `strategy`Strategy; 0- Mainly robots; 1- The extended axis is synchronized with the robot
Default param	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.65 Code example for setting the synchronous motion strategy of the extended axis and the robot

New in version python: SDK-v2.1.5

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
joint_pos1 = [-22.016, -49.217, 124.714, -161.100, -85.108, -0.333]
joint_pos2 = [-21.083, -46.613, 110.079, -147.796, -80.757, -0.330]
joint_pos3 = [-25.572, -60.090, 135.397, -163.889, -82.489, -0.345]
desc_pos1 = [2.637, -0.001, 30.673, 178.786, -4.134, 68.326]
desc_pos2 = [213.812, -1.440, 47.311, 177.410, 0.166, 68.946]
desc_pos3 = [444.342, -12.723, 82.470, -177.701, -1.325, 65.151]
epos1 = [0.001, 0.000, 0.000, 0.000]
epos2 = [299.977, 0.000, 0.000, 0.000]
epos3 = [399.969, 0.000, 0.000, 0.000]
offset_pos = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
rtn = robot.SetExAxisRobotPlan(0)
print(f"SetExAxisRobotPlan rtn is {rtn}")
time.sleep(1)
rtn = robot.ExtAxisSyncMoveL(desc_pos=desc_pos1,tool=1,user=0,vel=100,acc=100,ovl=100,
↪blendR=-1,exaxis_pos=epos1,offset_flag=0,offset_pos=offset_pos)
print(f"ExtAxisSyncMoveL 1 rtn is {rtn}")
rtn = robot.ExtAxisSyncMoveL(desc_pos=desc_pos2,tool=1,user=0,vel=100,acc=100,ovl=100,
```

(continues on next page)

(continued from previous page)

```
↪blendR=-1,exaxis_pos=epos2,offset_flag=0,offset_pos=offset_pos)
print(f"ExtAxisSyncMoveL 2 rtn is {rtn}")
rtn = robot.ExtAxisSyncMoveL(desc_pos=desc_pos3,tool=1,user=0,vel=100,acc=100,ovl=100,
↪blendR=-1,exaxis_pos=epos3,offset_flag=0,offset_pos=offset_pos)
print(f"ExtAxisSyncMoveL 3 rtn is {rtn}")
time.sleep(8)
robot.CloseRPC()
```

2.4.13.66 Control array type sucker

New in version python: SDK-v2.1.5

Prototype	`SetSuckerCtrl(slaveID, len, ctrlValue)`
Description	Control array type sucker
Mandatory par	• `slaveID`Slave station number • `len`length • `ctrlValue`Control value 1- Suction according to the maximum vacuum degree 2- suction according to the set vacuum degree 3- stop suction
Default param	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.67 Get the state of the array suction cup

New in version python: SDK-v2.1.5

Prototype	`GetSuckerState(slaveID)`
Description	Get the state of the array suction cup
Mandatory par	• `slaveID`Slave station number
Default param	NULL
Return Value	• Error Code Success-0 Failure- errcode • `state`Adsorption state 0 -released object 1 -successful adsorption of the workpiece was detected 2 -no adsorption to the object 3 -object detachment • `pressValue`The current vacuum is in kpa • `error`The current error code for the sucker

2.4.13.68 Wait for the sucker state

New in version python: SDK-v2.1.5

Prototype	`WaitSuckerState(slaveID, state, ms)`
Description	Wait for the sucker state
Mandatory par	• `slaveID`Slave station number • `state`Adsorption state 0 -released object 1 -successful adsorption of the workpiece was detected 2 -no adsorption to the object 3 -object detachment • `ms`Maximum waiting time
Default param	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.69 Array type sucker control command code example

New in version python: SDK-v2.1.5

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
robot.OpenLuaUpload("C://Project/PeripheralSDK/CtrlDev_sucker.lua")
time.sleep(2)
robot.UnloadCtrlOpenLUA(1)
robot.LoadCtrlOpenLUA(1)
time.sleep(1)
ctrl = bytearray(20)
ctrl[0] = 1
robot.SetSuckerCtrl(0, 1, ctrl)
for i in range(100):
    rtn, state, press_value, error = robot.GetSuckerState(1)
    print(f"sucker1 state is {state}, pressValue is {press_value}, error num is {error}")
    rtn, state, press_value, error = robot.GetSuckerState(12)
    print(f"sucker12 state is {state}, pressValue is {press_value}, error num is {error}")
    time.sleep(0.1)
ret = robot.WaitSuckerState(1, 1, 100)
print(f"WaitSuckerState result is {ret}")
ctrl[0] = 3
robot.SetSuckerCtrl(1, 1, ctrl)
robot.SetSuckerCtrl(12, 1, ctrl)
robot.CloseRPC()
```

2.4.13.70 Upload an open protocol Lua file

New in version python: SDK-v2.1.5

Prototype	`OpenLuaUpload(filePath)`
Description	Upload an open protocol Lua file
Mandatory par	• `filePath`Local open protocol lua file path name
Default param	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.71 Get the slave board parameters

New in version python: SDK-v2.1.5

Prototype	`GetFieldBusConfig()`
Description	Get the slave board parameters
Mandatory par	NULL
Default param	NULL
Return Value	• Error Code Success-0 Failure- errcode • `type`0-Ethercat1-CClink, 3-Ethercat, 4-EIP • `version`Protocol version • `connState`0- Not connected 1- Connected

2.4.13.72 Write the slave DO

New in version python: SDK-v2.1.5

Prototype	`FieldBusSlaveWriteDO(DOIndex, wirteNum, status)`
Description	Write the slave DO
Mandatory par	• `DOIndex`DO number • `wirteNum`Number of writes • `status`The number to write, up to 8
Default param	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.73 Write the slave AO

New in version python: SDK-v2.1.5

Prototype	`FieldBusSlaveWriteAO(AOIndex, wirteNum, status)`
Description	Write the slave AO
Mandatory par	• `AOIndex`AO number • `wirteNum`Number of writes • `status`The number to write, up to 8
Default param	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.74 Read the slave DI

New in version python: SDK-v2.1.5

Prototype	`FieldBusSlaveReadDI(DOIndex, readeNum)`
Description	Read the slave DI
Mandatory par	• `DOIndex`DI number • `readeNum`Number of reads
Default param	NULL
Return Value	• Error Code Success-0 Failure- errcode • `status[8]`Read a maximum of 8 values

2.4.13.75 Read the slave AI

New in version python: SDK-v2.1.5

Prototype	`FieldBusSlaveReadAI(AOIndex, readeNum)`
Description	Read the slave AI
Mandatory par	• `AOIndex`AI number • `readeNum`Number of reads
Default param	NULL
Return Value	• Error Code Success-0 Failure- errcode • `status[8]`Read a maximum of 8 values

2.4.13.76 Wait for the extension DI input

New in version python: SDK-v2.1.5

Prototype	`FieldBusSlaveWaitDI(DIIndex, status, waitMs)`
Description	Wait for the extension DI input
Mandatory par	• `DIIndex`DI number • `status`0- low level; 1- High level • `waitMs`Maximum waiting time (ms)
Default param	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.77 Wait for extended AI input

New in version python: SDK-v2.1.5

Prototype	`FieldBusSlaveWaitAI(AIIndex, waitType, value, waitMs)`
Description	Wait for extended AI input
Mandatory par	• `AIIndex`AI number • `waitType`0- greater than; 1- less than • `value`AI value • `waitMs`Maximum waiting time (ms)
Default param	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.13.78 Example of slave mode related interface instruction code

New in version python: SDK-v2.1.5

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.OpenLuaUpload("D://zUP/Peripheral_equipment/CtrlDev_field.lua")
time.sleep(2)
robot.SetCtrlOpenLUAName(3,"CtrlDev_field.lua")
robot.UnloadCtrlOpenLUA(3)
robot.LoadCtrlOpenLUA(3)
time.sleep(8)
rtn,type, version, conn_state = robot.GetFieldBusConfig()
print(f"type is {type}, version is {version}, connState is {conn_state}")
# Write digital outputs
ctrl = [1, 0, 1]  # DO0=1, DO1=0, DO2=1
robot.FieldBusSlaveWriteDO(0, 3, ctrl)
# Write analog output
ctrl_ao = [0x1000]  # AO2 = 0x1000
robot.FieldBusSlaveWriteAO(2, 1, ctrl_ao)
```

(continues on next page)

(continued from previous page)

```
for i in range(100):
    rtn,di = robot.FieldBusSlaveReadDI(0, 4)
    print(f"DI0 is {di[0]}, DI1 is {di[1]}, DI2 is {di[2]}, DI3 is {di[3]}")
    rtn, ai = robot.FieldBusSlaveReadAI(0, 3)
    print(f"AI0 is {ai[0]}, AI1 is {ai[1]}, AI2 is {ai[2]}")
    time.sleep(0.01)
ret = robot.FieldBusSlaveWaitDI(0, 1, 100)
print(f"FieldBusSlaveWaitDI result is {ret}")
ret = robot.FieldBusSlaveWaitAI(0, 0, 400.00, 100)
print(f"FieldBusSlaveWaitAI result is {ret}")
robot.CloseRPC()
```

2.4.14 Weld

2.4.14.1 Setting Welding Process Curve Parameters

New in version python: SDK-v2.0.5

Prototype	`WeldingSetProcessParam(id, startCurrent, startVoltage, startTime, weldCurrent, weldVoltage, endCurrent, endVoltage, endTime)`
Description	Set weld process profile parameters
Mandatory parameters	• `id`: Welding process number (1-99) • `startCurrent`: Arc starting current (A) • `startVoltage`: startVoltage Arc-starting voltage (V) • `startTime`: startTime Arc start time (ms) • `weldCurrent`: weldCurrent Welding current (A) • `weldVoltage`: weldVoltage Welding voltage (V) • `endCurrent`: endCurrent Arc recovery current (A) • `endVoltage`: endVoltage Arc charging voltage (V) • `endTime`: endTime closing time (ms)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.2 Obtaining Welding Process Curve Parameters

New in version python: SDK-v2.0.5

Prototype	`WeldingGetProcessParam(id)`
Description	Get welding process curve parameters
Mandatory parameters	• `id`: Welding process number (1-99)
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `startCurrent`: arc starting current (A) • `startVoltage`: Arc starting voltage (V) • `startTime`: start time (ms) • `weldCurrent`: welding current (A) • `weldVoltage`: welding voltage (V) • `endCurrent`: arc closing current (A) • `endVoltage`: arc closing voltage (V) • `endTime`: arc closing time (ms)

2.4.14.3 Setting of welding current and output analog correspondences

Changed in version python: SDK-v2.0.5

Prototype	`WeldingSetCurrentRelation(currentMin, currentMax, outputVoltageMin, outputVoltageMax)`
Description	Setting the welding current to correspond to the output analog
Mandatory parameters	• `currentMin`: Welding current - analog output linear relationship left point current value (A) • `currentMax`: Welding current - analog output linear relationship right point current value (A) • `outputVoltageMin`: analog output voltage value (V) at the left point of the welding current-analog output linear relationship. • `outputVoltageMax`: analog output voltage value (V) at the point on the right side of the weld current-analog output linear relationship. • `AOIndex`: analog output port for welding current
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.4 Setting the welding voltage and output analog correspondence

Changed in version python: SDK-v2.0.5

Prototype	`WeldingSetVoltageRelation(weldVoltageMin, weldVoltageMax, outputVoltageMin, outputVoltageMax)`
Description	Setting the weld voltage to correspond to the output analog
Mandatory parameters	• `weldVoltageMin`: Welding voltage - analog output linear relationship left point welding voltage value (A) • `weldVoltageMax`: Welding voltage - analog output linear relationship right point welding voltage value (A) • `outputVoltageMin`: analog output voltage value (V) at the left point of the welding voltage-analog output linear relationship. • `outputVoltageMax`: analog output voltage value (V) at the point on the right side of the weld voltage-analog output linear relationship. • `AOIndex`: welding voltage analog output port
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.5 Acquiring the correspondence between welding current and output analog quantity

Changed in version python: SDK-v2.0.5

Prototype	`WeldingGetCurrentRelation()`
Description	Get weld current to output analog correspondence
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `currentMin`: welding current - analog output linear relationship left point current value (A) • `currentMax`: welding current - analog output linear relationship right point current value (A) • `outputVoltageMin`: analog output voltage value at the left point of the welding current-analog output linear relationship (V) • `outputVoltageMax`: analog output voltage value (V) at the point on the right side of the weld current-analog output linear relationship. • `AOIndex`: welding voltage analog output port

2.4.14.6 Getting welding voltage and output analog correspondence

Changed in version python: SDK-v2.0.5

Prototype	`WeldingGetVoltageRelation()`
Description	Get weld voltage to output analog correspondence
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `weldVoltageMin`: Welding voltage - analog output linear relationship left point welding voltage value (V) • `weldVoltageMax`: Welding voltage - analog output linear relationship of the right point welding voltage value (V) • `outputVoltageMin`: Analog output voltage value at the left point of the welding voltage-analog output linear relationship (V) • `outputVoltageMax`: Analog output voltage value (V) at the right point of the weld current-analog output linear relationship • `AOIndex`: welding voltage analog output port

2.4.14.7 Setting the welding current

Changed in version python: SDK-v2.0.5

Prototype	`WeldingSetCurrent(ioType, current, AOIndex, blend)`
Description	Setting the welding current
Mandatory parameters	• `ioType`: type 0 - controller IO; 1 - extended IO • `current`: Welding current value (A) • `AOIndex`: Analog output port (0-1) of the welding current control box • `blend`: smooth or not 0 - not smooth, 1 - smooth
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.8 Setting the welding voltage

Changed in version python: SDK-v2.0.5

Prototype	`WeldingSetVoltage(ioType, voltage, AOIndex, blend)`
Description	Set weld voltage
Mandatory parameters	• `ioType`: type 0 - controller IO; 1 - extended IO • `voltage`: Welding voltage value (V) • `AOIndex`: Analog output port (0-1) of the welding current control box • `blend`: smooth or not 0 - not smooth, 1 - smooth
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.9 Setting Oscillation Parameters

Changed in version python: SDK-v2.0.5

Prototype	`WeaveSetPara(weaveNum, weaveType, weaveFrequency, weaveIncStayTime, weaveRange, weaveLeftRange, weaveRightRange. additionalStayTime, weaveLeftStayTime, weaveRightStayTime, weaveCircleRadio, weaveStationary, weaveYawAngle=0)`
Description	Setting swing parameters
Mandatory parameters	• `weaveNum`: Pendulum welding parameter configuration number • `weaveType`: type of oscillation 0-planar triangular oscillation; 1-vertical L-shaped triangular oscillation; 2-clockwise circular oscillation; 3-counterclockwise circular oscillation; 4-planar sinusoidal oscillation; 5-vertical L-shaped sinusoidal oscillation; 6-vertical triangular oscillation; 7-vertical sinusoidal oscillation. • `weaveFrequency`: swing frequency (Hz) • `weaveIncStayTime`: wait mode 0-cycle without wait time; 1-cycle with wait time mandatory parameter • `weaveRange`: swing range (mm) • `weaveLeftRange`: vertical triangle swing left chord length(mm) • `weaveRightRange`: vertical triangle swing right chord length(mm) • `additionalStayTime`: vertical triangle swing vertical triangle point residence time(mm) • `weaveLeftStayTime`: wave left stay time (ms) • `weaveRightStayTime`: wave right stay time (ms) • `weaveCircleRadio`: Circle swing-back ratio (0-100%) • `weaveStationary`: swing position wait, 0 - position continues to move during wait time; 1 - position is stationary during wait time
Default parameters	• `weaveYawAngle`: azimuth angle of the swing direction (rotation around the swing Z-axis) in °, default 0 • `weaveRotAngle`: Swing direction azimuth (rotation around the X-axis of the swing), unit °, default 0
Return Value	Error Code Success-0 Failure- errcode

2.4.14.10 Example code for setting welding parameters

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.WeldingSetProcessParam(1, 177, 27, 1000, 178, 28, 176, 26, 1000)
robot.WeldingSetProcessParam(2, 188, 28, 555, 199, 29, 133, 23, 333)
start_current = 0
start_voltage = 0
start_time = 0
weld_current = 0
weld_voltage = 0
```

(continues on next page)

(continued from previous page)

```
end_current = 0
end_voltage = 0
end_time = 0
error, start_current, start_voltage, start_time, weld_current, weld_voltage, end_current, end_voltage, end_time = robot.WeldingGetProcessParam(1)
print(f"the Num 1 process param is {start_current} {start_voltage} {start_time} {weld_current} {weld_voltage} {end_current} {end_voltage} {end_time}")
error, start_current, start_voltage, start_time, weld_current, weld_voltage, end_current, end_voltage, end_time = robot.WeldingGetProcessParam(2)
print(f"the Num 2 process param is {start_current} {start_voltage} {start_time} {weld_current} {weld_voltage} {end_current} {end_voltage} {end_time}")
rtn = robot.WeldingSetCurrentRelation(0, 400, 0, 10, 0)
print(f"WeldingSetCurrentRelation rtn is: {rtn}")
rtn = robot.WeldingSetVoltageRelation(0, 40, 0, 10, 1)
print(f"WeldingSetVoltageRelation rtn is: {rtn}")
current_min = 0
current_max = 0
vol_min = 0
vol_max = 0
output_vmin = 0
output_vmax = 0
cur_index = 0
vol_index = 0
rtn,current_min, current_max, output_vmin, output_vmax, cur_index = robot.WeldingGetCurrentRelation()
print(f"WeldingGetCurrentRelation rtn is: {rtn}")
print(f"current min {current_min} current max {current_max} output vol min {output_vmin} output vol max {output_vmax}")
rtn,vol_min, vol_max, output_vmin, output_vmax, vol_index = robot.WeldingGetVoltageRelation()
print(f"WeldingGetVoltageRelation rtn is: {rtn}")
print(f"vol min {vol_min} vol max {vol_max} output vol min {output_vmin} output vol max {output_vmax}")
rtn = robot.WeldingSetCurrent(1, 100, 0, 0)
print(f"WeldingSetCurrent rtn is: {rtn}")
time.sleep(3)
rtn = robot.WeldingSetVoltage(1, 10, 0, 0)
print(f"WeldingSetVoltage rtn is: {rtn}")
rtn = robot.WeaveSetPara(0, 0, 2.000000, 0, 10.000000, 0.000000, 0.000000, 0, 0, 0, 0, 0, 0.0, 60.000000)
print(f"rtn is: {rtn}")
robot.WeaveOnlineSetPara(0, 0, 1, 0, 20, 0, 0, 0, 0)
rtn = robot.WeldingSetCheckArcInterruptionParam(1, 200)
print(f"WeldingSetCheckArcInterruptionParam {rtn}")
rtn = robot.WeldingSetReWeldAfterBreakOffParam(1, 5.7, 98.2, 0)
print(f"WeldingSetReWeldAfterBreakOffParam {rtn}")
enable = 0
length = 0
velocity = 0
move_type = 0
check_enable = 0
arc_interrupt_time_length = 0
```

(continues on next page)

(continued from previous page)

```
rtn,check_enable, arc_interrupt_time_length = robot.WeldingGetCheckArcInterruptionParam()
print(f"WeldingGetCheckArcInterruptionParam checkEnable {check_enable}␣
↪arcInterruptTimeLength {arc_interrupt_time_length}")
rtn,enable, length, velocity, move_type = robot.WeldingGetReWeldAfterBreakOffParam()
print(f"WeldingGetReWeldAfterBreakOffParam enable = {enable}, length = {length},␣
↪velocity = {velocity}, moveType = {move_type}")
robot.SetWeldMachineCtrlModeExtDoNum(17)
for i in range(5):
    robot.SetWeldMachineCtrlMode(0)
    time.sleep(1)
    robot.SetWeldMachineCtrlMode(1)
    time.sleep(1)
robot.CloseRPC()
```

2.4.14.11 Instant setup of swing parameters

New in version python: SDK-v2.0.1

Prototype	`WeaveOnlineSetPara (weaveNum, weaveType, weaveFrequency, weaveIncStayTime, weaveRange, weaveLeftStayTime, weaveRightStayTime. weaveCircleRadio, weaveStationary)`
Description	Set swing parameters on the fly
Mandatory parameters	• `weaveNum`: Pendulum welding parameter configuration number • `weaveType`: type of oscillation 0-planar triangular oscillation; 1-vertical L-shaped triangular oscillation; 2-clockwise circular oscillation; 3-counterclockwise circular oscillation; 4-planar sinusoidal oscillation; 5-vertical L-shaped sinusoidal oscillation; 6-vertical triangular oscillation; 7-vertical sinusoidal oscillation. • `weaveFrequency`: swing frequency (Hz) • `weaveIncStayTime`: wait mode 0-cycle without wait time; 1-cycle with wait time mandatory parameter • `weaveRange`: swing range (mm) • `weaveLeftStayTime`: wave left stay time (ms) • `weaveRightStayTime`: wave right stay time (ms) • `weaveCircleRadio`: Circle swing-back ratio (0-100%) • `weaveStationary`: swing position wait, 0 - position continues to move during wait time; 1 - position is stationary during wait time
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.12 The detection parameters of unexpected interruption of robot welding arc were obtained

New in version python: SDK-v2.0.8

Prototype	`WeldingGetCheckArcInterruptionParam()`
Description	The detection parameters of unexpected interruption of robot welding arc were obtained
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `checkEnable`: Whether to enable detection. 0: Indicates that the function is disabled. 1- Enable • `arcInterruptTimeLength`:Arc break confirmation time (ms)

2.4.14.13 Set the detection parameters of robot welding arc unexpected interruption

New in version python: SDK-v2.0.8

Prototype	`WeldingSetCheckArcInterruptionParam(checkEnable, arcInterruptTimeLength)`
Description	Set the detection parameters of robot welding arc unexpected interruption
Mandatory parameters	• `checkEnable`: Whether to enable detection. 0: Indicates that the function is disabled. 1- Enable • `arcInterruptTimeLength`:Arc break confirmation time (ms)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.14 Obtain the robot welding interrupt recovery parameters

New in version python: SDK-v2.0.8

Prototype	`WeldingGetReWeldAfterBreakOffParam()`
Description	Obtain the robot welding interrupt recovery parameters
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `enable`: Whether to enable recovery from welding interruption • `length`: weld overlap distance (mm) • `velocity`: Percentage of robot return to rearcing point (0-100) • `moveType`: The way the robot moves to the rearcing point; 0-LIN; 1-PTP

2.4.14.15 Set the robot welding interrupt recovery parameters

New in version python: SDK-v2.0.8

Prototype	`WeldingSetReWeldAfterBreakOffParam(enable, length, velocity, moveType)`
Description	Set the robot welding interrupt recovery parameters
Mandatory parameters	• `enable`:Whether to enable recovery from welding interruption • `length`: weld overlap distance (mm) • `velocity`: Percentage of robot return to rearcing point (0-100) • `moveType`: The way the robot moves to the rearcing point; 0-LIN; 1-PTP
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.16 Set welder control mode to expand DO port

New in version python: SDK-v2.0.5

Prototype	`SetWeldMachineCtrlModeExtDoNum(DONum)`
Description	Set welder control mode to extend DO port
Mandatory parameters	• `DONum`: welder control mode DO port (0-127)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.17 Setting the welder control mode

New in version python: SDK-v2.0.5

Prototype	`SetWeldMachineCtrlMode(mode)`
Description	Set welder control mode
Mandatory parameters	• `mode`: welder control mode; 0 - unitary
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.18 Welding Start

New in version python: SDK-v2.0.1

Prototype	`ARCStart(ioType, arcNum, timeout)`
Description	Welding Start
Required Parameters	• `ioType`: io type 0 - controller IO; 1 - extended IO • `arcNum`: Welder profile number • `timeout`: timeout for starting an arc
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.19 End of welding

New in version python: SDK-v2.0.1

prototype	`ARCEnd(ioType, arcNum, timeout)`
Description	End of welding
Mandatory parameters	• `ioType`: type 0 - controller IO; 1 - extended IO • `arcNum`: Welder profile number • `timeout`: timeout for starting an arc
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.20 swing start

New in version python: SDK-v2.0.1

Prototype	`WeaveStart(weaveNum)`
Description	Swing Start
Mandatory parameters	• `weaveNum`: type 0 - controller IO; 1 - extended IO
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.21 end of swing (math.)

New in version python: SDK-v2.0.1

Prototype	`WeaveEnd(weaveNum)`
description	end of swing
Mandatory parameter	• `weaveNum`: Pendulum welding parameter configuration number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.22 Positive wire feed

New in version python: SDK-v2.0.1

Prototype	`SetForwardWireFeed(ioType, wireFeed)`
Description	Positive Wire Feed
Mandatory parameters	• `ioType`: 0 - controller IO; 1 - extended IO • `wireFeed`: wire feed control 0 - stop wire feed; 1 - wire feed.
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.23 Reverse wire feed

New in version python: SDK-v2.0.1

Prototype	`SetReverseWireFeed(ioType, wireFeed)`
Description	Reverse Wire Feed
Mandatory parameters	• `ioType`: 0 - controller IO; 1 - extended IO • `wireFeed`: wire feed control 0 - stop wire feed; 1 - wire feed.
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.24 aspiration

New in version python: SDK-v2.0.1

Prototype	`SetAspirated(ioType, airControl)`
Description	Air delivery
Mandatory parameters	• `ioType`: 0 - controller IO; 1 - extended IO • `airControl`: air delivery control 0 - stop air delivery; 1 - air delivery
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.25 Set the robot to resume welding after welding interruption

New in version python: SDK-v2.0.8

Prototype	`WeldingStartReWeldAfterBreakOff()`
Description	Set the robot to resume welding after welding interruption
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.26 Set the robot to exit welding after welding interruption

New in version python: SDK-v2.0.8

Prototype	`WeldingAbortWeldAfterBreakOff()`
Description	Set the robot to exit welding after welding interruption
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.27 Sample code for robot welding control

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.SetForwardWireFeed(0, 1)
time.sleep(1)
robot.SetForwardWireFeed(0, 0)
robot.SetReverseWireFeed(0, 1)
time.sleep(1)
robot.SetReverseWireFeed(0, 0)
robot.SetAspirated(0, 1)
time.sleep(1)
robot.SetAspirated(0, 0)
robot.WeldingSetCurrent(1, 230, 0, 0)
robot.WeldingSetVoltage(1, 24, 0, 1)
p1Desc = [228.879, -503.594, 453.984, -175.580, 8.293, 171.267]
p1Joint = [102.700, -85.333, 90.518, -102.365, -83.932, 22.134]
p2Desc = [-333.302, -435.580, 449.866, -174.997, 2.017, 109.815]
p2Joint = [41.862, -85.333, 90.526, -100.587, -90.014, 22.135]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
robot.MoveJ(joint_pos=p1Joint, tool=13, user=0)
robot.ARCStart(1, 0, 10000)
robot.WeaveStart(0)
robot.MoveL(desc_pos=p2Desc, tool=13, user=0)
robot.ARCEnd(1, 0, 10000)
robot.WeaveEnd(0)
robot.WeldingStartReWeldAfterBreakOff()
robot.WeldingAbortWeldAfterBreakOff()
robot.CloseRPC()
```

2.4.14.28 Segmented welding startup

New in version python: SDK-v2.0.1

Prototype	`SegmentWeldStart(startDesePos, endDesePos, startJPos, endJPos, weldLength, noWeldLength, weldIOType, arcNum, weldTimeout, isWeave. weaveNum,tool,user,vel=20.0, acc=0.0, ovl=100. 0, blendR=-1.0,exaxis_pos=[0.0, 0.0, 0.0, 0.0], search=0, offset_flag=0, offset_pos=[0.0, 0.0, 0.0, 0.0, 0.0, 0.0])`
Description	Segmented Welding Initiation
Mandatory parameters	• `startDesePos`: Initial Cartesian position in [mm][°]. • `endDesePos`: target Cartesian position in [mm][°]. • `startJPos`: initial joint position in [°]. • `endJPos`: target joint position in [°] • `weldLength`: weld length in [mm] • `noWeldLength`: non-weld length in [mm] • `weldIOType`: weld IO type (0-control box IO; 1-expansion IO) arcNum welder profile number • `timeout`: timeout for arc extinguishing • `isWeave`: welded False – not welded • `weaveNum`: Pendulum welding parameter configuration number • `tool`: tool number, [0 to 14] • `user`: artifact number, [0 to 14]
Default parameters	• `vel`: percentage of speed, [0~100] default 20.0 • `acc`: acceleration [0~100] Not open Default 0.0 • `ovl`: velocity scaling factor, [0~100] default 100.0 • `blendR`: [-1.0] - motion in place (blocking), [0~1000] - smoothing radius (non-blocking), unit [mm] default -1.0 • `exaxis_pos`: external axis 1 position ~ external axis 4 position Default [0.0,0.0,0.0,0.0] • `search`: [0] - no wire seek, [1] - wire seek • `offset_flag`: [0] - no offset, [1] - offset in workpiece/base coordinate system, [2] - offset in tool coordinate system Default 0 • `offset_pos`: position offset in [mm][°] default [0.0,0.0,0.0,0.0,0.0,0.0]
Return Value	• errcode Success-0 Failure- errcode

2.4.14.29 Sample robot segment welding code

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.WeldingSetCurrent(1, 230, 0, 0)
robot.WeldingSetVoltage(1, 24, 0, 1)
p1Desc = [228.879, -503.594, 453.984, -175.580, 8.293, 171.267]
```

(continues on next page)

(continued from previous page)

```
p1Joint = [102.700, -85.333, 90.518, -102.365, -83.932, 22.134]
p2Desc = [-333.302, -435.580, 449.866, -174.997, 2.017, 109.815]
p2Joint = [41.862, -85.333, 90.526, -100.587, -90.014, 22.135]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
rtn = robot.SegmentWeldStart(p1Desc, p2Desc, p1Joint, p2Joint, 20, 20, 0, 0, 5000, 0, 0,␣
↪0, 0)
print(f"SegmentWeldStart rtn is {rtn}")
robot.CloseRPC()
```

2.4.14.30 Simulated swing start

New in version python: SDK-v2.0.5

Prototype	`WeaveStartSim(weaveNum)`
Description	Simulation of swing start
Mandatory parameters	• `weaveNum`: swing parameter number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.31 End of simulation swing

New in version python: SDK-v2.0.5

Prototype	`WeaveEndSim(weaveNum)`
description	end of simulation swing
Mandatory parameters	• `weaveNum`: swing parameter number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.32 Start trajectory detection warning (no movement)

New in version python: SDK-v2.0.5

Prototype	`WeaveInspectStart(weaveNum)`
Description	Start Trajectory Detection Warning (No Movement)
Mandatory parameters	• `weaveNum`: swing parameter number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.33 End trajectory detection warning (no movement)

New in version python: SDK-v2.0.5

Prototype	`WeaveInspectEnd(weaveNum)`
Description	End Trajectory Detection Warning (No Movement)
Mandatory parameters	• `weaveNum`: swing parameter number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.34 Wobble start

New in version python: SDK-v2.1.2

Prototype	`WeaveChangeStart(weaveNum)`
Description	Wobble start
Mandatory parameters	• `weaveChangeFlag`: Swing number 1- Variable swing parameters; 2- Variable swing parameters + welding speed • `weaveNum`: Swing number • `velStart`: Welding start speed, (cm/min) • `velEnd`: Welding end speed, (cm/min)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.35 Robot swing gradient welding code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
p1Desc = [228.879, -503.594, 453.984, -175.580, 8.293, 171.267]
p1Joint = [102.700, -85.333, 90.518, -102.365, -83.932, 22.134]
p2Desc = [-333.302, -435.580, 449.866, -174.997, 2.017, 109.815]
p2Joint = [41.862, -85.333, 90.526, -100.587, -90.014, 22.135]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
robot.MoveJ(joint_pos= p1Joint,tool= 13,user= 0)
robot.WeaveStartSim(0)
robot.MoveL(desc_pos= p2Desc,tool= 13,user= 0)
robot.WeaveEndSim(0)
robot.MoveJ(joint_pos= p1Joint,tool= 13,user= 0)
robot.WeaveInspectStart(0)
robot.MoveL(desc_pos= p2Desc,tool= 13,user= 0,)
robot.WeaveInspectEnd(0)
robot.WeldingSetVoltage(1, 19, 0, 0)
robot.WeldingSetCurrent(1, 190, 0, 0)
```

(continues on next page)

(continued from previous page)

```
robot.MoveL(desc_pos= p1Desc,tool= 1,user= 1,vel= 100,acc= 100,ovl= 50)
robot.ARCStart(1, 0, 10000)
robot.ArcWeldTraceControl(1, 0, 1, 0.06, 5, 5, 60, 1, 0.06, 5, 5, 80, 0, 0, 4, 1, 10, 0,␣
↪0)
robot.WeaveStart(0)
robot.WeaveChangeStart(1, 0, 50, 30)
robot.MoveL(desc_pos= p2Desc,tool= 1,user= 1,vel= 100)
robot.WeaveChangeEnd()
robot.WeaveEnd(0)
robot.ArcWeldTraceControl(0, 0, 1, 0.06, 5, 5, 60, 1, 0.06, 5, 5, 80, 0, 0, 4, 1, 10, 0,␣
↪0)
robot.ARCEnd(1, 0, 10000)
robot.CloseRPC()
```

2.4.14.36 Wobble end

New in version python: SDK-v2.0.9-3.7.9

Prototype	`WeaveChangeEnd()`
Description	Wobble end
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.37 Extended IO-Configuration Welder Gas Detection Signal

New in version python: SDK-v2.0.5

Prototype	`SetAirControlExtDoNum(DONum)`
Description	Extended IO-Configuration Welder Gas Detection Signal
Mandatory parameters	• `DONum`: gas detection signal extension DO number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.38 Extended IO-Configuration of welder arc start signal

New in version python: SDK-v2.0.5

Prototype	`SetArcStartExtDoNum(DONum)`
Description	Extended IO-Configuration Welder Arc Start Signal
Mandatory parameters	• `DONum`: gas detection signal extension DO number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.39 Extended IO-Configuration of the welder's reverse wire feed signal

New in version python: SDK-v2.0.5

Prototype	`SetWireReverseFeedExtDoNum(DONum)`
Description	Extended IO-Configuration Welder Reverse Wire Feed Signal
Mandatory parameters	• `DONum`: gas detection signal extension DO number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.40 Extended IO-Configuration of the welder's forward wire feed signal

New in version python: SDK-v2.0.5

Prototype	`SetWireForwardFeedExtDoNum(DONum)`
Description	Extended IO-Configure welder positive wire feed signal
Mandatory parameters	• `DONum`: gas detection signal extension DO number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.41 Extended IO-Configuration of the welder's arc start success signal

New in version python: SDK-v2.0.5

Prototype	`SetArcDoneExtDiNum(DINum)`
Description	Extended IO-Configuration Welder Arc Start Success Signal
Mandatory parameters	• `DINum`: Welder Ready Signal Expansion DI Number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.42 Extended IO-Configuration Welder Ready Signal

New in version python: SDK-v2.0.5

Prototype	`SetArcDoneExtDiNum(DINum)`
Description	Extended IO-Configuration Welder Ready Signal
Mandatory parameters	• `DINum`: Welder Ready Signal Expansion DI Number
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.43 Extended IO-Configuration Weld Interrupt Recovery Signal

New in version python: SDK-v2.0.5

Prototype	`SetExtDIWeldBreakOffRecover(reWeldDINum, abortWeldDINum)`
description	Extended IO-Configuration Weld Interrupt Recovery Signal
Mandatory parameters	• `reWeldDINum`: Weld signal extension DI number for resumption of welding after a weld interruption • `abortWeldDINum`: exit weld signal extension DI number after weld interruption
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.44 Example code for setting up extended IO solder signals

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
rtn = robot.SetArcStartExtDoNum(10)
print(f"SetArcStartExtDoNum rtn is {rtn}")
rtn = robot.SetAirControlExtDoNum(20)
print(f"SetAirControlExtDoNum rtn is {rtn}")
rtn = robot.SetWireForwardFeedExtDoNum(30)
print(f"SetWireForwardFeedExtDoNum rtn is {rtn}")
rtn = robot.SetWireReverseFeedExtDoNum(40)
rtn = robot.SetWeldReadyExtDiNum(50)
print(f"SetWeldReadyExtDiNum rtn is {rtn}")
rtn = robot.SetArcDoneExtDiNum(60)
print(f"SetArcDoneExtDiNum rtn is {rtn}")
rtn = robot.SetExtDIWeldBreakOffRecover(70, 80)
print(f"SetExtDIWeldBreakOffRecover rtn is {rtn}")
rtn = robot.SetWireSearchExtDIONum(0, 1)
print(f"SetWireSearchExtDIONum rtn is {rtn}")
robot.CloseRPC()
```

2.4.14.45 Arc tracking control

New in version python: SDK-v2.0.9-3.7.9

Prototype	`ArcWeldTraceControl(flag,delaytime, isLeftRight, klr, tStartLr, stepMaxLr, sumMaxLr, isUpLow, kud, tStartUd, stepMaxUd, sumMaxUd. axisSelect, referenceType, referSampleStartUd, referSampleCountUd, referenceCurrent, offsetType, offsetParameter)`
Description	Arc tracking control
Mandatory parameters	• `flag`: switch, 0-off; 1-on • `delayTime`: lag time in ms • `isLeftRight`: left-right deviation compensation 0 - off, 1 - on • `klr`: left/right adjustment factor (sensitivity) • `tStartLr`: left and right start compensation time cyc • `stepMaxLr`: Maximum compensation in mm for each left and right step. • `sumMaxLr`: total maximum compensation left and right mm • `isUpLow`: up and down deviation compensation 0 - off, 1 - on • `kud`: upward and downward adjustment factor (sensitivity) • `tStartUd`: up and down start compensation time cyc • `stepMaxUd`: Maximum compensation per step up and down mm • `sumMaxUd`: total maximum compensation up and down • `axisSelect`: upper and lower coordinate system selection, 0 - swing; 1 - tool; 2 - base • `referenceType`: upper and lower reference current setting mode, 0 - feedback; 1 - constant • `referSampleStartUd`: upper and lower reference current sampling start count (feedback), cyc • `referSampleCountUd`: upper and lower reference current sample cycle count (feedback), cyc • `referenceCurrent`: upper and lower reference currents mA • `offsetType`: bias tracking type, 0- no bias; 1- Sampling; 2- percent • `offsetParameter`: bias parameter; Sampling (offset sampling start time, default sampling cycle); Percentage (offset percentage (-100 ~ 100))
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.46 Arc tracking AI passband selection

New in version python: SDK-v2.0.5

Prototype	`ArcWeldTraceExtAIChannelConfig(channel)`
Description	Arc Tracking AI Passband Selection
Mandatory parameters	• `channel`: arc tracking AI passband selection, [0-3]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.47 Arc tracking + multi-layer multi-channel compensation on

New in version python: SDK-v2.0.5

prototype	`ArcWeldTraceReplayStart()`
Description	Arc Tracking + Multi-Layer Multi-Channel Compensation On
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.48 Arc Tracking + Multi-Layer Multi-Channel Compensation Off

New in version python: SDK-v2.0.5

Prototype	`ArcWeldTraceReplayEnd()`
Description	Arc Tracking + Multi-Layer Multi-Channel Compensation Shutdown
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.49 Offset Coordinate Change - Multi-layer Multi-pass Welding

New in version python: SDK-v2.0.5

Prototype	`MultilayerOffsetTrsfToBase(pointo, pointX, pointZ, dx, dy, db)`
Description	Offset Coordinate Change - Multi-Layer Multi-Pass Welding
Mandatory parameters	• `pointo`: datum Cartesian orientation • `pointX`: Cartesian position of point X in the offset direction from the reference point X. • `pointZ`: Cartesian position of point Z in the direction of offset from datum Z. • `dx`: x-direction offset (mm) • `dz`: z-direction offset (mm) • `dry`: offset around the y-axis (°)
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `offset`: the offset of the result of the calculation

2.4.14.50 Example code for multi-layer multi-pass welding arc tracking

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
mulitilineorigin1_joint = [-24.090, -63.501, 84.288, -111.940, -93.426, 57.669]
mulitilineorigin1_desc = [-677.559, 190.951, -1.205, 1.144, -41.482, -82.577]
mulitilineX1_desc = [-677.556, 211.949, -1.206]
mulitilineZ1_desc = [-677.564, 190.956, 19.817]
mulitilinesafe_joint = [-25.734, -63.778, 81.502, -108.975, -93.392, 56.021]
mulitilinesafe_desc = [-677.561, 211.950, 19.812, 1.144, -41.482, -82.577]
mulitilineorigin2_joint = [-29.743, -75.623, 101.241, -116.354, -94.928, 55.735]
mulitilineorigin2_desc = [-563.961, 215.359, -0.681, 2.845, -40.476, -87.443]
mulitilineX2_desc = [-563.965, 220.355, -0.680]
mulitilineZ2_desc = [-563.968, 215.362, 4.331]
epos = [0, 0, 0, 0]
offset = [0, 0, 0, 0, 0, 0]
time.sleep(0.01)
error = robot.MoveJ(joint_pos= mulitilinesafe_joint,tool= 13,user= 0,vel= 10)
print(f"MoveJ return: {error}")
error = robot.MoveL(desc_pos= mulitilineorigin1_desc,tool= 13,user= 0,vel= 10, speedPercent=100)
print(f"MoveL return: {error}")
error = robot.MoveJ(joint_pos= mulitilinesafe_joint,tool= 13,user= 0,vel= 10)
print(f"MoveJ return: {error}")
error = robot.MoveL(desc_pos= mulitilineorigin2_desc,tool= 13,user= 0,vel= 10, speedPercent=100)
print(f"MoveL return: {error}")
error = robot.MoveJ(joint_pos= mulitilinesafe_joint,tool= 13,user= 0,vel= 10)
print(f"MoveJ return: {error}")
error = robot.MoveL(desc_pos= mulitilineorigin1_desc,tool= 13,user= 0,vel= 10, speedPercent=100)
print(f"MoveL return: {error}")
error = robot.ARCStart(1, 0, 3000)
print(f"ARCStart return: {error}")
error = robot.WeaveStart(0)
print(f"WeaveStart return: {error}")
error = robot.ArcWeldTraceControl(1, 0, 1, 0.06, 5, 5, 50, 1, 0.06, 5, 5, 55, 0, 0, 4, 1, 10,0,0)
print(f"ArcWeldTraceControl return: {error}")
error = robot.MoveL(desc_pos= mulitilineorigin2_desc,tool= 13,user= 0,vel= 1, speedPercent=100)
print(f"MoveL return: {error}")
error = robot.ArcWeldTraceControl(0, 0, 1, 0.06, 5, 5, 50, 1, 0.06, 5, 5, 55, 0, 0, 4, 1, 10,0,0)
print(f"ArcWeldTraceControl return: {error}")
error = robot.WeaveEnd(0)
print(f"WeaveEnd return: {error}")
error = robot.ARCEnd(1, 0, 10000)
print(f"ARCEnd return: {error}")
error = robot.MoveJ(joint_pos= mulitilinesafe_joint,tool= 13,user= 0,vel= 10)
```

(continues on next page)

(continued from previous page)

```
print(f"MoveJ return: {error}")
error,offset = robot.MultilayerOffsetTrsfToBase(mulitilineorigin1_desc[:3], mulitilineX1_desc, mulitilineZ1_desc, 10.0, 0.0, 0.0)
print(f"MultilayerOffsetTrsfToBase return: {error}")
error = robot.MoveL(desc_pos= mulitilineorigin1_desc,tool= 13,user= 0,vel= 10, speedPercent=100)
print(f"MoveL return: {error}")
error = robot.ARCStart(1, 0, 3000)
print(f"ARCStart return: {error}")
error, offset = robot.MultilayerOffsetTrsfToBase(mulitilineorigin2_desc[:3], mulitilineX2_desc, mulitilineZ2_desc, 10, 0, 0)
print(f"MultilayerOffsetTrsfToBase return: {error}")
error = robot.ArcWeldTraceReplayStart()
print(f"ArcWeldTraceReplayStart return: {error}")
error = robot.MoveL(desc_pos= mulitilineorigin2_desc,tool= 13,user= 0,vel= 2, speedPercent=100)
print(f"MoveL return: {error}")
error = robot.ArcWeldTraceReplayEnd()
print(f"ArcWeldTraceReplayEnd return: {error}")
error = robot.ARCEnd(1, 0, 10000)
print(f"ARCEnd return: {error}")
error = robot.MoveJ(joint_pos= mulitilinesafe_joint,tool= 13,user= 0,vel= 10)
print(f"MoveJ return: {error}")
error, offset = robot.MultilayerOffsetTrsfToBase(mulitilineorigin1_desc[:3], mulitilineX1_desc, mulitilineZ1_desc, 0, 10, 0)
print(f"MultilayerOffsetTrsfToBase return: {error}")
error = robot.MoveL(desc_pos= mulitilineorigin1_desc,tool= 13,user= 0,vel= 10, speedPercent=100)
print(f"MoveL return: {error}")
error = robot.ARCStart(1, 0, 3000)
print(f"ARCStart return: {error}")
error, offset = robot.MultilayerOffsetTrsfToBase(mulitilineorigin2_desc[:3], mulitilineX2_desc, mulitilineZ2_desc, 0, 10, 0)
error = robot.ArcWeldTraceReplayStart()
print(f"ArcWeldTraceReplayStart return: {error}")
error = robot.MoveL(desc_pos= mulitilineorigin2_desc,tool= 13,user= 0,vel= 2, speedPercent=100)
print(f"MoveL return: {error}")
error = robot.ArcWeldTraceReplayEnd()
print(f"ArcWeldTraceReplayEnd return: {error}")
error = robot.ARCEnd(1, 0, 3000)
print(f"ARCEnd return: {error}")
error = robot.MoveJ(joint_pos= mulitilinesafe_joint,tool= 13,user= 0,vel= 10)
print(f"MoveJ return: {error}")
robot.CloseRPC()
```

2.4.14.51 Selection of AI channels for current feedback in arc tracking welding machines

New in version python: SDK-v2.1.2

Prototype	`ArcWeldTraceAIChannelCurrent(channel)`
Description	Selection of AI channels for current feedback in arc tracking welding machines
Mandatory parameters	• `channel`: Passage 0- Expand AI0; 1- Expand AI1; 2- Expand AI2; 3- Expand AI3; 4- Control Box AI0 5- Control Box AI1
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.52 Selection of AI channel for voltage feedback of arc tracking welding machine

New in version python: SDK-v2.1.2

Prototype	`ArcWeldTraceAIChannelVoltage(channel)`
Description	Selection of AI channel for voltage feedback of arc tracking welding machine
Mandatory parameters	• `channel`: Passage 0- Expand AI0; 1- Expand AI1; 2- Expand AI2; 3- Expand AI3; 4- Control Box AI0 5- Control Box AI1
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.53 Current feedback conversion parameters of arc tracking welding machine

New in version python: SDK-v2.1.2

Prototype	`ArcWeldTraceCurrentPara(AILow, AIHigh, currentLow, currentHigh)`
Description	Current feedback conversion parameters of arc tracking welding machine
Mandatory parameters	NULL
Default parameters	• `AILow`: Lower limit of AI channel, default value 0V, range [0-10V] • `AIHigh`: AI channel upper limit, default value 10V, range [0-10V] • `currentLow`: The lower limit of the AI channel corresponds to the current value of the welding machine. The default value is 0V, and the range is [0-200V] • `currentHigh`: The upper limit of the AI channel corresponds to the current value of the welding machine. The default value is 100V, and the range is [0-200V]
Return Value	Error Code Success-0 Failure- errcode

2.4.14.54 Voltage feedback conversion parameters of arc tracking welding machine

New in version python: SDK-v2.1.2

Prototype	`ArcWeldTraceVoltagePara(AILow, AIHigh, voltageLow, voltageHigh)`
Description	Voltage feedback conversion parameters of arc tracking welding machine
Mandatory parameters	NULL
Default parameters	• `AILow`: Lower limit of AI channel, default value 0V, range [0-10V] • `AIHigh`: AI channel upper limit, default value 10V, range [0-10V] • `voltageLow`: The lower limit of the AI channel corresponds to the welding machine voltage value. The default value is 0V, and the range is [0-200V] • `voltageHigh`: The upper limit of the AI channel corresponds to the voltage value of the welding machine. The default value is 100V, and the range is [0-200V]
Return Value	Error Code Success-0 Failure- errcode

2.4.14.55 Example arc tracking code

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')

safetydescPose = [-504.043,275.181,40.908,-28.002,-42.025,-14.044]
safetyjointPos = [-39.078,-76.732,87.227,-99.47,-94.301,18.714]
startdescPose = [-473.86,257.879,-20.849,-37.317,-42.021,2.543]
startjointPos = [-43.487,-76.526,95.568,-104.445,-89.356,3.72]
enddescPose = [-499.844,141.225,7.72,-34.856,-40.17,13.13]
endjointPos = [-31.305,-82.998,99.401,-104.426,-89.35,3.696]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
robot.MoveJ(joint_pos=safetyjointPos, tool=1, user=0, vel=20, acc=100)
robot.WeldingSetCurrentRelation(0, 495, 1, 10, 0)
robot.WeldingSetVoltageRelation(10, 45, 1, 10, 1)
robot.WeldingSetVoltage(0, 25, 1, 0)
robot.WeldingSetCurrent(0, 260, 0, 0)
rtn = robot.ArcWeldTraceAIChannelCurrent(4)
print("ArcWeldTraceAIChannelCurrent rtn is", rtn)
rtn = robot.ArcWeldTraceAIChannelVoltage(5)
print("ArcWeldTraceAIChannelVoltage rtn is", rtn)
rtn = robot.ArcWeldTraceCurrentPara(0, 5, 0, 500)
print("ArcWeldTraceCurrentPara rtn is", rtn)
rtn = robot.ArcWeldTraceVoltagePara(1.018, 10, 0, 50)
print("ArcWeldTraceVoltagePara rtn is", rtn)
robot.MoveJ(joint_pos=startjointPos, tool=1, user=0, vel=20, acc=100)
robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0)
robot.ARCStart(0, 0, 10000)
robot.WeaveStart(0)
robot.MoveL(desc_pos=enddescPose, tool=1, user=0, vel=100, ovl= 2, acc=100)
```

(continues on next page)

(continued from previous page)

```
robot.ARCEnd(0, 0, 10000)
robot.WeaveEnd(0)
robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10, 0, 0)
robot.MoveJ(joint_pos=safetyjointPos, tool=1, user=0, vel=20, acc=100)
```

2.4.14.56 Setting Up the Weld Wire Seek Expansion IO Port

New in version python: SDK-v2.0.5

Prototype	`SetWireSearchExtDIONum(searchDoneDINum, searchStartDONum)`
Description	Setting up the Weld Wire Seeker Expansion IO Port
Mandatory parameter	• `searchDoneDINum`: weld wire seek success DO port (0-127) • `searchStartDONum`: wire seek start/stop control DO port (0-127)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.57 code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
toolCoord = [0, 0, 200, 0, 0, 0]
robot.SetToolCoord(1, toolCoord, 0, 0, 1, 0)
wobjCoord = [0, 0, 0, 0, 0, 0]
robot.SetWObjCoord(1, wobjCoord, 0)
robot.ExtDevSetUDPComParam("192.168.58.88", 2021, 2, 50, 5, 50, 1, 50, 10)
robot.ExtDevLoadUDPDriver()
robot.SetWireSearchExtDIONum(0, 0)
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
descStart = [216.543, 445.175, 93.465, 179.683, 1.757, -112.641]
jointStart = [-128.345, -86.660, 114.679, -119.625, -89.219, 74.303]
descEnd = [111.143, 523.384, 87.659, 179.703, 1.835, -97.750]
jointEnd = [-113.454, -81.060, 109.328, -119.954, -89.218, 74.302]
error = robot.MoveL(desc_pos=descStart,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos=descEnd,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
descREF0A = [142.135, 367.604, 86.523, 179.728, 1.922, -111.089]
jointREF0A = [-126.794, -100.834, 128.922, -119.864, -89.218, 74.302]
descREF0B = [254.633, 463.125, 72.604, 179.845, 2.341, -114.704]
jointREF0B = [-130.413, -81.093, 112.044, -123.163, -89.217, 74.303]
descREF1A = [92.556, 485.259, 47.476, -179.932, 3.130, -97.512]
jointREF1A = [-113.231, -83.815, 119.877, -129.092, -89.217, 74.303]
```

(continues on next page)

(continued from previous page)

```
descREF1B = [203.103, 583.836, 63.909, 179.991, 2.854, -103.372]
jointREF1B = [-119.088, -69.676, 98.692, -121.761, -89.219, 74.303]
error = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchStart return: {error}")
error = robot.MoveL(desc_pos=descREF0A,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos=descREF0B,tool= 1,user= 1,vel= 100,search=1)
print(f"MoveL return: {error}")
error = robot.WireSearchWait("REF0")
print(f"WireSearchWait return: {error}")
error = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchEnd return: {error}")
error = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchStart return: {error}")
error = robot.MoveL(desc_pos= descREF1A,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos= descREF1B,tool= 1,user= 1,vel= 100,search=1)
print(f"MoveL return: {error}")
error = robot.WireSearchWait("REF1")
print(f"WireSearchWait return: {error}")
error = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0)
error = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchStart return: {error}")
error = robot.MoveL(desc_pos= descREF0A,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos= descREF0B,tool= 1,user= 1,vel= 100,search=1)
print(f"MoveL return: {error}")
error = robot.WireSearchWait("RES0")
print(f"WireSearchWait return: {error}")
error = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchEnd return: {error}")
error = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchStart return: {error}")
error = robot.MoveL(desc_pos= descREF1A,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos= descREF1B,tool= 1,user= 1,vel= 100,search=1)
print(f"MoveL return: {error}")
error = robot.WireSearchWait("RES1")
print(f"WireSearchWait return: {error}")
error = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchEnd return: {error}")
varNameRef = ["REF0", "REF1", "#", "#", "#", "#"]
varNameRes = ["RES0", "RES1", "#", "#", "#", "#"]
offectFlag = 0
offectPos = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
error, offectFlag, offectPos = robot.GetWireSearchOffset(0, 0, varNameRef, varNameRes)
print(f"GetWireSearchOffset return: {error}")
error = robot.PointsOffsetEnable(0, offectPos)
print(f"PointsOffsetEnable return: {error}")
error = robot.MoveL(desc_pos= descStart,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos= descEnd,tool= 1,user= 1,vel= 100,search=1)
```

(continues on next page)

(continued from previous page)

```
print(f"MoveL return: {error}")
error = robot.PointsOffsetDisable()
robot.CloseRPC()
```

2.4.14.58 Welding wire position finding start

New in version python: SDK-v2.0.5

Prototype	`WireSearchStart(refPos,searchVel,searchDis,autoBackFlag, autoBackVel,autoBackDis,offectFlag)`
Description	Welding wire seek start
Mandatory parameters	• `refPos`: 1 - reference point 0 - contact point • `searchVel`: search velocity % • `searchDis`: search distance mm • `autoBackFlag`: autoBackFlag, 0 - not auto; - auto • `autoBackVel`: Automatic Back Velocity % • `autoBackDis`: automatic back distance mm • `offectFlag`: 1 - seek with offset; 0 - teach point seek
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.59 End of wire position finding

New in version python: SDK-v2.0.5

Prototype	`WireSearchEnd(refPos,searchVel,searchDis,autoBackFlag, autoBackVel,autoBackDis,offectFlag)`
Description	Welding wire seek end
Mandatory parameters	• `refPos`: 1 - reference point 2 - contact point • `searchVel`: search velocity % • `searchDis`: search distance mm • `autoBackFlag`: autoBackFlag, 0 - not auto; - auto • `autoBackVel`: Automatic Back Velocity % • `autoBackDis`: automatic back distance mm • `offectFlag`: 1 - seek with offset; 2 - teach point seek
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.60 Calculate the wire finding offset

New in version python: SDK-v2.0.5

Prototype	`GetWireSearchOffset(seamType, method,varNameRef,varNameRes)`
Description	Calculate the welding wire seek offset
Mandatory parameters	• `seamType`: weld type • `method`: method of calculation • `varNameRef`: datums 1-6, "#" denotes a pointless variable • `varNameRes`: Contacts 1-6, "#" for dotless variables
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `offsetFlag`: 0 - the offset is superimposed directly on the instruction point; 1 - the offset requires a coordinate transformation of the instruction point • `offset`: offset position [x, y, z, a, b, c]

2.4.14.61 Waiting for wire seek to complete

New in version python: SDK-v2.0.5

Prototype	`WireSearchWait(varname)`
Description	Waiting for wire seek to complete.
Mandatory parameters	• `varName`: contact name "RES0" ~ "RES99"
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.62 Wire seek contact points written to database

New in version python: SDK-v2.0.5

Prototype	`SetPointToDatabase(varName,pos)`
Description	Welding wire seeker contact points written to database
Mandatory parameters	• `varName`: contact name "RES0" ~ "RES99" • `pos`: contact point data [x, y, x, a, b, c]
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.63 Example of robot wire locating code

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
toolCoord = [0, 0, 200, 0, 0, 0]
robot.SetToolCoord(1, toolCoord, 0, 0, 1, 0)
wobjCoord = [0, 0, 0, 0, 0, 0]
robot.SetWObjCoord(1, wobjCoord, 0)
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
descStart = [216.543, 445.175, 93.465, 179.683, 1.757, -112.641]
jointStart = [-128.345, -86.660, 114.679, -119.625, -89.219, 74.303]
descEnd = [111.143, 523.384, 87.659, 179.703, 1.835, -97.750]
jointEnd = [-113.454, -81.060, 109.328, -119.954, -89.218, 74.302]
error = robot.MoveL(desc_pos=descStart,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos=descEnd,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
descREF0A = [142.135, 367.604, 86.523, 179.728, 1.922, -111.089]
jointREF0A = [-126.794, -100.834, 128.922, -119.864, -89.218, 74.302]
descREF0B = [254.633, 463.125, 72.604, 179.845, 2.341, -114.704]
jointREF0B = [-130.413, -81.093, 112.044, -123.163, -89.217, 74.303]
descREF1A = [92.556, 485.259, 47.476, -179.932, 3.130, -97.512]
jointREF1A = [-113.231, -83.815, 119.877, -129.092, -89.217, 74.303]
descREF1B = [203.103, 583.836, 63.909, 179.991, 2.854, -103.372]
jointREF1B = [-119.088, -69.676, 98.692, -121.761, -89.219, 74.303]
error = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchStart return: {error}")
error = robot.MoveL(desc_pos=descREF0A,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos=descREF0B,tool= 1,user= 1,vel= 100,search=1)
print(f"MoveL return: {error}")
error = robot.WireSearchWait("REF0")
print(f"WireSearchWait return: {error}")
error = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchEnd return: {error}")
error = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchStart return: {error}")
error = robot.MoveL(desc_pos= descREF1A,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos= descREF1B,tool= 1,user= 1,vel= 100,search=1)
print(f"MoveL return: {error}")
error = robot.WireSearchWait("REF1")
print(f"WireSearchWait return: {error}")
error = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0)
error = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchStart return: {error}")
error = robot.MoveL(desc_pos= descREF0A,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos= descREF0B,tool= 1,user= 1,vel= 100,search=1)
```

(continues on next page)

(continued from previous page)

```
print(f"MoveL return: {error}")
error = robot.WireSearchWait("RES0")
print(f"WireSearchWait return: {error}")
error = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchEnd return: {error}")
error = robot.WireSearchStart(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchStart return: {error}")
error = robot.MoveL(desc_pos= descREF1A,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos= descREF1B,tool= 1,user= 1,vel= 100,search=1)
print(f"MoveL return: {error}")
error = robot.WireSearchWait("RES1")
print(f"WireSearchWait return: {error}")
error = robot.WireSearchEnd(0, 10, 100, 0, 10, 100, 0)
print(f"WireSearchEnd return: {error}")
varNameRef = ["REF0", "REF1", "#", "#", "#", "#"]
varNameRes = ["RES0", "RES1", "#", "#", "#", "#"]
offectFlag = 0
offectPos = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
error, offectFlag, offectPos = robot.GetWireSearchOffset(0, 0, varNameRef, varNameRes)
print(f"GetWireSearchOffset return: {error}")
error = robot.PointsOffsetEnable(0, offectPos)
print(f"PointsOffsetEnable return: {error}")
error = robot.MoveL(desc_pos= descStart,tool= 1,user= 1,vel= 100)
print(f"MoveL return: {error}")
error = robot.MoveL(desc_pos= descEnd,tool= 1,user= 1,vel= 100,search=1)
print(f"MoveL return: {error}")
error = robot.PointsOffsetDisable()
robot.CloseRPC()
```

2.4.14.64 Set the welding voltage to start gradually

New in version python: SDK-v2.1.2

Prototype	`WeldingSetVoltageGradualChangeStart(IOType, voltageStart, voltageEnd, AOIndex, blend)`
Description	Set the welding voltage to start gradually
Mandatory parameters	• `IOType`: Control type 0- Control Box IO 1- Digital Communication Protocol (UDP) 2- Digital Communication Protocol (ModbusTCP) • `voltageStart`: Initial welding voltage (V) • `voltageEnd`: Termination welding voltage (V) • `AOIndex`: The AO port number of the control box (0-1) • `blend`: Whether smooth 0- Not smooth; 1- Smooth
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.65 Set the welding voltage gradient to end

New in version python: SDK-v2.1.2

Prototype	`WeldingSetVoltageGradualChangeEnd()`
Description	Set the welding voltage gradient to end
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.66 Set the welding current to start gradually

New in version python: SDK-v2.1.2

Prototype	`WeldingSetCurrentGradualChangeStart(IOType, currentStart, currentEnd, AOIndex, blend)`
Description	Set the welding current to start gradually
Mandatory parameters	• `IOType`: Control type 0- Control Box IO 1- Digital Communication Protocol (UDP) 2- Digital Communication Protocol (ModbusTCP) • `currentStart`: Initial welding current (A) • `currentEnd`: Termination welding current (A) • `AOIndex`: The AO port number of the control box (0-1) • `blend`: Whether smooth 0- Not smooth; 1- Smooth
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.67 Set the welding current to gradually end

New in version python: SDK-v2.1.2

Prototype	`WeldingSetCurrentGradualChangeEnd()`
Description	Set the welding current to gradually end
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.68 Robot welding current voltage gradient code example

```
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
startdescPose = [-484.707, 276.996, -14.013, -37.657, -40.508, -1.548]
startjointPos = [-45.421, -75.673, 93.627, -104.302, -87.938, 6.005]
enddescPose = [-508.767, 137.109, -13.966, -37.639, -40.508, -1.559]
endjointPos = [-32.768, -80.947, 100.254, -106.201, -87.201, 18.648]
```

(continues on next page)

(continued from previous page)

```
safedescPose = [-484.709, 294.436, 13.621, -37.660, -40.508, -1.545]
safejointPos = [-46.604, -75.410, 89.109, -100.003, -88.012, 4.823]
exaxisPos = [0, 0, 0, 0]
offdese = [0, 0, 0, 0, 0, 0]
robot.WeldingSetCurrentRelation(0, 495, 1, 10, 0)
robot.WeldingSetVoltageRelation(10, 45, 1, 10, 1)
robot.WeldingSetVoltage(0, 25, 1, 0)
robot.WeldingSetCurrent(0, 260, 0, 0)
robot.MoveJ(joint_pos=safejointPos, tool=1, user=0, vel=5, acc=100)
rtn = robot.WeldingSetCurrentGradualChangeStart(0, 260, 220, 0, 0)
print("WeldingSetCurrentGradualChangeStart rtn is", rtn)
rtn = robot.WeldingSetVoltageGradualChangeStart(0, 25, 22, 1, 0)
print("WeldingSetVoltageGradualChangeStart rtn is", rtn)
rtn = robot.ArcWeldTraceControl(1, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1,
↪ 10, 0, 0)
print("ArcWeldTraceControl rtn is", rtn)
robot.MoveJ(joint_pos=startjointPos, tool=1, user=0, vel=5, acc=100)
robot.ARCStart(0, 0, 10000)
robot.WeaveStart(0)
robot.WeaveChangeStart(2, 1, 24, 36)
robot.MoveL(desc_pos=enddescPose, tool=1, user=0, vel=100, ovl=2, acc=100)
robot.ARCEnd(0, 0, 10000)
robot.WeaveChangeEnd()
robot.WeaveEnd(0)
robot.ArcWeldTraceControl(0, 0, 1, 0.08, 5, 5, 300, 1, 0.06, 4, 4, 300, 1, 0, 4, 1, 10,␣
↪0, 0)
robot.WeldingSetCurrentGradualChangeEnd()
robot.WeldingSetVoltageGradualChangeEnd()
```

2.4.14.69 Set custom swing parameters

New in version python: SDK-v2.1.6

Prototype	`CustomWeaveSetPara(id, pointNum, point, stayTime, frequency, incStayType, stationary)`
Description	Set custom swing parameters
Mandatory parameters	• `id`Custom swing number: 0-2 • `pointNum`The number of swing points is 0-10 • `point`Move endpoint data x,y,z • `stayTime`The swing residence time is ms • `frequency`Swing frequency Hz • `incStayType`Waiting mode: 0-cycle does not contain waiting time; The 1-cycle contains the waiting time • `stationary`Swing position waiting: 0- continue to move within the waiting time; 1 -The position is stationary during the waiting time
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.14.70 Gets custom swing parameters

New in version python: SDK-v2.1.6

Prototype	`CustomWeaveGetPara(id)`
Description	Gets custom swing parameters
Mandatory parameters	• `id`Custom swing number: 0-2
Default parameters	NULL
Return Value	• Error Code Success-0 Failure- errcode • `pointNum`The number of swing points is 0-10 • `point`Move endpoint data x,y,z • `stayTime`The swing residence time is ms • `frequency`Swing frequency Hz • `incStayType`Waiting mode: 0-cycle does not contain waiting time; The 1-cycle contains the waiting time • `stationary`Swing position waiting: 0- continue to move within the waiting time; 1 -The position is stationary during the waiting time

2.4.14.71 Custom swing parameter code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
point = [0.0] * 30
point[0] = -3.0
point[1] = -3.0
point[2] = 0.0
point[3] = -6.0
point[4] = 0.0
point[5] = 0.0
point[6] = -3.0
point[7] = 3.0
point[8] = 0.0
point[9] = 0.0
point[10] = 0.0
point[11] = 0.0
stayTime = [0.0] * 10
rtn = robot.CustomWeaveSetPara(2, 4, point, stayTime, 1.000, 0, 0)
print(f"CustomWeaveSetPara rtn is {rtn}")
time.sleep(1)
pointNum = 0
frequency = 0.0
incStayType = 0
stationary = 0
rtn, pointNum, point, stayTime, frequency, incStayType, stationary = robot.CustomWeaveGetPara(2)
print(f"pointNum is {pointNum}")
```

(continues on next page)

(continued from previous page)

```
for i in range(pointNum):
    print(f"point {i}, point x y z {point[i * 3 + 0]},{point[i * 3 + 1]},{point[i * 3 + 2]}")
print(f"fre is {frequency}, stay is {incStayType},{stationary}")
robot.WeaveSetPara(0, 9, 1.000000, 1, 5.000000, 6.000000, 5.000000, 50, 100, 100, 0, 1, 0.000000, 0.000000)
desc_p1 = [-288.650, 367.807, 288.404, 0.000, -0.001, 0.001]
desc_p2 = [-431.714, 367.815, 288.415, 0.001, 0.001, 0.000]
desc_p3 = [-348.666, 427.798, 288.404, -0.000, -0.000, 0.001]
j1 = [140.656, -84.560, -91.707, -93.734, 90.000, 50.655]
j2 = [149.873, -98.298, -77.599, -94.103, 90.000, 59.873]
j3 = [139.773, -96.173, -80.014, -93.814, 90.000, 49.772]
epos = [0.0] * 4
offset_pos = [0.0] * 6
robot.MoveJ(joint_pos=j1, tool=3, user=0, vel=100)
robot.WeaveStart(0)
robot.Circle(desc_pos_p=desc_p3, tool_p=3, user_p=0, vel_p=50, desc_pos_t=desc_p2, tool_t=3, user_t=0, vel_t=50, oacc=10)
robot.WeaveEnd(0)
robot.MoveJ(joint_pos=j1, tool=3, user=0, vel=100)
robot.WeaveStart(0)
robot.MoveC(desc_pos_p=desc_p3, tool_p=3, user_p=0, vel_p=50, desc_pos_t=desc_p2, tool_t=3, user_t=0, vel_t=50)
robot.WeaveEnd(0)
robot.MoveJ(joint_pos=j1, tool=3, user=0, vel=100)
robot.WeaveStart(0)
robot.MoveL(desc_pos=desc_p2, tool=3, user=0, vel=100, ovl=10, speedPercent=100)
robot.WeaveEnd(0)
robot.CloseRPC()
```

2.4.15 Others

2.4.15.1 Get SSH public key

Prototype	`GetSSHKeygen()`
description	Get SSH public key
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • keygen: public key

2.4.15.2 Issue the SCP command

New in version python: SDK-v2.1.3

Prototype	`SetSSHScpCmd(mode, sshname, sship, usr_file_url, robot_file_url)`
Description	Issue the SCP command
Mandatory parameters	• `mode`: 0- Upload (Host Computer -> Controller), 1- Download (Controller -> Host Computer) • `sshname`: User name of the upper computer • `sship`: The ip address of the upper computer • `usr_file_url`: File path of the upper computer • `robot_file_url`: Robot controller file path
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.3 Calculate the MD5 value of a file in a specified path

Prototype	`ComputeFileMD5(file_path)`
Description	Calculates the MD5 value of a file in the specified path.
Mandatory parameter	• `file_path`: file path including filename, default Traj folder path is :/fruser/traj/, e.g. /fruser/traj/trajHelix_aima_1.txt
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `md5`: the MD5 value of the file.

2.4.15.4 Robot SSH, MD5 instruction code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
file_path = "/fruser/airlab.lua"
md5 = ""
emerg_state = 0
si0_state = 0
si1_state = 0
sdk_com_state = 0
ssh_keygen = ""
retval,ssh_keygen = robot.GetSSHKeygen()
print(f"GetSSHKeygen retval is: {retval}")
print(f"ssh key is: {ssh_keygen}")
ssh_name = "fr"
ssh_ip = "192.168.58.45"
```

(continues on next page)

(continued from previous page)

```
ssh_route = "/home/fr"
ssh_robot_url = "/root/robot/dhpara.config"
retval = robot.SetSSHScpCmd(1, ssh_name, ssh_ip, ssh_route, ssh_robot_url)
print(f"SetSSHScpCmd retval is: {retval}")
print(f"robot url is: {ssh_robot_url}")
error, md5 = robot.ComputeFileMD5(file_path)
print(f"md5 is: {md5}")
robot.CloseRPC()
```

2.4.15.5 Setting the Robot 20004 Port Feedback Cycle

New in version python: SDK-v2.0.5

Prototype	`SetRobotRealtimeStateSamplePeriod(period)`
Description	Setting the robot 20004 port feedback cycle
Mandatory parameter	• `period`: robot 20004 port feedback period (ms)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.6 Get robot 20004 port feedback cycle

New in version python: SDK-v2.0.5

Prototype	`GetRobotRealtimeStateSamplePeriod()`
Description	Get robot 20004 port feedback cycle
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `period`: robot 20004 port feedback period (ms)

2.4.15.7 Robot 20004 port state feedback cycle configuration code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
robot.SetRobotRealtimeStateSamplePeriod(10)
error,getPeriod = robot.GetRobotRealtimeStateSamplePeriod()
print(f"period is {getPeriod}")
time.sleep(1)
robot.CloseRPC()
```

2.4.15.8 Robot software upgrade

New in version python: SDK-v2.0.5

Prototype	`SoftwareUpgrade(filePath, block)`
Description	Robot Software Upgrade
Required parameters	• `filePath`: full path of the software upgrade package • `block`: whether to block until the upgrade is complete true:blocking; false:non-blocking
Default parameters	NULL
Return Value	• errcode Success-0 Failure- errcode

2.4.15.9 Get robot software upgrade status

New in version python: SDK-v2.0.5

Prototype	`GetSoftwareUpgradeState()`
Description	Get robot software upgrade status
Mandatory parameters	NULL
Default parameters	NULL
Return Value	• errorcode Success-0 Failure- errcode • `state`: robot package upgrade status, 0: idle in progress or uploading upgrade package in progress, 1~100: percentage of upgrade completed, -1: upgrade software failure, -2: checksum failure, -3: version checksum failure, -4: unpacking failure, -5: user configuration upgrade failure, -6: peripheral configuration upgrade failure, -7: extended axis configuration upgrade failure, -8: robot configuration upgrade failure, -9: DH parameter configuration upgrade failure

2.4.15.10 Robot software upgrade code example

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the␣
↪connection is successful
robot = Robot.RPC('192.168.58.2')
error = robot.SoftwareUpgrade("D://zUP/QNX382/software.tar.gz", False)
print(f"SoftwareUpgrade error is {error}")
while True:
    curState = robot.GetSoftwareUpgradeState()
    print(f"upgrade state is {curState}")
    time.sleep(3)
robot.CloseRPC()
```

2.4.15.11 Download Point Table Database

New in version python: SDK-v2.0.1

Prototype	`PointTableDownLoad(point_table_name, save_file_path)`
Description	Download Points Table Database
Required Parameters	• `point_table_name`: name of the point table to be downloaded pointTable1.db; • `save_file_path`: storage path to download the point table C://test/;
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.12 Upload point table database

New in version python: SDK-v2.0.1

Prototype	`PointTableUpLoad(point_table_file_path)`
Description	Upload point table database
Required Parameters	• `point_table_file_path`: full pathname of uploaded point table C://test/pointTable1.db
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.13 Point table update lua file

New in version python: SDK-v2.0.1

prototype	`PointTableUpdateLua(point_table_name, lua_file_name)`
Description	Point table update lua file
Required Parameters	• `point_table_name`: the name of the point table to be switched, pointTable1.db, when the point table is empty, i.e. ‘‘, it means updating the lua program to the initial program with no point table applied • `lua_file_name`: name of the lua file to update testPointTable.lua
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.14 Robot point table operation code example

```python
from fairino import Robot
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
save_path = "D://zDOWN/"
point_table_name = "point_table_FR5.db"
rtn = robot.PointTableDownLoad(point_table_name, save_path)
print(f"download : {point_table_name} fail: {rtn}")
upload_path = "D://zDOWN/point_table_FR5.db"
rtn = robot.PointTableUpLoad(upload_path)
print(f"retval is: {rtn}")
point_tablename = "point_table_FR5.db"
lua_name = "test0610.lua"
rtn,error = robot.PointTableUpdateLua(point_tablename, lua_name)
print(f"retval is: {rtn}")
robot.CloseRPC()
```

2.4.15.15 Controller log download

New in version python: SDK-v2.1.1

Prototype	`RbLogDownload(savePath)`
Description	Controller log download
Mandatory parameters	• `savePath`: Save the file path D://zDown/
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.16 Download all data sources

New in version python: SDK-v2.1.1

Prototype	`AllDataSourceDownload(savePath)`
Description	Download all data sources
Mandatory parameters	• `savePath`: Save the file path D://zDown/
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.17 Data backup package download

New in version python: SDK-v2.1.1

Prototype	`DataPackageDownload(savePath)`
Description	Data backup package download
Mandatory parameters	• `savePath`: Save the file path D://zDown/
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.18 Download the controller data code example

```python
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
rtn = robot.RbLogDownload("D://zDOWN/")
print(f"RbLogDownload rtn is {rtn}")
rtn = robot.AllDataSourceDownload("D://zDOWN/")
print(f"AllDataSourceDownload rtn is {rtn}")
rtn = robot.DataPackageDownload("D://zDOWN/")
print(f"DataPackageDownload rtn is {rtn}")
robot.CloseRPC()
```

2.4.15.19 Set up encoder upgrade

New in version python: SDK-v2.1.4

Prototype	`SetEncoderUpgrade(path)`
Description	Set up encoder upgrade
Mandatory parameters	• `path`Local upgrade package full path(D://zUP/XXXXX.bin)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.20 Set up joint firmware upgrade

New in version python: SDK-v2.1.4

Prototype	`SetJointFirmwareUpgrade(type, path)`
Description	Set up joint firmware upgrade
Mandatory parameters	• `type`Updating file types 1- Upgrade firmware; 2- Upgrade the slave profile • `path`Local upgrade package full path(D://zUP/XXXXX.bin)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.21 Set up control box firmware upgrade

New in version python: SDK-v2.1.4

Prototype	`SetCtrlFirmwareUpgrade(type, path)`
Description	Set up control box firmware upgrade
Mandatory parameters	• `type`Updating file types 1- Upgrade firmware; 2- Upgrade the slave profile • `path`Local upgrade package full path(D://zUP/XXXXX.bin)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.22 Set up the end firmware upgrade

New in version python: SDK-v2.1.4

Prototype	`SetEndFirmwareUpgrade(type, path)`
Description	Set up the end firmware upgrade
Mandatory parameters	• `type`Updating file types 1- Upgrade firmware; 2- Upgrade the slave profile • `path`Local upgrade package full path(D://zUP/XXXXX.bin)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.23 Joint full parameter profile upgrade

New in version python: SDK-v2.1.4

Prototype	`JointAllParamUpgrade(path)`
Description	Joint full parameter profile upgrade
Mandatory parameters	• `path`Local upgrade package full path(D://zUP/XXXXX.bin)
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.24 Example of robot slave firmware upgrade code

```
from fairino import Robot
import time
# Establish a connection with the robot controller and return a robot object if the connection is successful
robot = Robot.RPC('192.168.58.2')
robot.RobotEnable(0)
time.sleep(0.2)
rtn = robot.JointAllParamUpgrade("D://zUP/MT/joint0603/jointallparameters.db")
print(f"robot JointAllParamUpgrade rtn is {rtn}")
rtn = robot.SetCtrlFirmwareUpgrade(2, "D://zUP/MT/FAIR_Cobot_Cbd_Asix_V2.0.bin")
print(f"robot SetCtrlFirmwareUpgrade rtn is {rtn}")
rtn = robot.SetEndFirmwareUpgrade(2, "D://zUP/MT/FAIR_Cobot_Axle_Asix_V2.4.bin")
print(f"robot SetEndFirmwareUpgrade rtn is {rtn}")
robot.SetSysServoBootMode()
time.sleep(0.2)
rtn = robot.SetCtrlFirmwareUpgrade(1, "D://zUP/MT/FR_CTRL_PRIMCU_FV201412_MAIN_U4_T01_20250630(MT).bin")
print(f"robot SetCtrlFirmwareUpgrade rtn is {rtn}")
rtn = robot.SetEndFirmwareUpgrade(1, "D://zUP/MT/FR_END_FV2010010_MAIN_U1_T01_20250603.bin")
print(f"robot SetEndFirmwareUpgrade rtn is {rtn}")
rtn = robot.SetJointFirmwareUpgrade(1, "D://zUP/MT/FR_SERVO_FV504215_MAIN_U7_T07_20250603.bin")
print(f"robot SetJointFirmwareUpgrade rtn is {rtn}")
robot.CloseRPC()
```

2.4.15.25 Robot Operating System upgrade (LA control box)

New in version python: SDK-v2.1.6

Prototype	`KernelUpgrade(filePath)`
Description	Robot Operating System upgrade (LA control box)
Mandatory parameters	• `filePath`Operating system upgrade package full path
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.26 Obtain upgrade results of robot operating system (LA control box)

New in version python: SDK-v2.1.6

Prototype	`GetKernelUpgradeResult()`
Description	Obtain upgrade results of robot operating system (LA control box)
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.15.27 Robot MCU log generation

New in version python: SDK-v2.1.7

Prototype	`RobotMCULogCollect()`
Description	Robot MCU log generation
Mandatory parameters	NULL
Default parameters	NULL
Return Value	Error Code Success-0 Failure- errcode

2.4.16 Appendix

2.4.16.1 Source Code Download

Find the "Downloads" section in the FAIRINO documentation (https://fairino-doc-en.readthedocs.io/latest/), click the "Python SDK" button, and then click "FAIRINO Python SDK" on the right-hand page. Wait for the browser to complete the download.



Figure 16.1-1 Python SDK Source Code Download

Download and extract the Python SDK. The project directory is shown below. The "windows" folder contains the Python SDK for Windows systems, and the "linux" folder contains the Python SDK for Linux systems.

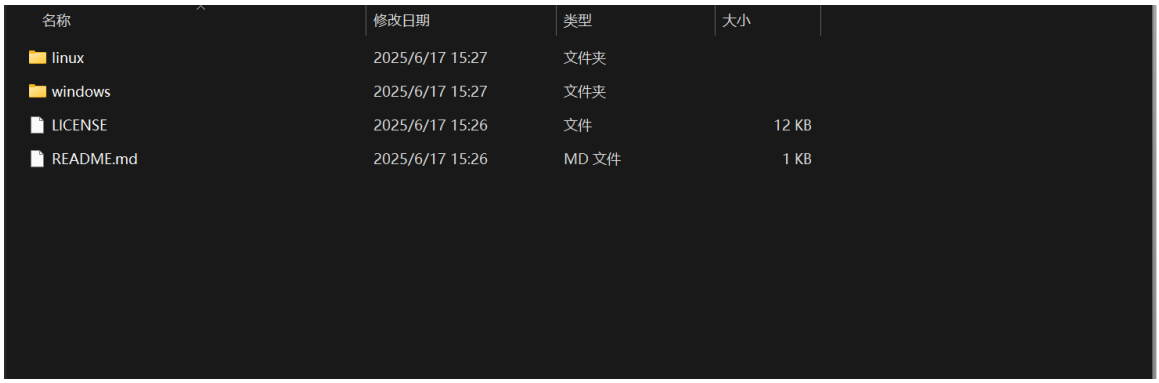


Figure 16.1-2 Python SDK File Structure Example

Taking the Windows system as an example, open the "windows" folder. The directory is shown below. The "example" folder contains test samples, the "fairino" folder contains the Python SDK source code, and "libfairino" contains library files.



Figure 16.1-3 Python SDK File Structure Example for Windows Systems

Open the "windows" folder in PyCharm. The structure is shown below.

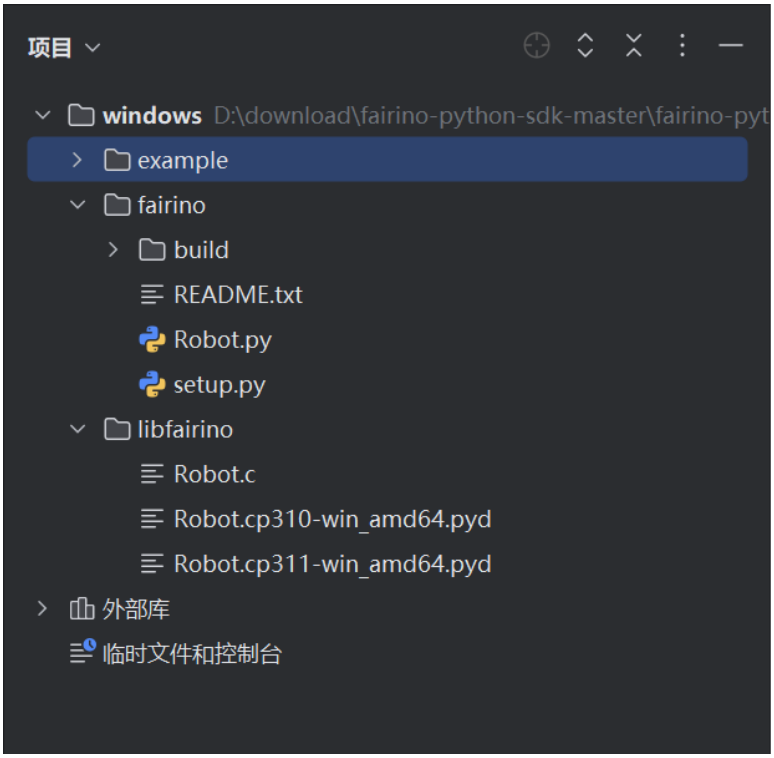


Figure 16.1-4 Project File Structure Example in PyCharm

2.4.16.2 Source Code Compilation

The generation of Python dynamic libraries varies depending on the system type and Python version. For example, on Windows platforms, the library file suffix is ".pyd", while on Linux platforms, it is ".so". Additionally, dynamic libraries generated for different Python versions cannot be mixed. Therefore, before generating dynamic libraries, ensure the Python version and platform are correctly selected. This manual uses Python 3.10, Windows 11, and Ubuntu 22.04 for compilation instructions.

2.4.16.2.1 Windows Platform Python SDK Compilation

First, open the downloaded Python SDK files in PyCharm and open the "setup.py" file.

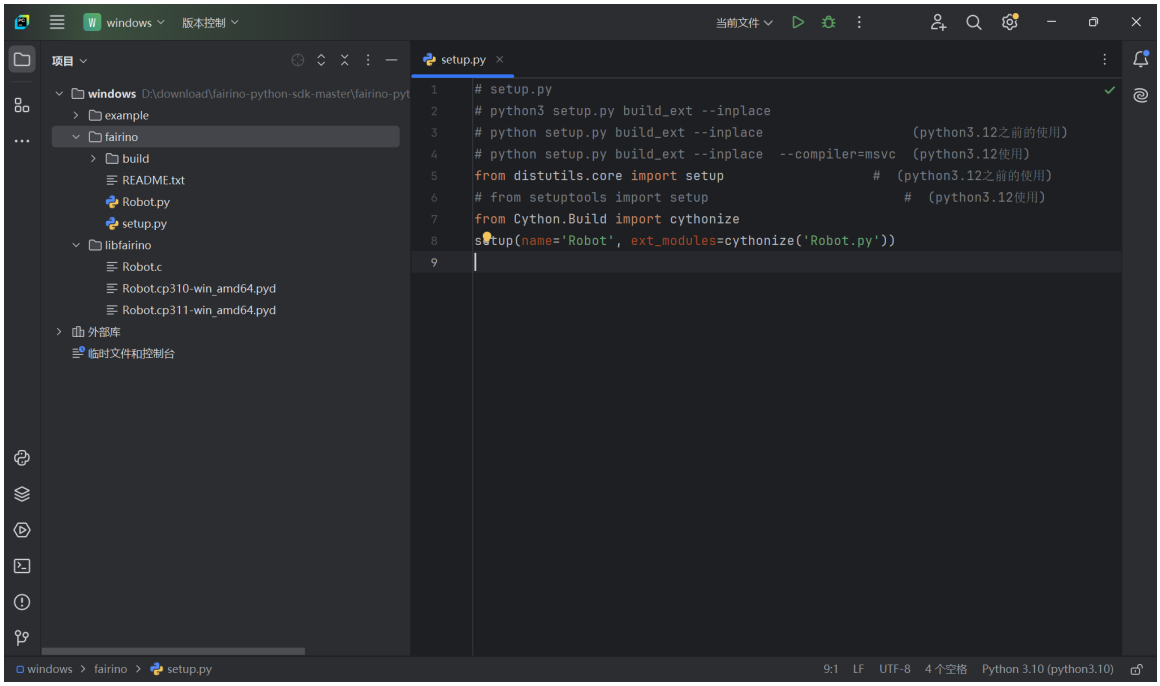


Figure 16.2-1 Opening the Project File

Then, select the Python interpreter at the bottom-right corner. This example uses Python 3.10.

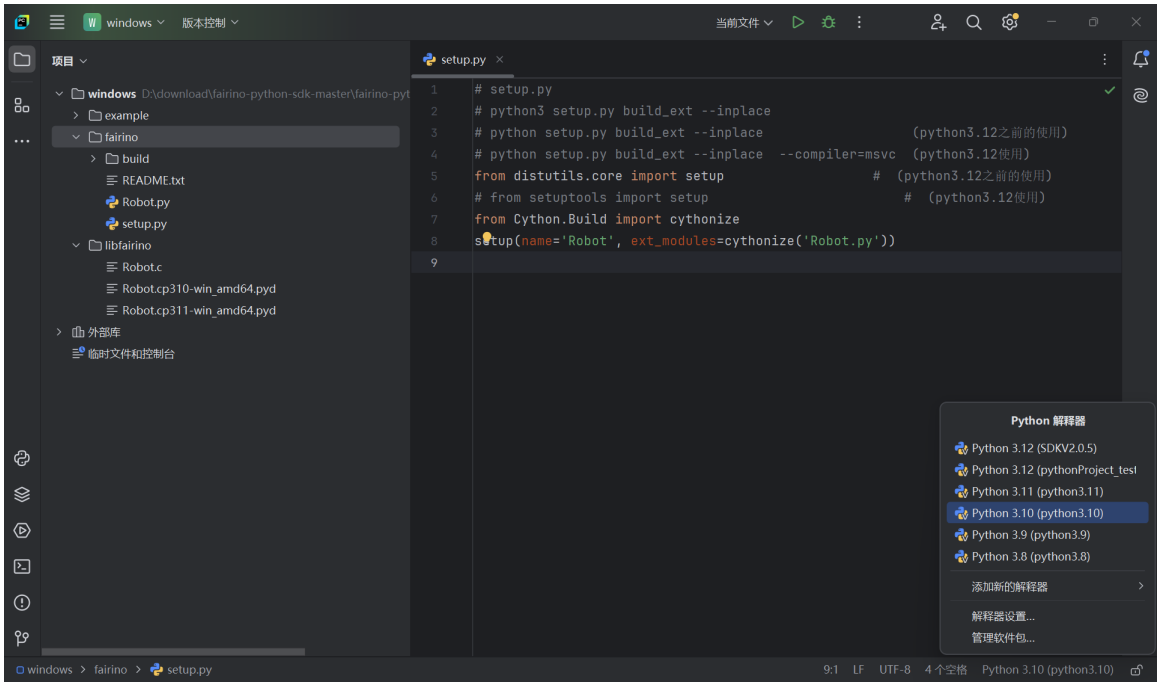


Figure 16.2-2 Selecting the Python Version

Right-click the “fairino” folder, click “Open In,” and then click “Terminal.”

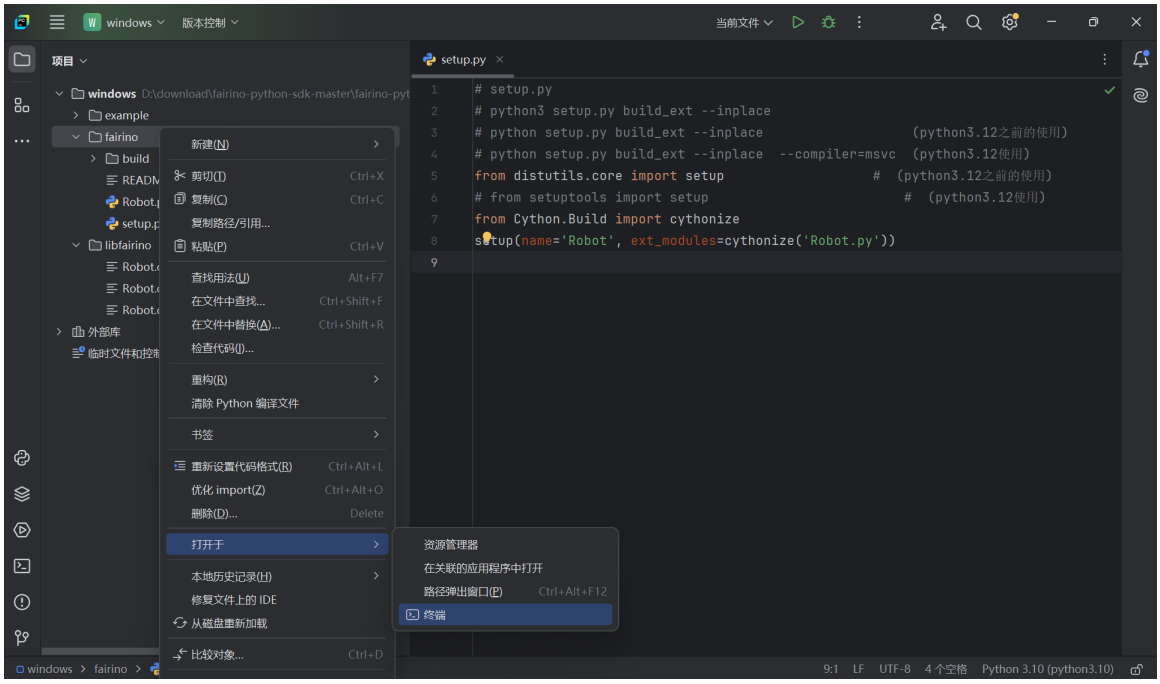


Figure 16.2-3 Opening the Terminal

In the terminal, enter “python setup.py build_ext –inplace” and press “Enter” to generate the Python SDK dynamic library.

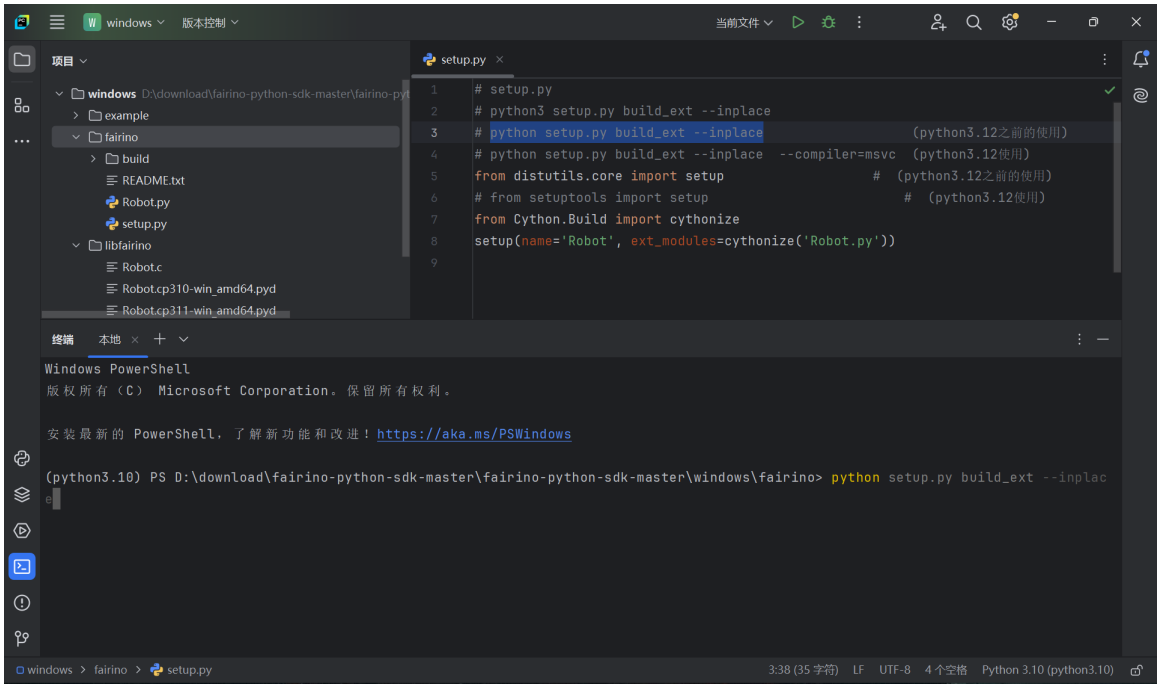


Figure 16.2-4 Running the Dynamic Library Generation Command

After the dynamic library is generated, the “fairino” folder will contain “Robot.c” and “Robot.cp310-win_amd64.pyd”. “Robot.c” is the C language file converted from “Robot.py”, and “Robot.cp310-win_amd64.pyd” is the Python SDK dynamic library. Here, “cp310” indicates compatibility with Python 3.10, and “win_amd64” indicates compatibility with Windows platforms.

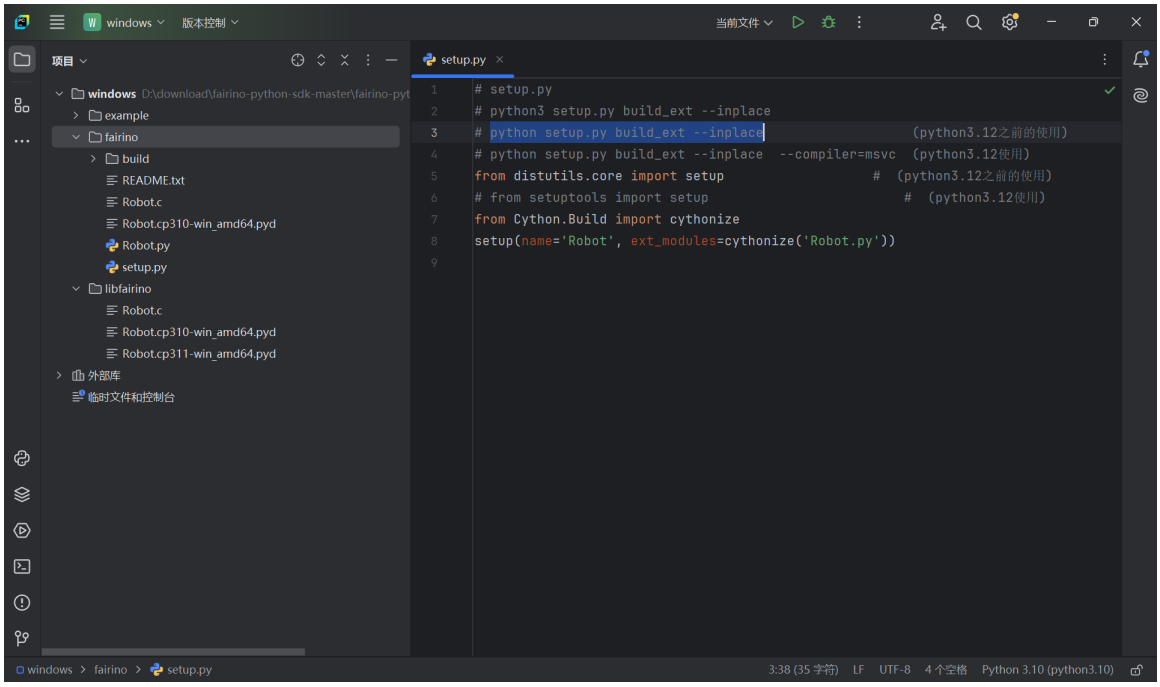


Figure 16.2-5 Generated .pyd Dynamic Library

2.4.16.2.2 Linux Platform Python SDK Compilation

First, check the Python version. This manual uses the "pyenv" tool to manage Python versions on Linux systems. Run the "pyenv versions" command to view the current Python version.

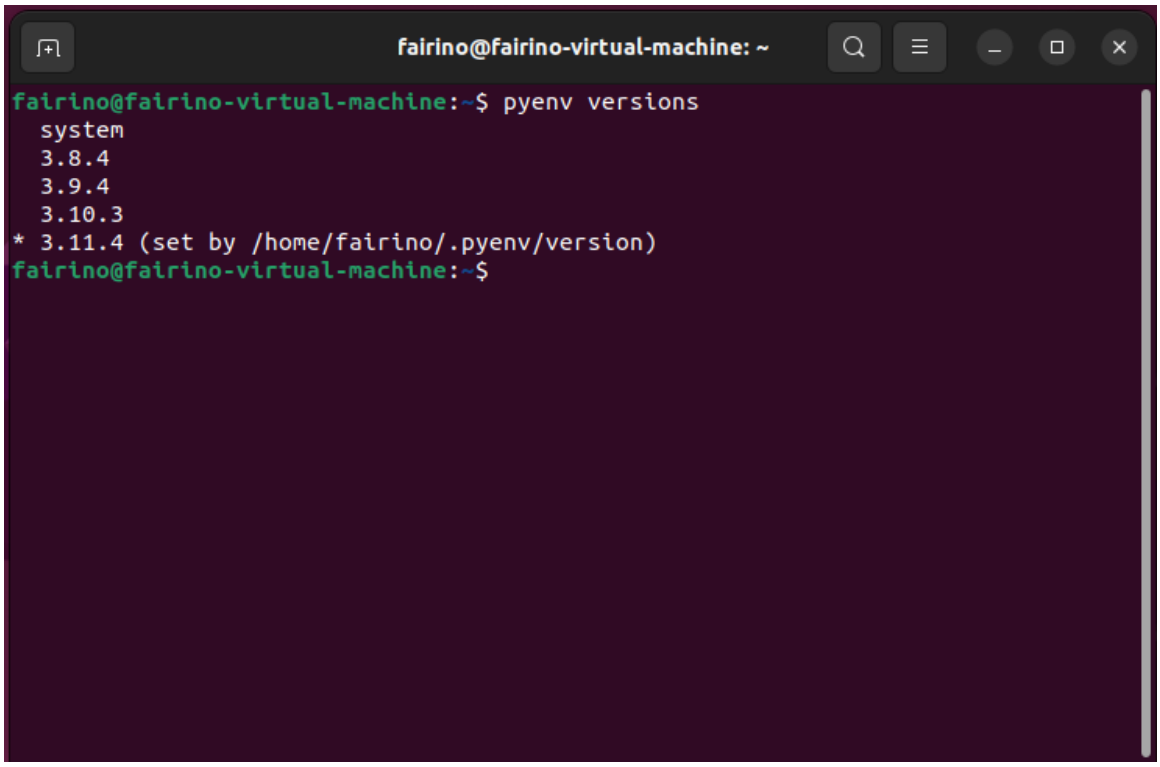


Figure 16.2-6 Checking the Python Version

Switch to the target Python version. For example, to use Python 3.10, run the "pyenv global 3.10.3" command.

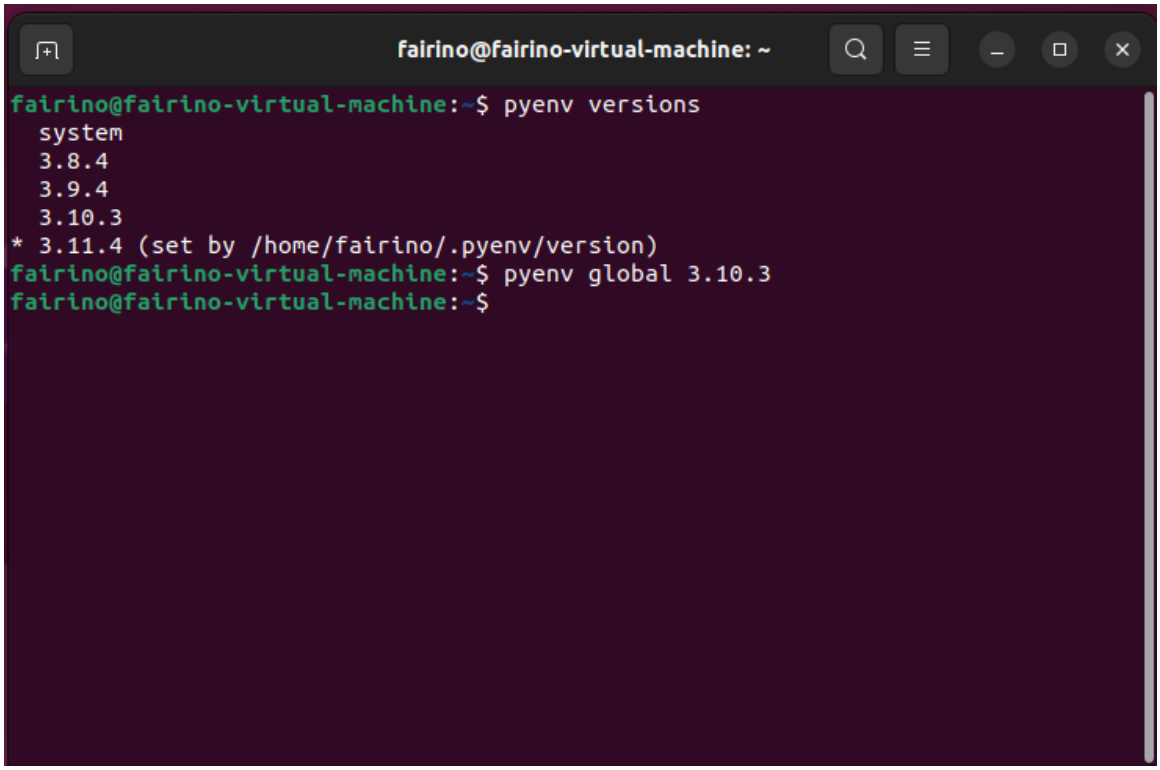


Figure 16.2-7 Selecting the Python Version

Navigate to the directory containing "Robot.py" by running the command "cd /home/fairino/fairino-python-sdk-master/fairino-python-sdk-master/linux/fairino".

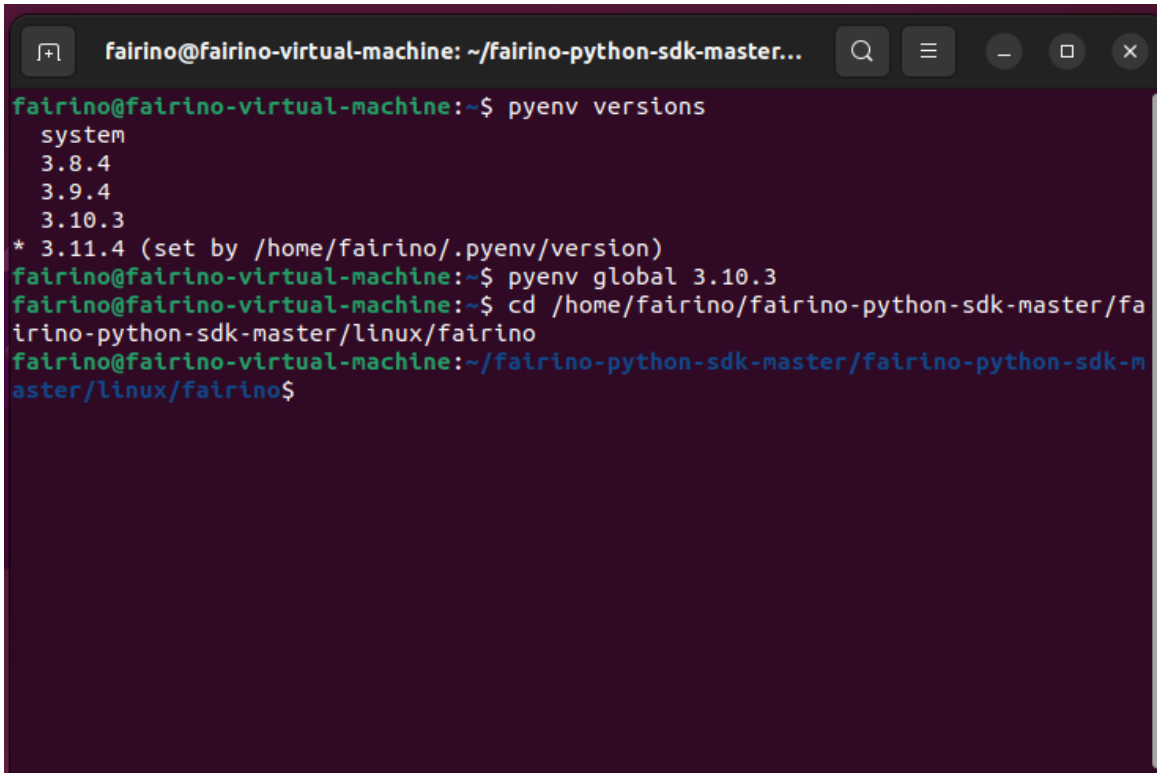


Figure 16.2-8 Navigating to the “Robot.py” Directory

Confirm the Python version by running the “python –version” command.

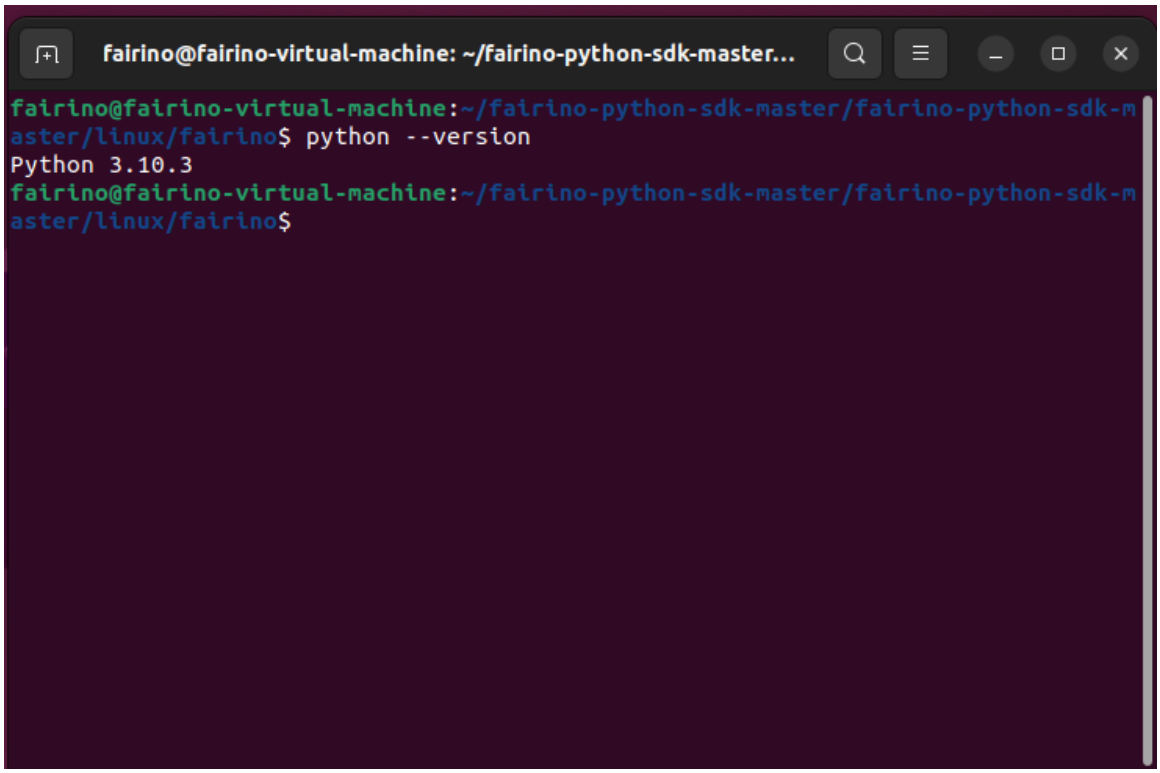


Figure 16.2-9 Checking the Python Version

In the terminal, enter “python setup.py build_ext –inplace” and press “Enter” to generate the Python SDK dynamic library.

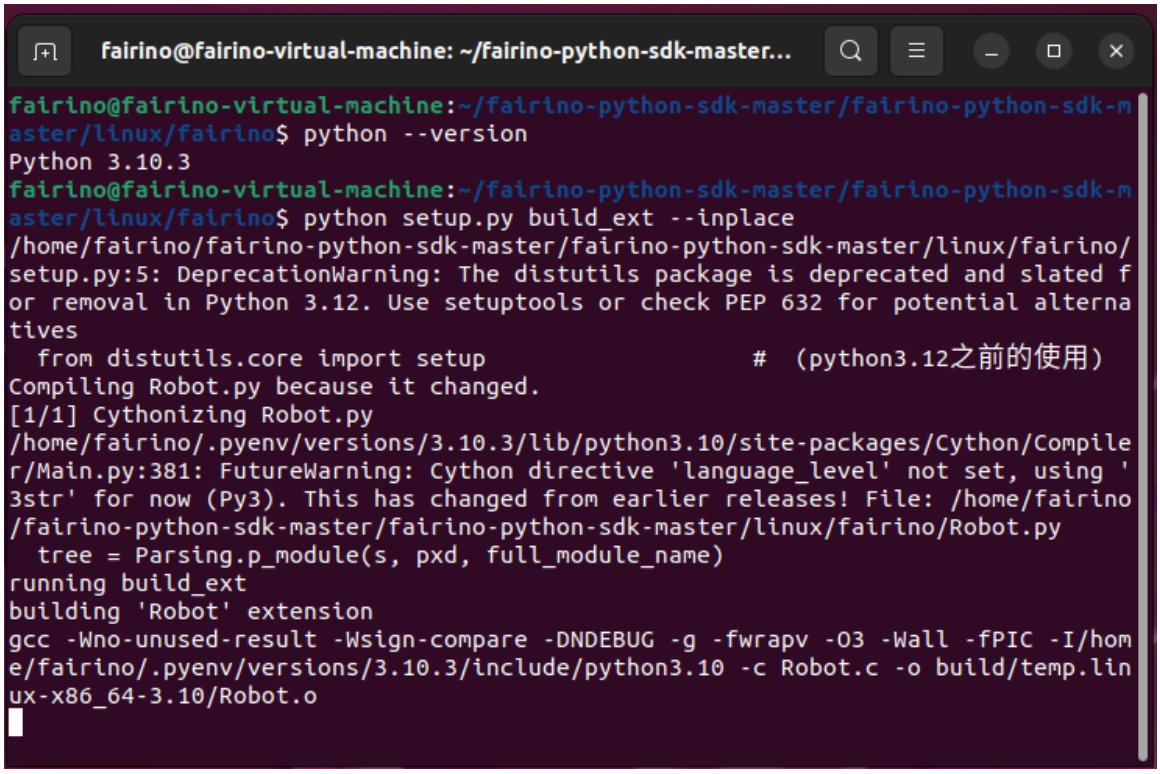


Figure 16.2-10 Running the Dynamic Library Generation Command

After the dynamic library is generated, the “fairino” folder will contain “Robot.c” and “Robot.cpython-310-x86_64-linux-gnu.so”. “Robot.c” is the C language file converted from “Robot.py”, and “Robot.cpython-310-x86_64-linux-gnu.so” is the Python SDK dynamic library. Here, “python-310” indicates compatibility with Python 3.10, and “linux-gnu” indicates compatibility with Linux platforms.

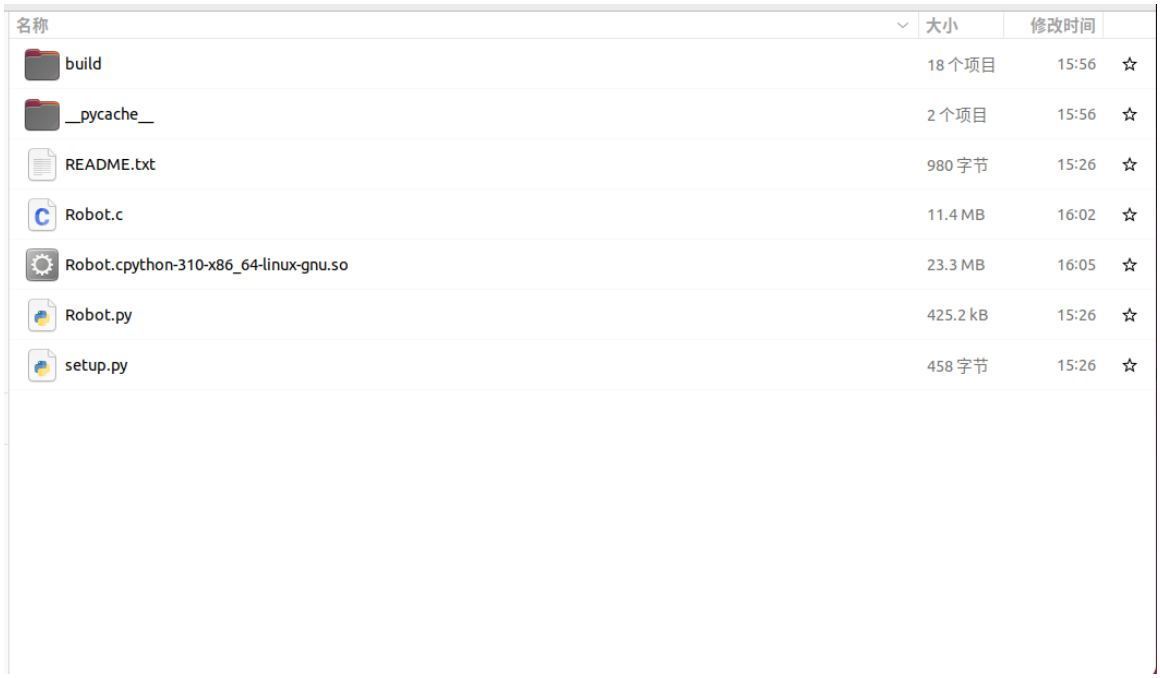


Figure 16.2-11 Generated .so Dynamic Library

2.4.16.3 Notes

2.4.16.3.1 Potential Issues

2.4.16.3.1.1 Version Compatibility

Python dynamic libraries depend on the generation environment and Python version. Therefore, when using Python dynamic libraries, ensure the library matches the system type and Python version.

2.4.16.3.1.2 Error Codes

A return value of 0 indicates normal operation. If the return value is not 0, refer to the error code table.

2.5 Error Code Comparison Table

Errcode	Describe	Processing method
-7	Upload file does not exist	Check that the file name is correct
-6	The save file path does not exist	Check whether the file path is correct
-5	The LUA file does not exist	Check that the lua file name is correct
-4	xmlrpc interface execution failed	Contact the after-sales engineer to view the controller log
-3	xmlrpc communication failed	Please check whether the network connection and server IP address are correct
-2	Abnormal communication with the controller	Check the software and hardware connection with the controller
-1	Other errors	Contact the after-sales engineer to view the controller log
0	Successful call	/
3	The number of interface parameters is inconsistent	Check the number of interface parameters
4	Interface parameter value exception	Check parameter value type or range
5	The tpd timer fails to start	Check the TPD timer parameter Settings
6	The tpd timer fails to close	Check the TPD timer parameter Settings
7	Failed to create the tpd file	Check the TPD file name
8	Failed to open track file	Check if the TPD track file exists or the track name is correct
9	TPD file name sending failed	Check if the TPD track name is correct
10	Failed to send TPD file content	Check whether the contents of the TPD file are correct
11	Program exception, parsing stopped	Check the contents of the program, check the contents of the instructions executed when the program terminates
12	The contents of the tpd file are in consistent	Check TPD file contents
13	The number of tpd files is abnormal	Check TPD file contents
14	Interface execution failed	Check whether the web interface reports a fault or status feedback reports a fault
15	TDP record points exceeded	Reblog
16	Protocol loaded	Do not duplicate load protocols
17	Protocol unload	Please load the protocol first
18	The robot program is running, please stop it first	Stop the program before performing other operations

continues on next page

Table 2 – continued from previous page

Errcode	Describe	Processing method
19	The communication of the posesensor is abnormal	Check the connection between robot and sensor network and the loading of sensor protocol
20	No tool is set for swing welding	Swing welding requires a tool coordinate system with tool number other than 0
21	External axis not removed active	Please remove the activation expansion shaft first
22	The three-point method does not set a tool	Please set the tool coordinate system first
23	Failed to obtain the posesensor data	Check the connection between robot and sensor network and the loading of sensor protocol
24	Eight point method-The first four points change too much	Please reduce the attitude change values of the first four points
25	Data exception, calculation failed	Re-calibration or identification
26	Eight-point method-did not switch to the base system, did not switch to exaxis 0	Please switch to the base system first (tool coordinate system with tool number 0)
27	Eight-point method-Calculation results are abnormal	Please re-calibrate
28	Inverse kinematics calculation results are abnormal	Check if the pose is reasonable
29	ServoJ joint overrun	Check whether the joint data is within a reasonable range
30	Non-resettable fault, please power off and restart the control box	Please power off and restart the control box
31	The emergency stop button is released, please power off and restart the control box	Please power off and restart the control box
32	joint over limit	Switch to drag mode and move the joint to the soft limit range
33	External axis is not in zero position, lead and resolution Settings fail	Please check the external axis settings
34	Wrong workpiece number	Please check that the workpiece number is reasonable
35	Please switch to work piece number0	Switch to the job coordinate system with job number 0
36	Filename too long	Please shorten the filename length
37	Wrong tool number	Please check whether the tool number is reasonable
38	Singular pose, calculation failed	Please change pose
40	Speed percentage exceeds limit	Check if the speed percentage is reasonable
41	The external axis does not return to zero	Please check the external axis settings
42	Posture changes too much	Insert intermediate pose to transition
43	The conveyor belt detection switch DI is not configured	Configure the DI detection port on the conveyor belt
44	The robot attitude Angle exceeds the limit	Please check the target attitude Settings
45	The external axis is not active	Please check the external axis settings
46	The synchronization function requires calibration of the external axis	Please check the external axis settings
47	Failed to configure the external drive information	Please check the external axis settings
48	Obtaining external shaft drive information timed out	Please check the external axis settings

continues on next page

Table 2 – continued from previous page

Errcode	Describe	Processing method
49	External axis error can not be enabled	Please check the external shaft status
50	Failed to obtain external shaft drive information	Please check the external shaft status
51	Obtaining external shaft driver information timed out	Please check the external shaft status
52	The synchronization function cannot be performed in a single step	Please check the external axis settings
59	Force/torque sensor not activated	Activate force sensor
60	Force/torque sensor reference frame not switched to tool	Switch force sensor coordinate system to tool
61	The force/torque sensor is not homing	Please set the force/torque sensor to zero first
62	Force torque sensor load is not set to zero	Please reset the load of the force/torque sensor first
63	Failed to obtain the system time	Contact after-sales engineers to view controller logs
64	Not added to the instruction queue	Contact the after-sales engineer to view the controller log
66	The middle point 1 of the full circle/helix command is wrong	Check whether the middle point 1 data is correct
67	The middle point 2 of the full circle/helix command is wrong	Check whether the middle point 2 data is correct
68	The middle point 3 of the full circle/helix command is wrong	Check whether the middle point 3 data is correct
69	The middle point of the arc command is wrong	Check if the intermediate point data is correct
70	Arc instruction target point error	Check if the target point data is correct
73	Gripper movement error	Check whether the communication status of the gripper is normal
74	Line instruction point error	Check whether the point data is correct
75	Channel error	Check if IO number is in range
76	Wait timeout	Check whether the IO signal is input or the wiring is correct
82	TPD instruction point error	Re-record the teaching track
83	TPD instruction tool does not match current tool	Change the tool coordinate system used when teaching to TPD
84	Weld locating failed	Check the laser sensor parameter configuration to ensure that the sensor can recognize the weld
85	Line command error	Please check whether the command point position parameters, tool number and job number are configured
90	Failed to check the external axis configuration file	Contact after-sales engineers to view controller logs
91	The version of the peripheral configuration file does not match	Contact after-sales engineers to view controller logs
92	Failed to read the peripheral configuration file	Contact after-sales engineers to view controller logs
93	The number of spline instructions exceeds the limit	Reduce the number of spline instruction points
94	Spline cue point error	Check whether the point data is correct
95	Spline parameter error	Check whether the spline parameters are reasonable
96	Welding wire locating failed	Check the locating parameters of the welding wire

continues on next page

Table 2 – continued from previous page

Errcode	Describe	Processing method
97	The recorded data is empty	Please wire for data recording
98	The main program is not configured	Configure the main program first
99	A safe stop has been triggered	Please check the safety stop signal status
100	The job origin is not configured	Configure the job origin first
101	The robot is not enabled	Please enable the robot first
106	The external axis is not enabled	Enable the external axis first
108	Wrong starting point for helix command	Check whether the starting point data is correct
110	Please turn off Drag Configuration-Impedance call back into the interference zone	Disable the drag configuration into the intervention zone
111	Turn off the drag teaching lock freedom function	Turn off the drag teaching lock freedom function
112	The given pose cannot be reached	Check if the target pose is reasonable
130	The point table does not exist	Check that the point table name is correct
131	Modbus master not connected	Check if the Modbus master is connected
132	Modbus wait timeout	Check if the Modbus connection is correct
133	Please first enter Drag mode/Force Sensor Assisted drag mode	Please first enter Drag mode/Force Sensor Assisted drag mode
134	Force/torque sensor no load	Please check the load of the force/torque sensor
135	Inversea nomaly of matrix under force/torque sensor	Contact after-sales engineers to view controller logs
138	The detection switch DI is not configured	You need to configure the detection switch DI port first
139	Please apply the extended axis coordinate system first	Please first apply an extended axis coordinate system with a number other than 0
140	Please activate the current extension axis first	Please activate the current extension axis first
143	The teaching point information does not exist	Please check that the teaching point information exists
144	The LUA file does not exist	Check whether the LUA file exists
151	Changes in joint configuration	Check whether the joint configuration has changed between the target location and the current location
152	The spacing between welding command points is too small	Check whether the distance between the welding command point and the current point is too small
153	The distance between arc command points is too small	Please check the arc command point spacing
154	Joint Command Point Error	Please check the joint command
155	Background Program Startup Failed	Please check the background program
156	Background Program Pause Failed	Please check the background program
157	Background Program Resume Failed	Please check the background program
158	Background Program Delete Failed	Please check the background program
159	Number of Running Background Programs Exceeds 8	Please check the number of running background programs
160	Speed Configuration File Check Failed	Please check the Speed configuration file
161	Load Configuration File Check Failed	Please check the Load configuration file

continues on next page

Table 2 – continued from previous page

Errcode	Describe	Processing method
162	Asynchronous Motion Not Allowed During Laser Tracking	Please check the laser tracking
163	Multiple WHILE Loops in WHILE(1) Welding Process	Please check the WHILE command
164	Segment Welding Information Not Received	Please check the segment welding information
165	CNC Communication Exception	Please check the CNC communication
166	CNC Wait Timeout	Please check the CNC communication
167	CNC Communication Connection Exception	Please check the CNC communication connection
168	Number of Numeric Variable Monitors Exceeds Limit	Please check the numeric variables
169	Number of Character Variable Monitors Exceeds Limit	Please check the character variables
170	Workpiece Coordinate System Not Applied	Please check the workpiece coordinate system
171	Exceeded Joint Soft Limit, Soft Limit Protection Failed to Start, Move all joints within the preset soft limit range	Please check the joint soft limits
172	Motion Speed Cannot Be 0	Please check the motion speed
173	Extended Axis Blending Motion Requires Acceleration Smoothing to Be Enabled	Please check if acceleration smoothing is enabled
174	Current Robot Model Does Not Support This Function	Please check the robot model
175	Socket Not Configured	Please check the Socket communication
176	Communication Connected	Please check the communication connection status
177	Communication Not Connected	Please check the communication connection status
178	Firmware Version Too High, Not Supported by Current Software Version	Please check the firmware version
179	Firmware Version Too Low, Not Supported by Current Software Version	Please check the firmware version
180	G-code File Cannot Be Opened	Please check the G-code
181	G-code Parsing Failed	Please check the G-code
182	Given Physical Speed Exceeds Limit	Please check the given physical speed
183	Given Physical Acceleration Exceeds Limit	Please check the given physical acceleration
184	External Axis Enable Failed	Please check the external axis
185	Fault Signal Triggered, Motion Stopped	Please check the fault signal
186	Emergency Stop Signal Triggered, Motion Stopped	Please check the emergency stop signal
187	Current Collision Level Too Low	Please check the collision level
188	Force Control and Impedance Control Started Simultaneously, Please Check Force Control or Impedance Control	Please check the force control or impedance control

continues on next page

Table 2 – continued from previous page

Errcode	Describe	Processing method
189	Kernel Upgrade Version Verification Failed	Please check the kernel version
190	Kernel Version Upgrade Failed	Please check the kernel version
191	Linear Rack Guide Not Activated	Please check if the linear rack guide is activated
192	Current Linear Rack Collision Detection is Only for Upright Installation	Please check the robot installation pose
193	Torque Control Speed Exceeds Limit	Please check the torque control speed
194	Torque Control Power Exceeds Limit	Please check the torque control power
195	FR30L Base Tilt Angle Must Be 0	Please check the base tilt angle
196	Spot sensor signal not triggered	Please check the spot sensor
197	Sensor zero point and/or sensitivity calibration not completed	Please check the sensor
198	Abnormality detected in joint torque sensor	Please check the joint torque sensor
199	When multiple cube interference zones are enabled, only interference within the configured range is allowed	Please check the interference zone(s)
200	Robot is currently jogging	Please check the robot
201	Weave configuration file check failed	Please check the Weave configuration file
202	Interfere configuration file check failed	Please check the Interfere configuration file

CHAPTER

THREE

FAIRINO SIMMACHINE

3.1 Virtual Machine

3.1.1 Overview

This manual aims to introduce how to use the FAIRINO SimMachine virtual machine.

3.1.2 Operating Instructions

3.1.2.1 Installing VMware Workstation

VMware Workstation demo version: 17.6.3 (skip this step if already installed).

Directly search for the VMware official website in your browser or click the URL https://www.vmware.com, then download the installation package and install it using the default path.

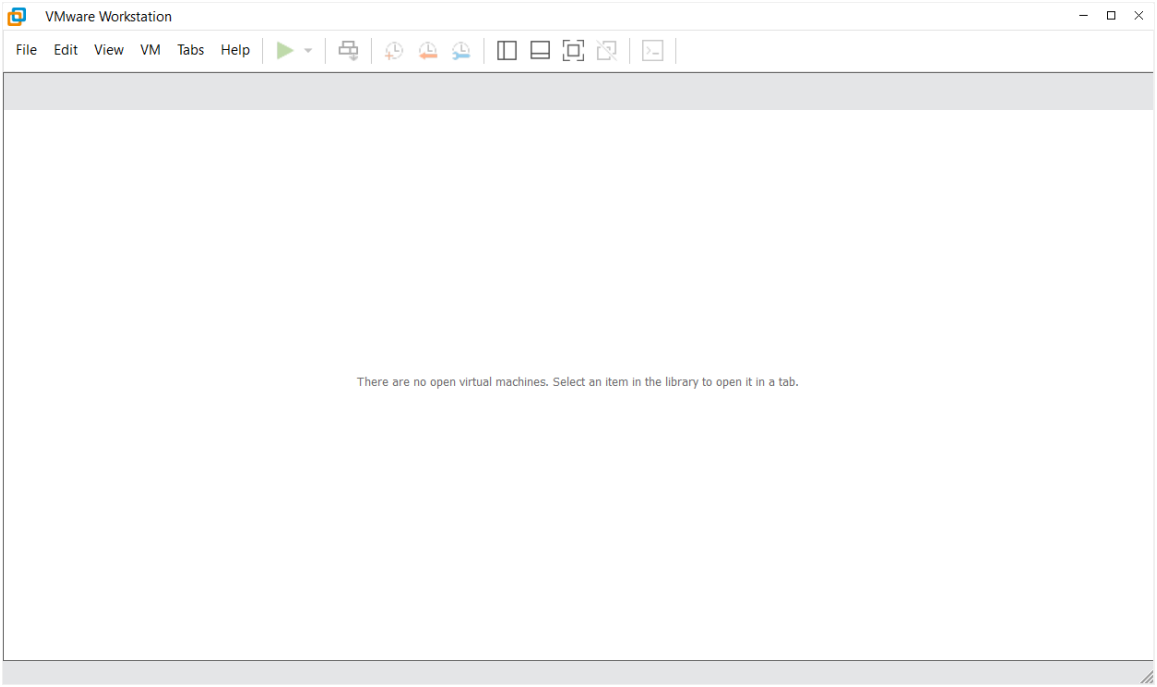


Figure 6.2-1 VMWare Interface

3.1.2.2 Opening the Virtual Machine Image

1. Download the virtual machine image FAIRINO_SimMachine.zip and extract it.
2. Open VMware, click File->Open. As shown in Figure 2-2 below:

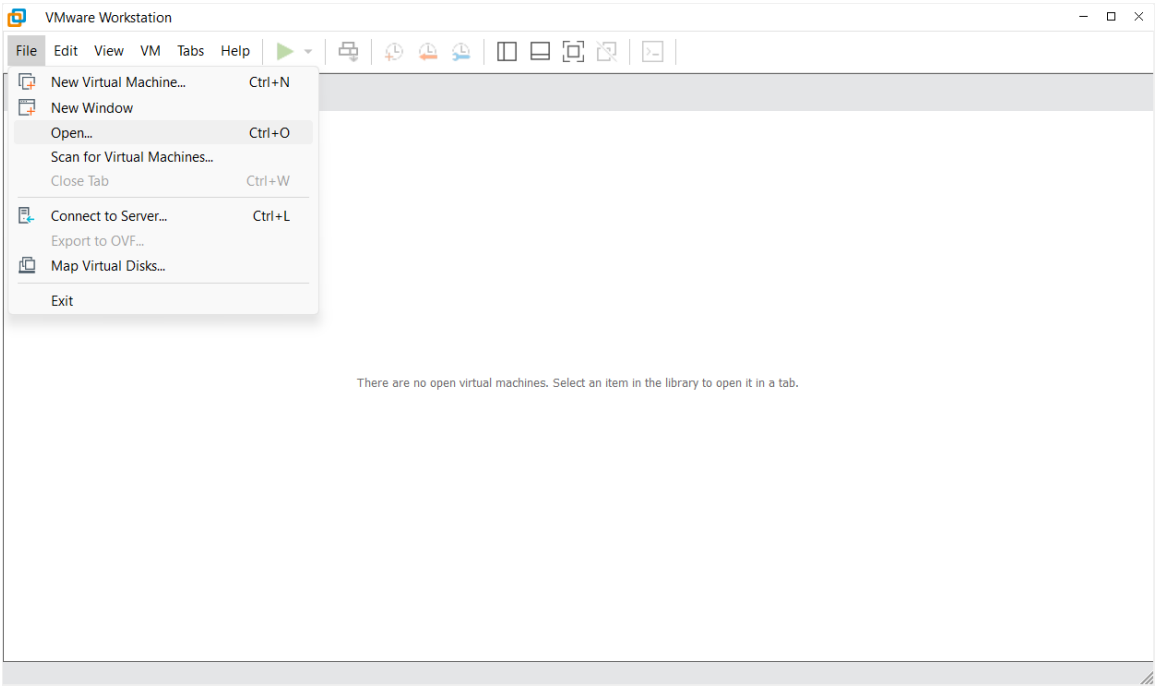


Figure 6.2-2 Open Image

3. Locate the extracted folder and select the file with the .vmx extension. As shown in Figure 2-3 below:

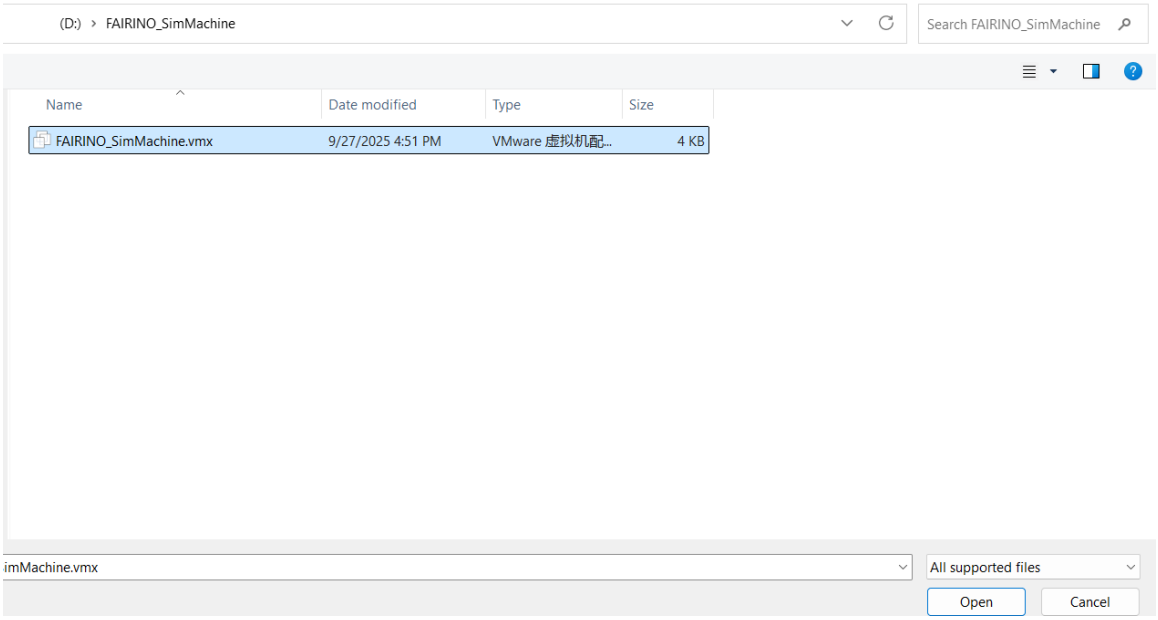


Figure 6.2-3 Select File

4. Click “Power on this virtual machine” to start the virtual machine. As shown in Figure 2-4 below:

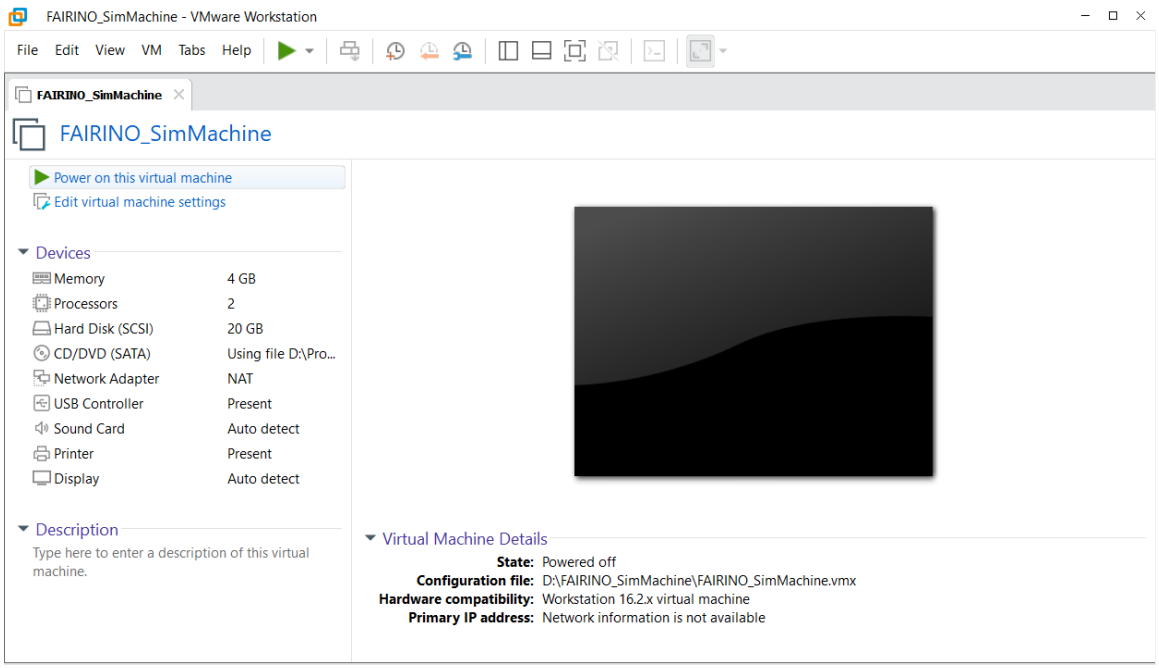


Figure 6.2-4 Start Virtual Machine

5. Find “fr_get_vm_net” in the extracted folder and double-click to open it. As shown in Figure 2-5 below. The output content is the virtual machine IP. As shown in Figure 2-6 below.

Note: If the acquisition fails, please go to the virtual machine and obtain it by executing the “ifconfig” command.

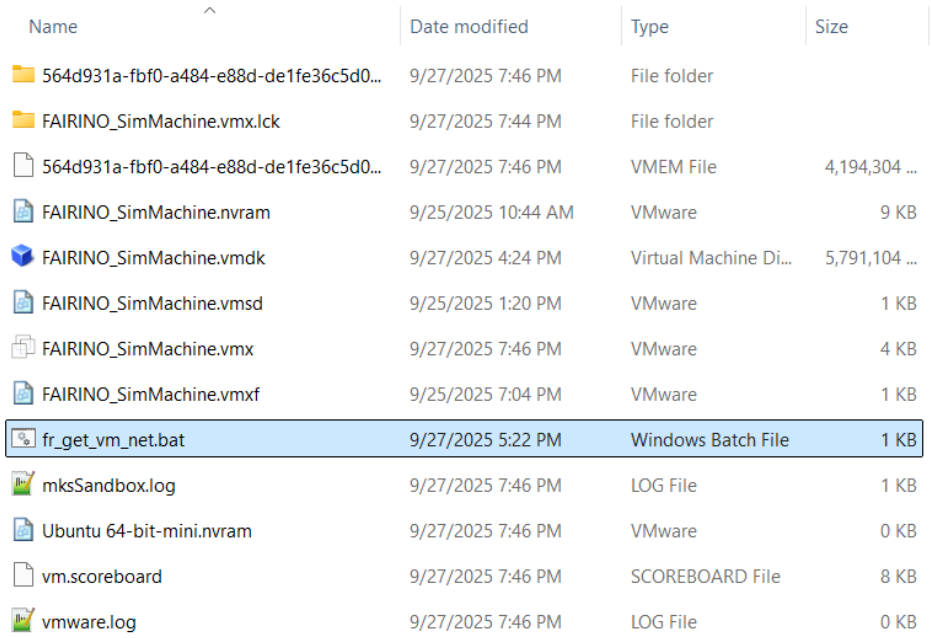


Figure 6.2-5 fr_get_vm_net.bat

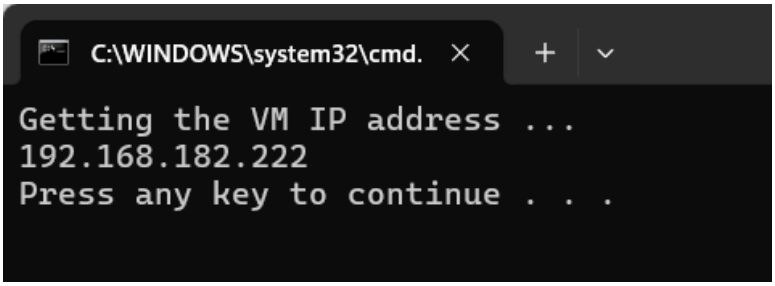


Figure 6.2-6 Virtual Machine IP

3.1.2.3 Accessing WebApp from Windows

1. After obtaining the virtual machine IP, directly access the virtual machine IP in the Windows browser to enter WebApp. For example, enter: 192.168.182.222, as shown in Figure 2-7:

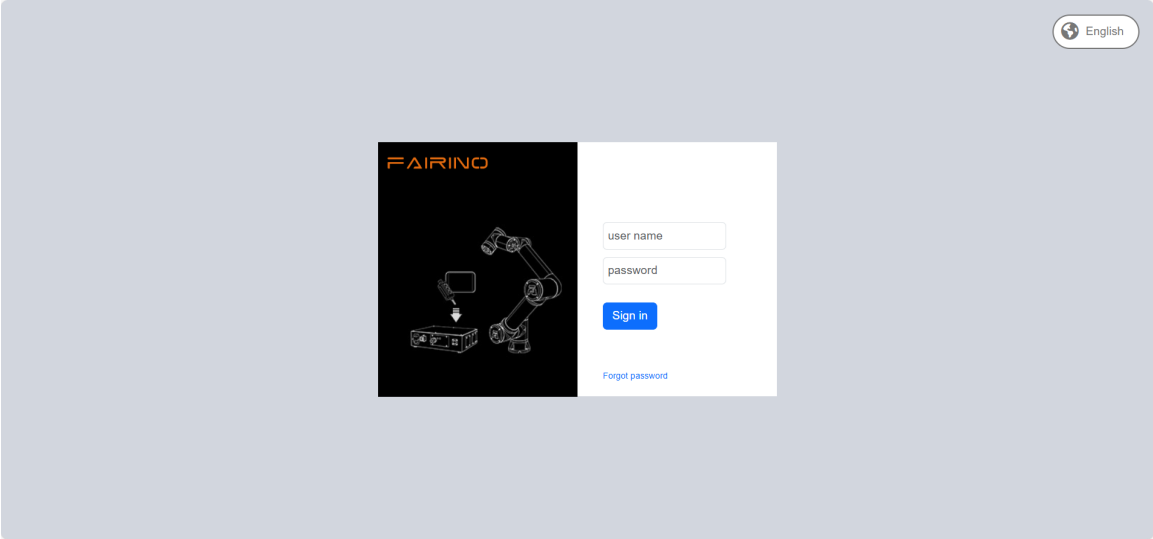


Figure 6.2-7 Access WebApp via Virtual Machine IP

3.2 Virtual Machine - Docker

3.2.1 Linux Docker Image Deployment

3.2.1.1 Operating Environment

Virtual machine operating system: Ubuntu 18.04.6;

Virtual machine operating environment: RAM 4G, ROM 50G, 6-core CPU;

Operation permissions: Use super administrator root privileges. Setting method see Appendix 3;

Docker installation file: fr_docker.tar.gz;

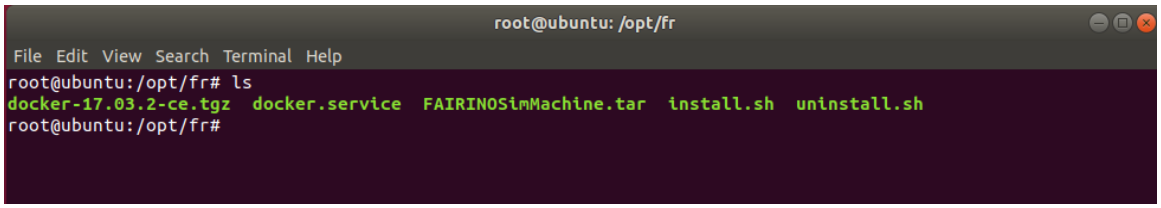
FAIRINO SimMachine image: FAIRINOSimMachine.tar;

3.2.1.2 Install Docker

If the user has already installed and deployed Docker, skip this section and proceed to section 1.3 Image Deployment.

1. Download fr_docker.tar.gz and place it in the Ubuntu file path /opt/.
2. Extract fr_docker.tar.gz. Using /opt/ directory as an example:

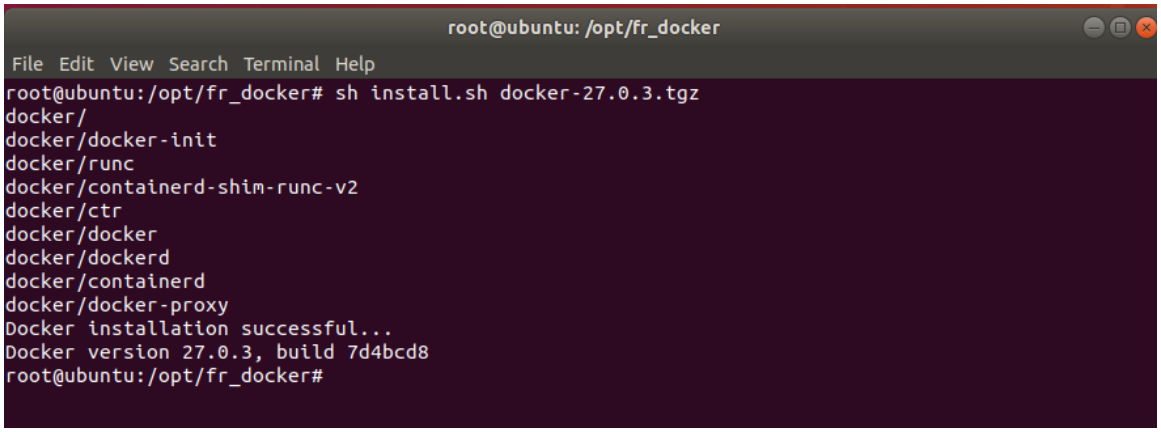
```
cd /opt/ && tar -zxvf fr_docker.tar.gz
```



3. Execute the Docker installation script:

```
sh install.sh docker-27.0.3.tgz
```

After the script finishes executing, if the version number appears, it indicates successful installation.

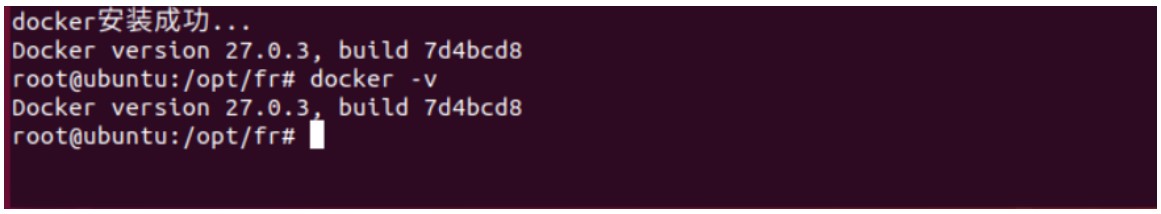


3.2.1.3 Image Configuration

3.2.1.3.1 Import Docker Image

1. Download the virtual machine image FAIRINOSimMachine.tar and extract it.
2. Check the Docker version to confirm installation.

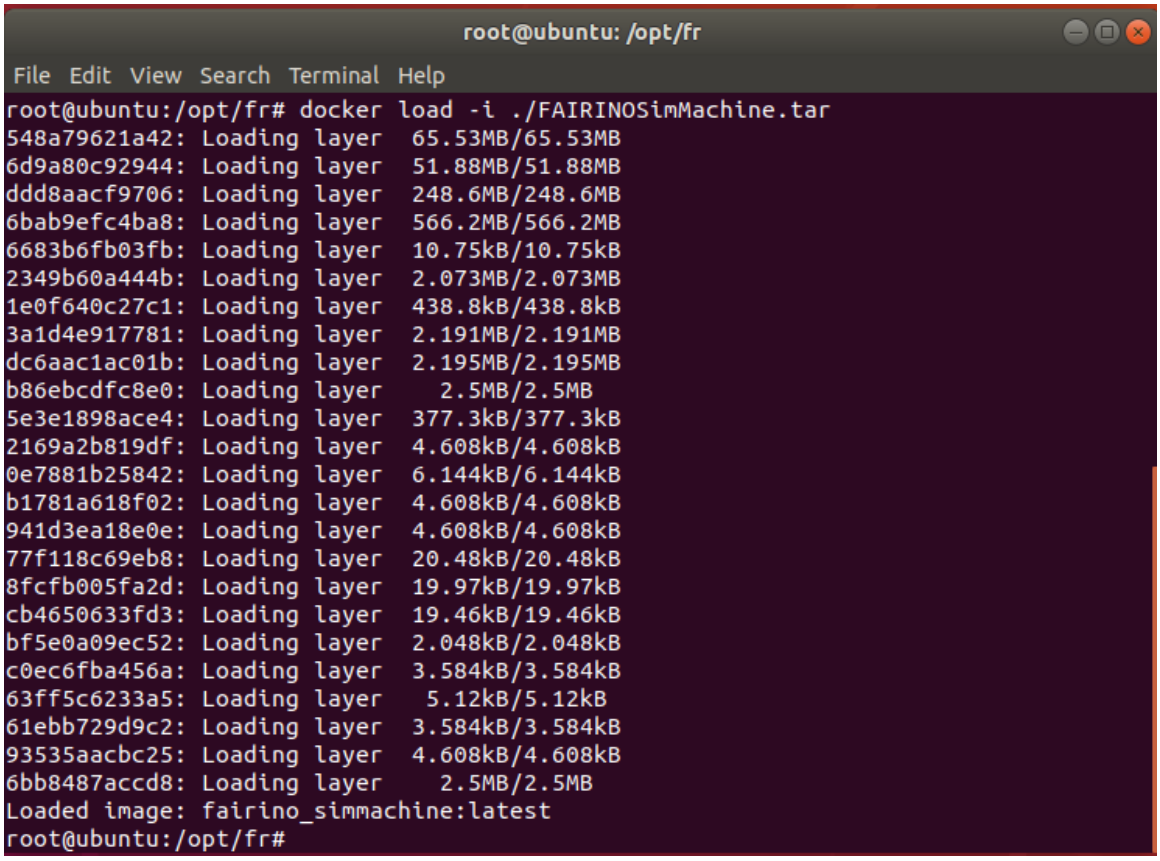
```
docker -v
```



3. Import the image

```
docker load -i ./FAIRINOSimMachine.tar
```

The appearance of *fairno_simmachine:latest* indicates the import is complete.



4. Execute *docker images* to check if the import was successful.

3.2.1.3.2 Create Custom Bridge Network

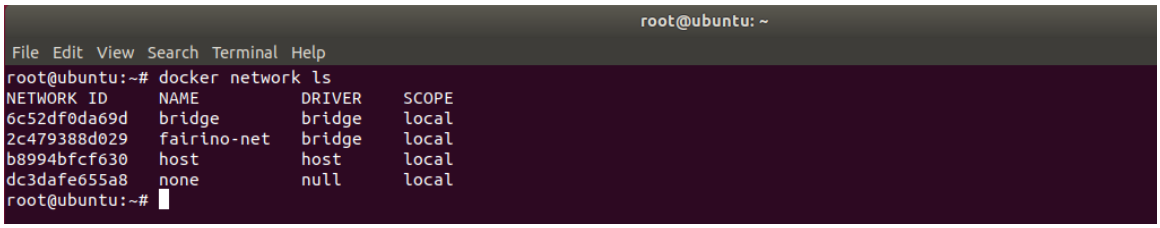
1. Execute the following command to create a bridge network named *fairino-net* with subnet *192.168.58.0/24*.

```
docker network create --driver bridge --subnet 192.168.58.0/24 --gateway 192.168.58.1␣
↪fairino-net
```

2. View the network

```
docker network ls
```

The presence of the *fairino-net* network indicates successful creation.



3.2.1.3.3 First-time Docker Container Startup

1. Create and start the container

Start the container using the *fairino-net* network and the *fairino_simmachine* image.

```
docker run -d -P --name fairino-container --privileged -u root --net fairino-net fairino_
↪simmachine
```

```
6bb8487acc08: Loading layer    2.5MB/2.5MB
Loaded image: fairino_simmachine:latest
root@ubuntu:/opt/fr# docker network create --driver bridge --subnet 192.168.58.0/24 --gateway 1
92.168.58.1 fairino-net
2c479388d029316837283 76e4f0918fe07df2ab39d0128f417dbcd88d11bca15
root@ubuntu:/opt/fr# docker run -d -P --name fairino-container --net fairino-net fairino_simmac
hine
585bc648bbcb2e416e6a24f38a7738739622bd8d344a861122582c806ea7bc05
root@ubuntu:/opt/fr#
```

```
docker ps
```

Check if the container started successfully. The appearance of *fairino-container* indicates successful startup.

```
Loaded image: fairino_simmachine:latest
root@ubuntu:/opt/fr# docker network create --driver bridge --subnet 192.168.58.0/24 --gateway 192.168.58.1 fairino-net
2c479388d02931683728376e4f0918fe07df2ab39d0128f417dbcd88d11bca15
root@ubuntu:/opt/fr# docker run -d -P --name fairino-container --net fairino-net fairino_simmachine
585bc648bbcb2e416e6a24f38a7738739622bd8d344a861122582c806ea7bc05
root@ubuntu:/opt/fr# docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED          STATUS          PORTS     NAMES
585bc648bbcb   fairino_simmachine   "/bin/bash -c '/etc/…"   23 seconds ago   Up 23 seconds             fairino-container
root@ubuntu:/opt/fr#
```

3.2.2 Web Operation of Virtual Robot

3.2.2.1 Container Normal Startup

This section applies to non-first-time container startups, such as when the container is not running in the background due to computer restart or Docker shutdown.

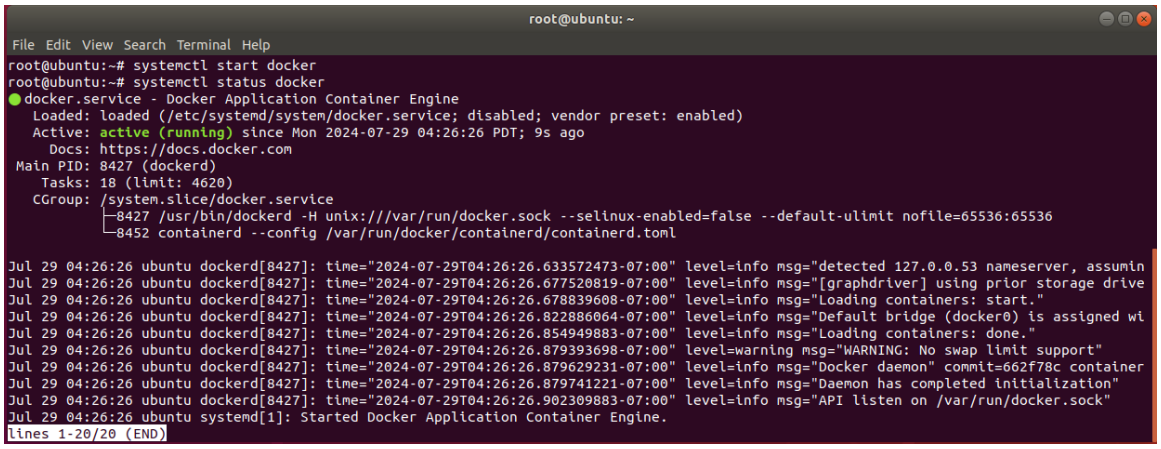
1. Start Docker:

```
systemctl start docker
```

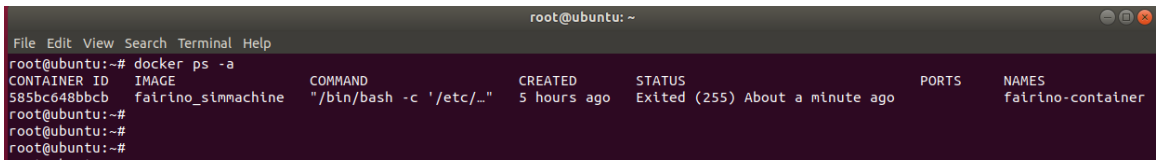
2. Check Docker status:

```
systemctl status docker
```

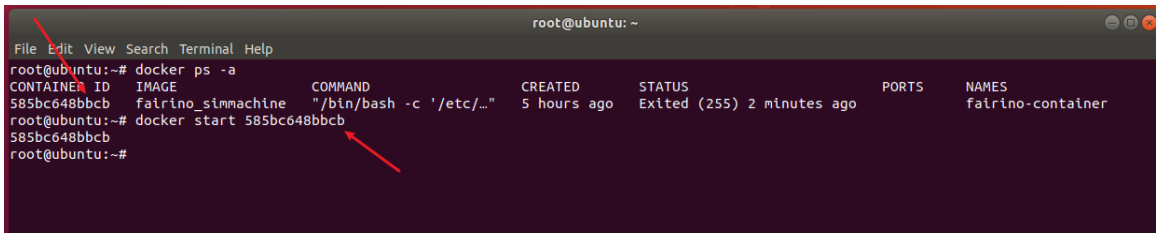
Green *active(running)* indicates successful startup.



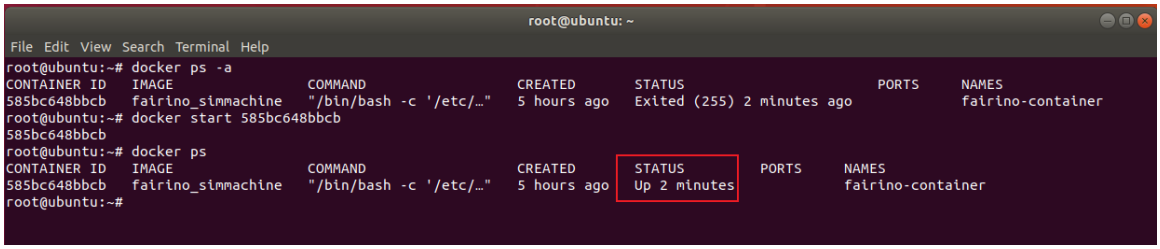
3. Execute *docker ps -a* to view the container ID.



4. Execute *docker start [Container ID]*.



5. After successful execution, execute *docker ps* again to confirm the container is running.

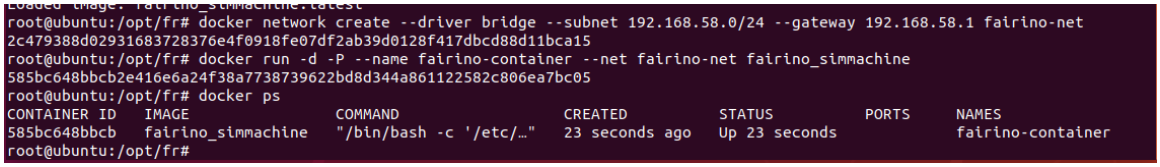


3.2.2.2 Operating the Virtual Robot

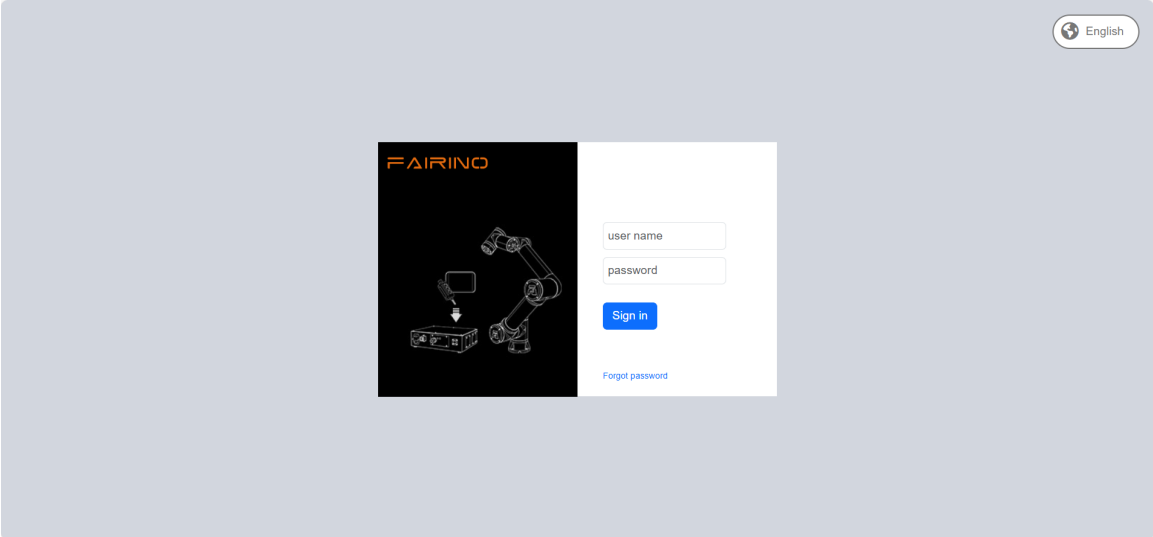
1. Confirm the Docker container is running.

```
docker ps
```

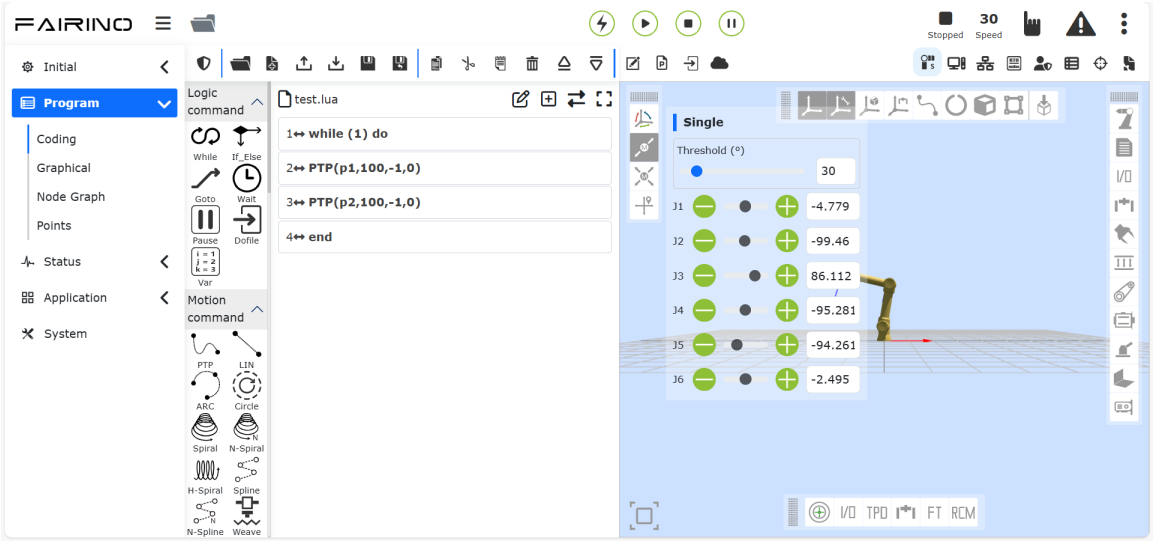
The appearance of *fairino-container* indicates it is running.



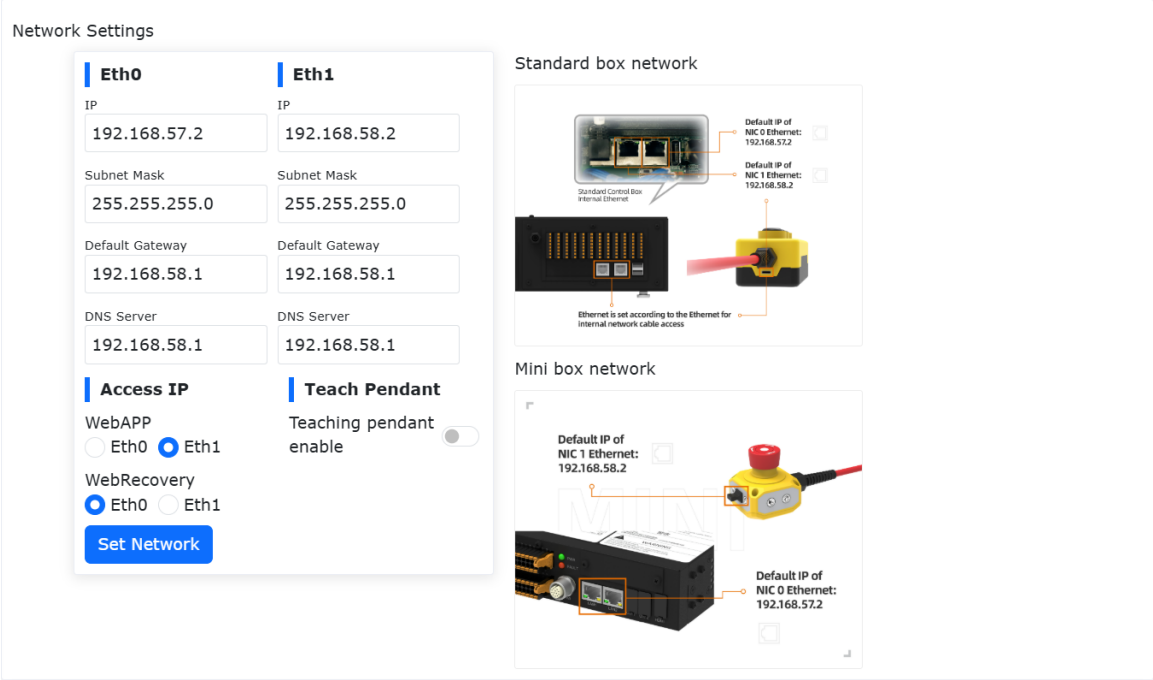
2. Open a browser, enter the default IP: 192.168.58.2, to access the web interface and operate the virtual robot.



3. Log in with the admin account, password: 123.



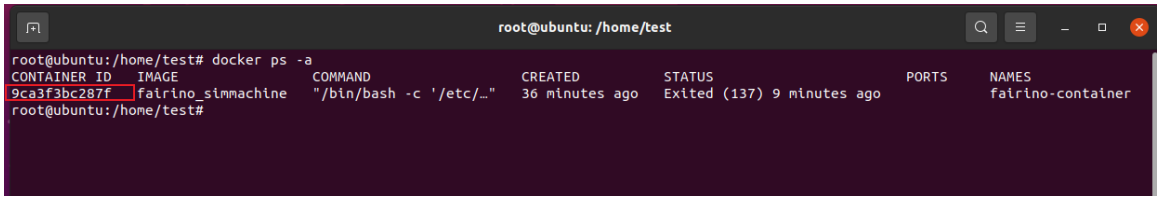
3.2.2.3 User IP Address Modification



1. Open a browser, enter the default IP: 192.168.58.2, to open the web page;
2. Log in with the admin account, password: 123;
3. Navigate to “System Settings” → “General Settings” → “Network Settings”, modify the IP to the target IP address, subnet mask, and gateway. Click “Set Network”;
4. Open the terminal, stop the container;

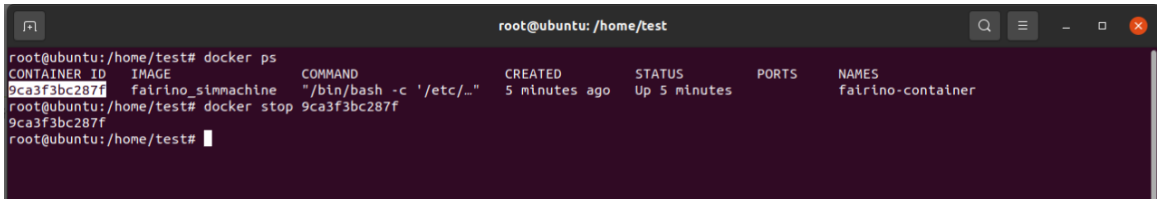
View the container ID:

```
docker ps -a
```



Stop the container:

```
docker stop [Container ID]
```



5. Reconfigure the container network;

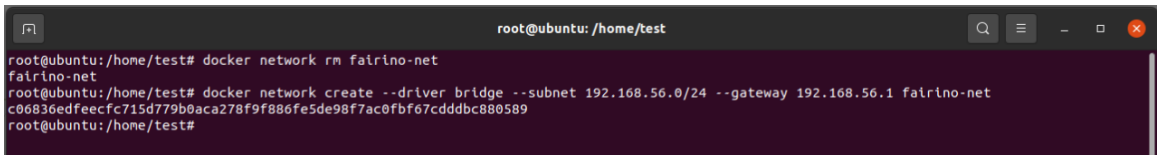
Delete the original network:

```
docker network rm fairino-net
```

Create a new network:

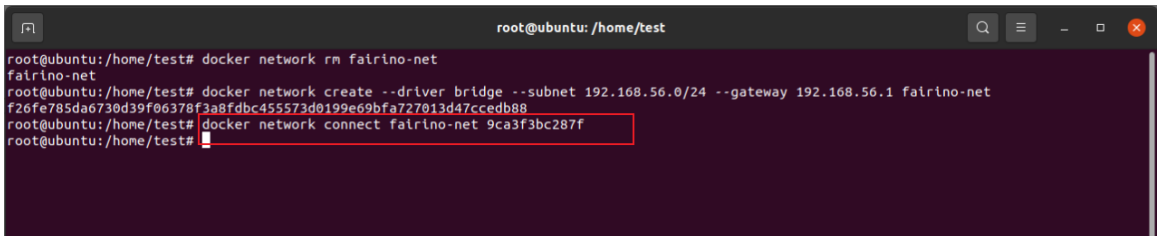
```
docker network create --driver bridge --subnet [Target IP/Subnet Mask] --gateway␣
↪[Gateway IP] fairino-net
```

Using 192.168.56.0/24 as an example: *docker network create –driver bridge –subnet 192.168.56.0/24 –gateway 192.168.56.1 fairino-net*



6. Reconnect the container to the newly created network;

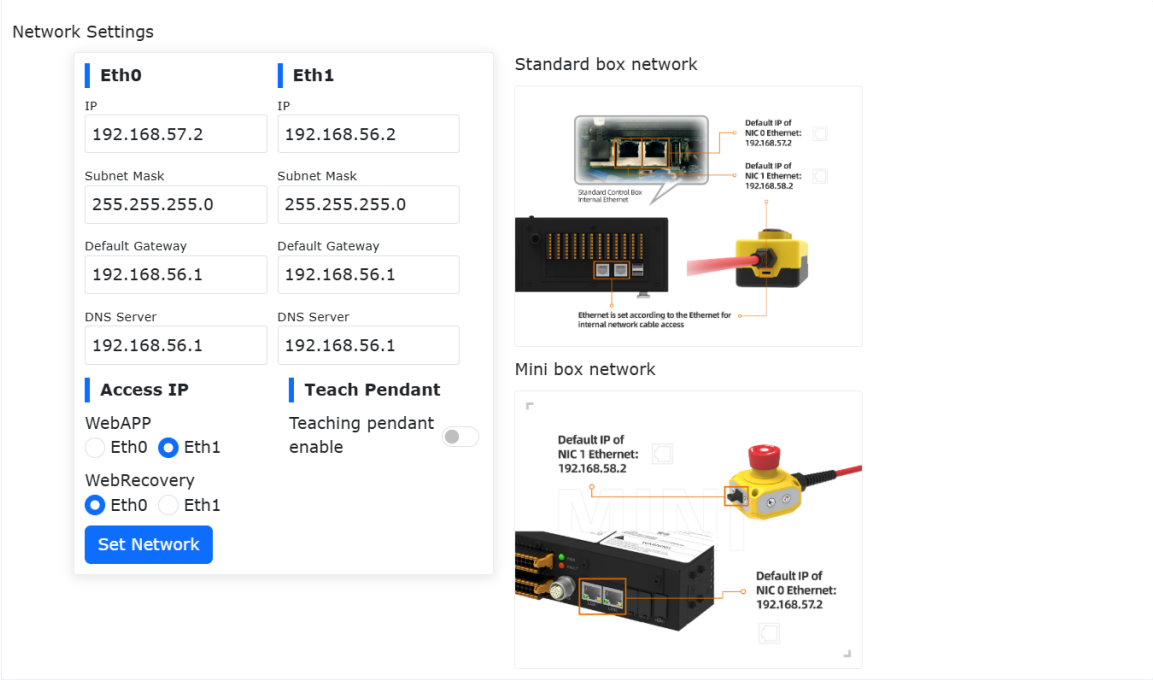
```
docker network connect fairino-net [Container ID]
```



7. Restart the container;

```
docker start [Container ID]
```

8. Now open the browser, enter the modified IP address, to access the web interface and operate the virtual robot.



3.2.3 Virtual Machine Version Upgrade/Downgrade

3.2.3.1 Overview

This manual details the standard procedure for performing software upgrades and downgrades when using the FAIRINO SimMachine Docker virtual machine, and systematically outlines key considerations during the version change process.

3.2.3.2 Upgrade/Downgrade Preparation and Considerations

3.2.3.2.1 Operation Preparation

1. A deployed and normally functioning FAIRINO SimMachine Docker virtual machine. Deployment tutorial see “User Manual - Linux Docker Image Deployment”;
2. Software upgrade package for the Docker virtual machine version. Download location see “Data Download - FAIRINO SimMachine Docker”. After extraction, contents include the latest version docker image FAIRINOSimMachine.tar and the software upgrade package software.tar.gz.

3.2.3.2.2 Considerations

1. Data Backup: It is recommended to perform a backup before upgrading, method see “Data Backup” chapter, to avoid data loss due to upgrade anomalies.
2. Version Restrictions:

Table 2.3-1 Upgrade/Downgrade Version Restrictions

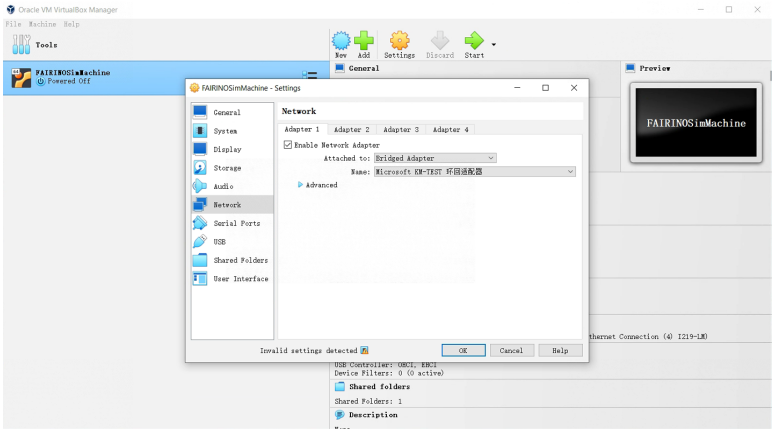
Operation Type	Condition/Restriction	Step Description
Version Upgrade	Current Version >= 3.7.8	Can upgrade directly
Version Upgrade	Current Version < 3.7.8	Must first upgrade to version 3.7.5 or use the compatibility solution
Version Downgrade	Current AND Target Version >= 3.7.8	Can downgrade directly
Version Downgrade	Current OR Target Version < 3.7.8	Use the compatibility solution
Compatibility Solution	Applicable for both upgrade/downgrade anomalies	See "Compatibility Solution" chapter for detailed steps

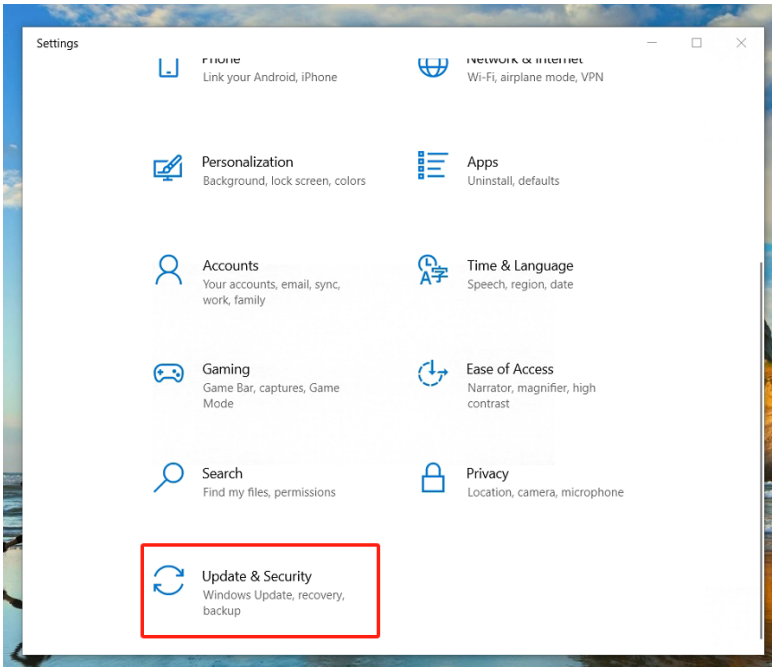
3.3 Appendix

3.3.1 Appendix 1: Enabling Virtualization in the BIOS

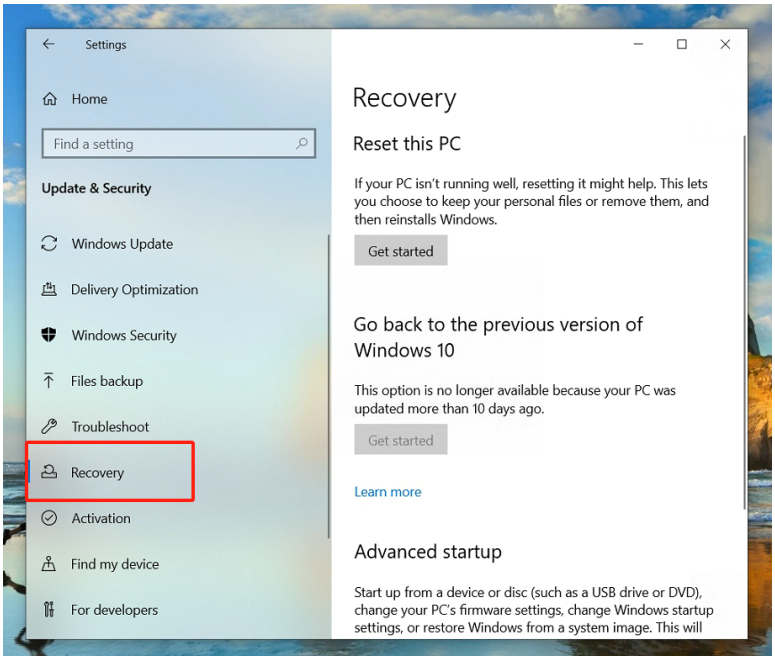
Different models of computers may have different processes to enable virtualization. Take Lenovo ThinkPad series Windows 10 as an example:

- Open PC Settings and select Update and Security.

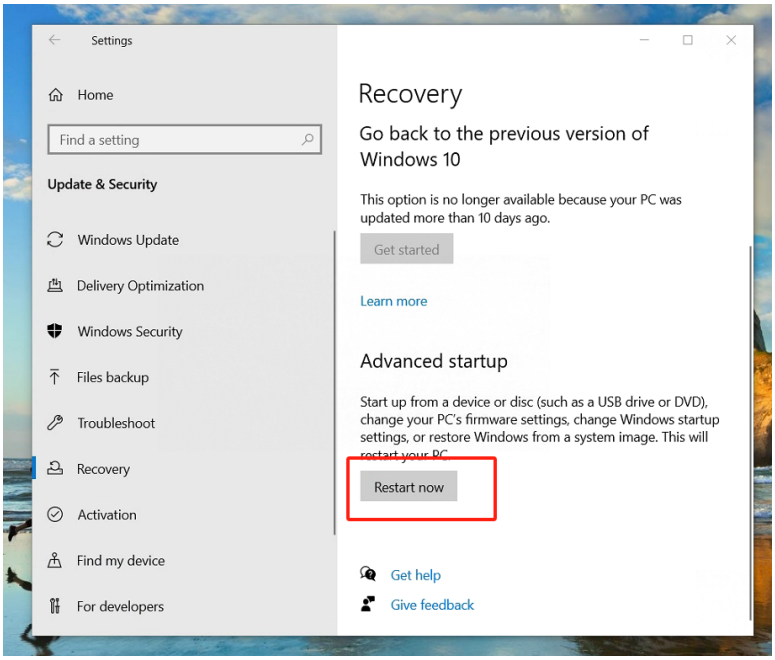




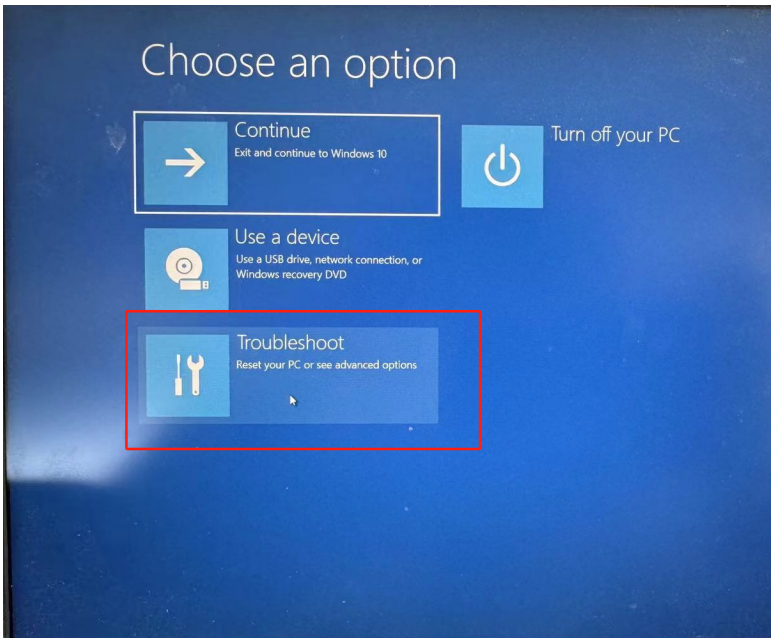
- Select “Recover”.



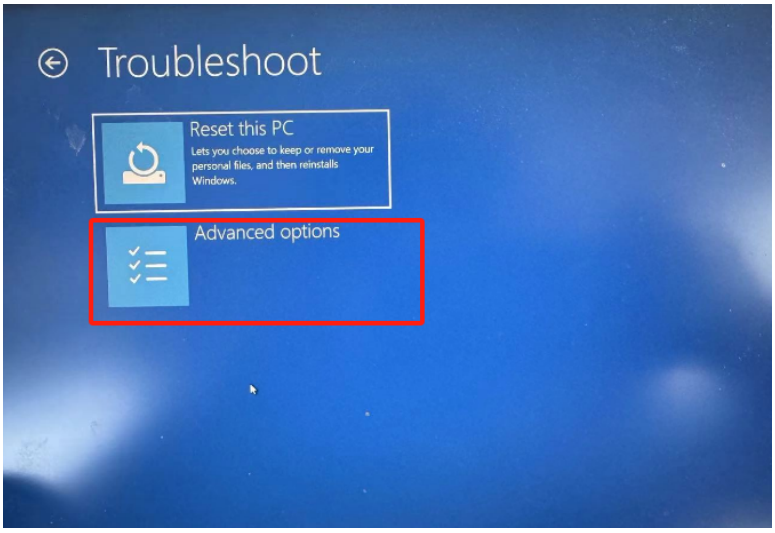
- Select “Restart Now”.



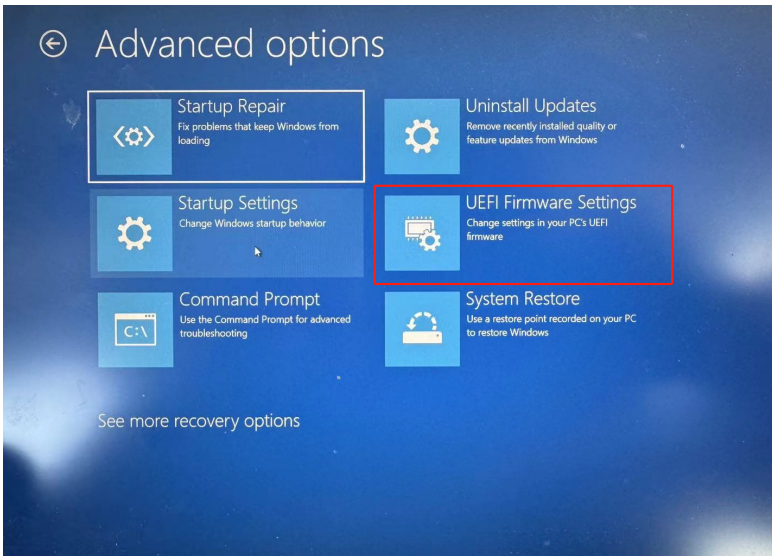
- Select “Troubleshoot”.



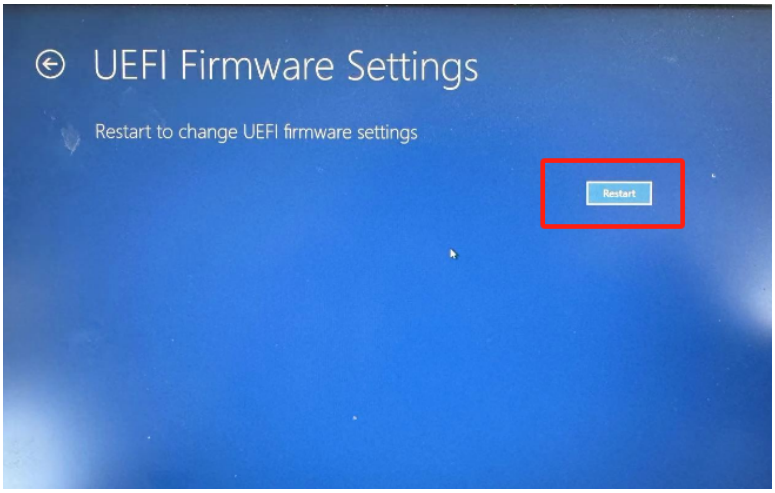
- Select “Advanced Options”.



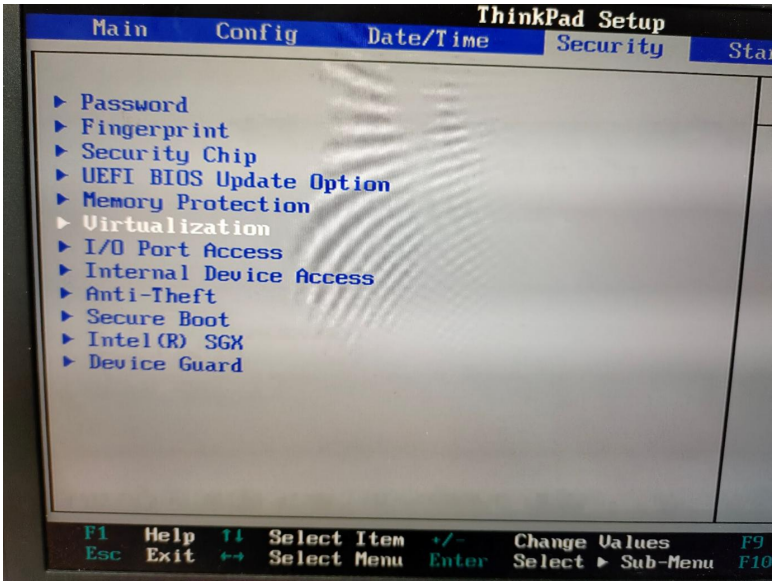
- Select “UEFI Firmware Settings”.



- Select “Restart”.



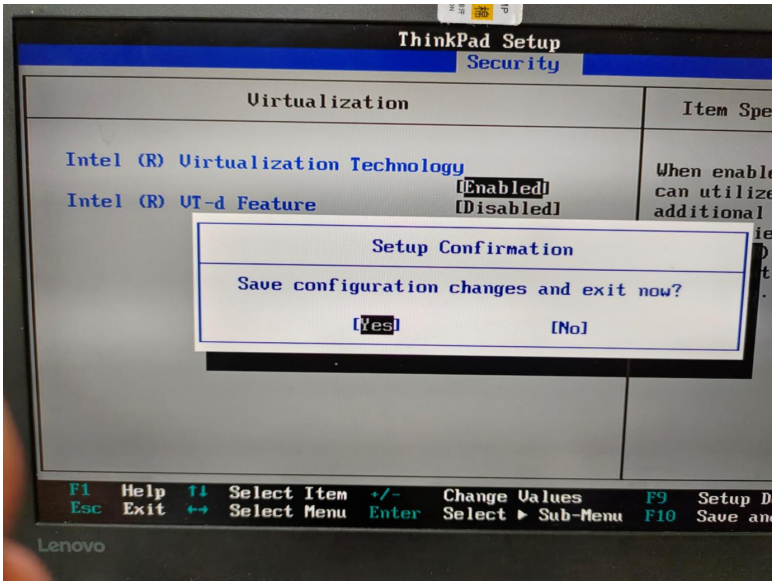
- Select “Virtualization” under Security.



- Select “Enabled” and press “Enter” to confirm.

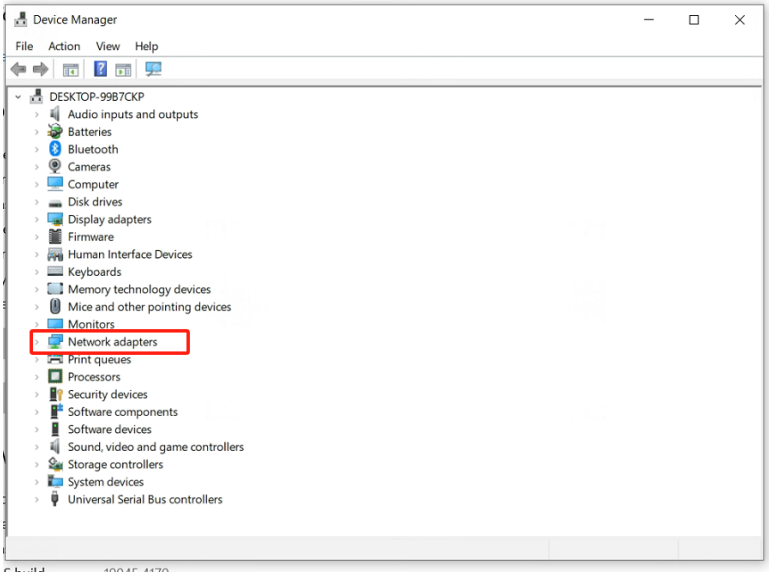


- Press “F10”, select “Yes”, press “Enter” to save changes.

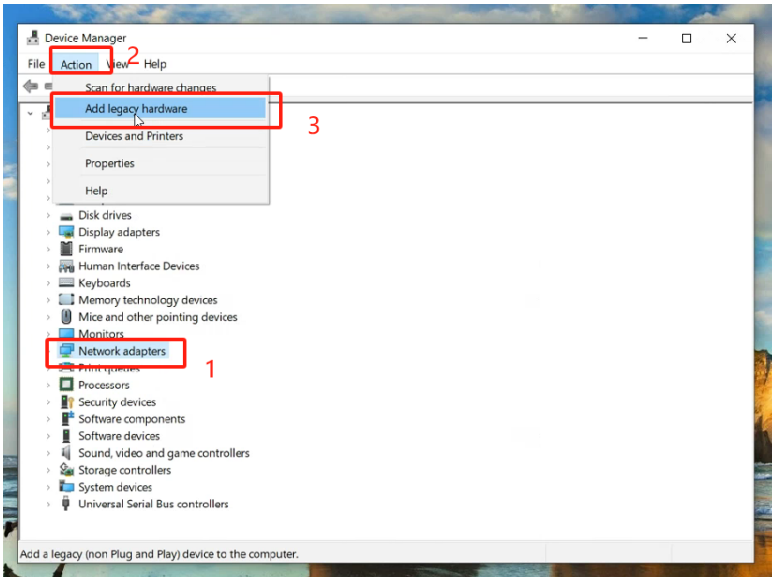


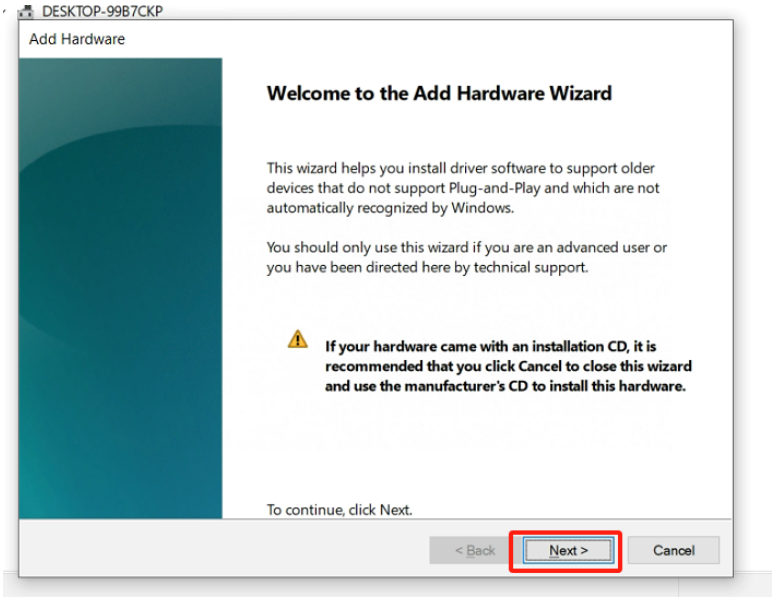
3.3.2 Appendix 2: Adding a Virtual NIC (Loopback Network Adapter)

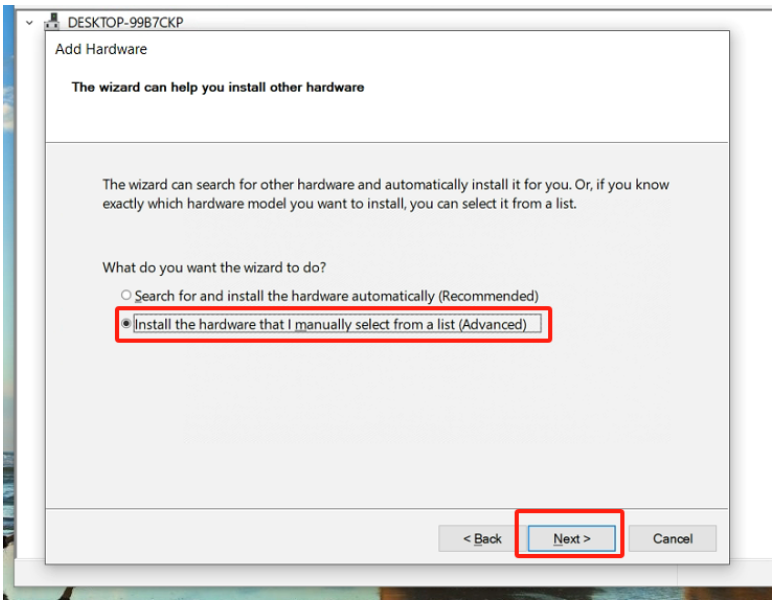
1. Open Device Manager, press “Windows Key - X”, select “Network adapters”.

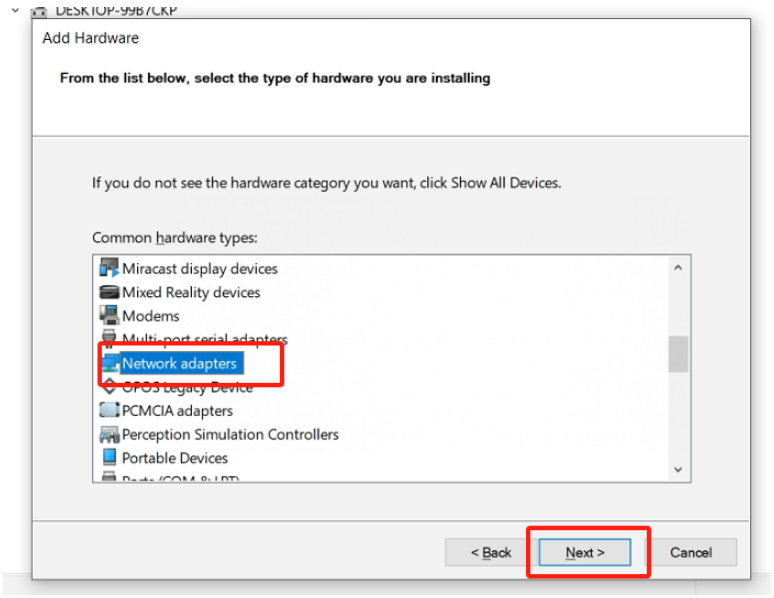


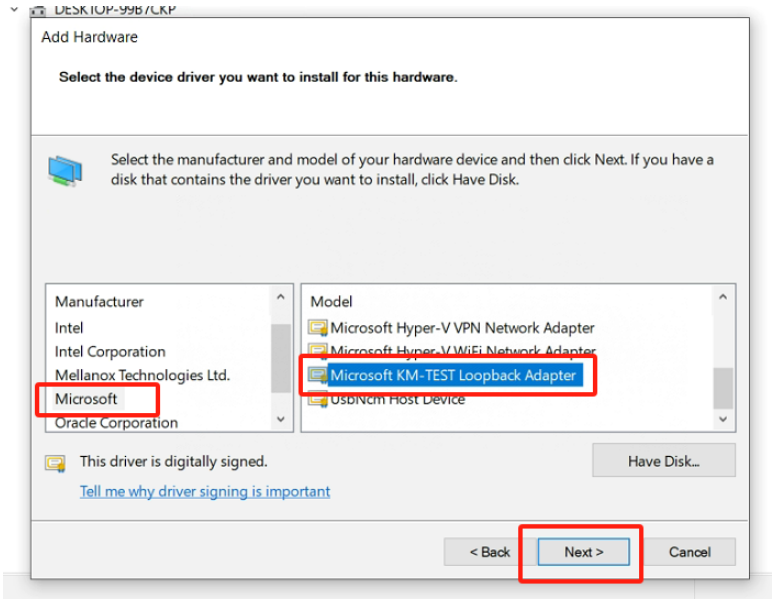
2. Add a network.

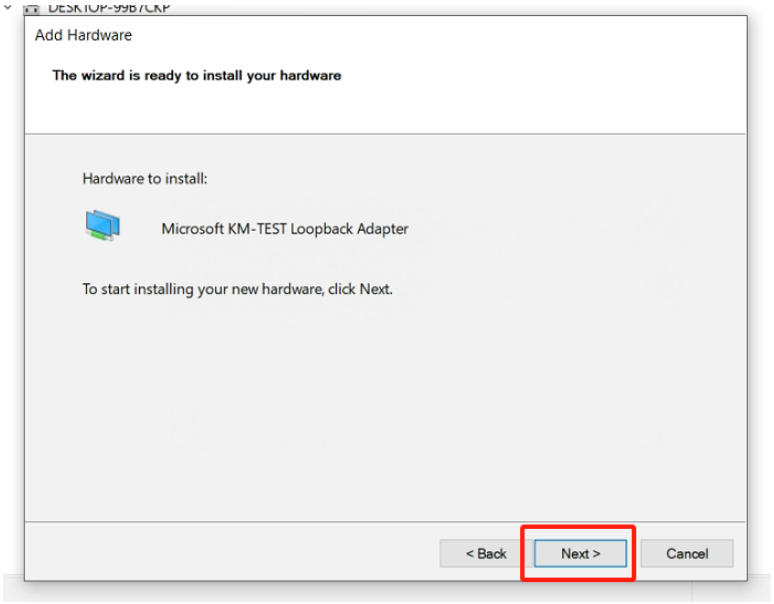

Device Manager
File
Action
View
Help
2
Scan for hardware changes
Add legacy hardware
Devices and Printers
Properties
Help
3
Disk drives
Display adapters
Firmware
Human Interface Devices
Keyboards
Memory technology devices
Mice and other pointing devices
Monitors
Network adapters
1
Processors
Security devices
Software components
Software devices
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers
Add a legacy (non Plug and Play) device to the computer.

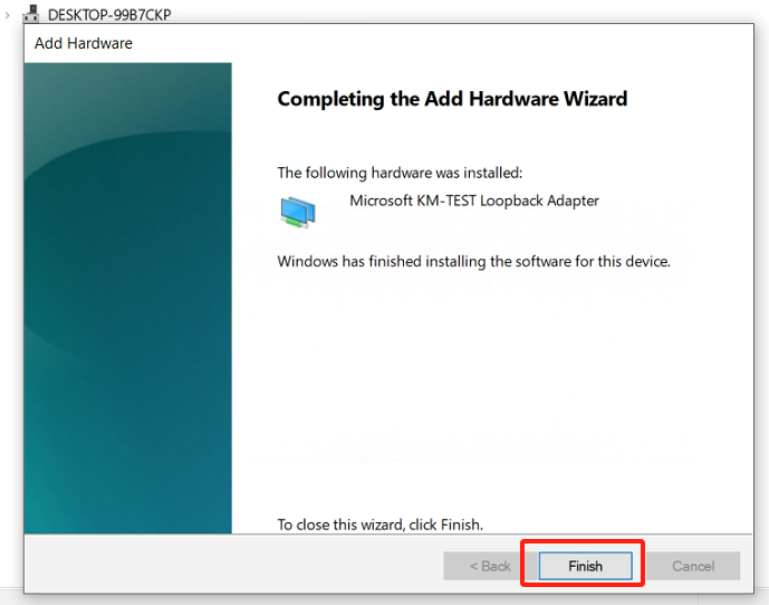

DESKTOP-99B7CKP
Add Hardware
Welcome to the Add Hardware Wizard
This wizard helps you install driver software to support older devices that do not support Plug-and-Play and which are not automatically recognized by Windows.
You should only use this wizard if you are an advanced user or you have been directed here by technical support.
If your hardware came with an installation CD, it is recommended that you click Cancel to close this wizard and use the manufacturer's CD to install this hardware.
To continue, click Next.
< Back
Next >
Cancel


DESKTOP-99B7CKP
Add Hardware
The wizard can help you install other hardware
The wizard can search for other hardware and automatically install it for you. Or, if you know exactly which hardware model you want to install, you can select it from a list.
What do you want the wizard to do?
Search for and install the hardware automatically (Recommended)
Install the hardware that I manually select from a list (Advanced)
< Back
Next >
Cancel

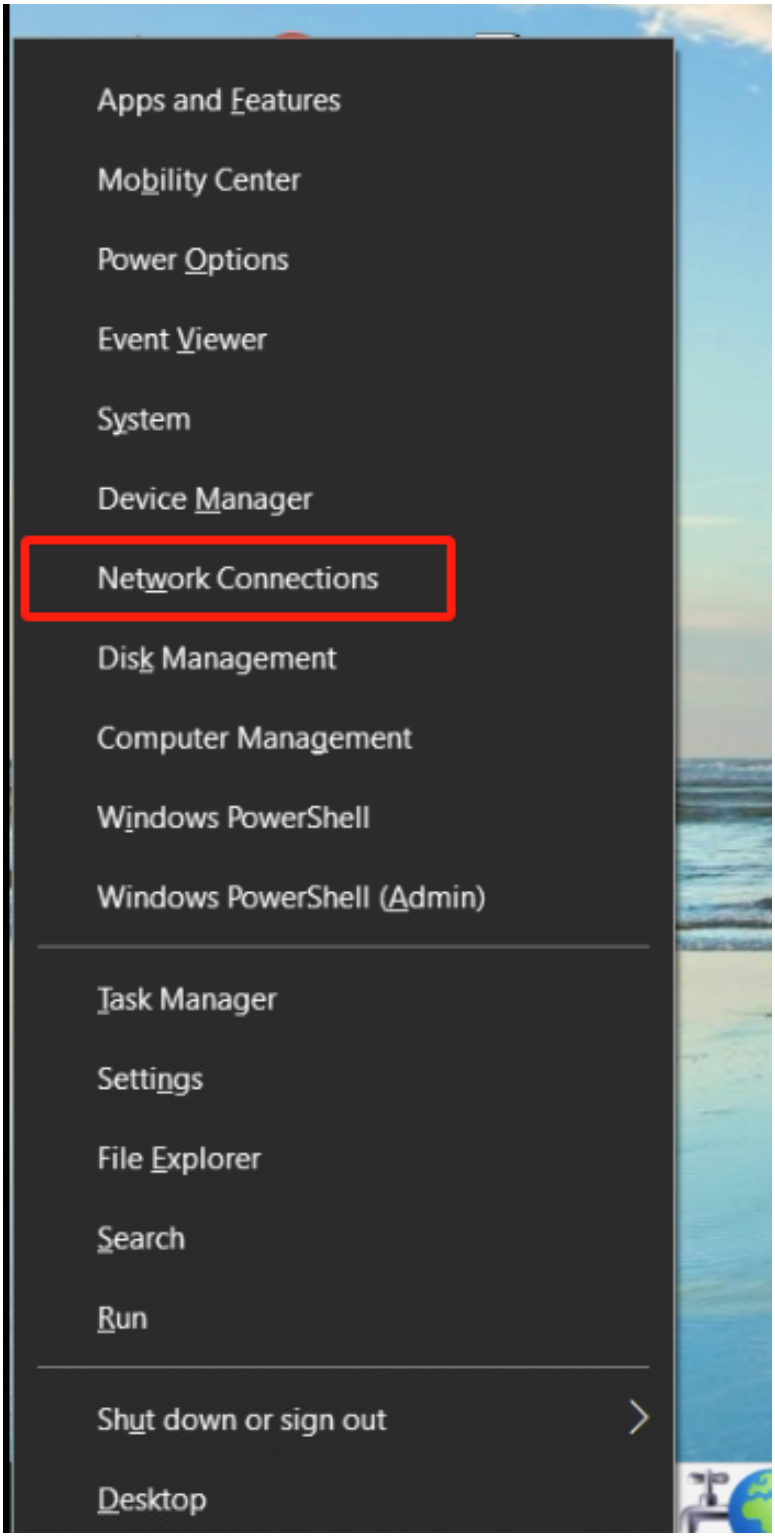

Add Hardware
From the list below, select the type of hardware you are installing
If you do not see the hardware category you want, click Show All Devices.
Common hardware types:
Miracast display devices
Mixed Reality devices
Modems
Network adapters
PCMCIA adapters
Perception Simulation Controllers
Portable Devices
< Back
Next >
Cancel

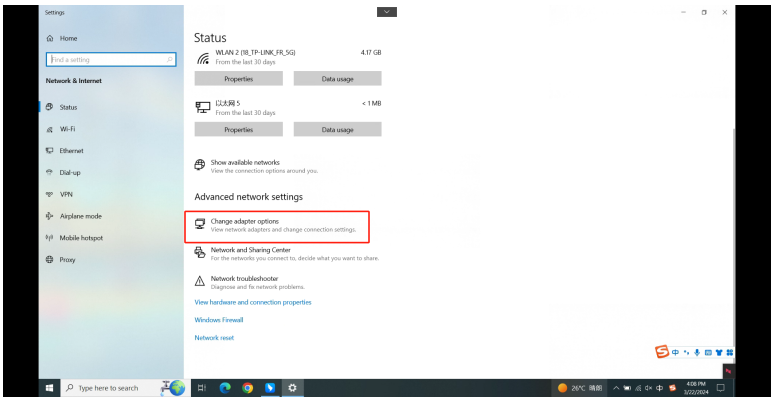

Add Hardware
Select the device driver you want to install for this hardware.
Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.
Manufacturer
Intel
Intel Corporation
Mellanox Technologies Ltd.
Microsoft
Oracle Corporation
Model
Microsoft Hyper-V VPN Network Adapter
Microsoft KM-TEST Loopback Adapter
This driver is digitally signed.
Tell me why driver signing is important
Have Disk...
< Back
Next >
Cancel

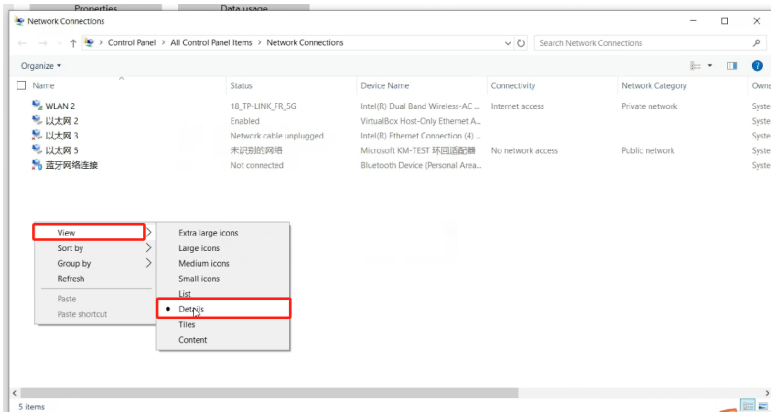

Add Hardware
The wizard is ready to install your hardware
Hardware to install:
Microsoft KM-TEST Loopback Adapter
To start installing your new hardware, click Next.
< Back
Next >
Cancel

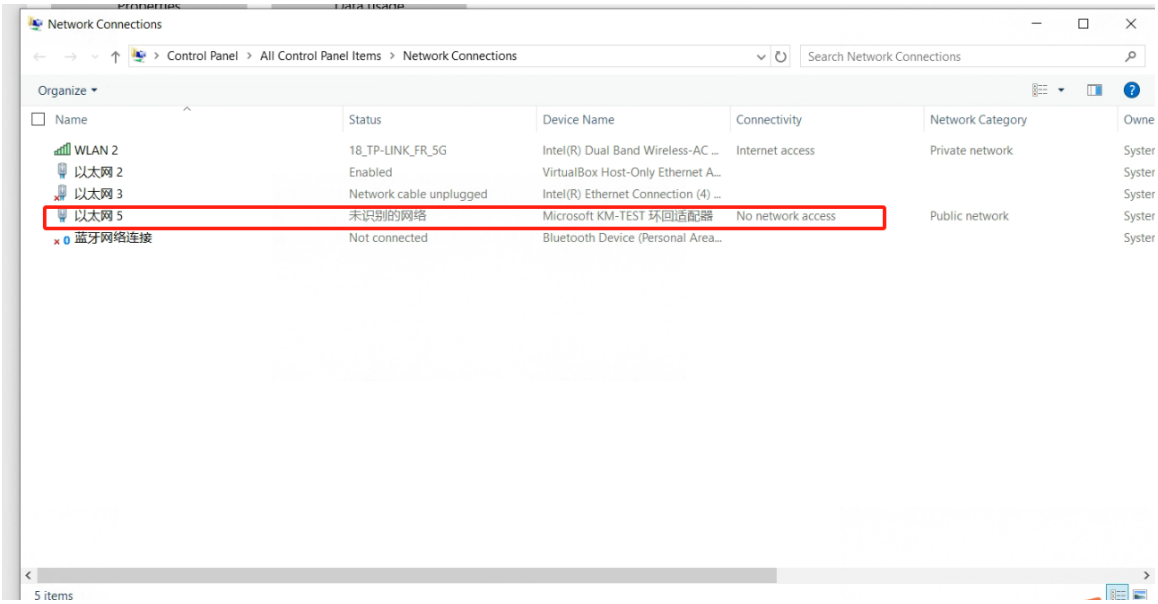


3. View the virtual network card, press the “Windows key - X” , select “Network Connection”.


Apps and Features
Mobility Center
Power Options
Event Viewer
System
Device Manager
Network Connections
Disk Management
Computer Management
Windows PowerShell
Windows PowerShell (Admin)
Task Manager
Settings
File Explorer
Search
Run
Shut down or sign out
Desktop

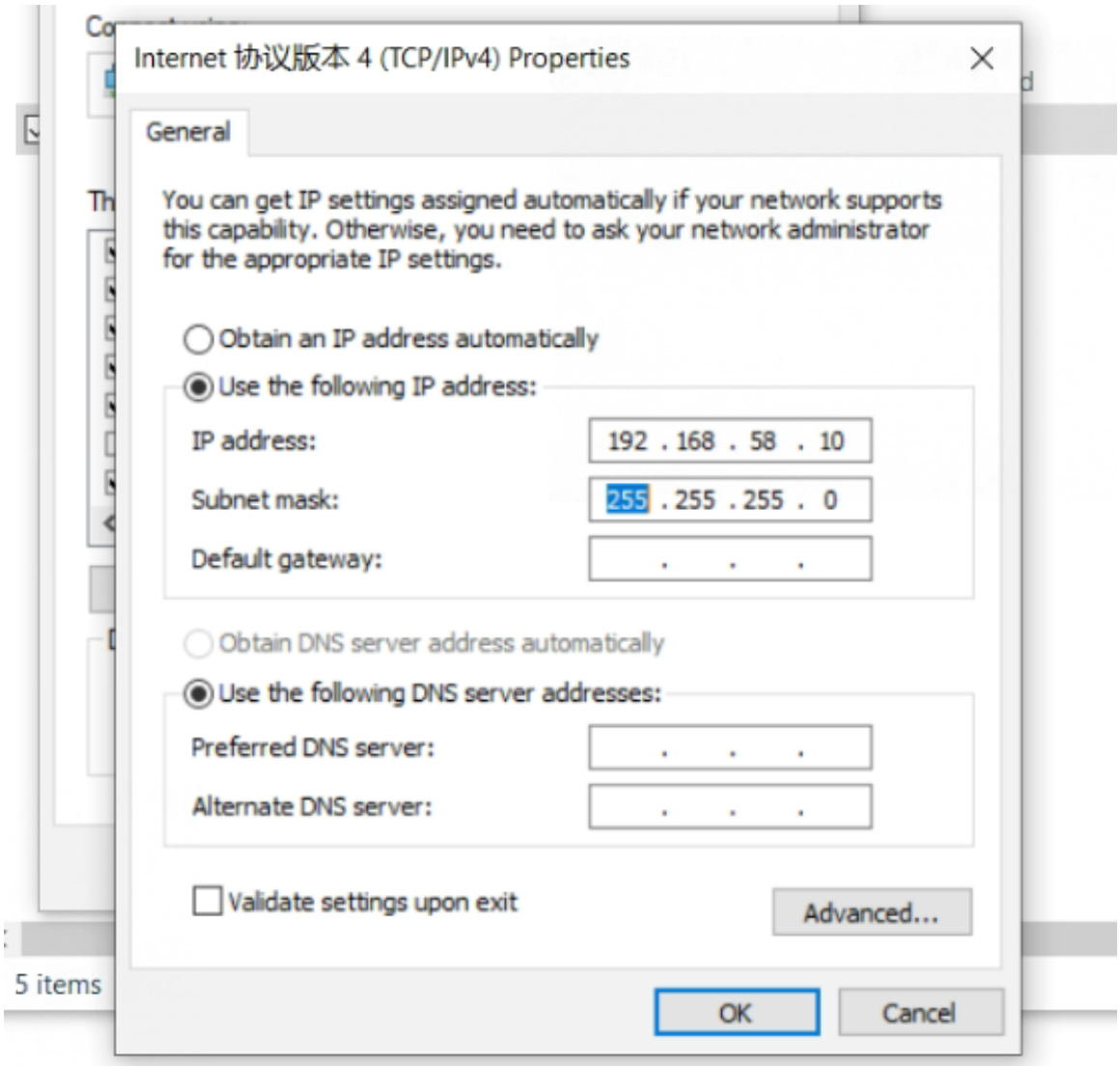




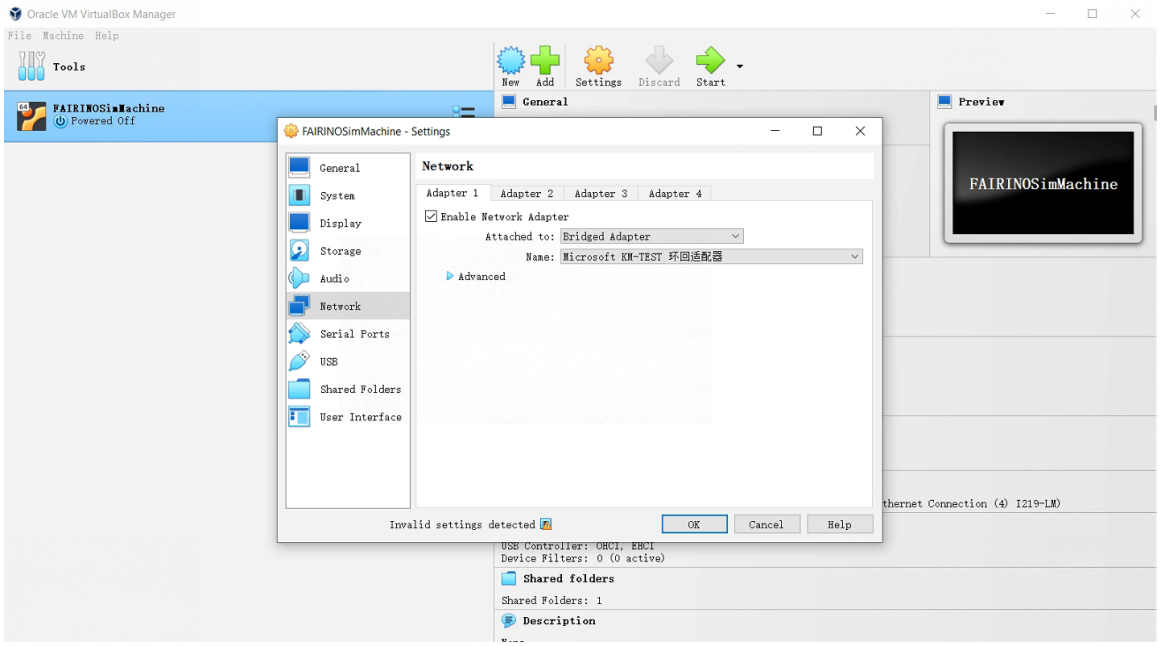


4. Configuring a Loopback Adapter Network.

- IP address: 192.168.58.XXX (same network segment as 192.168.58.2) .
- Subnet mask: 255.255.255.0.



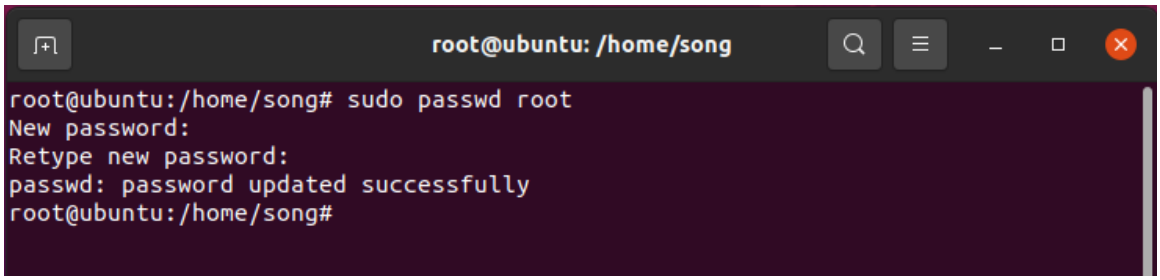
5. Open the Virtualbox network configuration, select “Loopback Adapter Network” for the network card name, and start the virtual machine.



3.3.3 Appendix 3: Root Permissions

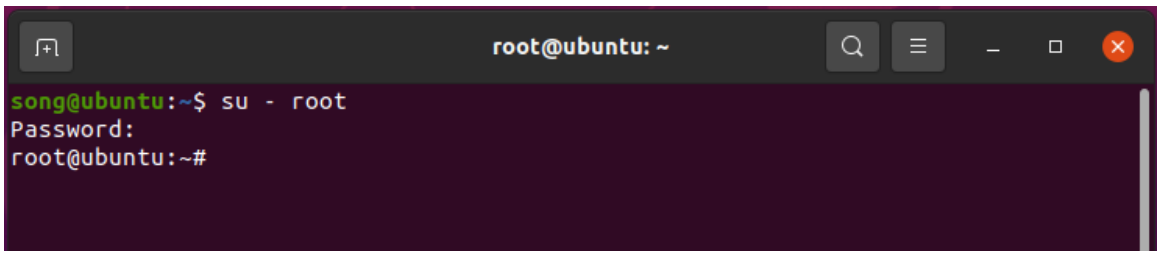
After Ubuntu is installed, the root user is not allowed to log in by default, and the password is empty. If you want to log in as the root user, you must first set a password for the root user.

1. Open the terminal, type *sudo passwd root*, press Enter, and then enter the password several times. A success message will be displayed once the password is set.



2. Continue in the terminal by entering the command *su - root* to switch users, and press Enter to input the password.

Warning: When entering the command, be sure to include the hyphen in "su -". The option "-" signifies that the environment variables should be switched along with the user, and it is crucial not to omit it.



3.3.4 Appendix 4: Docker Basic Commands

1. Docker help command:

```
docker --help
```

2. Start Docker :

```
systemctl start docker
```

3. Stop Docker :

```
systemctl stop docker
```

4. Restart Docker :

```
systemctl restart docker
```

5. Set Docker to start automatically with the service :

```
systemctl enable docker
```

6. Check the running status of Docker :

```
systemctl status docker
--If it's running, you'll see "active" in green
```

7. Docker container :

```
docker images: List the downloaded images.
docker rmi [image_id_or_name]: Remove a local image.
docker rmi -f [image_id_or_name]: Force remove an image.
docker build: Build an image.
docker search [image_id_or_name]: Search for an image by keyword in the Hub.
docker pull [image_id_or_name]: Download an image
docker images: List the downloaded images.
docker rmi [image_id_or_name]: Remove a local image.
docker rmi -f [image_id_or_name]: Force remove an image.
docker build: Build an image.
```

8. Docker container :

```
docker ps: List running containers.
docker ps -a: List all containers, including those not running.
docker stop [container_id_or_name]: Stop a container.
docker kill [container_id]: Force stop a container.
docker start [container_id_or_name]: Start a stopped container.
docker inspect [container_id]: View all information about a container.
docker container logs [container_id]: View container logs.
docker top [container_id]: View processes inside the container.
docker exec -it [container_id] /bin/bash: Enter the container.
exit: Exit the container.
docker rm [container_id_or_name]: Remove a stopped container.
docker rm -f [container_id]: Remove a running container.
docker exec -it [container_ID] sh: Enter the container using the shell.
```

CHAPTER

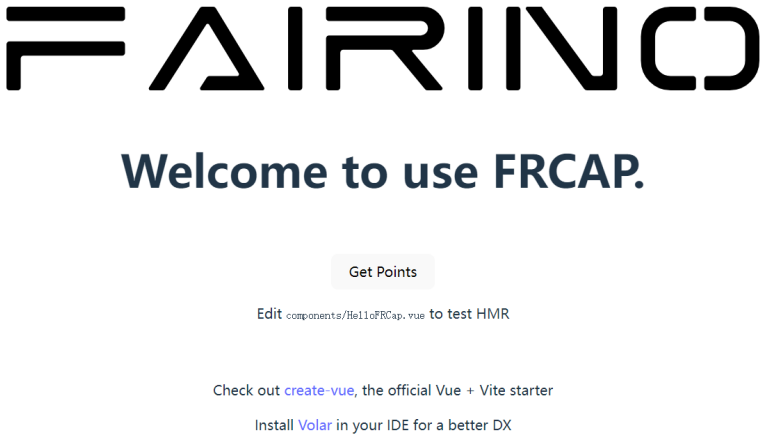
FOUR

FRCAP PLUG-IN SYSTEM

4.1 Introduction

FRCap is a web-based plug-in that can be integrated into the collaborative robot WebApp. FRCap expands robot functions and application scenarios by building a collaborative robot WebApp configuration page or application based on No de.js and Vue3's Element plus, frcap-ui and frcap-api modules.

Essentially, FRCap is a Web application running in the Node.js environment, independent of WebApp, and WebApp provides management and access services. FRCap can interact with the robot controller through the official interface provided, or customers can write custom interface instructions and processing logic according to actual needs for personalized development.



WebApp + FRCap display diagram

4.2 Quick start

4.2.1 I don't have FRCap

If you don't currently have an FRCap, you can quickly create one in this section.

First, we need to connect to the robot and access the WebApp, open a browser on the local computer and enter the robot's default IP address (http://192.168.58.2) and log in to the WebApp.

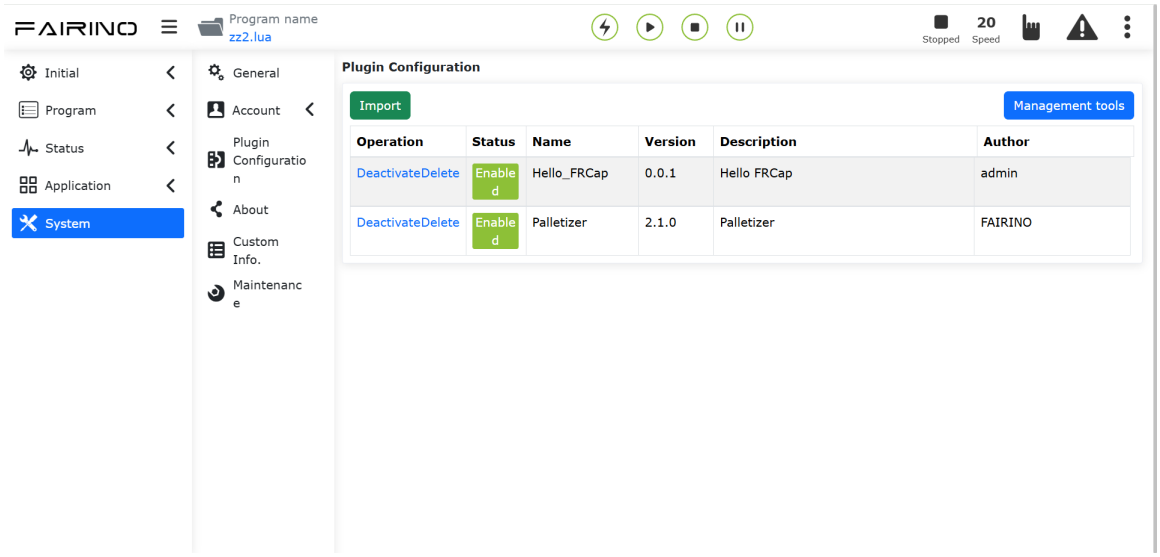


Figure 2-1 "FRCap Management" page of WebApp

In WebApp, a new tab will open in the browser and access the "FRCap Management Tools".

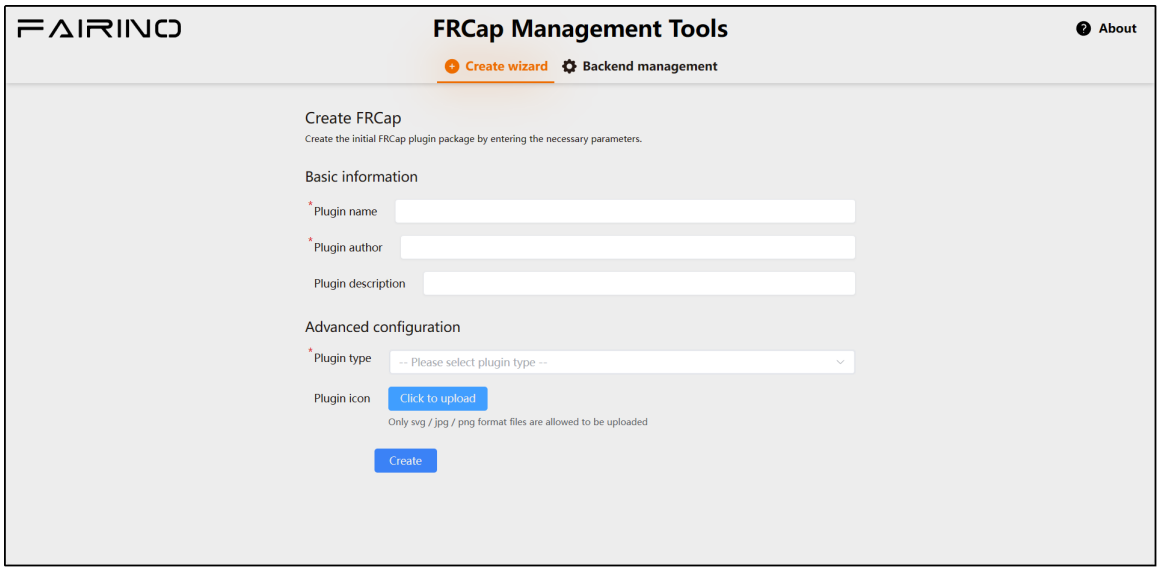


Figure 2-2 FRCap management tool

In the FRCap management tools, and enter or select the following plug-in content:

- Plug-in name: Hello_FRCap.
- Plug-in author: admin.

- Plug-in description: Hello FRCap.
- Plug-in type: Configuration.

In this section, There is no need to upload the plug-in icon. After entering or selecting parameters, click “Create” to complete the creation of FRCap.

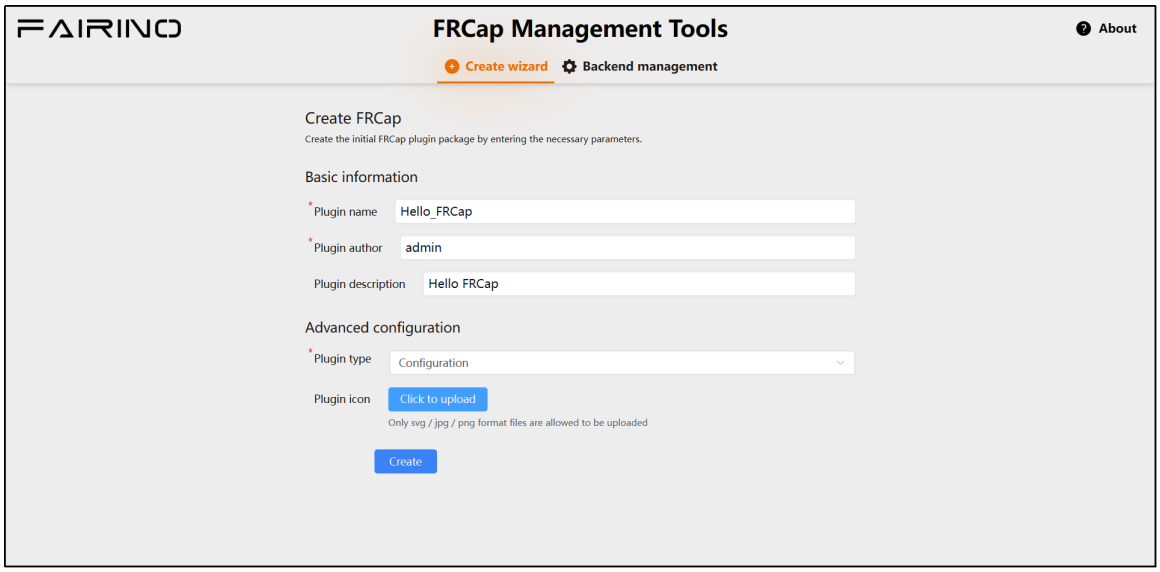


Figure 2-3 FRCap creation wizard

After the creation is successful, jump to the creation success page and display the name of the currently successfully created FRCap. Click “Download” to download the created FRCap to the local computer.

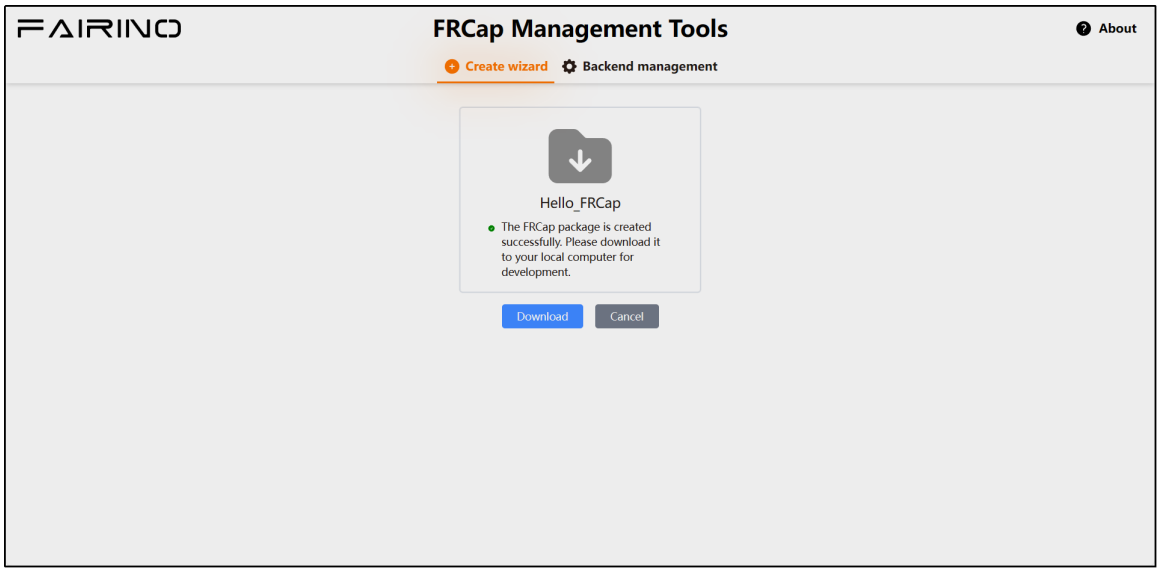


Figure 2-4 Download the Hello FRCap plug-in package

4.2.2 I already have FRCap

If you already have an FRCap project folder and it conforms to the FRCap project structure , please read Build FRCap directly FRCap directly.

If you already have a complete plug-in package with the file extension name ".plugin", please read Hello FRCap.

4.2.3 Build FRCap

If you already have a complete plug-in package with the file extension name ".plugin", please read Hello FRCap directly.

Depending on the system you are currently using, first open the build script, modify the buildName parameter to the name you want, then save and close, and execute the corresponding script in the terminal.

- Start the terminal in Windows and run the following commands:

```
./build.bat
```

- Start a terminal in Linux and run the following commands:

```
./build.sh
```

After the build is completed, a package file with the file name FRCap and the file suffix ". frcap " is generated in the FRCap project directory.

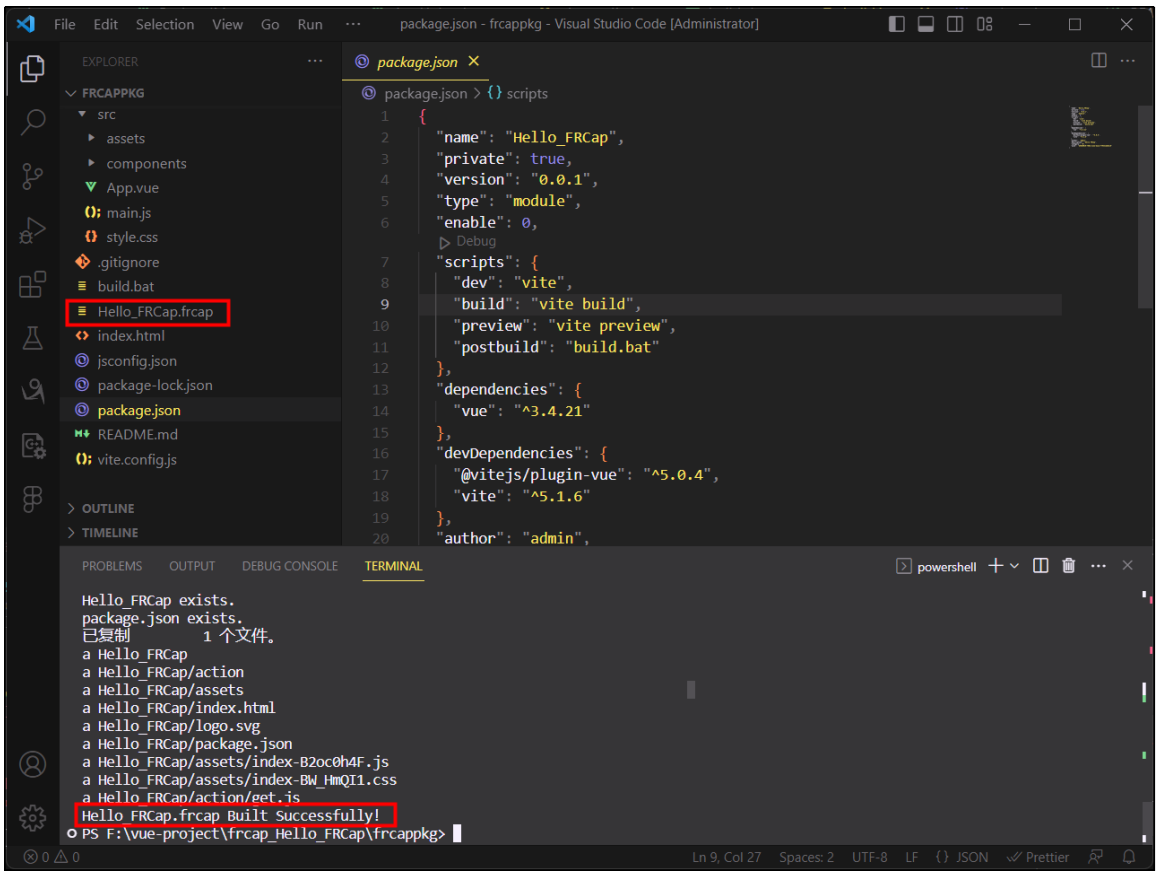


Figure 2-5 The completed FRCap package file

4.2.4 Hello FRCap

After the FRCap project is built, open a browser on your local computer and enter the robot's default IP address (http://192.168.58.2) and log in to the WebApp, click "System Settings" -> "FRCap Management" -> "Import" . Select the built FRCap package file with the ". frcap " suffix , open it and upload it. After the upload is successful, the imported FRCap information will be displayed in the plug-in information list below.

Use the operation bar in the list to control whether FRCap is enabled and deleted, and check the enablement status of FRCap in the start/stop status bar.

After Hello FRCap is enabled, it can be used in "Auxiliary Applications"-> "FRCap" -> "Hello FRCap". This page carries the configuration class FRCap , which can be full-width or half-width. By default, it is displayed in half-width.

At this point, you have completed the entire plug-in quick creation and usage process.

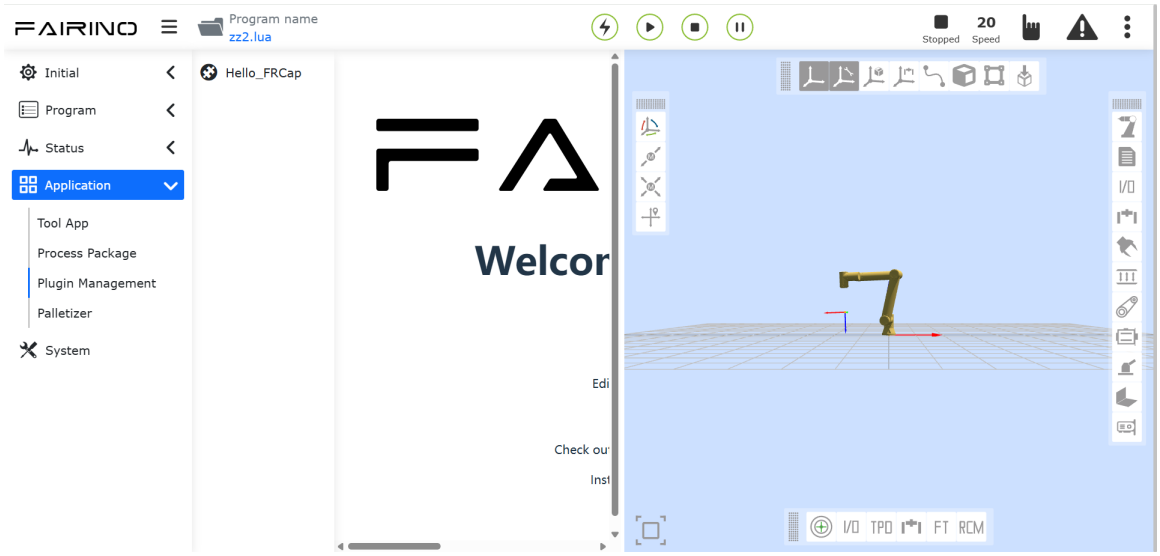


Figure 2-6 Hello FRCap content

If you want to know the detailed creation wizard guidance, you can continue to view the creation wizard.

To learn about the tool environment and guidance required to develop FRCap, please view the Development Guide.

For specific guidance on using FRCap in WebApp, please see Using FRCap in WebApp.

4.3 Create wizard

"Creation Wizard" is a tool in FRCap-Tools, through which you can easily and quickly initialize an FRCap project by entering a small number of parameters.

4.3.1 Parameter configuration

Creating FRCap mainly requires two types of parameters, basic information of FRCap and configurations at all levels. They will be explained separately below.

4.3.1.1 Basic information

The basic information includes "plug-in name", "plug-in author" and "plug-in description".

Plugin name:

- Required fields;
- There are no restrictions on input characters and character length, and no spaces are allowed;
- It is recommended that the name should not exceed 7 CKJ characters (Chinese, Japanese, Korean, etc.) or 10 all uppercase Latin letters or 14 all lowercase Latin letters (English, French, etc.).
- Recommended examples:
  - Palletizer;
  - Polishing process software;
  - Device Config;
  - HELLO FRCAP.

Plugin author:

- Required fields;
- There are no restrictions on input characters and character length. For example, you can enter your personal name, company name, etc.;
- Recommended examples:
  - Zhang San
  - Franklin Peter
  - FAIR Innovation (Suzhou) Robot Systems Co., Ltd.

Plugin description:

- Optional fields;
- There are no restrictions on input characters or character length, just briefly describe your plug-in.

4.3.2 Advanced configuration

Plug-in type:

- Required fields;
- The type options are “Configuration” and “Application” respectively.
- “Configuration” is recommended for FRCap that implements relatively simple configuration and control operations such as setting parameters and button operations. After importing, use it in “Auxiliary Application” -> “FRCap” in WebApp.
- “Application” recommends FRCap that implements complex process scenarios, such as palletizing scenarios, welding processes and other industry applications. After importing, use it directly under “Auxiliary Applications” in WebApp.

Plug-in icon:

- Optional fields;
- You can upload the company logo or any icon you want to use for the icon. Please pay attention to the copyright. Our company is not responsible for any copyright issues caused by any reason;
- If you do not upload an icon, the company’s “FAIRINO” Logo icon will be used by default in the exported FRCap project. You can replace and modify it in the public folder under the project folder. This icon is for initialization purposes only. Please do not use the “FAIRINO” Logo directly in any commercial scenarios.

4.3.3 Download

After all the above parameters are configured and the FRCap is successfully created, you will be redirected to the download page. You need to confirm that the name is correct before you can download the created FRCap project to your local computer for subsequent development work and construction.

The downloaded plug-in is in “.tar.gz” compressed format.

On Windows systems we recommend using 7-Zip software to decompress.

On Linux systems, you can use the following command in the terminal to decompress:

```
tar -zxvf frcap_{FRCapName}.tar.gz
```

4.4 Backstage management

4.4.1 Backend upgrade

In the “Software Download” section of this document, you can download the latest FRCap background system upgrade package. After the downloaded upgrade package is uploaded successfully, the background will automatically upgrade.

The upgrade is successful as shown in the picture below.

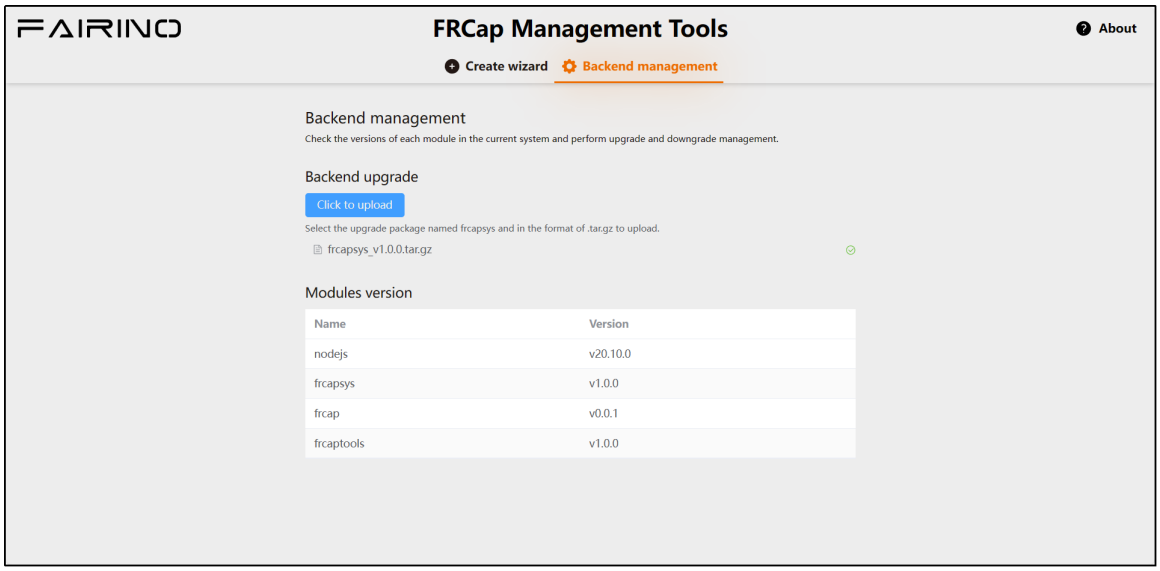


Figure 4-1 FRCap background system upgrade successful

Note: After the upgrade is successful, you need to restart the robot control box for it to take effect.

4.4.2 Modules version

The initial versions of each module of the FRCap backend system in the first version:

- Node.js:v20.10.0.
- FRCapSys:v1.0.0.
- FRCap:v0.0.1.
- FRCapTools:v1.0.0.

For specific version update content, please see the version update instructions.

Note: Node.js will not be upgraded with versions at this time.

4.5 Development guidance

4.5.1 Development environment and conditions

The development environment must meet the following minimum configuration:

- CPU:1.6 GHz or faster processor;
- RAM:>=1 GB(more than 2 GB recommended);
- ROM:>=128GB;
- OS:Requires Windows 10 or higher, macOS 10.1 5 or higher, Linux (x64) system (Ubuntu, Debian, etc.).

- Controller Version: Check in WebApp “System Settings-About”. Note the distinction between QX and LA in development environment. Avoid using ES6+ syntax and other modern JavaScript features in QX environment examples.

We have encapsulated some interfaces and modules, but if you want to achieve a better development effect, it is recommended to have a certain understanding of web development, and it is best to be familiar with the following technologies:

- HTMLJavaScript/TypeScriptCSS;
- Vue3;
- Vite;
- Node.js.

4.5.2 Development tools

We recommend using the latest Visual Studio Code (VSCode) software for development. To download, please visit the VSCode official download page and select the corresponding system to download.

At the same time, the Node.js runtime environment needs to be installed on the local computer. When installing Node.js, tools such as npm will be installed to facilitate package management. Visit the Node.js official download page and select the corresponding system version v20 to download.

When developing in VSCode, you may also use the following VSCode plug-ins, which can be installed and configured as needed .

- Vue;
- ESlint;
- npm Intellisense;
- Vue Language Features (Volar);
- TypeScript Vue Plugin (Volar)Vue.volar;
- Tailwind CSS IntelliSense.

4.5.3 FRCap project structure

FRCap project file structure:

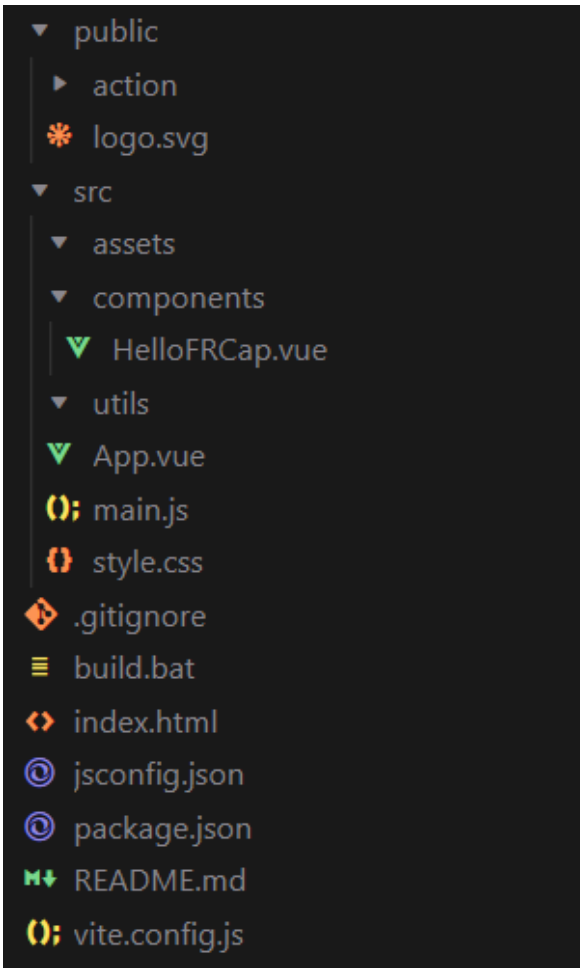


Figure 5-1 FRCap project structure

- Public:

In the public resource folder, the internal files will not be built during the build process, but will be copied directly to the build directory.

The action folder and logo.svg are included by default .

The Action folder is used to store custom command background interface logic files.

Logo.svg is the plug-in icon.

- Src:

Assets folder is mainly used to place static resources.

The Components folder is mainly used to place components.

The Utils folder is mainly used to place utility classes.

App.vue home page code.

Main.js is mainly responsible for global resource introduction.

Vue framework creation, etc.

Style.css project basic style file.

- Build.bat:Windows platform build script.
- Index.html:The main frame of the page UI.
- Package.json:package description file and compilation strategy, etc.
- Vite.config.js:Vite configuration file.

4.5.4 Front-end frcap-ui and frcap-api use

Frcap-ui provides some HTML controls that have been encapsulated through Vue components, which can be imported into projects for use, reducing the difficulty of page UI development and the amount of code, and improving code readability. Of course, you can also choose some excellent open source UI component libraries, such as Element plus, etc.

First open the terminal in your project path and install frcap-ui.

```
npm install frcap-ui -s
```

After successful installation, introduce it into the components that need to use frcap-ui , taking the button control as an example.

```
import { AppButton } from 'frcap-ui'
```

Then use it in the <template> element of the component.

```
<AppButton button-text="Start" button-type="primary"></AppButton>
```

Preview the development project results in the browser.



Figure 5-2 AppButton

Currently we provide 4 common control components.

- AppButton:Button component.
  - buttonType: button type, String, corresponding to different button styles, the default is primary.
    - primary:blue;
    - secondery:gray;
    - safety:green;
    - warning:yellow;

        * serious:red.
    - buttonText:Button text, String, default value is “primary”.
- AppInput:Input component.
    - Type:required item, String, default value text. Indicates the type of input box.
        * Number:number input box;
        * Text:text input box.
    - inputLabel:required item, String, input box label text.
    - inputUnit:String, input box unit text.
    - hasUnit:Boolean, default false, indicates whether unit text is required.
    - isEmptyErr:Boolean, whether the input box is empty.
    - isReadonly:Boolean, whether the input box is read-only.
- AppSelect:Selection box component.
    - selectionLabel:required item, String, selection box label text.
    - optionsData:required items, Array, option data.
- Modal:modal window component.
    - show:Boolean, whether to pop up the modal window.
    - title:String, modal window title.

In order to facilitate the development of custom instructions that may be created in FRCap , we have built Http requests and APIs into the initial FRCap project downloaded in the “Creation Wizard”. In this way, both custom instructions and default provided instructions can be placed in the api.js file in frcap-api . The specific path of api.js is “./src/api/api.js”.

The use of F rcap-api is similar to frcap-ui , as follows:

1. api in files that need to use api , such as components .

```
import api from '@/api/api';
```

2. Call the default instructions provided in the interface.

```
api.getRobotStatus()
```

3. Write processing logic in the returned promise.

```
api. getRobotStatus ()
.then((res) => {
console.log(res.data);
})
.catch((err) => {
    console.error(err);
});
```

4.5.5 Backend Custom Command Development

4.5.5.1 Database Operation Example (LA)

1. Import database module

```
var node = "/usr/local/etc/node/sys"
var Sqlite3_Action = require(node + '/better-sqlite3/better-sqlite3.js');
var sqlite = new Sqlite3_Action();
```

2. Get content from point database

```
// Match cmd
case 'get_points':
// Write SQL statement to return data to frontend sorted by numeric ascending + first␣
↪letter ascending + Chinese characters ascending
var sql = "select * from points order by name ASC";
var sql_data = sqlite.queryall(DB_POINTS, sql);
// Format json data
for (var i = 0; i < sql_data.length; i++) {
    response_data[sql_data[i].name] = sql_data[i];
}
// Return json data to frontend
event_socket.emit('response', res, response_status, response_data);
break;
```

4.5.5.2 Database Operation Example (QX)

Note: QX version uses JSON format files for data storage.

1. Import database module

```
var node = "/usr/local/etc/node/sys"
var sqlite_adapter = require(node + '/jsdb/sqlite_adapter');
var db = new sqlite_adapter.Database(palletizing_db);
```

2. Database usage example

```
// Execute SELECT query and get all rows
var rows = db.queryall('SELECT * FROM box_cfg');
console.log('result:', rows);

// Execute SELECT query and get single row
var row = db.queryget('SELECT * FROM box_cfg WHERE flag=1');
console.log('result:', row);

// Execute UPDATE statement
db.run('UPDATE box_cfg SET height=100 WHERE flag=1', function(err) {
   if (err) {
      console.error('Update failed:', err);
   } else {
```

(continues on next page)

(continued from previous page)

```
      console.log('Update success');
   }
});

// Execute parameterized query
var params = [100, 200, 300, 1];
db.run('UPDATE box_cfg SET height=?, width=?, length=? WHERE flag=?', params,␣
↪function(err) {
   if (err) {
      console.error('update failed:', err);
   } else {
      console.log('update success');
   }
});

// Close database connection
db.close();
```

4.5.5.3 Socket Communication Example

- Import socket communication module

```
var node = "/usr/local/etc/node/sys"
var Socket_Cmd = require(node + '/socket/socket_cmd');
var socket_cmd = new Socket_Cmd();
```

- Send system variable setting command

```
// Match cmd
case 511:
// Get sent data content
content = data_json.content;
// Get sent data length
len = data_json.content.length;
// Assemble sending data
send_content = '/f/bIII1III511III' + len + 'III' + content + 'III/b/f'
// socket send
socket_cmd.send(send_content);
// socket recv (note LA/QX difference)
// LA Version:
socket_cmd.recv().then((recv_data)=>{
   response_data = recv_data;
event_socket.emit('response', res, response_status, response_data);
}).catch((err)=>{
   console.log(err);
})
// QX Version
// socket_cmd.recv().then(function(recv_data){
//     response_data = recv_data;
// event_socket.emit('response', res, response_status, response_data);
// }).catch (function(err){
```

(continues on next page)

(continued from previous page)

```
//     console.log(err);
// })
break;
```

4.6 FRCap in WebApp

4.6.1 FRCap management

Select “System Settings” -> “FRCap Management” in the WebApp. The page ribbon contains:

1. FRCap import: Click the “Import” button to upload;
2. Backend management: includes “Creation Wizard” and “Backend Management” tools;
3. FRCaps information list and operations.

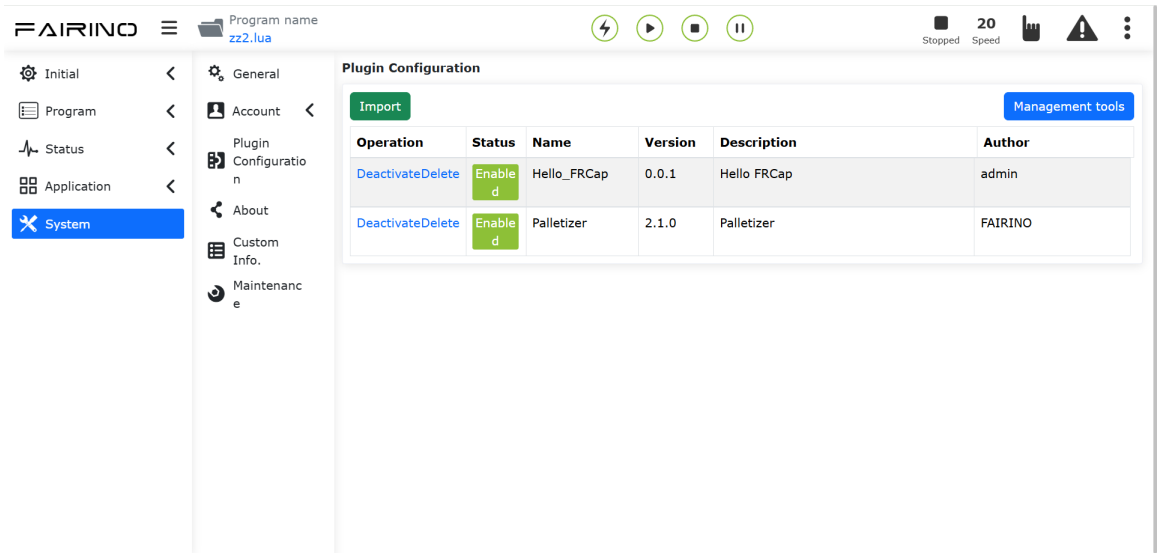


Figure 6-1 FRCap management page functional area

4.6.2 Configuration class usage

The path to configure class FRCap is “auxiliary application” -> “FRCap” -> “[FRCap name]” in the robot WebApp.

4.6.3 Application class use

Application class FRCap is “auxiliary application” -> “[FRCap name]” in the robot WebApp .

4.7 FRCap case

4.7.1 FAIRINO Palletizer

It can be used after uploading the "Palletizer.plugin" under the build folder in the project to the WebApp and registering it.

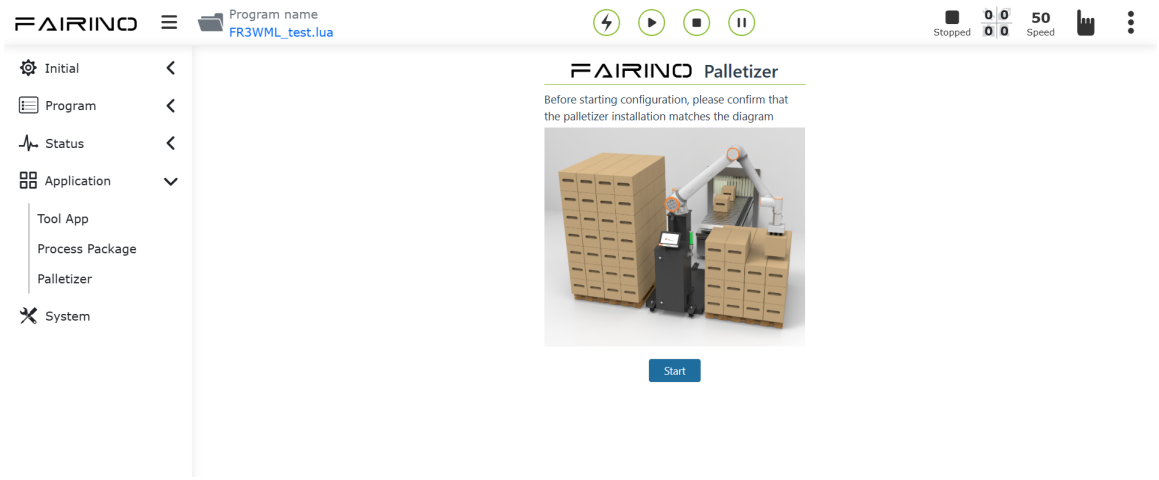


Figure 7-1 Use of FRCap for palletizing

4.7.1.1 Palletizing workpiece configuration

Command name: palletizing_config_box.

Command parameters:

```
/**
* @param  int length piece length
* @param  int width workpiece speed
* @param  int height Work piece height
* @param  int payload artifact payload
* @param  string grip_point workpiece grip point
* /
```

Instruction example:

```
{
   cmd: "palletizing_config_box",
   data: {
      length: 800,
      width: 615,
      height: 312,
      payload: 2.34,
      grip_point: "grippoint"
   }
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @return status:404 "fail"
*/
```

4.7.1.2 Palletizing pallet configuration

Command name: palletizing_config_pallet.

Command parameters:

```
/**
* @param  int front tray front
* @param  int side tray side
* @param  int height Pallet height
* @param  int left_pallet left tray enabled
* @param  int right_pallet Right tray enabled
*/
```

Instruction example:

```
{
   cmd: "palletizing_config_pallet",
   data: {
         front: 1200,
         side: 1000,
         height: 110,
         left_pallet: 0,
         right_pallet: 1
      }
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @return status:404 "fail"
*/
```

4.7.1.3 Advanced palletizing configuration

Command name: palletizing_advanced_cfg.

Command parameters:

```
/**
* @param  string height grab point lifting height
* @param  string x1 palletizing progressive point 1: x offset in x direction, unit mm
* @param  string y1 palletizing progressive point 1: y offset in x direction, unit mm
* @param  string z1 palletizing progressive point 1: z offset in x direction, unit mm
* @param  string x2 palletizing progressive point 2: x offset in x direction, unit mm
* @param  string y2 palletizing progressive point 2: y offset in x direction, unit mm
```

(continues on next page)

(continued from previous page)

```
* @param  string z2 palletizing progressive point 2: z offset in x direction, unit mm
* @param  string time suction waiting time, unit ms
*/
```

Instruction example:

```
{
   cmd: "palletizing_advanced_cfg",
   data: {
   height: "1000",
         x1: "100",
         y1: "100",
         z1: "100",
         x2: "10",
         y2: "10",
         z2: "10",
         time: "1"
      }
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @return status:404 "fail"
*/
```

4.7.1.4 Palletizing equipment size configuration

Command name: palletizing_config_device.

Command parameters:

```
/**
* @param  int x absolute value of the upper right corner of the left pallet relative to the coordinate axis of the robot base coordinate system in the x direction
* @param  int y absolute value of the upper right corner of the left pallet relative to the coordinate axis of the robot base coordinate system in the y direction
* @param  int z absolute value of the upper right corner of the left pallet relative to the coordinate axis of the robot base coordinate system in the z direction
* @param  int angle angle when the robot is installed
*/
```

Instruction example:

```
{
   cmd: "palletizing_config_device",
   data: {
      x: 2400,
      y: 1800,
      z: 120,
      angle: 0
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @return status:404 "fail"
*/
```

4.7.1.5 Palletizing mode configuration

Command name: palletizing_config_pattern.

Command parameters:

```
/**
* @param  int layers palletizing layers
* @param  int box_gap Workpiece pixel spacing, unit: mm
* @param  string sequence palletizing working mode
* @param  int pattern_b_enable Whether mode b is turned on, 1: turned on, 0: not turned on
* @param  string left_pattern_a Left station mode a Cartesian coordinates
* @param  string left_pattern_b Left station mode b Cartesian coordinates
* @param  string right_pattern_a Right station mode a Cartesian coordinates
* @param  string right_pattern_b Right station mode b Cartesian coordinates
* @param  string origin_pattern_a Initial mode a Cartesian coordinates
* @param  string origin_pattern_b Initial mode b Cartesian coordinates
*/
```

Instruction example:

```
{
   cmd: "palletizing_config_pattern",
   data: {
      layers: 8,
      box_gap: 0,
      sequence: "a,b,a,b,a,b,a,b",
      pattern_b_enable: 1,
      left_pattern_a: "{\"1\": [[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3]]}",
      "left_pattern_b": "{\"1\": [[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3]]}",
      "right_pattern_a": "{\"1\": [[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3]]}",
      "right_pattern_b": "{\"1\": [[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3]]}",
      "origin_pattern_a": "[]",
      "origin_pattern_b": "[]"
   }
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @return status:404 "fail"
*/
```

4.7.1.6 Palletizing program generation

Command name: generate_palletizing_program.

Command parameters:

```
/**
* @param  string palletizing_name name
* @param  string depalletizing_name Destacking name
* @param  string flag the palletizing or depalletizing program is generated, 0-not␣
↪generated, 1 generated
*/
```

Instruction example:

```
{
   cmd: "generate_palletizing_program",
   data: {
      palletizing_name: "palletizing_1",
      depalletizing_name:"depalletizing_1",
      flag:"[0,1]"
   }
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @return status:404 "fail"
*/
```

4.7.1.7 Get palletizing recipe

Command name: get_palletizing_formula.

Command parameters:

```
/**
* @param  string name palletizing recipe name
*/
```

Instruction example:

```
{
   cmd: "get_palletizing_formula",
   data: {
      name: "palletizing_1"
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Command feedback:

```
/**
* @return status:200
* @param  object box_config artifact configuration
* @param  object pallet_config pallet configuration
* @param  object device_config Installation device location
* @param  object pattern_config Mode configuration
* @param  object program_config Procedural build configuration
* @param  object lefttransitionpoint left transition point
* @param  object righttransitionpoint Right transition point Cartesian coordinates
* @param  object advanced_config advanced configuration
* @return status:404 "fail"
*/
```

Instruction feedback case:

```
{
   "box_config": {
     "flag": 1,
     "length": 200,
     "width": 400,
     "height": 300,
     "payload": 2.34,
     "grip_point": "grippoint"
   },
   "pallet_config": {
     "flag": 1,
     "front": 1000,
     "side": 1200,
     "height": 110,
      "left_pallet": 0,
      "right_pallet": 1
   },
   "device_config": {
   "flag": 1,
   "x": 2400,
   "y": 1800,
   "z": 120,
   "angle": 0
   },
   "pattern_config": {
   "flag": 1,
   "layers": 8,
   "box_gap": 0,
   "sequence": "a,b,a,b,a,b,a,b",
   "pattern_b_enable": 1,
   "left_pattern_a": "{\"1\": [[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.
↪3]]}",
```

(continues on next page)

(continued from previous page)

```
   "left_pattern_b": "{\"1\": [[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.
↪3]]}",
   "right_pattern_a": "{\"1\": [[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.
↪3]]}",
   "right_pattern_b": "{\"1\": [[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.3],[1,2,3,0.1,0.2,0.
↪3]]}",
   "origin_pattern_a": "[]",
   "origin_pattern_b": "[]"
   },
   "program_config": {
   "palletizing_name": "palletizing_1",
   "depalletizing_name":"depalletizing_1",
   "flag":"[0,1]"
   },
   "lefttransitionpoint":{
   "j1":"120",
   "j2":"120",
   "j3":"120",
   "j4":"120",
   "j5":"120",
   "j6":"120"
   },
   "righttransitionpoint":{
   "j1":"120",
   "j2":"120",
   "j3":"120",
   "j4":"120",
   "j5":"120",
   "j6":"120"
   },
   "advanced_config":{
   "height": "1000",
   "x1": "100",
   "y1": "100",
   "z1": "100",
   "x2": "10",
   "y2": "10",
   "z2": "10",
   "time": "1"
   }
}
```

4.7.1.8 Get the list of existing formula names for palletizing

Command name: get_palletizing_formula_list.

Command parameters: None

Instruction example:

```
{
   cmd: "get_palletizing_formula_list"
}
```

Command feedback:

```
/**
* @return status:200
* @param  Array ${name} palletizing name list
* @return status:404 "fail"
*/
```

Instruction feedback case:

```
["palletizing1"]
```

4.8 API description

4.8.1 act command

All the following act instructions use POST, and the URL is /action/act.

4.8.1.1 Save teaching points

Command name: save_point.

Command parameters:

```
/**
* @param  string name The name of the teaching point recorded by string name
* @param  string speed speed
* @param  string elbow_speed elbow speed
* @param  string acc acceleration
* @param  string elbow_acc elbow acceleration
* @param  string toolnum Tool number
* @param  string workpiecenum Workpiece number
*/
```

Instruction example:

```
{
    cmd: "save_point",
    data:{
        name: "point1",
```

(continues on next page)

(continued from previous page)

```
        speed: "100",
        elbow_speed: "100",
        acc: "100",
        elbow_acc: "100",
        toolnum: "1",
        workpiecenum: "1"
    }
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @return status:404 "fail"
*/
```

4.8.2 sta command

All the following sta instructions use POST, and the URL is /action/ sta .

4.8.2.1 Get robot status data

Command name: basic.

Command parameters: none.

Instruction example:

```
{
    cmd: "basic",
}
```

Command feedback:

```
/**
* @return status:200
* @param  object joints joint position
* @param  object tcp Descartes pose
* @param  array exAxisPos External axis position
* @return status:404 "fail"
*/
{
    joints: {
        j1: "90",
        j2: "90",
        j3: "90",
        j4: "90",
        j5: "90",
        j6: "90",
    },
    tcp: {
```

(continues on next page)

(continued from previous page)

```
        x: "100",
        x: "100",
        z: "100",
        rx: "90",
        ry: "90",
        rz: "90",
    },
    exAxisPos: [0,0,0,0]
}
```

4.8.3 get command

All the following get instructions use POST, and the URL is /action/get.

4.8.3.1 Get teaching points

Command name: get_points().

Command parameters: none.

Instruction example:

```
{
    cmd: "get_points"
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @param  ${point_name}: object teaching point related information
* @return status:404 "fail"
*/
```

Instruction feedback case:

```
{
    "localpoint1": {
        "name":"localpoint1",
        "elbow_speed":"1",
        "elbow_acc":"1",
        "x": "1",
        "y": "1",
        "z": "1",
        "rx": "1",
        "ry": "1",
        "rz": "1",
        "j1": "1",
        "j2": "1",
        "j3": "1",
        "j4": "1",
```

(continues on next page)

(continued from previous page)

```
        "j5": "1",
        "j6": "1",
        "toolnum": "1",
        "workpiecenum": "1",
        "speed": "1",
        "acc": "1",
        "E1": "1",
        "E2: "1",
        "E3": "1",
        "E4": "1"
    }
}
```

4.8.3.2 Get system configuration

Command name: get_syscfg().

Command parameters: none.

Instruction example:

```
{
    cmd: "get_syscfg"
}
```

Command feedback:

```
/**
* @return status:200 "success"
* @param  string log_count Maximum number of log days recorded
* @param  string language Currently using language pack
* @param  string lifespan overtime time
* * @return status:404 "fail"
*/
```

Instruction feedback case:

```
{
    log_count:"10",
    language:"zh",
    lifespan:"1800"
}
```

4.8.4 set command

All the following set instructions use POST, and the URL is /action/set.

4.8.4.1 Issue system variable instructions

Command name: 511.

Command parameters:

```
/**
* @param int index system variable serial number: 1-20
* @param int value system variable value
*/
```

Instruction example:

```
{
    cmd: 511,
    data:{
        content:"SetSysVarValue(2,1)"
    }
}
```

Command feedback:

```
/**
* @return status:200 1: represents success, 0: represents
* @return status:404 "fail"
*/
```

Instruction feedback case:

```
1
```

4.8.4.2 Get system variable instructions

Command name: 512.

Command parameters:

```
/**
* @param int index system variable serial number: 1-20
* /
```

Instruction example:

```
{
    cmd: 512,
    data:{
        content:"GetSysVarValue(2)"
    }
}
```

Command feedback:

```
/**
* @return status:200
* @param int value system variable value
* @return status:404 "fail"
* /
```

Instruction feedback case:

```
1
```

4.8.5 better-sqlite3 directive

4.8.5.1 Query the first row of records in the database

Command parameters:

```
/**
* @param string db_name Database name (including absolute path)
* @param string sql sql statement
* @return string result The first row of records queried
*/
```

Instruction content:

```
queryget(string db_name, string sql);
```

4.8.5.2 Query all records in the database

Command parameters:

```
/**
* @param string db_name Database name (including absolute path)
* @param string sql sql statement
* @return string result All records queried
*/
```

Instruction content:

```
queryall(string db_name, string sql);
```

4.8.5.3 Execute database statements

```
/**
* @param string db_name Database name (including absolute path)
* @param string sql sql statement
* @param object obj sql Parameters required for sql statement execution
* @return \
*/
```

Command parameters:

```
exec(string db_name, string sql, object obj);
```

Instruction content:

4.8.6 socket command

4.8.6.1 socket send

Command parameters:

```
/**
* @param string send_content socket Socket communication command sends content
* @return \
*/
```

Instruction content:

```
socket_cmd.send(string send_content);//8065
socket_file.send(string send_content);//8067
```

4.8.6.2 socket recv

Command parameters:

```
/**
* @return string recv_content socket Socket communication command reply content
*/
```

Instruction content:

```
socket_cmd.recv();//8065
socket_file.recv();//8067
```

4.8.7 File Operation Commands

4.8.7.1 Write File Content

```
/**
* @param String filename File path
* @param string content Content to write
* @param Function callback Callback function with (error) parameter (not needed for LA␣
↪version)
* @return true/false
*/

write(filename, content, callback);
```

4.8.7.2 Read File Content

```
/**
* @param String filename File path
* @param string content Content to write
* @param Function callback Callback function with (error) parameter (not needed for LA␣
↪version)
* @return String File content
*/

read(filename, callback);
```

4.8.7.3 Modify File Permissions

```
/**
* @param String filename File path
* @param Number mode Permission mode (e.g. 0644)
* @param Function callback Callback function with (error) parameter (not needed for LA␣
↪version)
* @return true/false
*/

chmod(filename, mode, callback);
```

4.8.7.4 Read Directory Contents (Including Subdirectories)

```
/**
* @param String path File path
* @param Function callback Callback function with (error) parameter (not needed for LA␣
↪version)
* @return Array Filename array
*/

readdir(path, callback);
```

4.8.8 Compression/Decompression Commands

Note: Distinguish between LA and QX versions (LA version doesn't need callback parameter).

LA module import: var execSync = require('child_process').execSync;

QX module import: var tar_utils = require('/usr/local/etc/node/sys/tools/tar_utils');

4.8.8.1 Create tar.gz Archive

Create tar.gz example (LA):

```
var cmd = 'cd / && tar -zcvf ' + FILENAME + '-C ' + DIR;
execSync(cmd);
```

Create tar.gz command description (QX):

```
/**
* @param {Array|String} sourcePaths Source file/directory path array or single path
* @param String targetFile Target archive path
* @param Function callback Callback function with (error) parameter (not needed for LA␣
↪version)
* @param String basePath Base path, defaults to '/'
* @return \
*/

createTarGz(sourcePaths, targetFile, callback, basePath);
```

4.8.8.2 Extract tar.gz File

Extract tar.gz example (LA):

```
var cmd = 'cd / && tar -zxvf ' + FILENAME;
execSync(cmd);
```

Extract tar.gz command description (QX):

```
/**
* @param String sourceFile Source archive path
* @param String targetDir Target extraction directory
* @param Function callback Callback function with (error) parameter (not needed for LA␣
↪version)
* @return \
*/
extractTarGz(sourceFile, targetDir, callback);
```

4.9 Version update instructions

Frcapsys-v1.0.0 update content:

- Initial Release;
- Frcapsys background environment and related interfaces;
- FrcapTools management tool.

4.10 Palletizing FRCap

4.10.1 Palletizing FRCap plug-in package management

In the collaborative robot WebApp, “System Settings - FRCap Config.” pageclick the “Import” button and select the palletizing FRCap plug-in package (name format: plug-in package name + version number. frcap, example: palletizer Palletizer-v0.0.0.plugin) to upload. After the upload is successful, the list displays the successfully imported palletized FRCap plug-in package, including Status, Name, Version, Description and Author, etc. The palletizing FRCap plug-in package can be “Deactivate”, “Enabled” and “Delete” in the operation bar.

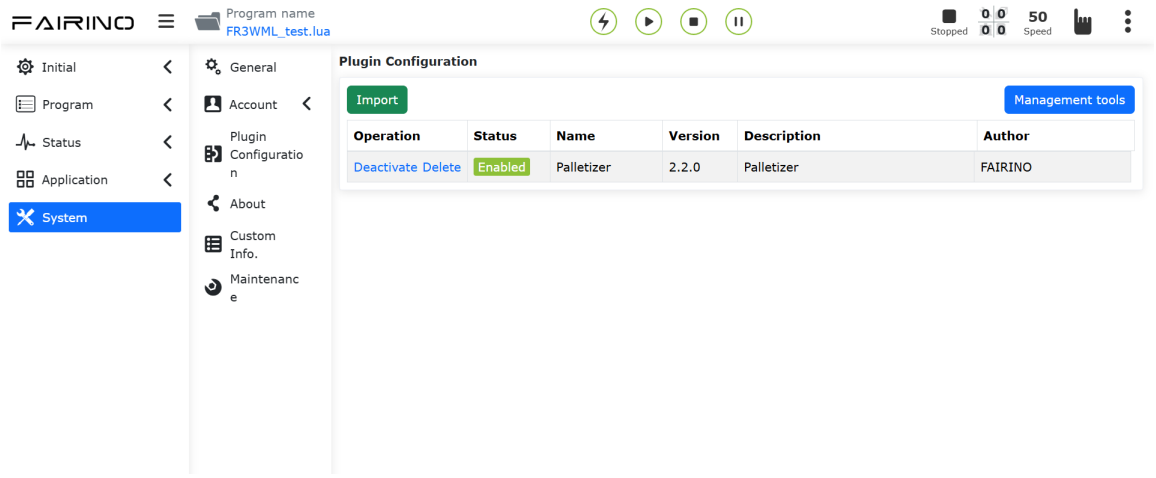


Figure 10-1-1 WebApp plug-in configuration interface

After successfully importing the palletizing FRCap plug-in package for the first time, the status of the plug-in package is “Disabled”. Click the “Enable” button. After successful activation, the “Auxiliary Application” module of the collaborative robot WebApp adds the palletizing FRCap plug-in package start page (For example: the page module name corresponding to Palletizer-v0.0.0.plugin is “Palletizer”). Click the “Start” button to enter the homepage, view the currently configured palletizing formula, and use it according to your needs.

Note: If the recipe is empty, please add/import the recipe first.

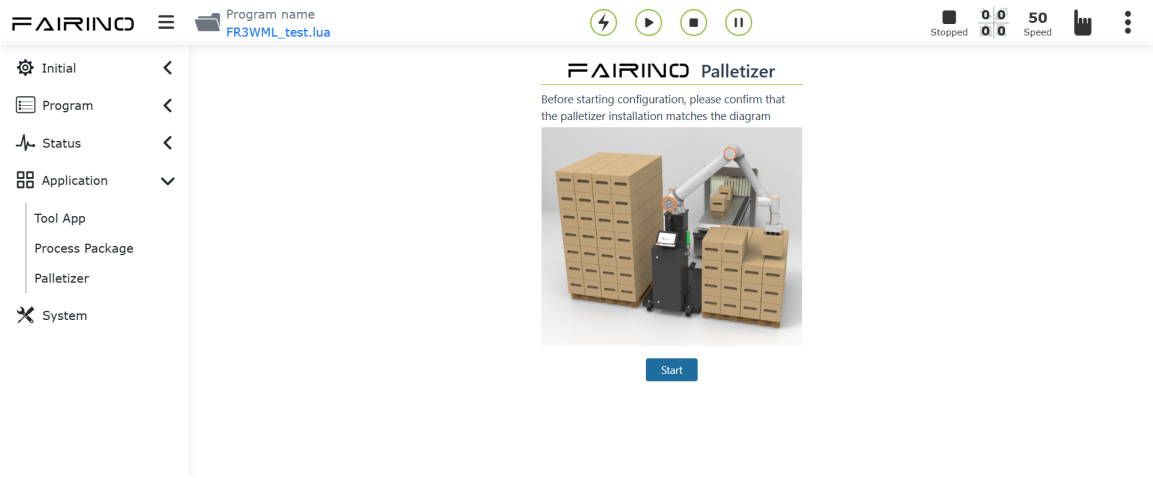


Figure 10-1-2 WebApp+ palletizing FRCap display diagram

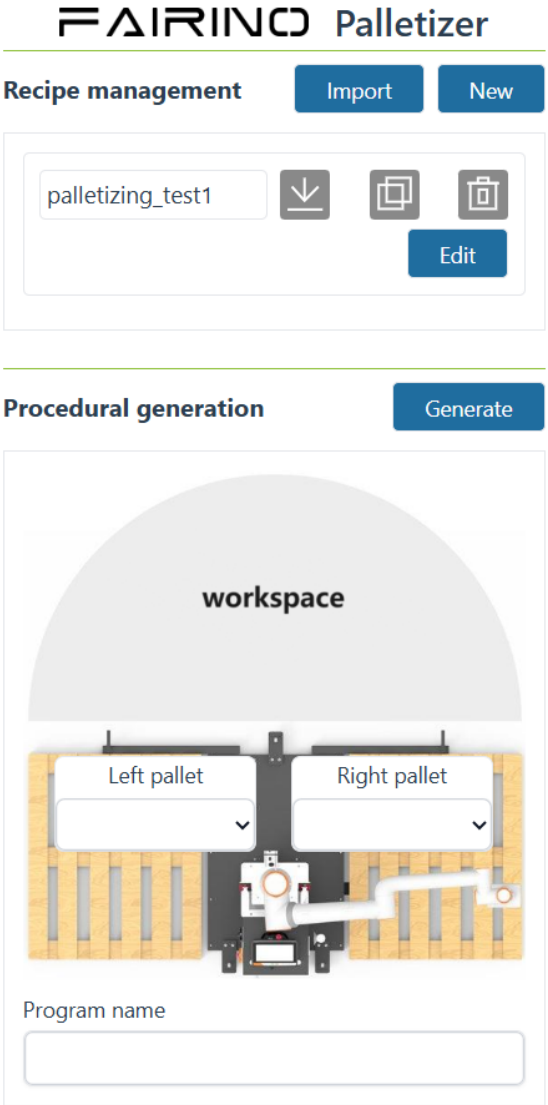


Figure 10-1-3 Palletizing FRCap Home Page

4.10.2 Recipe management

Each recipe is divided into three major areas: recipe name, recipe operation and recipe editing. The operation area buttons are: Rename, Export, Copy and Delete.



Figure 10-2-1 Recipe area division

Note: 

Name: Export recipe
Function: Export the data of the current recipe

Note: 

Name: Copy recipe
Function: Copy the data of the current recipe

Note: 

Name: Delete recipe
Function: Delete the current recipe

4.10.2.1 Obtain

After entering the homepage of the palletizing plug-in package, you can obtain all current recipes. When the number of recipes is greater than four, a scroll bar appears in the recipe display area, and the user can scroll up and down to view the recipes.

Note: All recipe names start with "palletizing", for example "palletizing_test1".

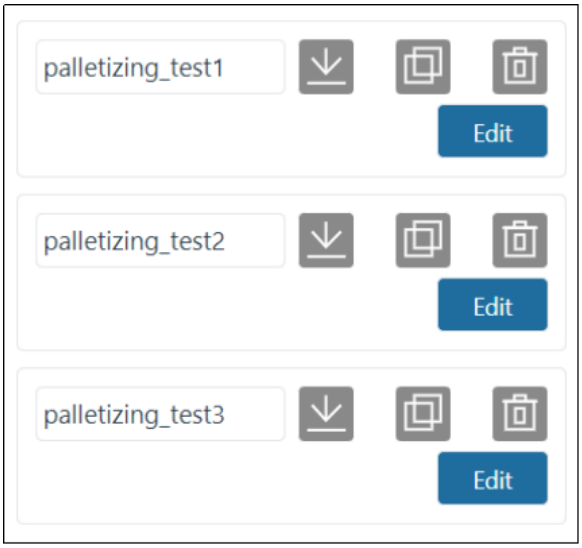
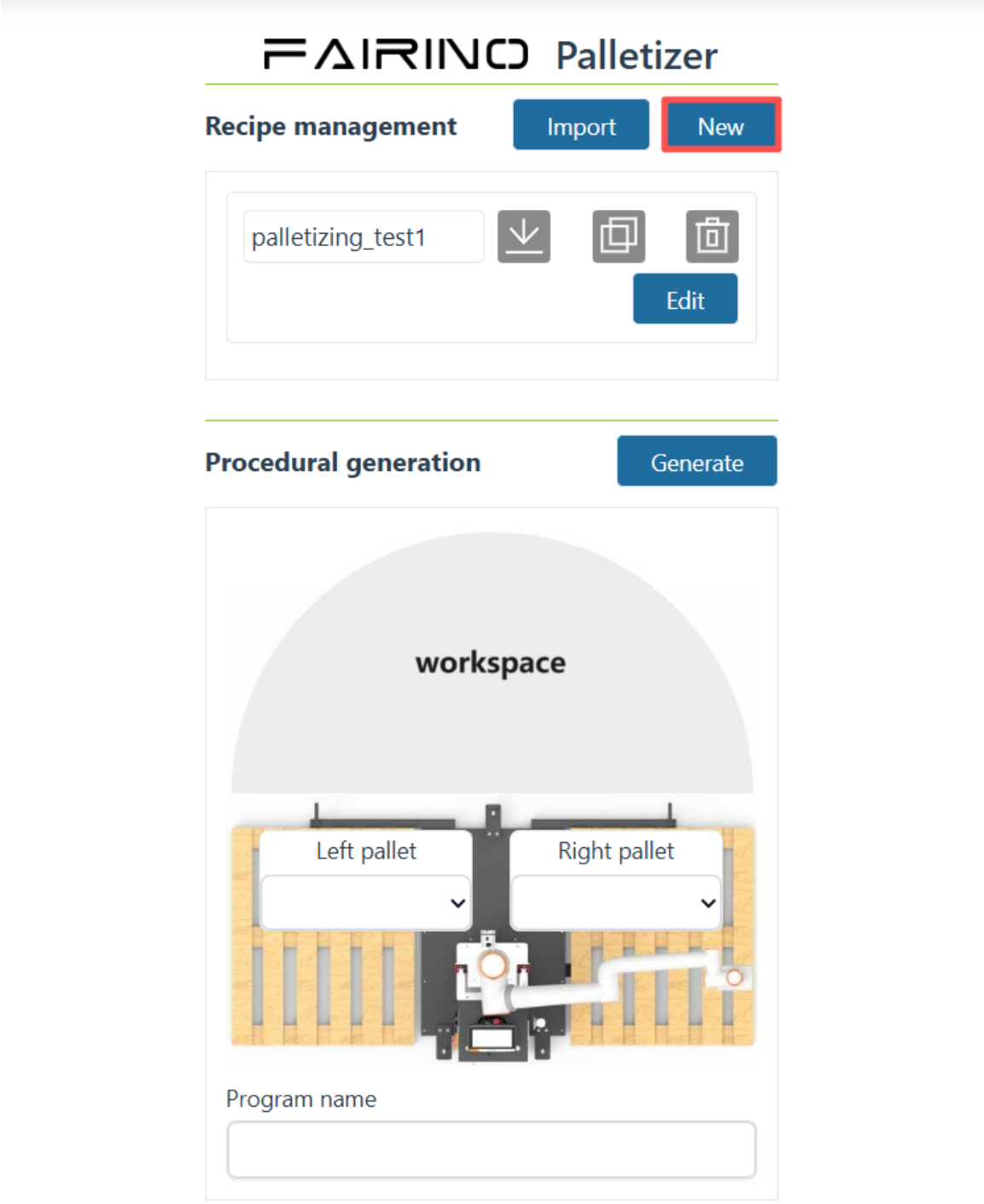


Figure 10-2-2 Recipe acquisition

4.10.2.2 New

In the operation area of any formula, click the "Add" button to enter the "New Formula" pop-up window, enter the name of the palletizing formula, and click the "Confirm" button. After the addition is successful, the new palletizing formula will be added to the formula display area.

Note: All recipe names start with "palletizing". There is no need to enter "palletizing". You only need to enter the name after "_". For example, "palletizing_add", enter "add".


FAIRINO Palletizer
Recipe management
Import
New
palletizing_test1
Edit
Procedural generation
Generate
workspace
Left pallet
Right pallet
Program name

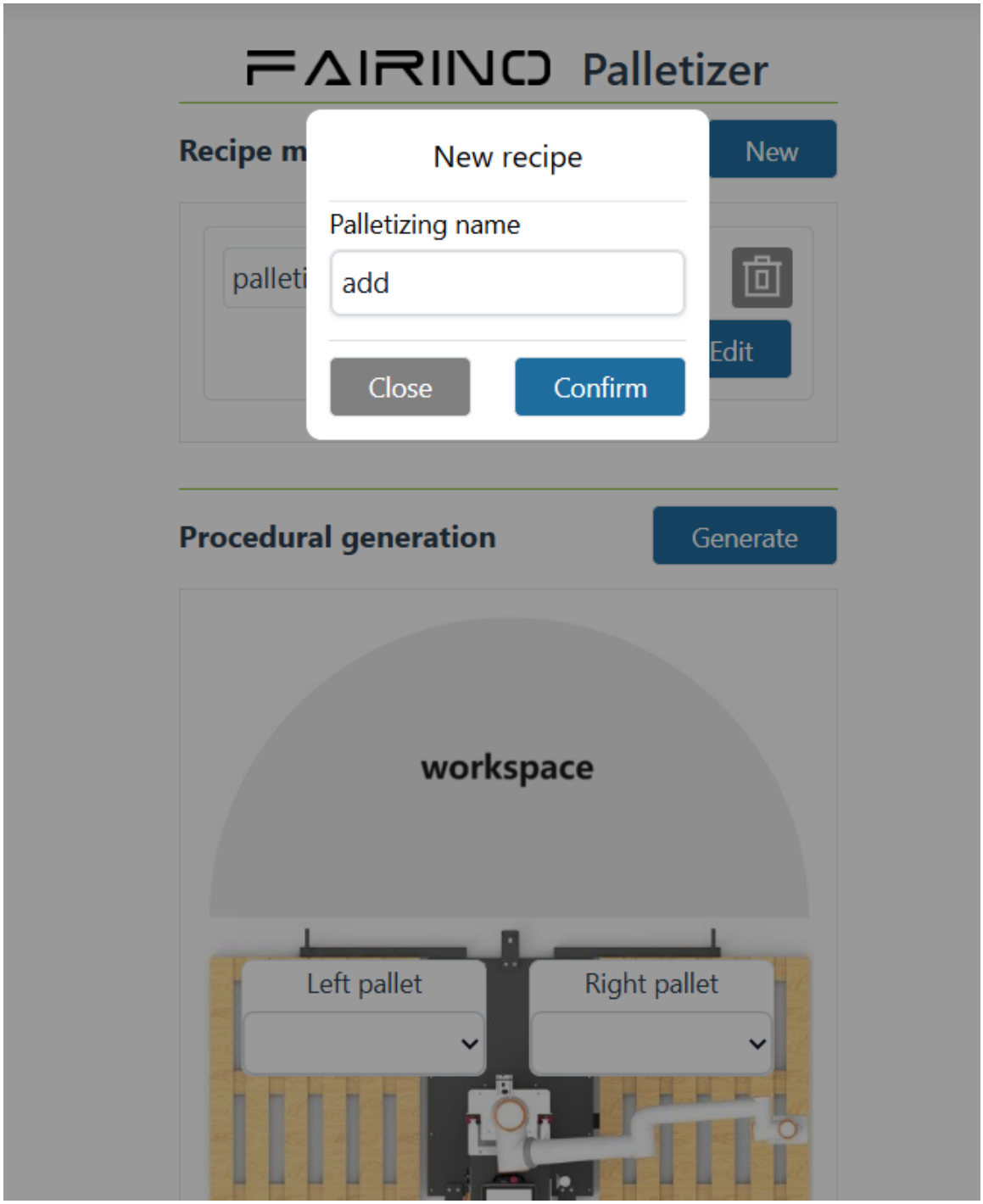


Figure 10-2-3 New recipe added

4.10.2.3 Rename

In the operation area of any formula, click the “Rename” icon to enter the “Palletizing Recipe Rename” pop-up window, enter the name of the palletizing formula, and click the “Confirm” button. After the renaming is successful, the original palletizing recipe name in the recipe display area is renamed.

Note: All recipe names start with “palletizing”. There is no need to enter “palletizing”. The modal window will automatically bring out the name after “_”. For example, “palletizing_rename” will automatically bring out “rename”.

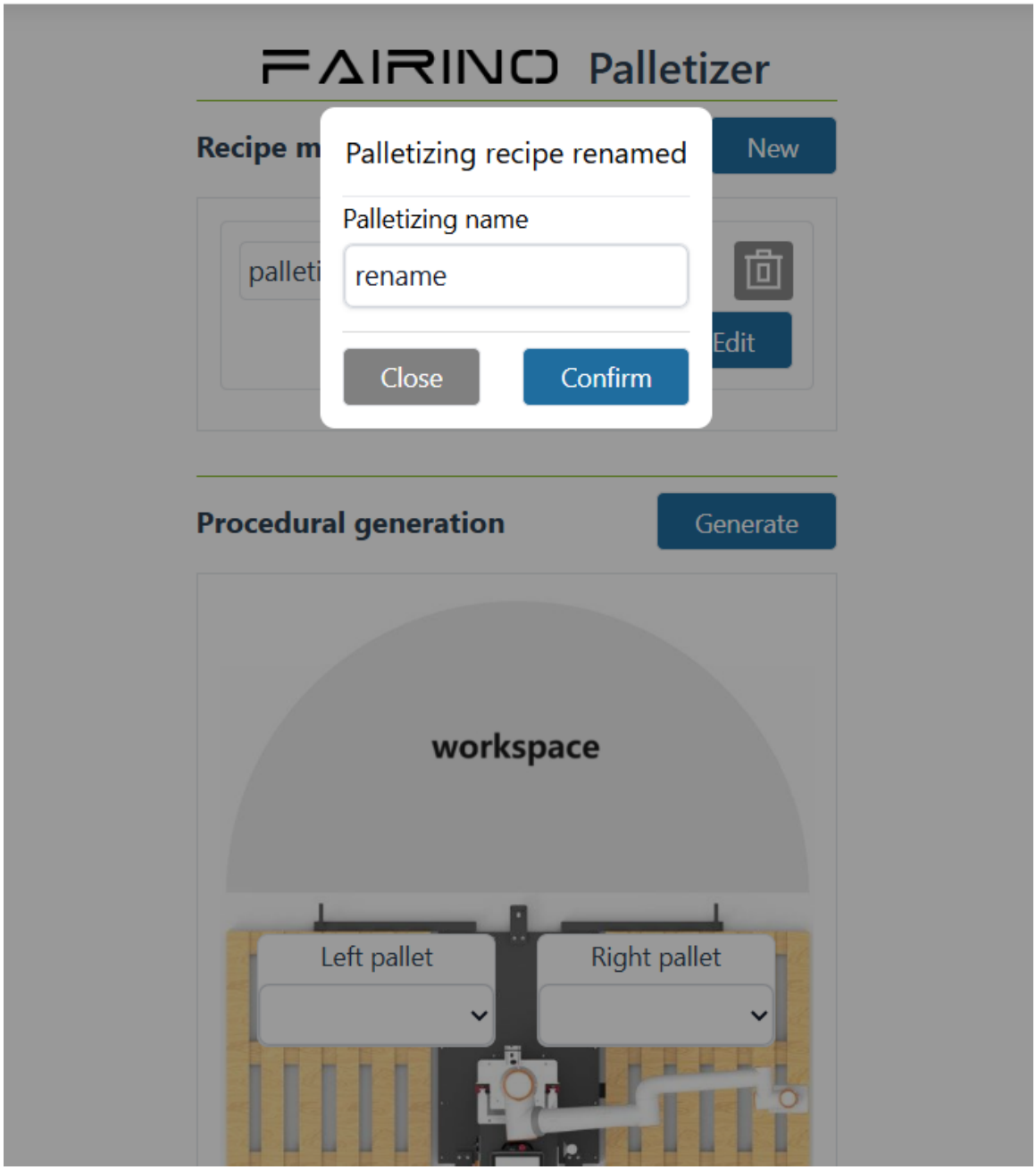


Figure 10-2-4 Recipe rename

4.10.2.4 Export

In the operation area of any recipe, click the “Export” icon to download all data of the current recipe.

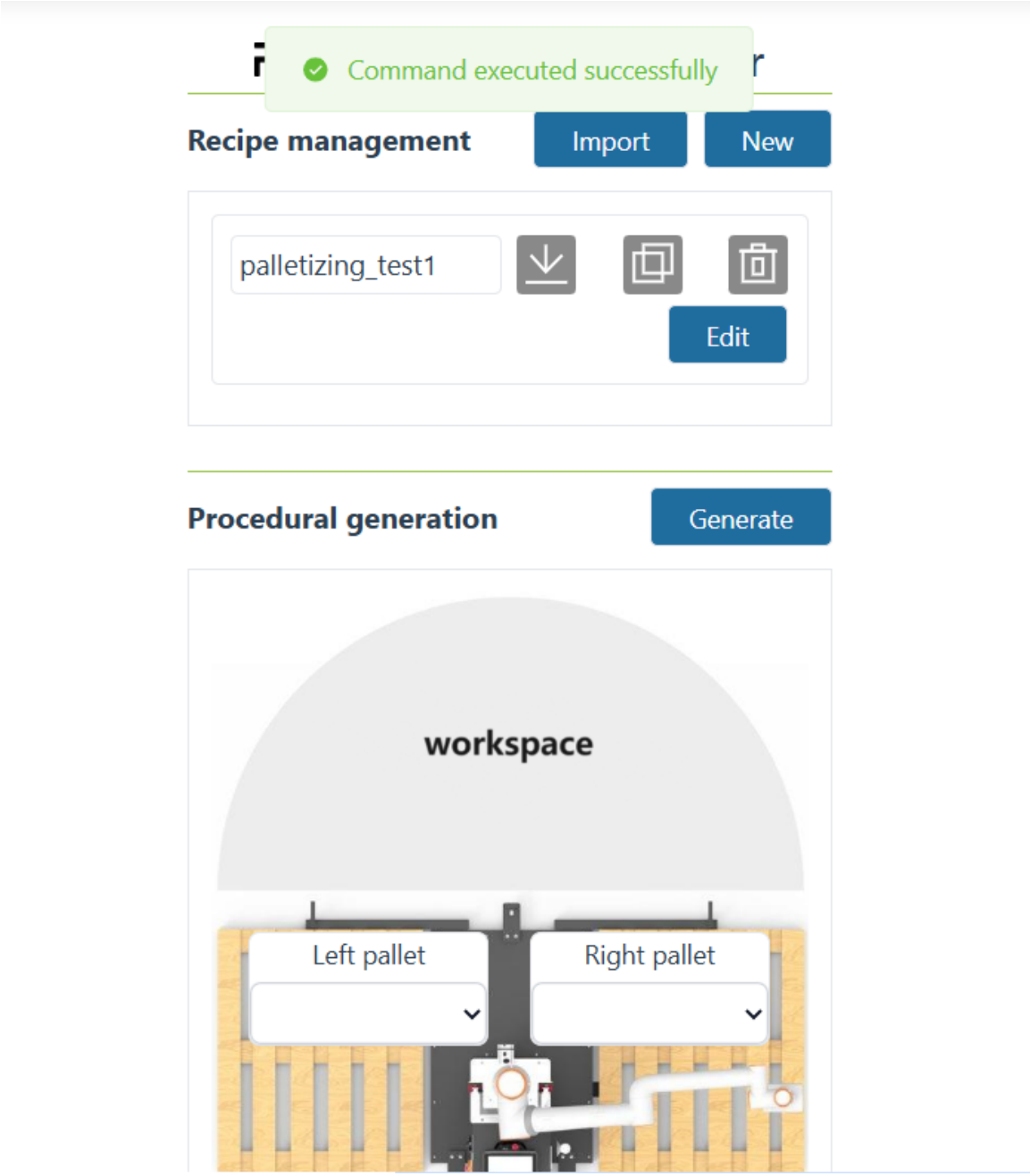


Figure 10-2-5 Recipe export

4.10.2.5 Copy

In the operation area of any formula, click the “Copy” icon to enter the “Palletizing Formula Copy” pop-up window, enter the name of the palletizing formula, and click the “Confirm” button. After the copy is successful, the copied palletizing formula will be added to the formula display area.

Note: All recipe names start with “palletizing”. There is no need to enter “palletizing”. The modal window will automatically bring out the names after “_”. For example, “palletizing_copy” will automatically bring out “copy”.

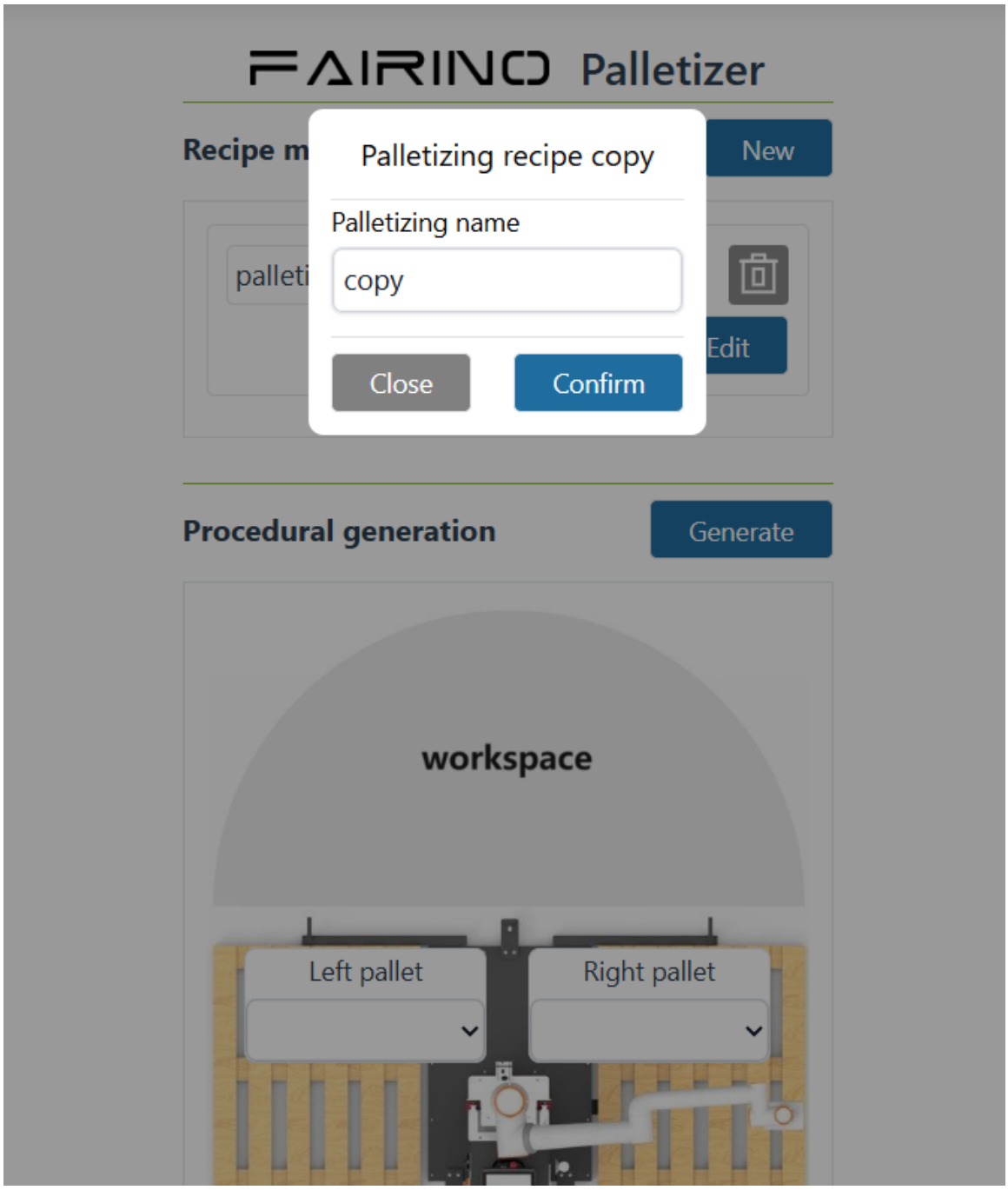
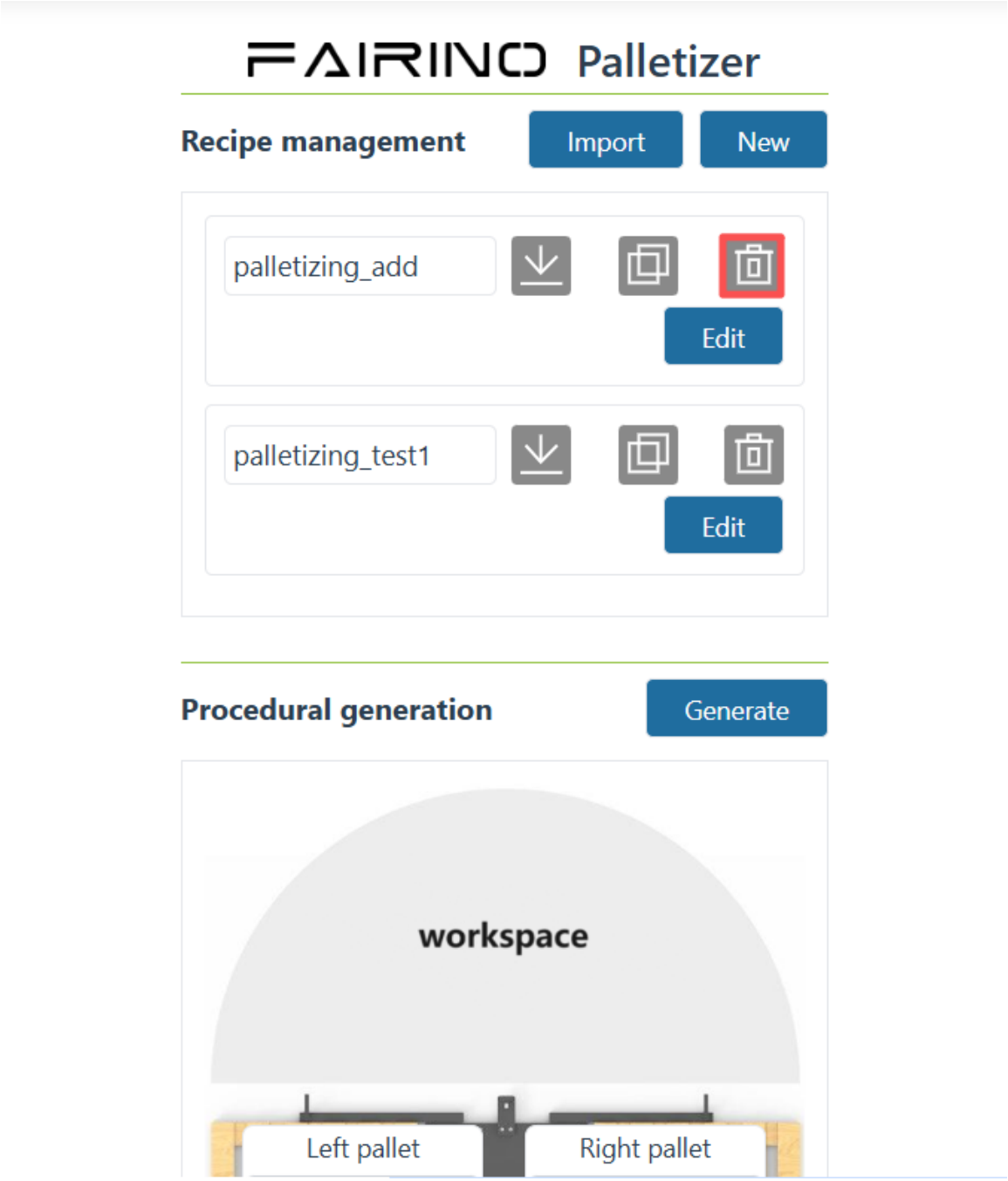


Figure 10-2-6 Recipe copy

4.10.2.6 Delete

In the operation area of any recipe, click the “Delete” icon to delete the current recipe.



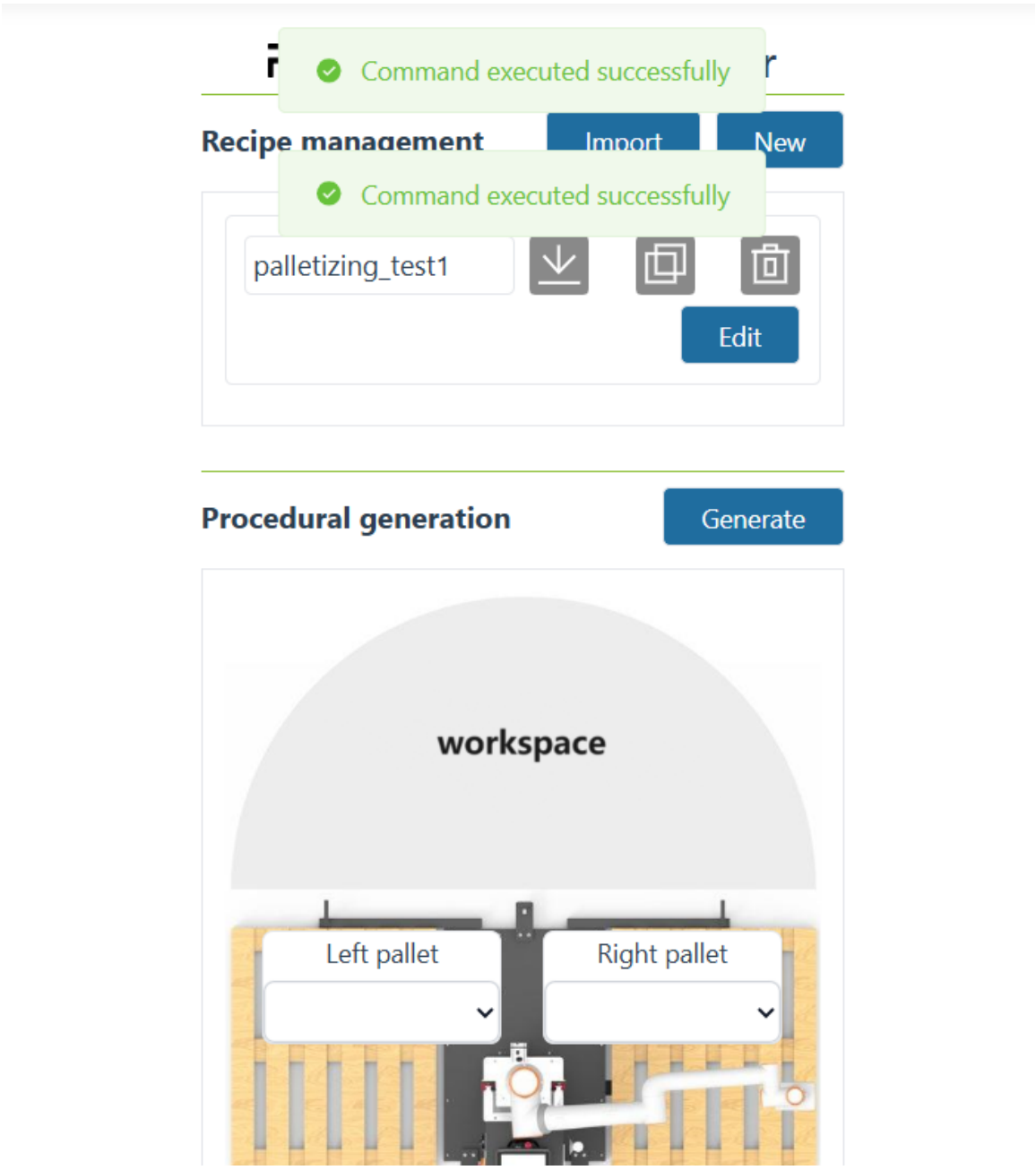


Figure 10-2-7 Recipe delete

4.10.2.7 Edit

For any recipe, click the “Edit” button to enter the configuration interface of the current recipe.

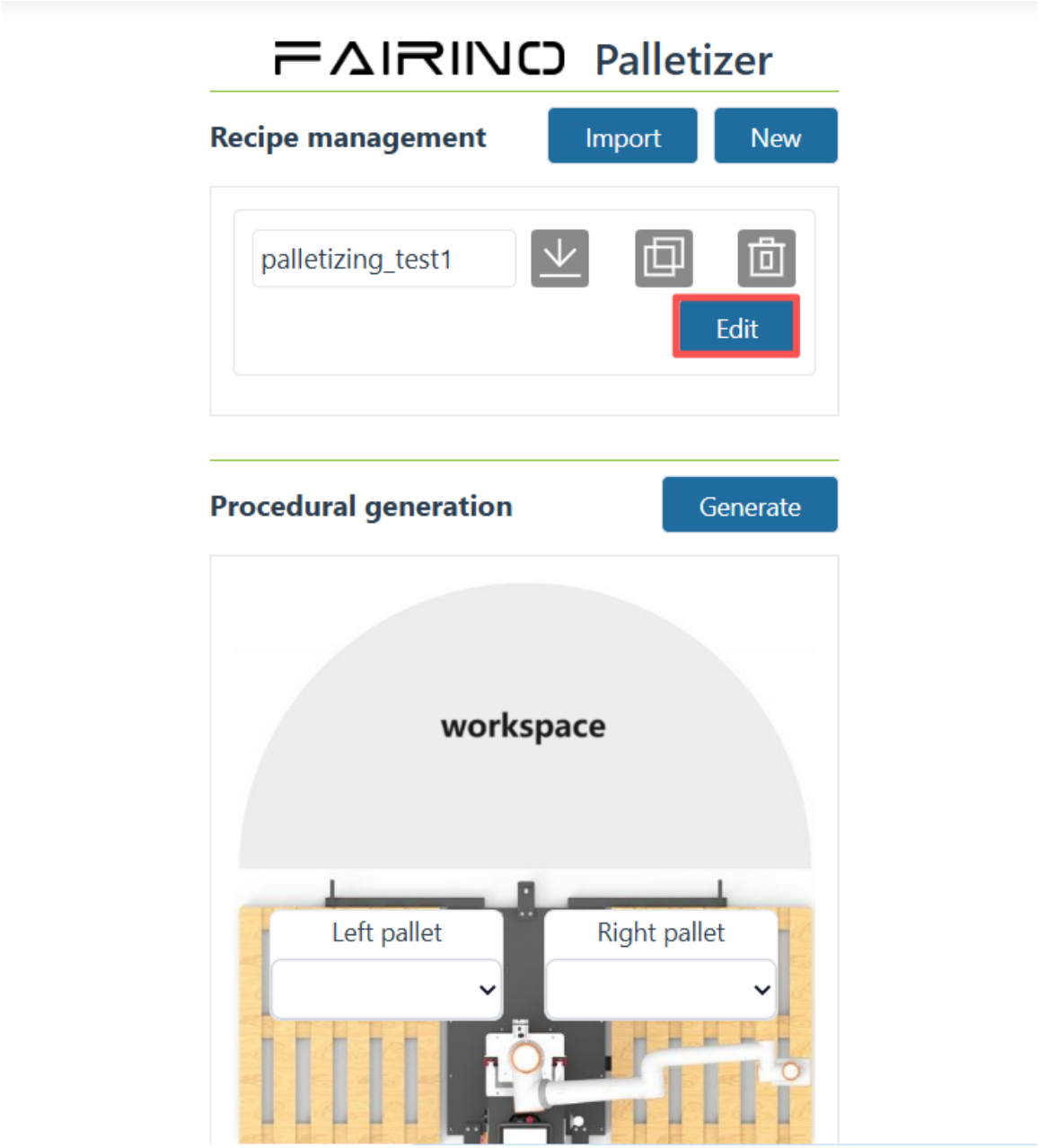


Figure 10-2-8 Palletizing recipe editor

4.10.2.8 Import

Click the “Import” button, select the compressed package of the palletizing formula and upload it. After the import is successful, the imported formula will be added to the palletizing formula.

Note: All recipe compressed package names start with “palletizing” and end with “.tar.gz”, such as “palletizing_import.tar.gz”.

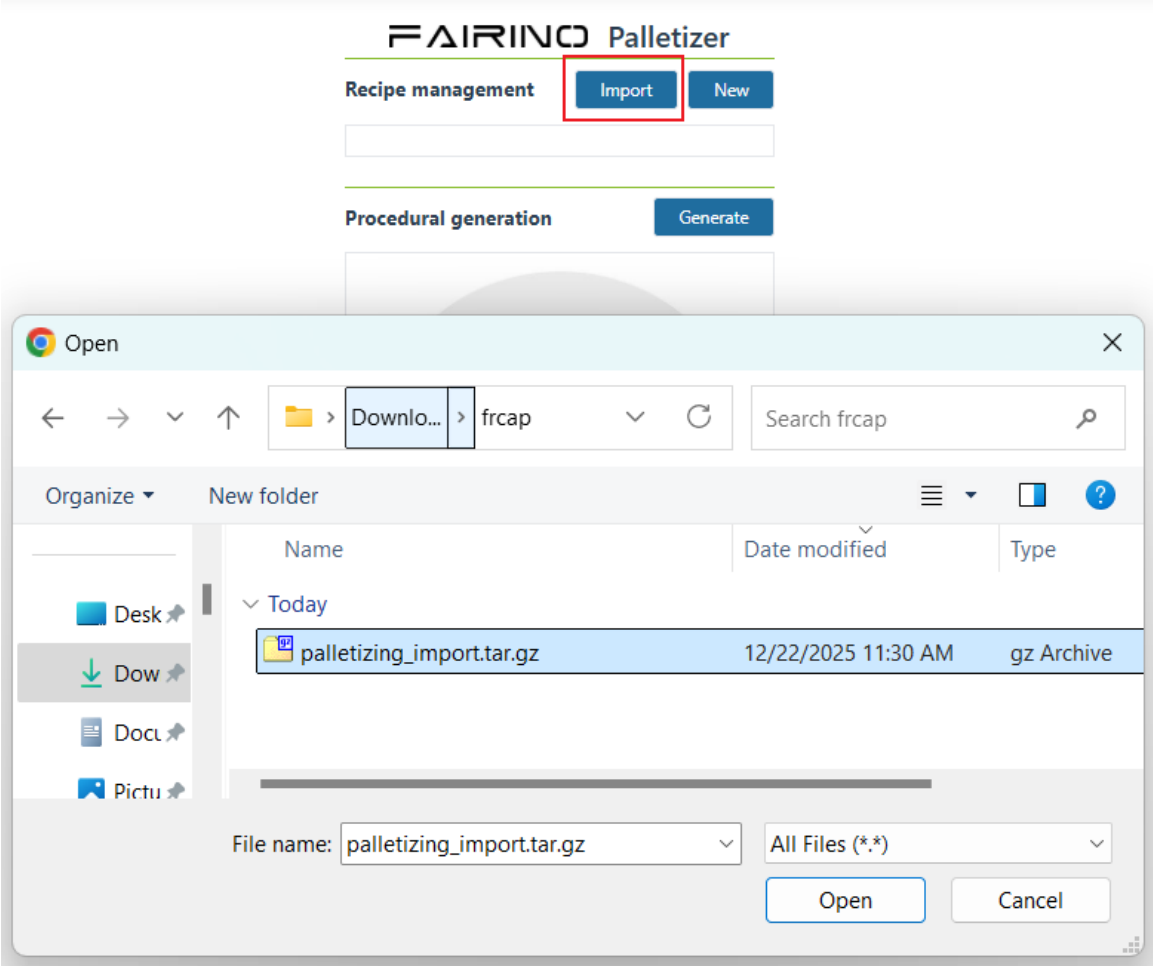


Figure 10-2-9 Recipe import

Important: When “adding”, “renaming” and “copying” a palletizing formula, entering an existing formula name will prompt “There is already a formula with the same name”.

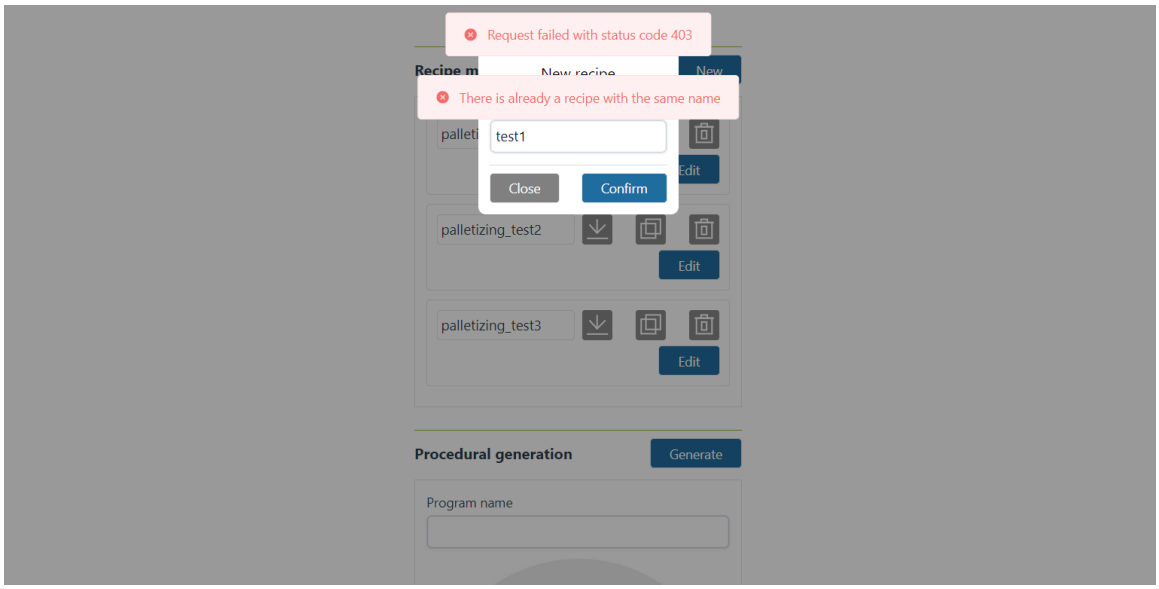


Figure 10-2-10 Tips for recipes with the same name

4.10.3 Recipe configuration

The configuration interface of any recipe displays the basic information of boxes, trays, modes and advanced configurations, and configures specific parameters in the corresponding configuration column.

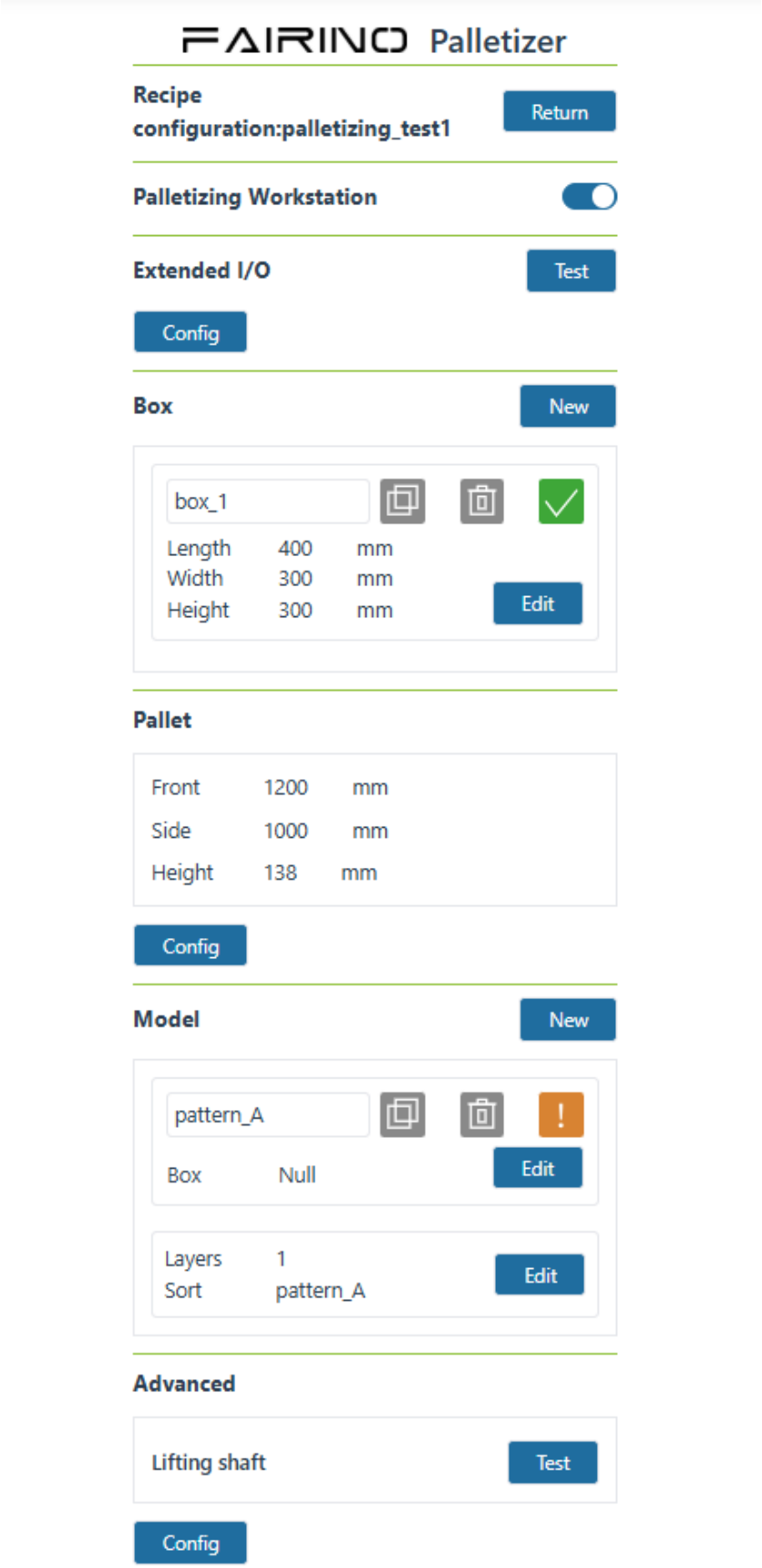

FAIRINO Palletizer
Recipe configuration:palletizing_test1
Return
Palletizing Workstation
Extended I/O
Test
Config
Box
New
box_1
Length 400 mm
Width 300 mm
Height 300 mm
Edit
Pallet
Front 1200 mm
Side 1000 mm
Height 138 mm
Config
Model
New
pattern_A
Box Null
Edit
Layers 1
Sort pattern_A
Edit
Advanced
Lifting shaft
Test
Config

Figure 10-3-1 Palletizing recipe editing interface

4.10.3.1 Workstation Settings

In recipe editing, you can choose whether to use a palletizing workstation, as shown in Figure 2-2. If a palletizing workstation is used, the corresponding palletizing functions will be completed using I/O signals from the workstation PLC; if no palletizing workstation is selected, the palletizing functions will default to using I/O signals from the control box.

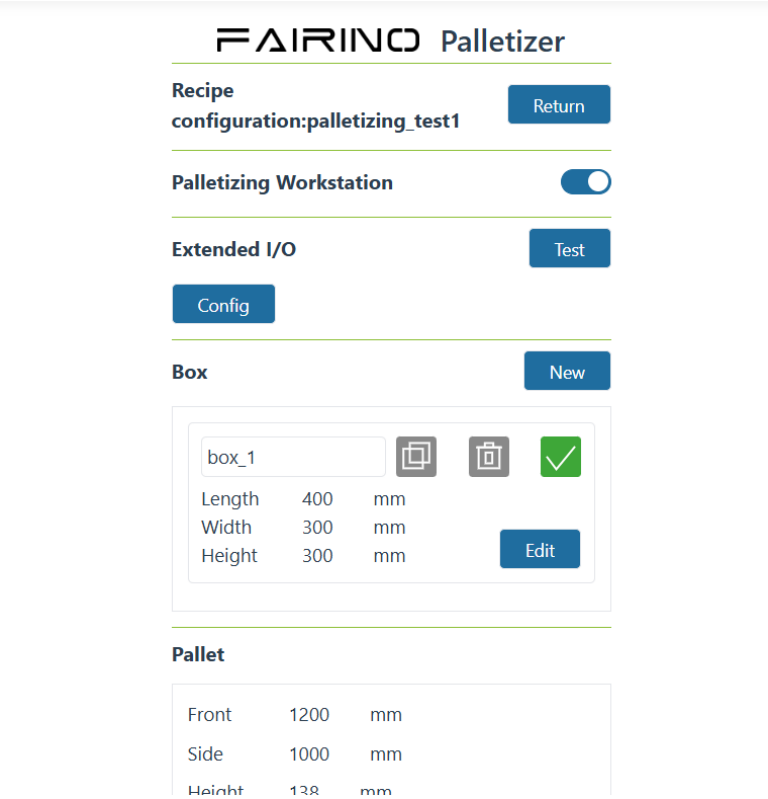


Figure 10-3-1-1 Recipe Editing Page

4.10.3.2 Palletizing Function I/O Wiring Configuration

(1) After selecting to use a palletizing workstation, click on Extended I/O Configuration. Based on the corresponding functions and the actual wiring of the I/O interfaces with the PLC, you can customize the selection of I/O signal configuration for palletizing functions. Figure shows the default wiring configuration for the palletizing workstation.

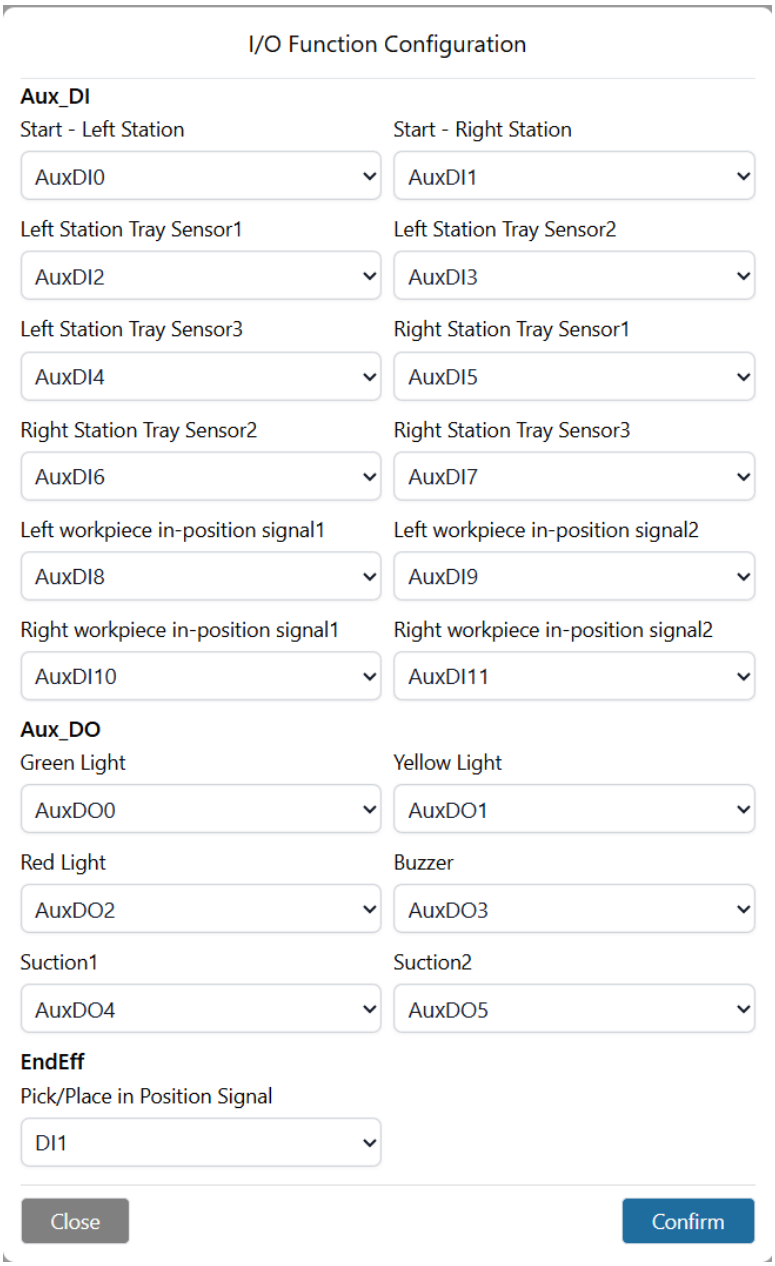


Figure 10-3-1-2 Default Wiring Configuration for Palletizing Workstation

(2) If you choose not to use a palletizing workstation, the system defaults to using control box I/O signals. Based on the corresponding functions and the actual wiring of the I/O interfaces with the control box, you can customize the selection of I/O signal configuration for palletizing functions. Figure shows the default wiring configuration when no palletizing workstation is used (using control box I/O).

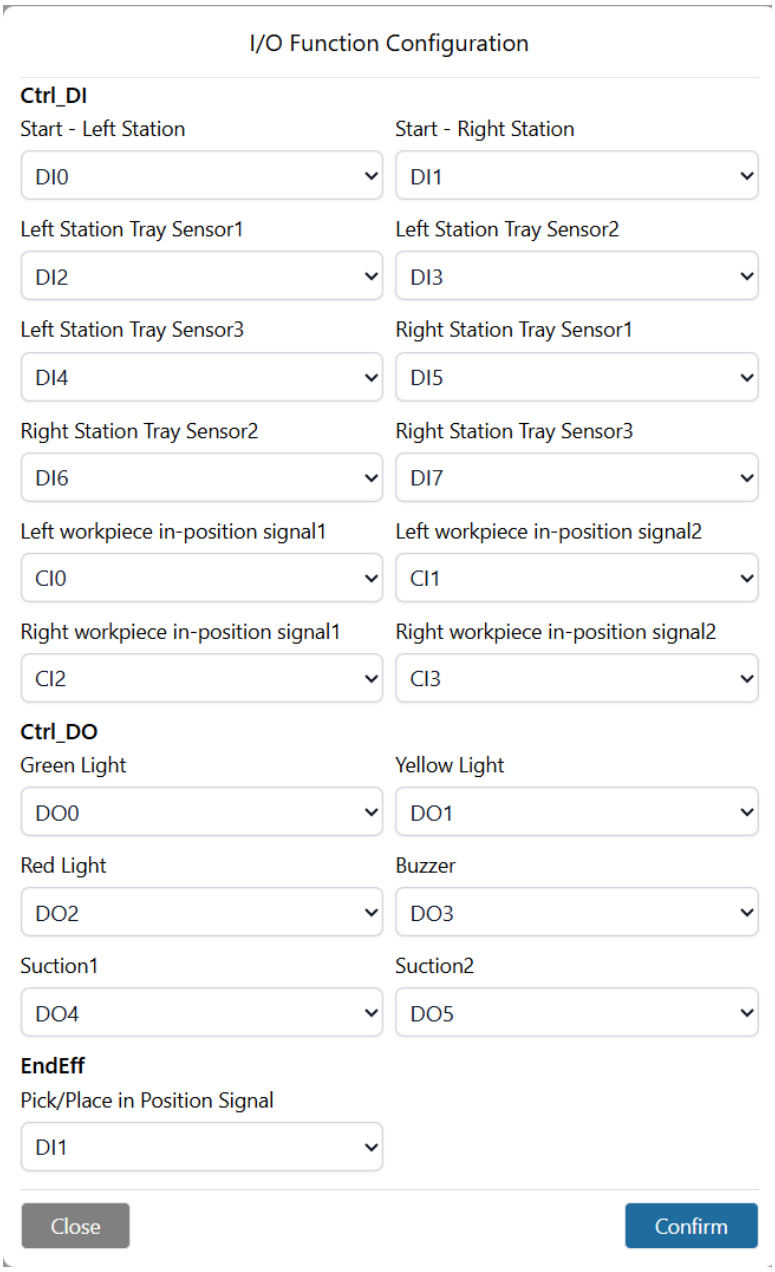


Figure 10-3-1-3 Default Wiring Configuration for No Palletizing Workstation (Control Box I/O)

4.10.3.3 Palletizing Function I/O Communication Test

(1) When using a palletizing workstation, after configuring the extended I/O signals for the workstation palletizing workstation, you can click “Test” to verify the functionality of the wired I/O;

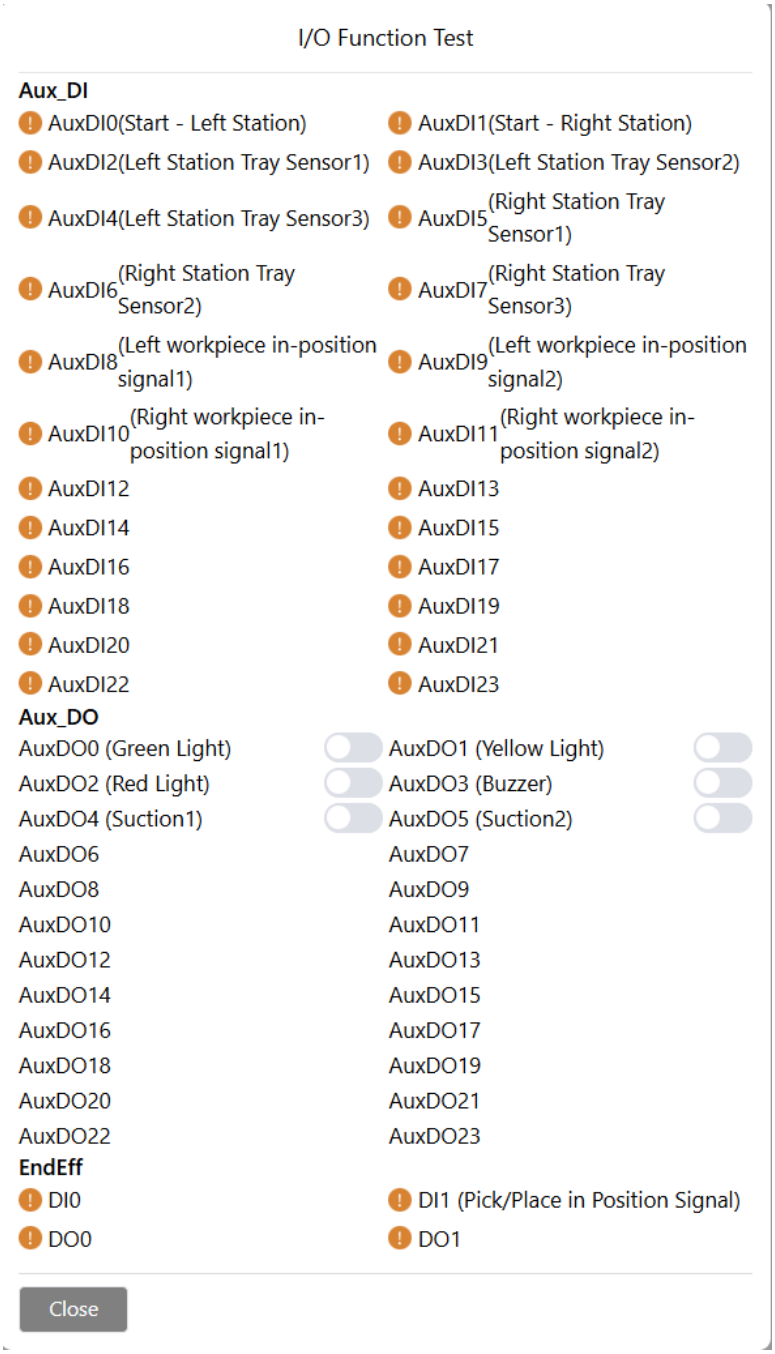


Figure 10-3-1-4 Palletizing Workstation I/O Wiring Test

(2) When no palletizing workstation is selected, after configuring the control box I/O signals corresponding to the palletizing functions, you can click “Test” to verify the functionality of the wired I/O;

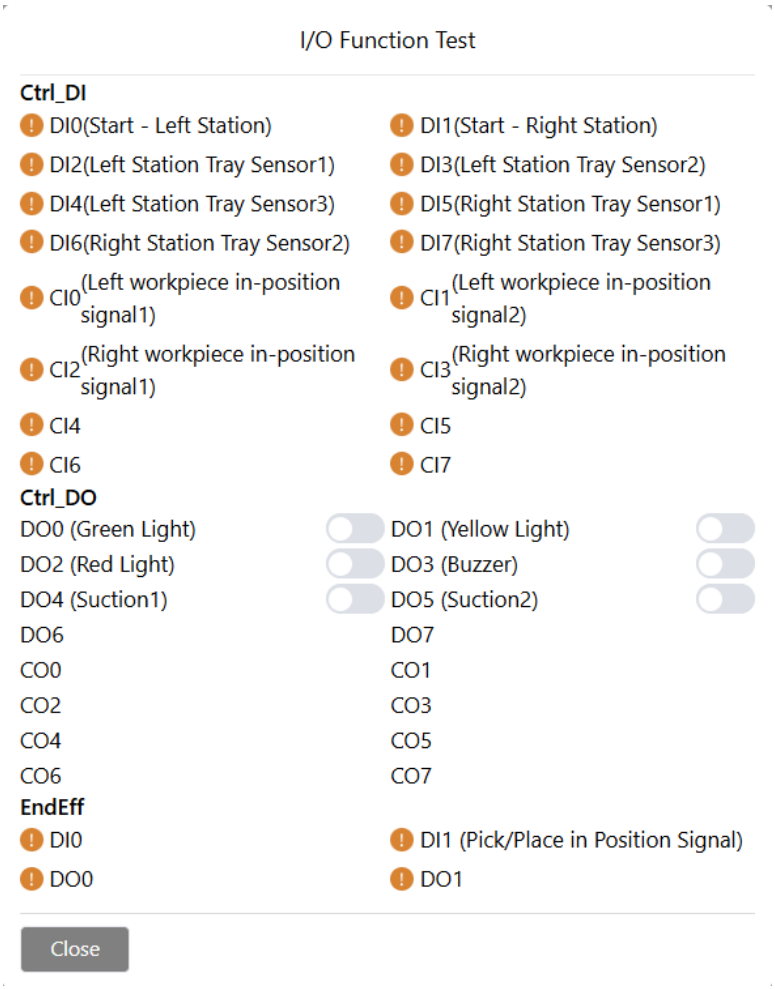
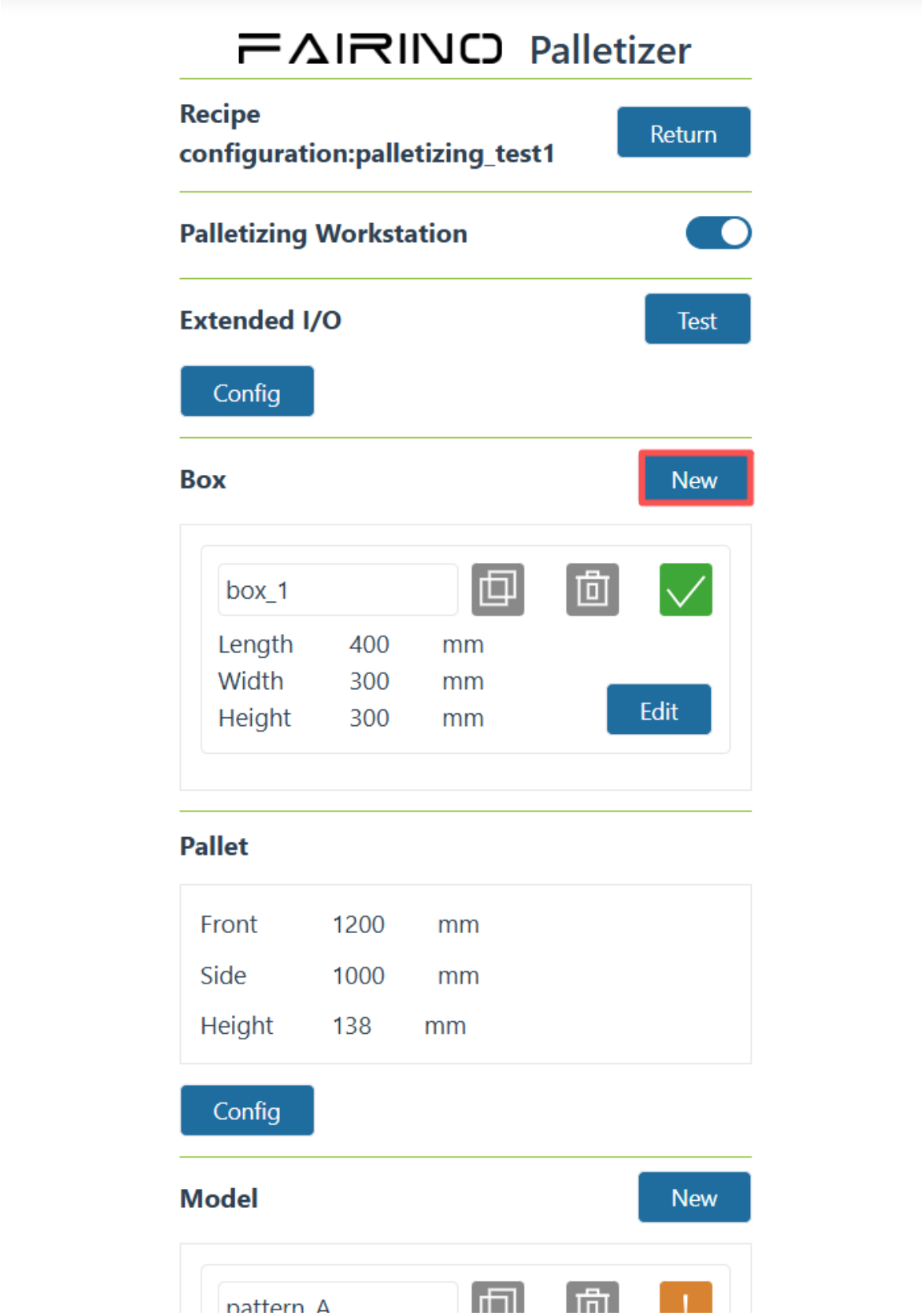


Figure 10-3-1-5 No Palletizing Workstation (Control Box I/O) Wiring Test

4.10.3.4 Box operation

Boxes can be configured with multiple boxes of different types.

Click the “Add” button. After the addition is successful, a new box will be added in the current order.


FAIRINO Palletizer
Recipe configuration:palletizing_test1
Return
Palletizing Workstation
Extended I/O
Test
Config
Box
New
box_1
Length 400 mm
Width 300 mm
Height 300 mm
Edit
Pallet
Front 1200 mm
Side 1000 mm
Height 138 mm
Config
Model
New

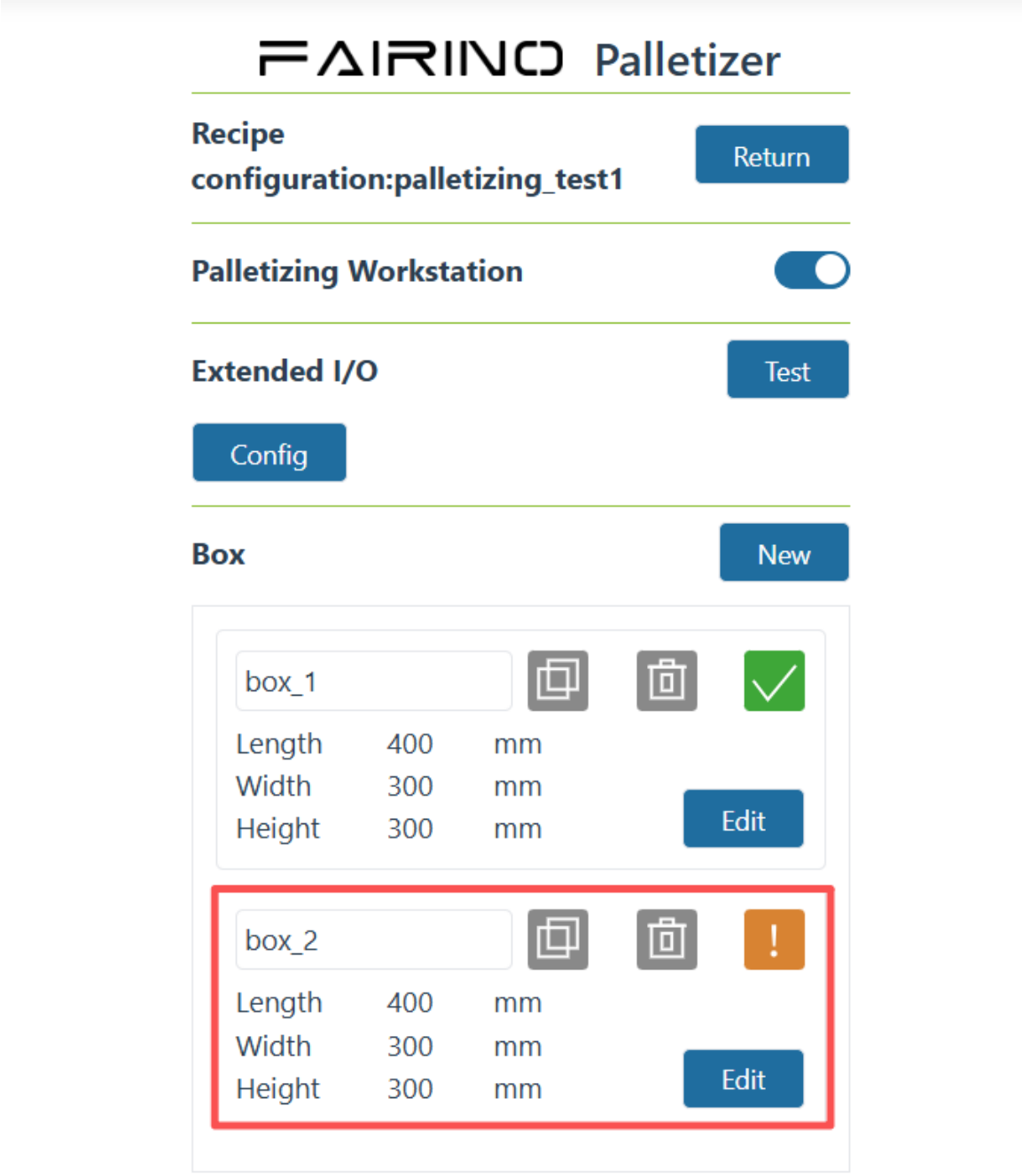


Figure 10-3-2 Add new box

Click the input box area where the box name is displayed, and the “Box Rename” modal window will pop up. After entering the name, click the “Confirm” button to confirm the renaming.

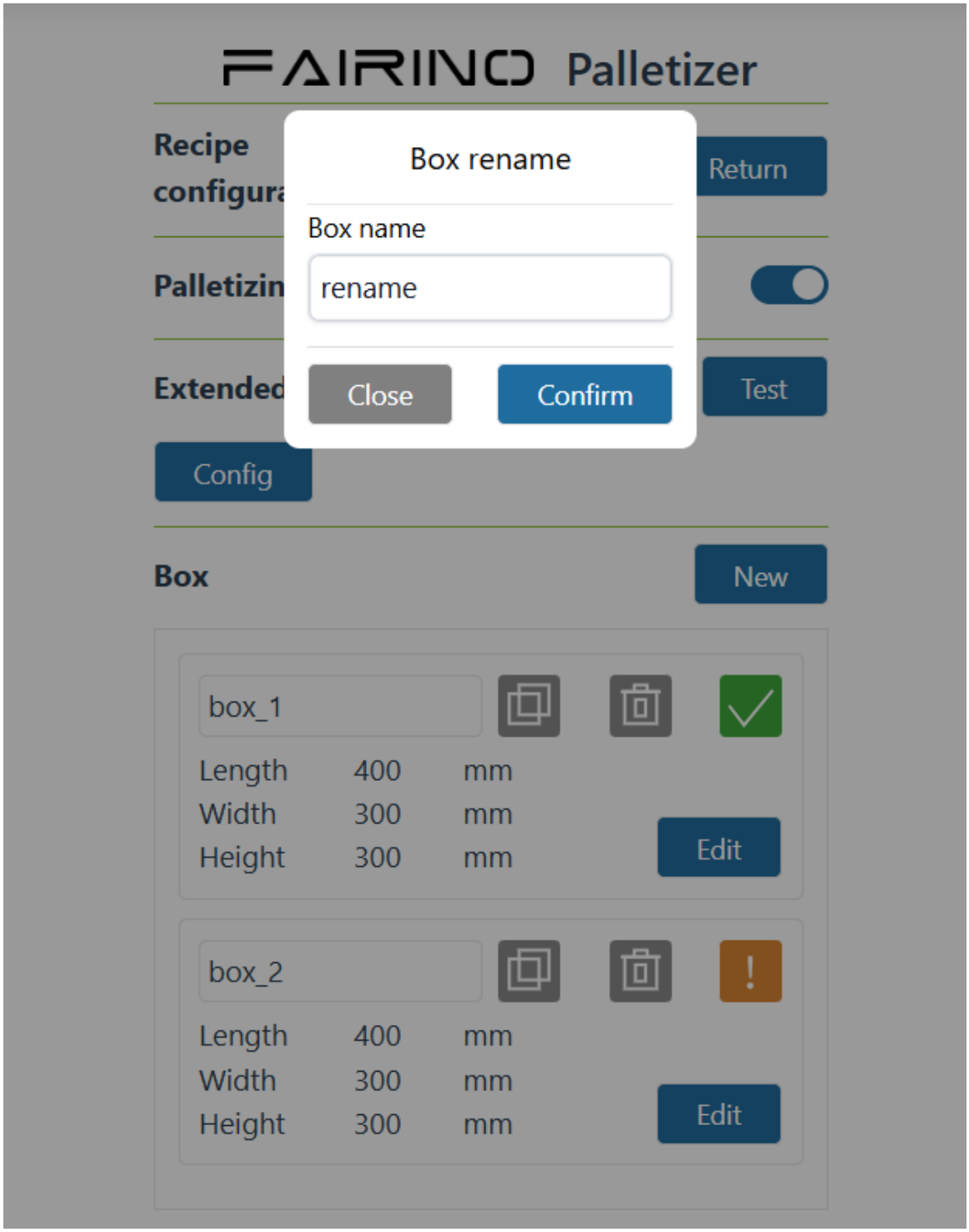
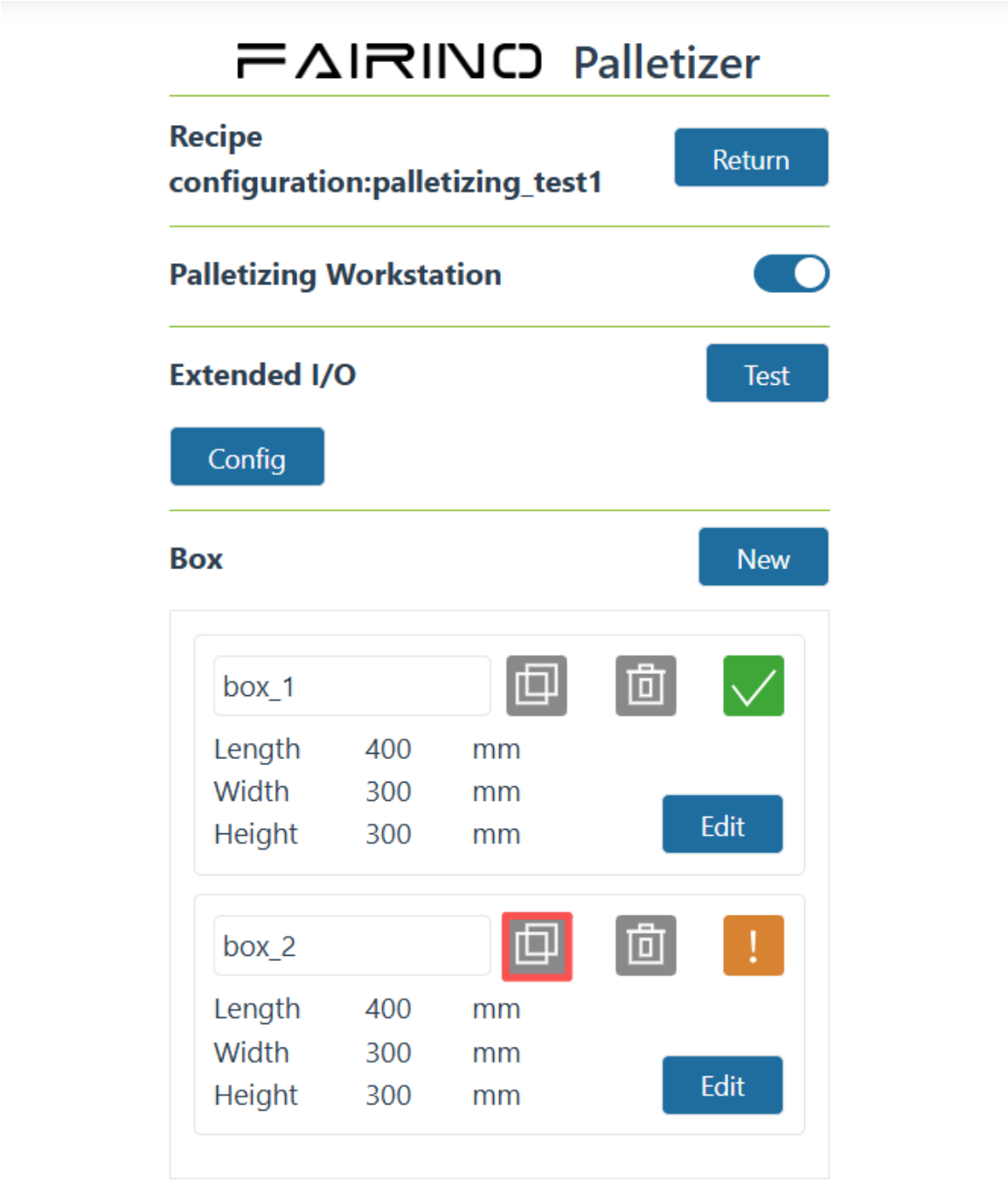


Figure 10-3-3 Rename box

Click the “Copy” icon. After the copy is successful, copy a box based on the current box name.


FAIRINO Palletizer
Recipe configuration:palletizing_test1
Return
Palletizing Workstation
Extended I/O
Test
Config
Box
New
box_1
Length 400 mm
Width 300 mm
Height 300 mm
Edit
box_2
Length 400 mm
Width 300 mm
Height 300 mm
Edit

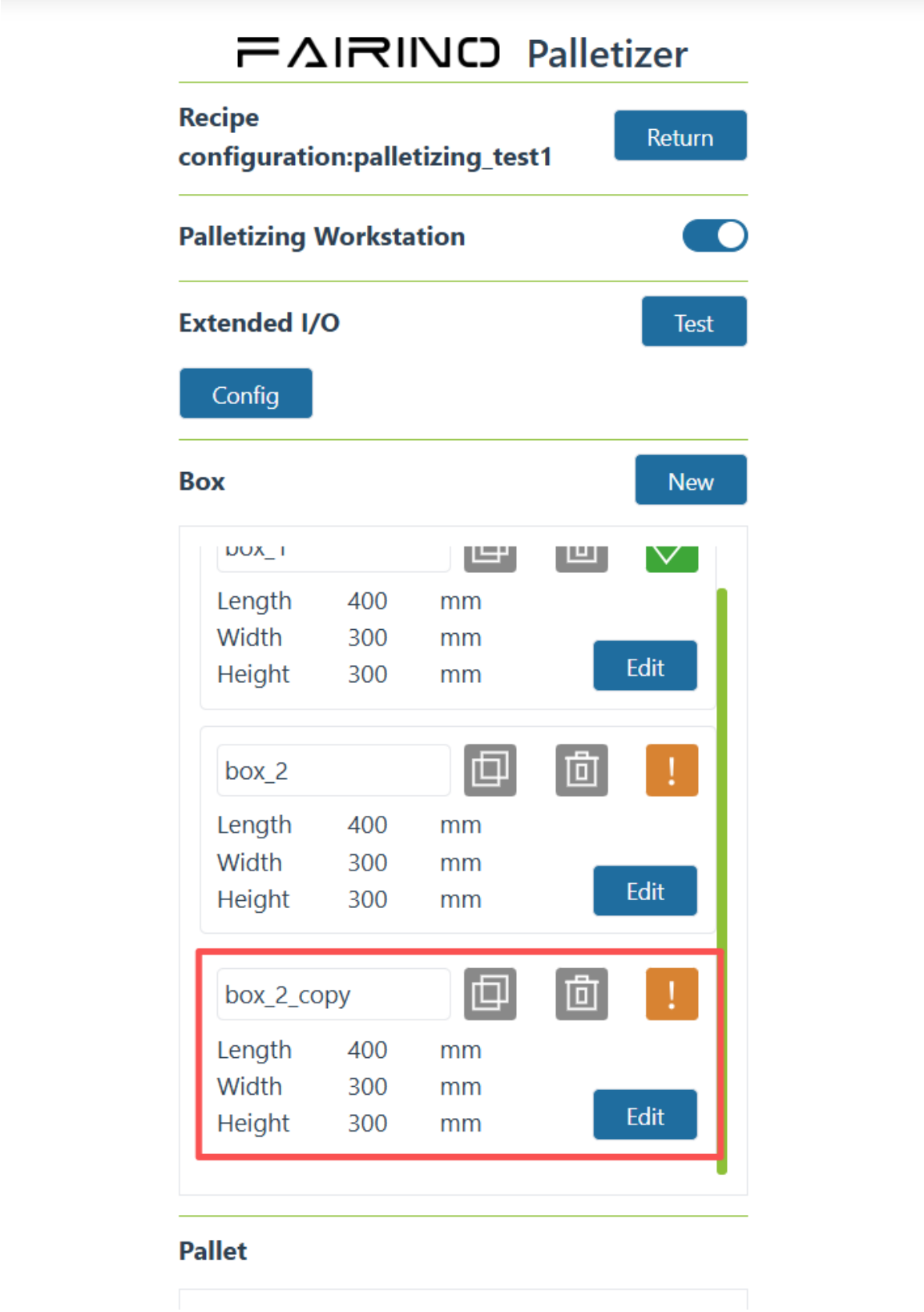
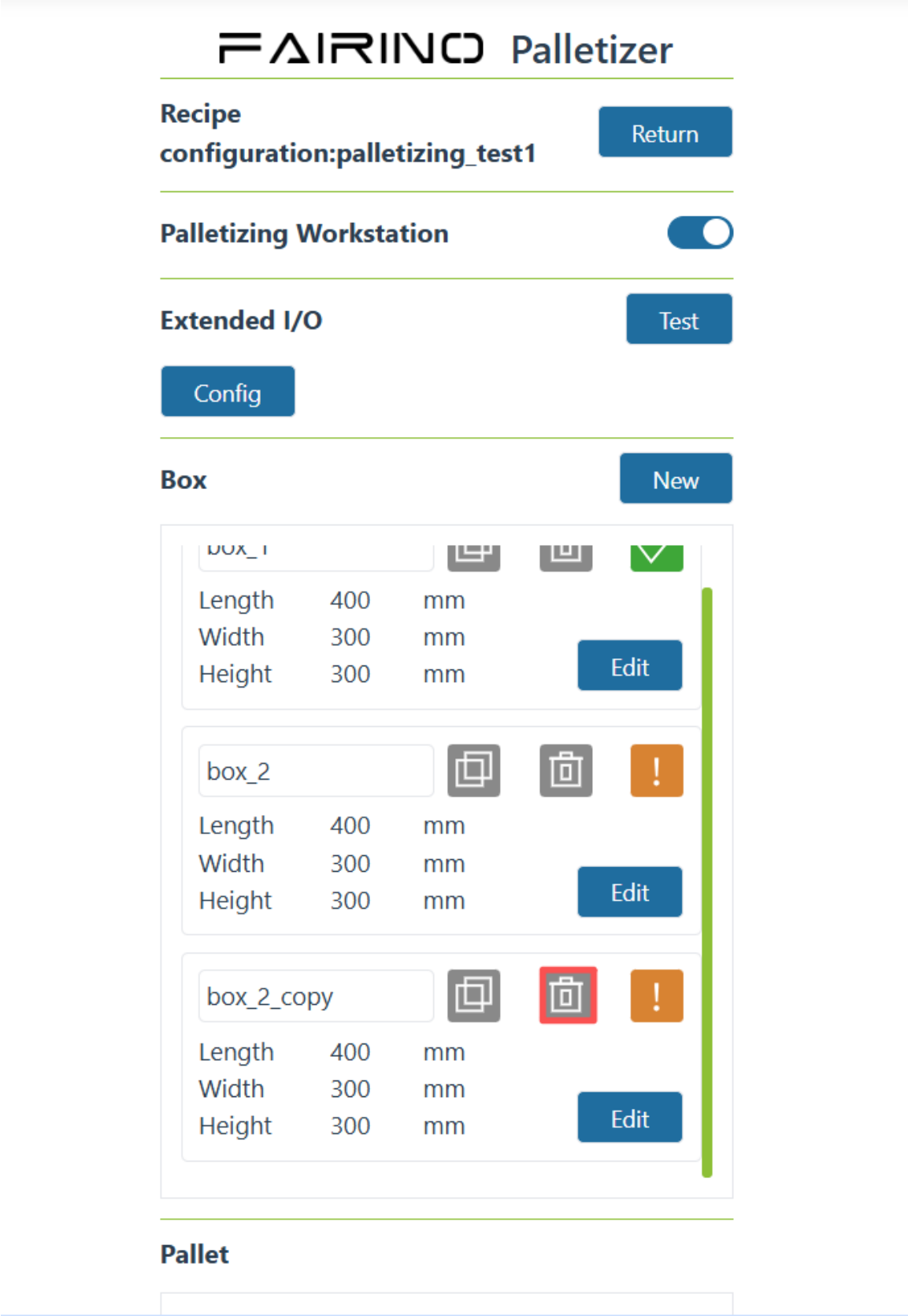


Figure 10-3-4 Copy box

Click the “Delete” icon to delete the box data.

Note: Do not delete boxes that have been configured in the mode configuration.


FAIRINO Palletizer
Recipe configuration:palletizing_test1
Return
Palletizing Workstation
Extended I/O
Test
Config
Box
New
Length 400 mm
Width 300 mm
Height 300 mm
Edit
box_2
Length 400 mm
Width 300 mm
Height 300 mm
Edit
box_2_copy
Length 400 mm
Width 300 mm
Height 300 mm
Edit
Pallet

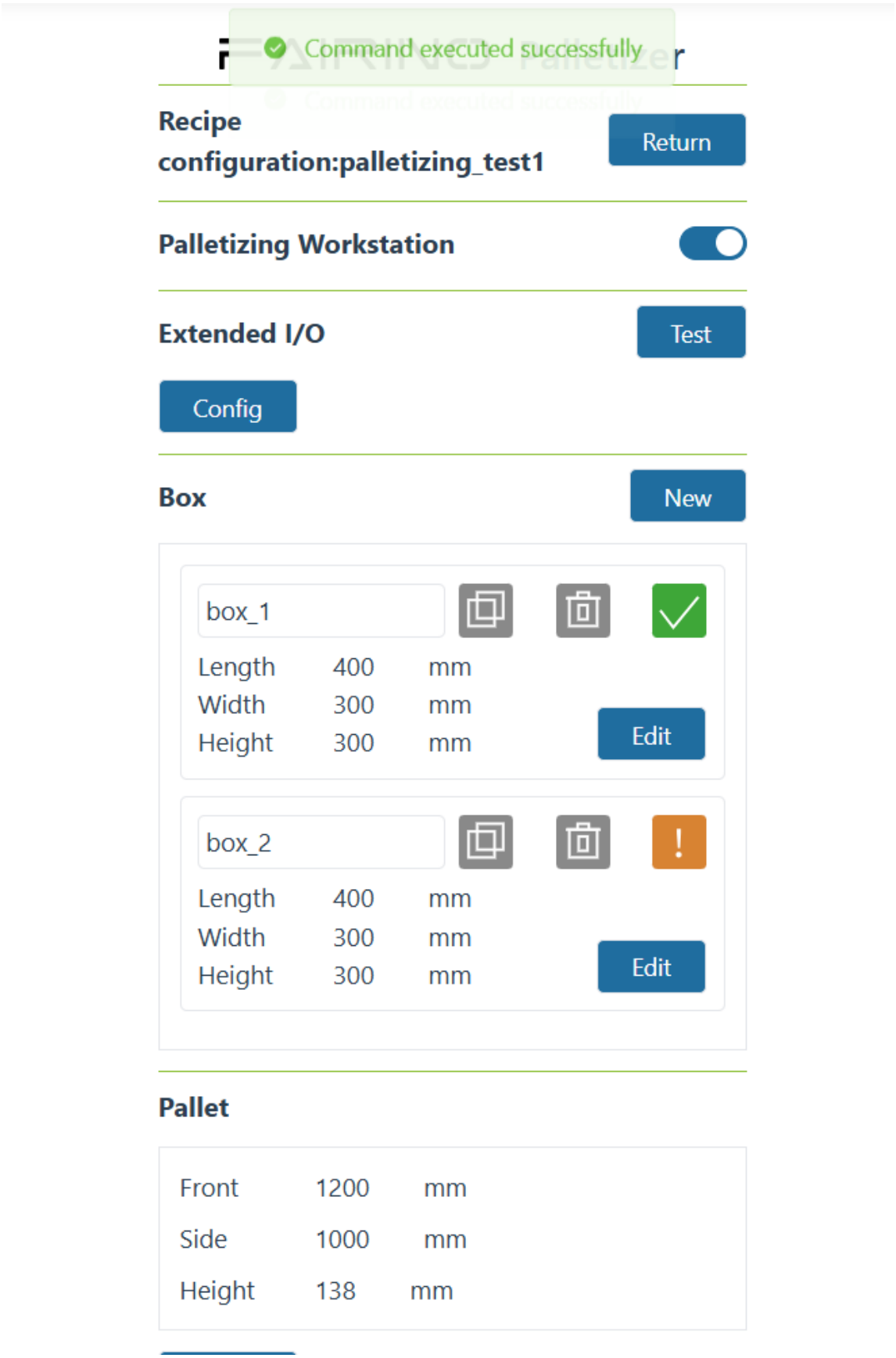

Command executed successfully
Recipe configuration:palletizing_test1
Return
Palletizing Workstation
Extended I/O
Test
Config
Box
New
box_1
Length 400 mm
Width 300 mm
Height 300 mm
Edit
box_2
Length 400 mm
Width 300 mm
Height 300 mm
Edit
Pallet
Front 1200 mm
Side 1000 mm
Height 138 mm

Figure 10-3-5 Delete box

For any box, click the "Edit" button to enter the box parameter configuration interface. After the configuration is successful, the box configuration status chart is green; when the configuration is not completed, the box configuration status icon is yellow.

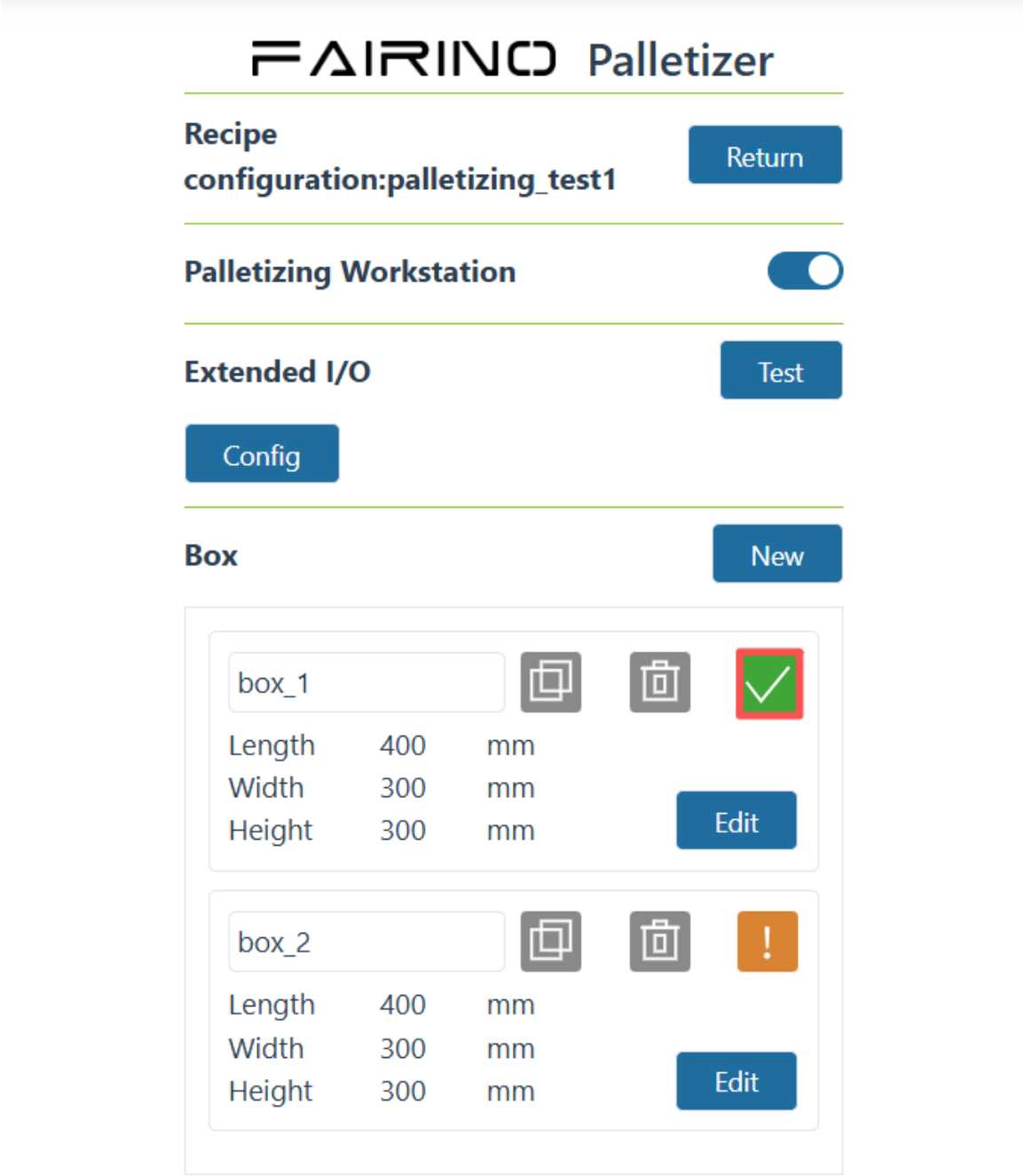


Figure 10-3-6 Box parameter configuration completed

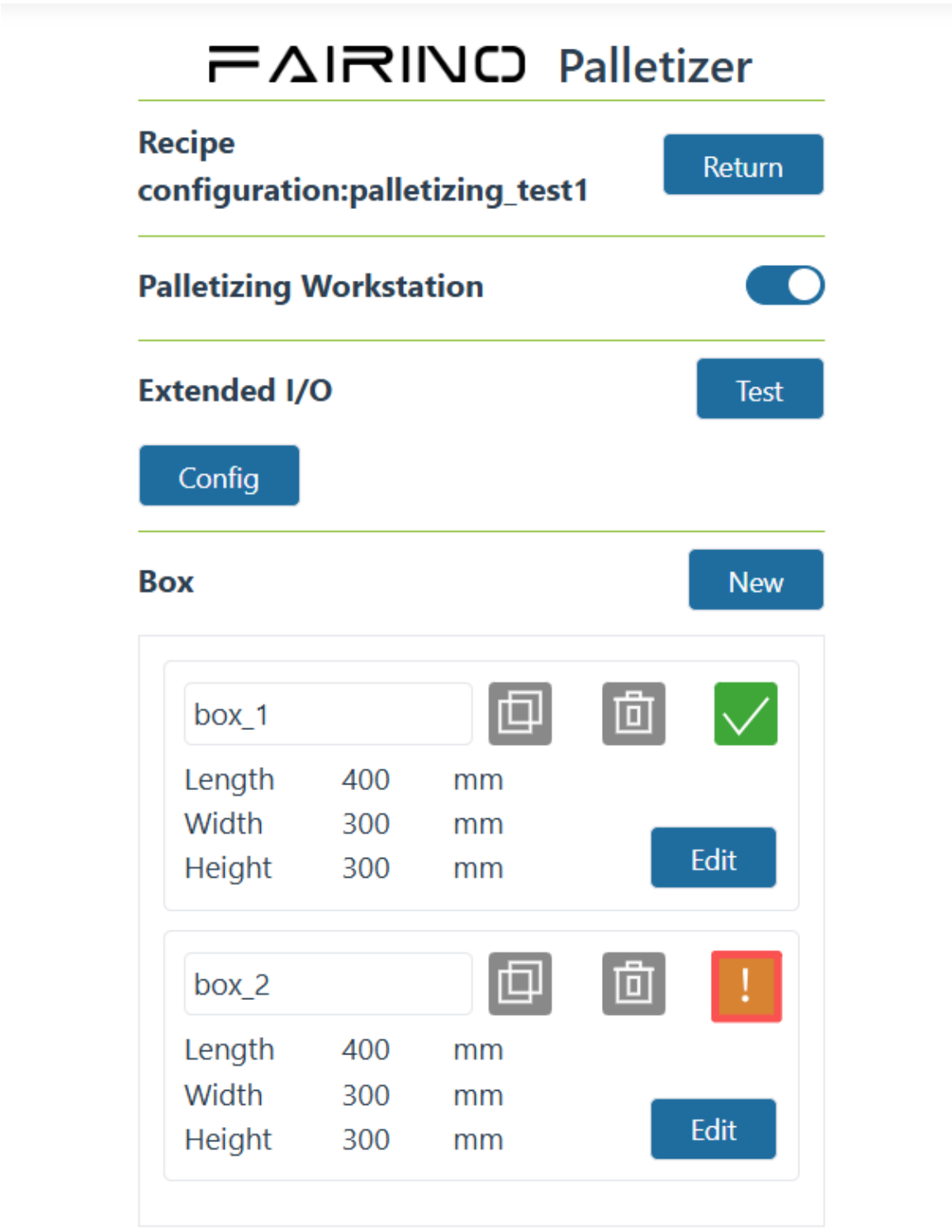
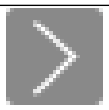


Figure 10-3-7 Box parameter configuration is not completed

4.10.3.5 Box parameters

Note: 

Name: Previous box
Function: Switch to select the previous box. When the first box is selected, switch the selection to the last box again.

Note: 

Name: Next box
Function: Switch to select the next box. When the last box is selected, switch to the first box again.

Click “Edit” in the box configuration column to enter the “Box Configuration” pop-up window, set the “length”, “width”, “height”, “load” and “workpiece label orientation”, click “Confirm” button to complete the box information configuration; set the grab point of the box (keep the grab point at the center of the box, and the bottom of the suction cup will be in a squeeze state when in contact with the box), and click the “Record” button to complete the setting.

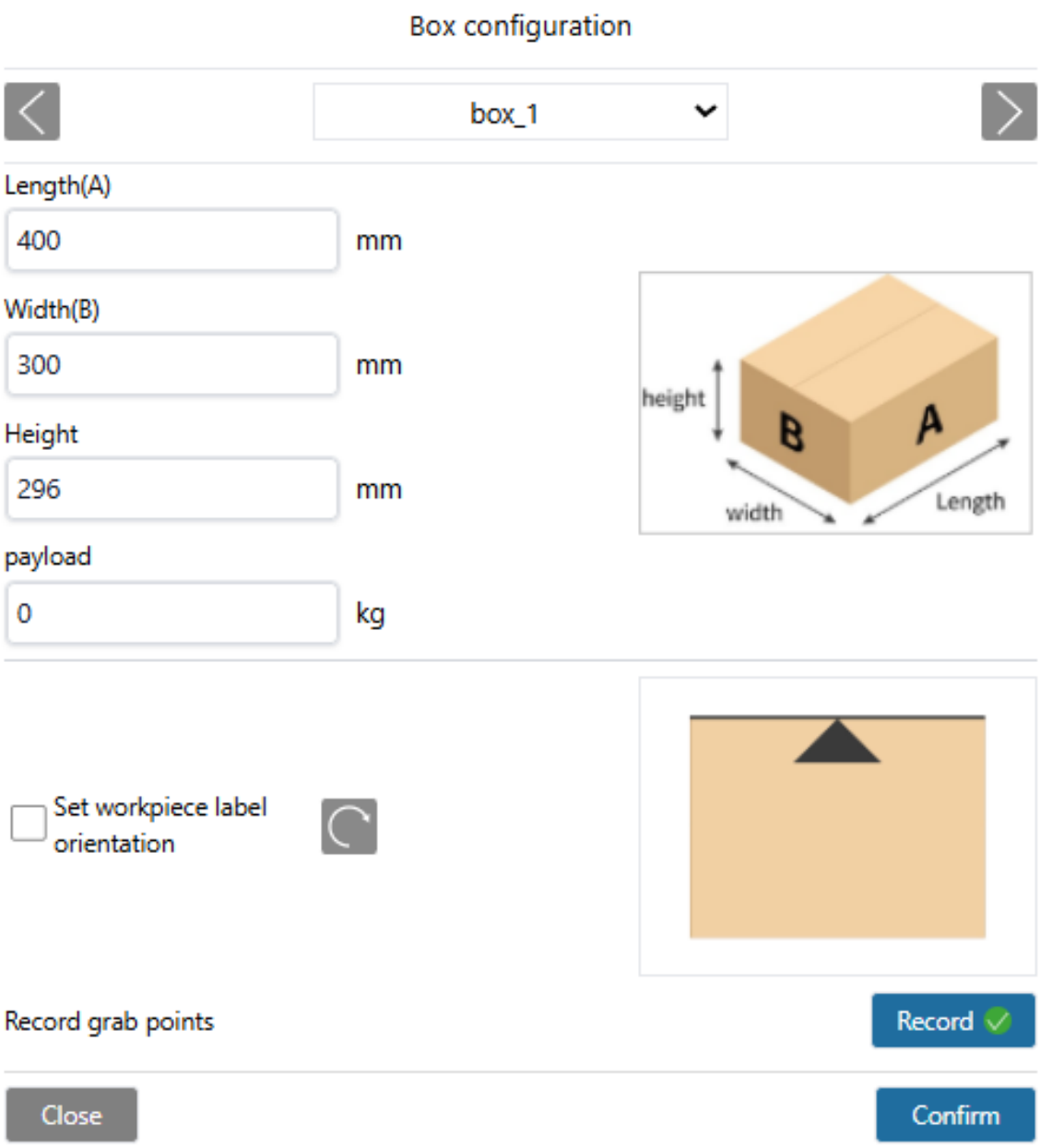


Figure 10-3-8 Box configuration



Figure 10-3-9 Box grab point

Important: The box grab point must be recorded, otherwise the length, width and height of the box cannot be configured.

4.10.3.6 Pallet configuration

Click “Config” in the pallet configuration column to enter the “Pallet Configuration” pop-up window, set the “front”, “side” and “height” of the pallet, then set the workstation transition point, and click “Confirm Configuration” to complete the pallet information setting.

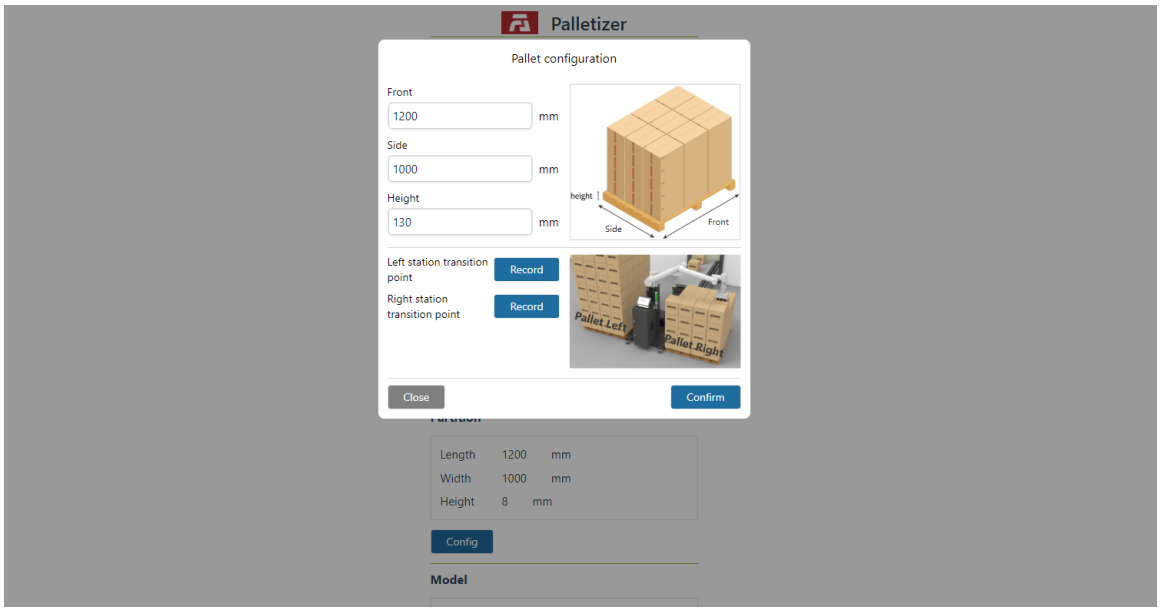


Figure 10-3-10 Pallet configuration



Figure 10-3-11 Left station transition point



Figure 10-3-12 Right station transition point

Important: Station transition points must be recorded, otherwise the program that cannot be generated cannot be saved.

4.10.3.6.1 Pattern configuration

4.10.3.7 Pattern operation

When selecting boxes in a pattern configuration, you can select boxes of the same height but different lengths and widths. The mode display area is divided into: mode addition (configuring palletizing type) and palletizing layer configuration.

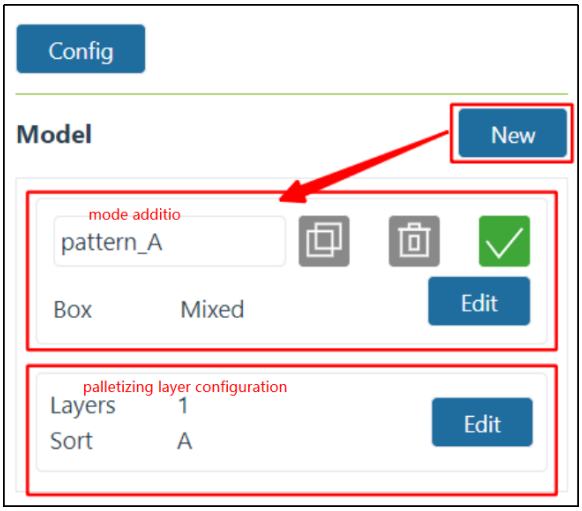
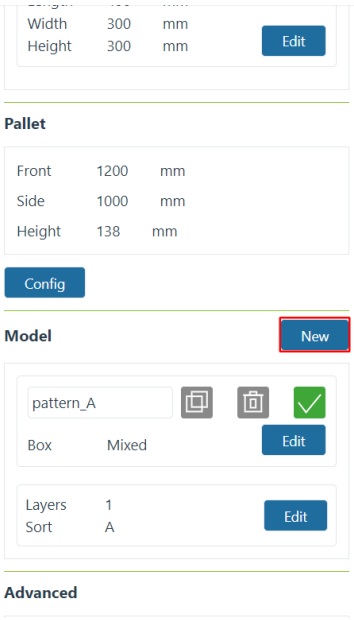


Figure 10-3-13 Mode display area

Click the “Add” button. After the addition is successful, a new mode will be added in the current order.



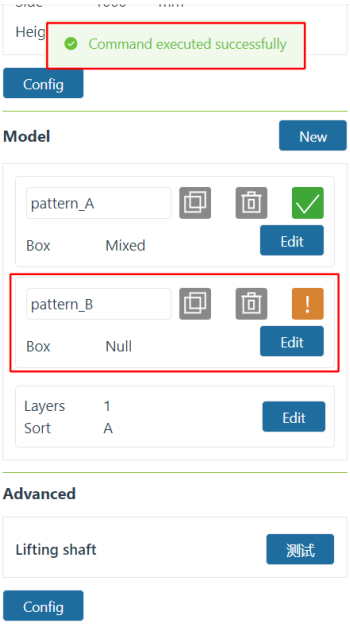


Figure 10-3-14 New pattern

In any mode in the mode adding area, click the input box area where the mode name is displayed, and the “Mode Rename” modal window will pop up. After entering the name, click the “Confirm” button to confirm the renaming.

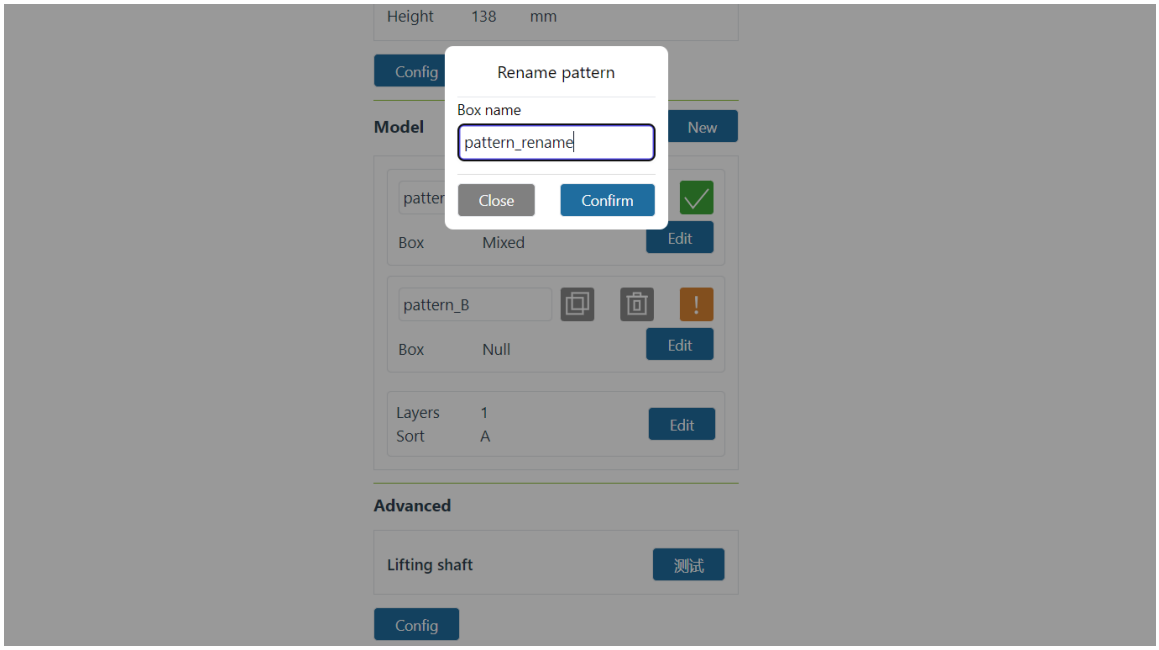


Figure 10-3-15 Rename pattern

In any mode in the mode adding area, click the “Copy” icon. After the copy is successful, a mode will be copied according to the current mode name.

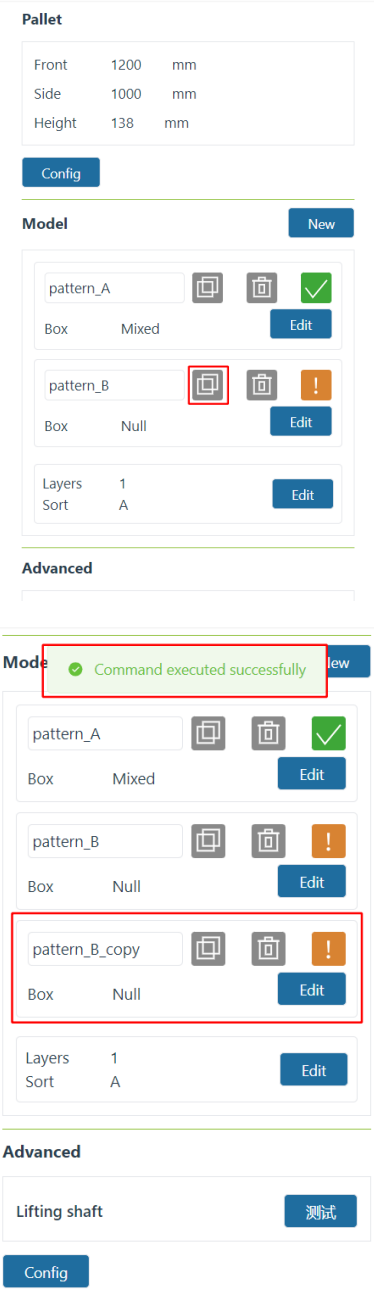


Figure 10-3-16 Copy pattern

In any mode in the mode adding area, click the “Delete” icon to delete the current mode data.

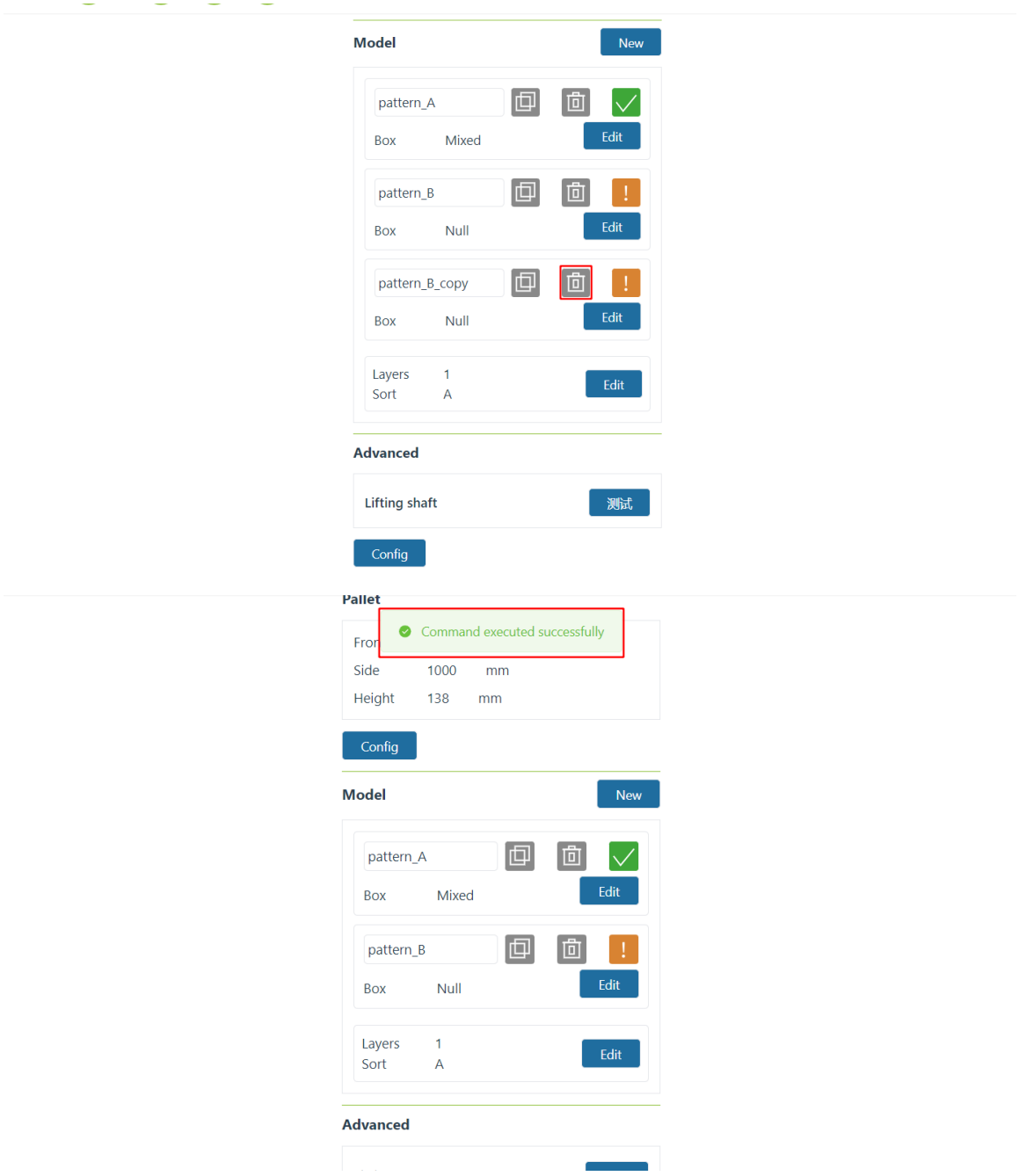


Figure 10-3-17 Delete pattern

In any mode in the mode adding area, click the “Edit” button to enter the “Mode Configuration” modal window and configure the palletizing type of the current mode. After the configuration is successful, the box configuration status chart is green; when the configuration is not completed, the box configuration status icon is yellow.

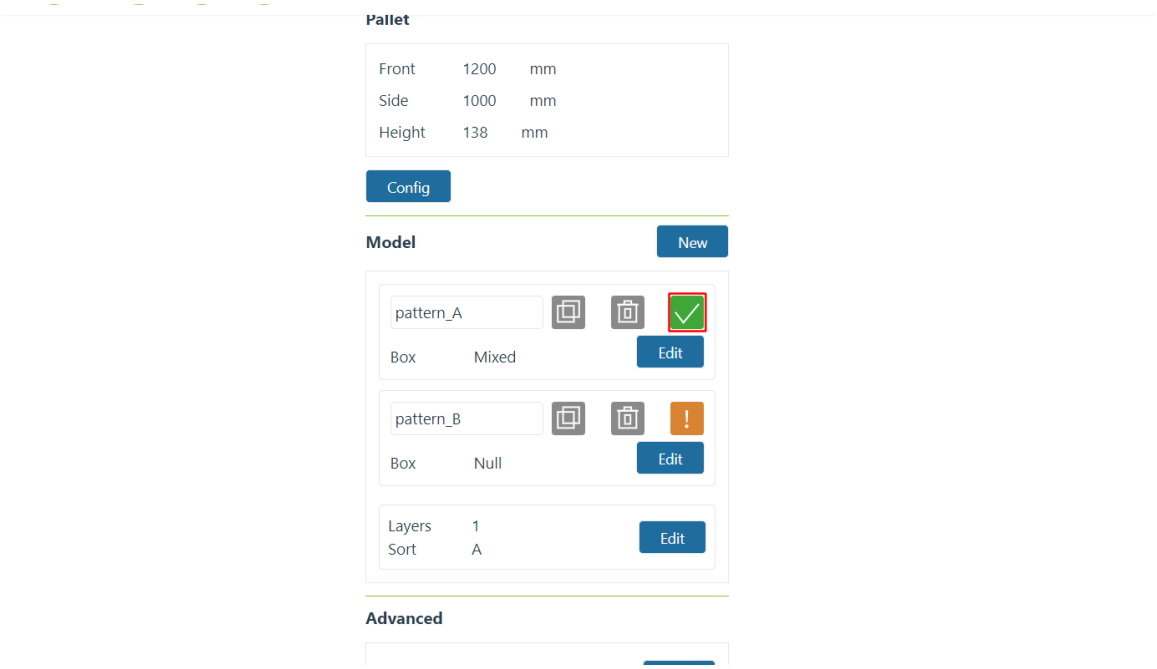


Figure 10-3-18 Pattern parameter configuration completed

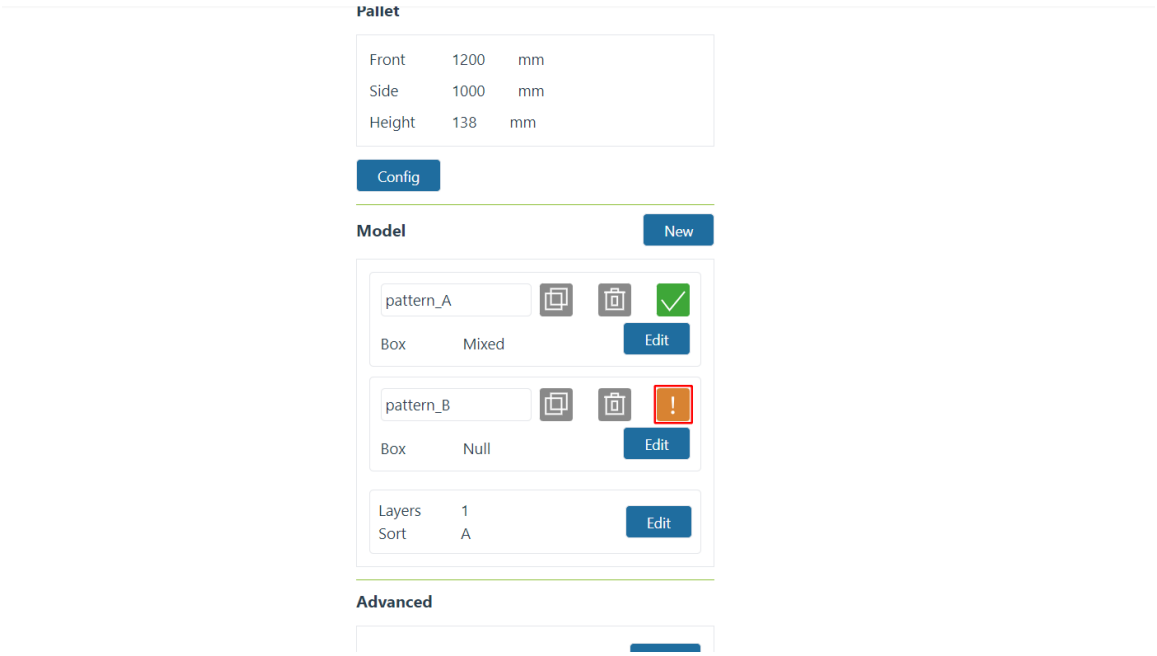


Figure 10-3-19 Mode parameter configuration is not completed

In the palletizing layer configuration area, the number and sorting of the palletizing layers are displayed. Click the “Edit” button to enter the “Palletizing Sequence Configuration” modal window, enter the “Number of Palletizing Layers”, select the mode of each layer, and click the “Confirm” button to complete the configuration.

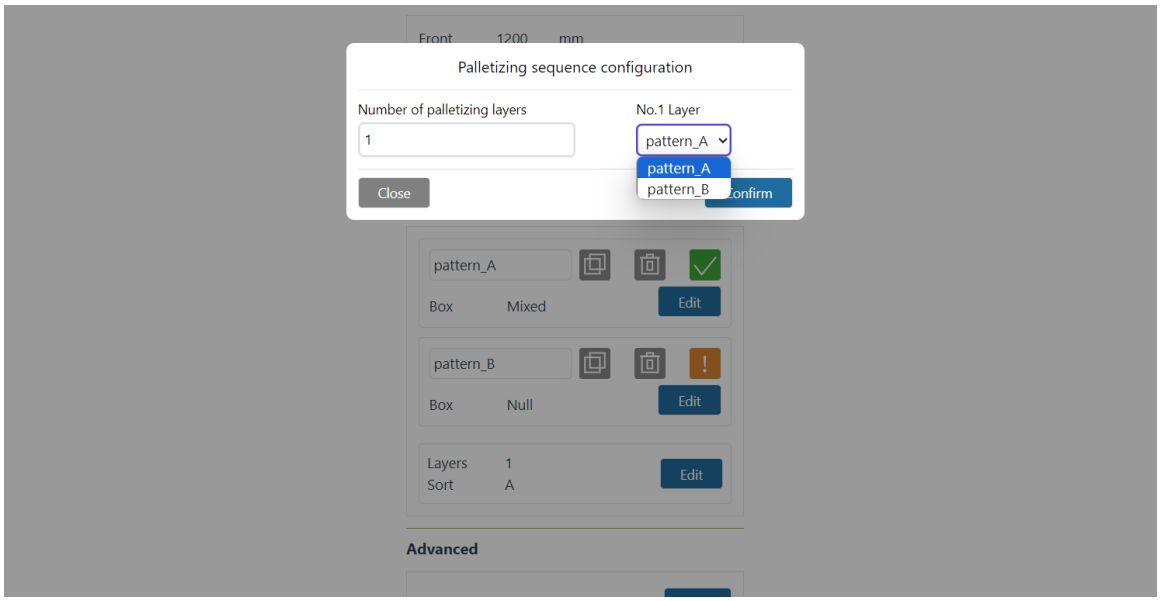
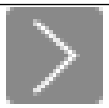


Figure 10-3-20 Palletizing layer configuration

4.10.3.8 Pattern parameters

Note: 

Name: Previous mode
Function: Switch to select the previous mode. When the first mode is selected, switch to the last mode again.

Note: 

Name: Next mode
Function: Switch to select the next mode. When the last mode is selected, switch to the first mode again.

Click “Edit” in the mode configuration column to enter the “Mode Configuration” pop-up window. It is mainly divided into three areas: pattern selection, box operation and stacking simulation.

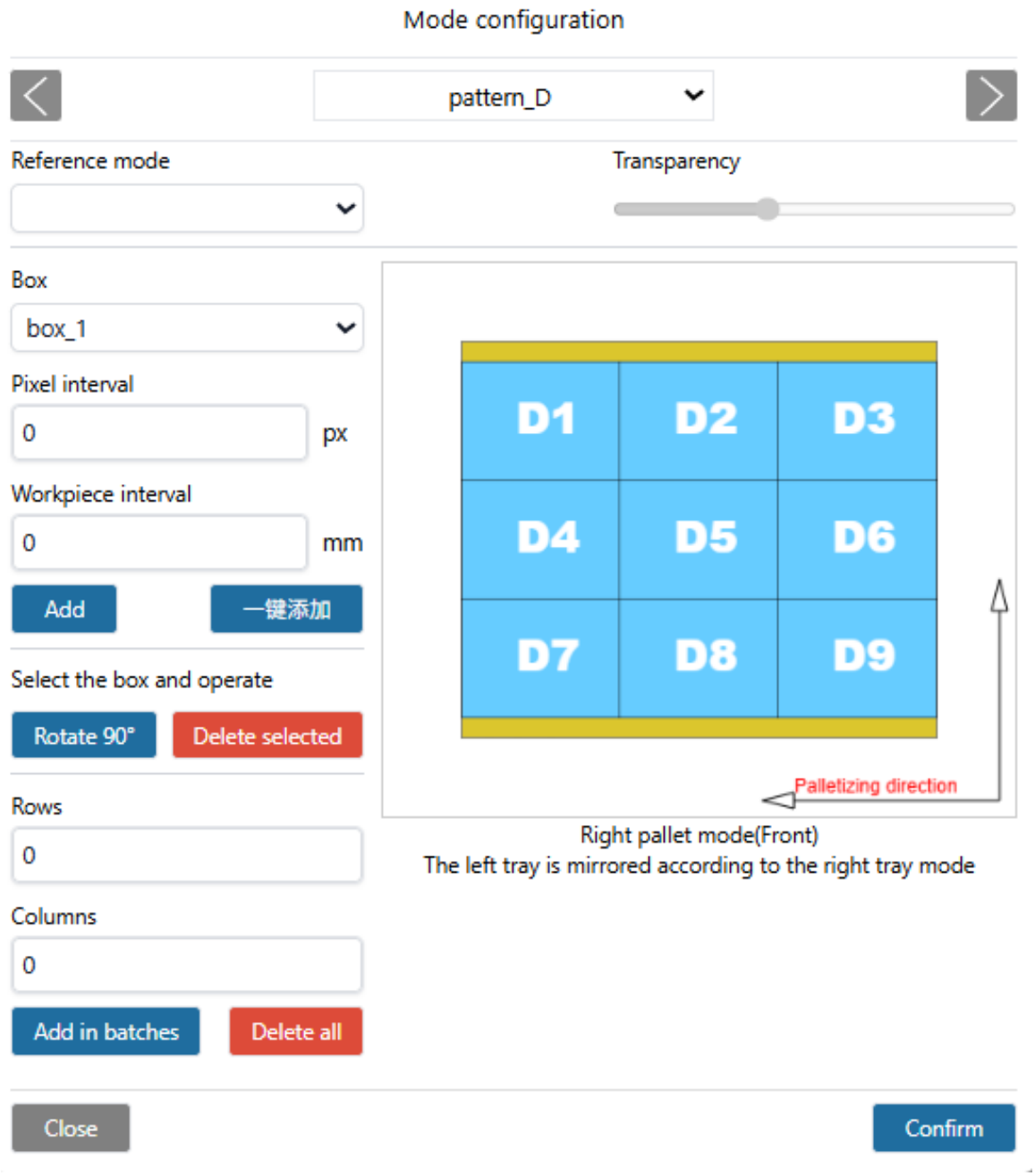


Figure 10-3-21 Pattern configuration

Important: When adding boxes, the background color of the workpiece turns red when there is a collision between boxes, and the above operation cannot be performed at this time. If necessary, please adjust the box to be collision-free.

In the pop-up header, select a mode, then choose boxes in the box operation area to add them under this mode. You can choose one-click addition, which by default fills the centered pallet with boxes without any gaps. Customize box spacing; boxes can be added individually or in batches. Click "Confirm" to complete the mode information setup. When the selected boxes have inconsistent heights, configuration cannot be completed, and the prompt "Box type heights are inconsistent and cannot be added to the same mode" will appear.

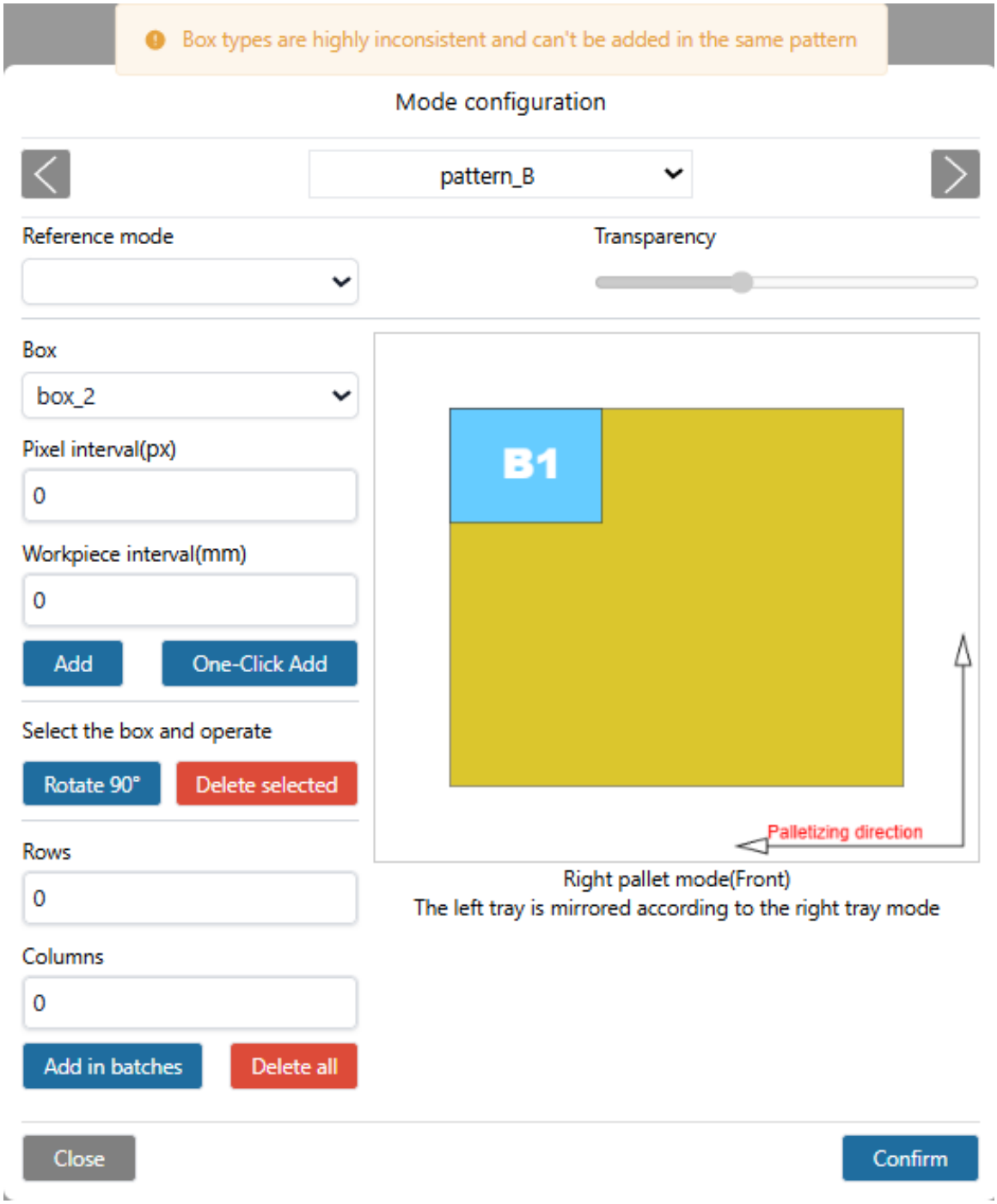


Figure 10-3-22 Tips for inconsistent heights of selected boxes

Select the reference mode (the selected mode cannot be selected), compare and see whether the current mode configuration can be palletized based on the reference mode, so that customers can intuitively check the box stacking types in different modes.

Note: Currently only two modes, A and B, are supported.

Important: Palletizing direction: Taking the right pallet as an example, the lower right corner is the farthest point. Place a row of workpieces vertically or horizontally from the lower right corner, then place workpieces horizontally or vertically in the upper row, and so on (Web page The palletizing direction has been marked, please check carefully). The left pallet places workpieces mirroring the right pallet mode.

4.10.3.8.1 Advanced configuration

Click "Config" in the advanced configuration column to enter the "Advanced Configuration" pop-up window. The configuration items are as follows:

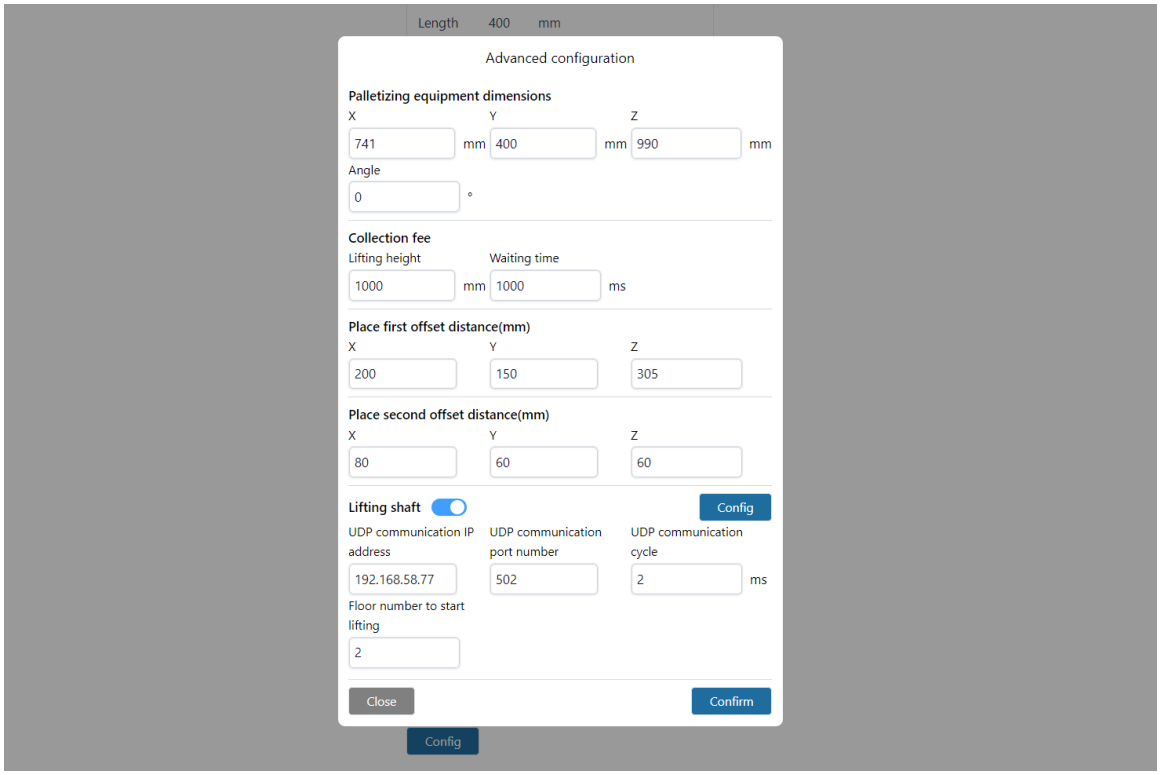
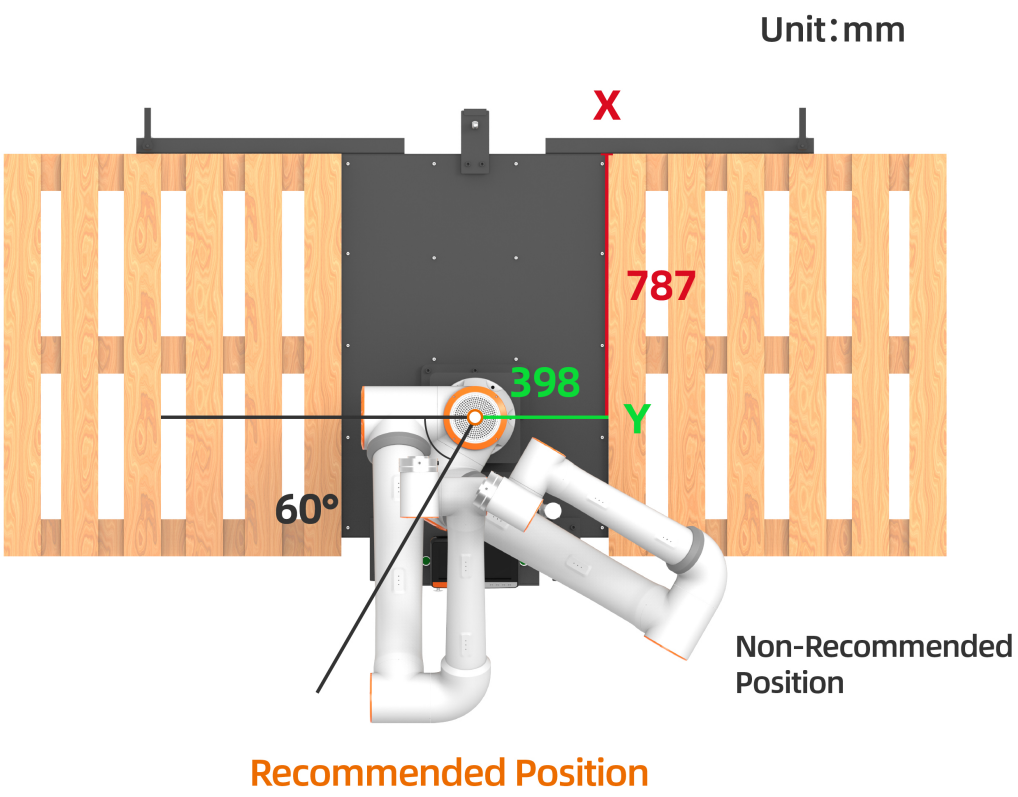


Figure 10-3-23 Advanced configuration

1) Dimensions of palletizing equipment: Dimensions of palletizing workbench.


Unit:mm
X
787
398
Y
60°
Non-Recommended Position
Recommended Position

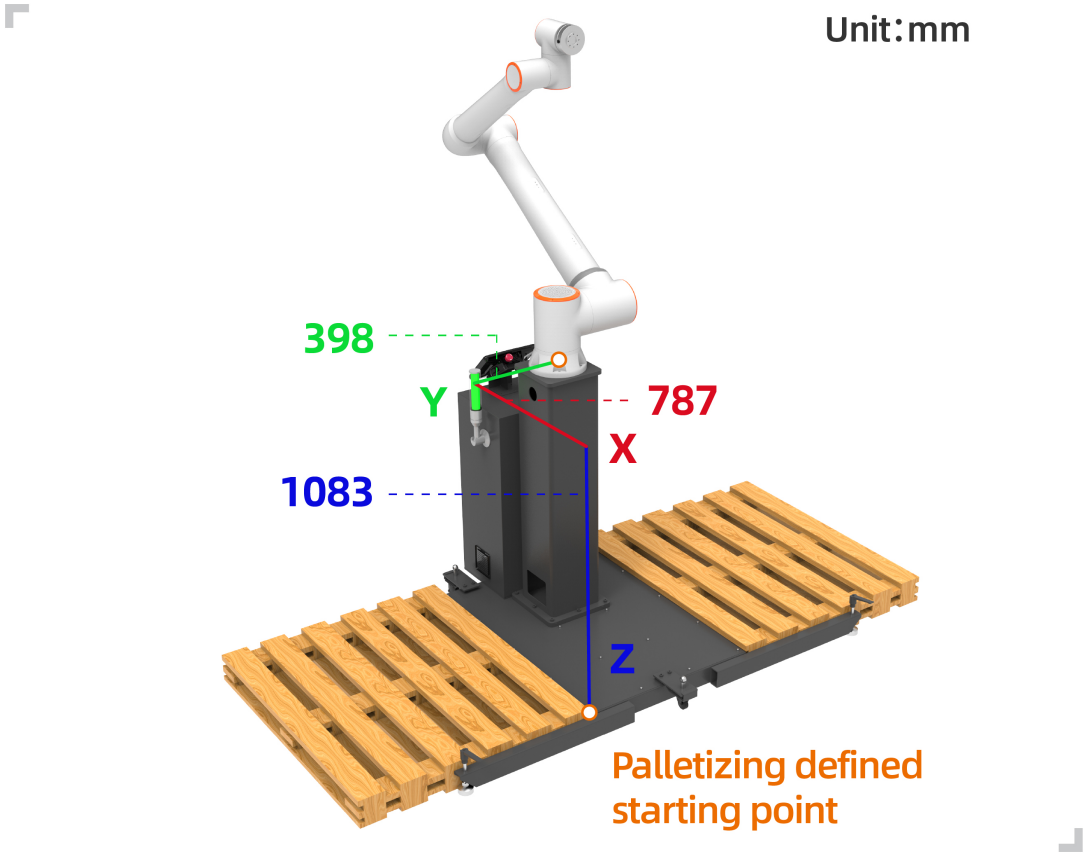


Figure 10-3-24 Palletizing Station

Important: X, Y, and Z are the absolute values of the coordinates of the point at the upper right corner of the left tray or the upper left corner of the right tray relative to the robot's base coordinate system. Angle is the rotation angle during the robot's installation, which is recommended to be 0 during installation.

2) Lifting height for picking up materials: User-defined lifting height after successfully picking up materials from the grabbing point.

3) Waiting time for picking up materials: The user can customize the waiting time for monitoring the negative pressure arrival signal after picking up materials, and repeat the picking up action when it is not in place.

4) First/second offset distance: User-defined offset distance for tilting the robot to the target point. (Note: The first offset parameter Z must be greater than the height of the box, otherwise it will collide with the already placed boxes during the stacking process).

5) Partition configuration: Click "Configuration" in the partition configuration column to enter the "Partition Configuration" pop-up window, set the partition size "length", "width" and "height" and select the start and stop of the partition.

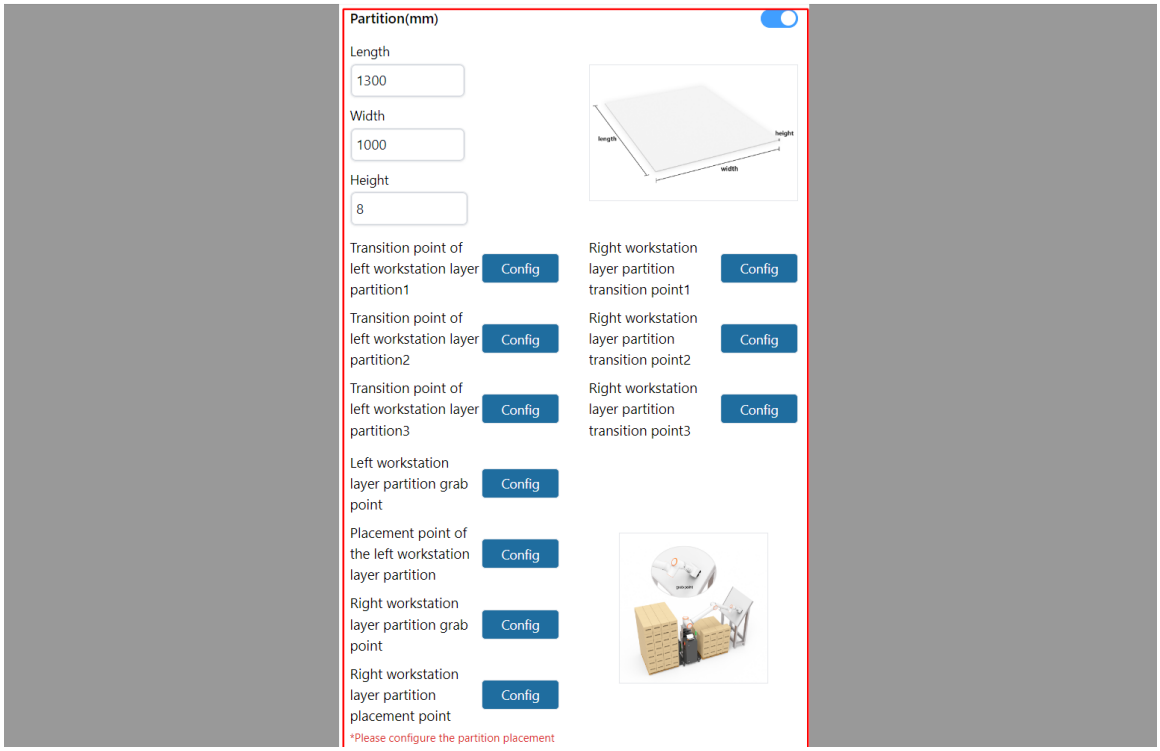


Figure 10-3-25 Partition configuration

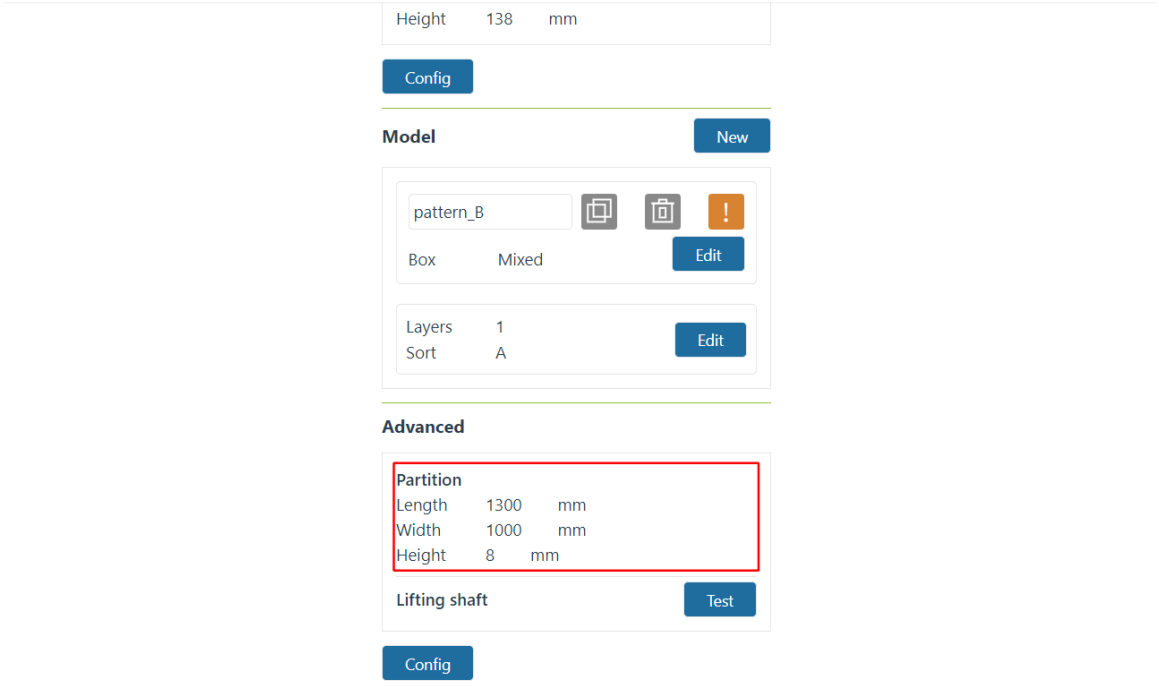


Figure 10-3-26 Recipe Management - Advanced Configuration Displays Partition Configuration

Then set the transition points of the partitions. There are three transition points for the partitions. The purpose of setting them is to roughly plan a movement path after grabbing the partitions to avoid collisions and being unable to complete the action of placing the partitions.

Note: Transition point 1 starts to move a certain distance from the box grab point and then teaches; transition point 2 starts to move a certain distance from transition point 1 and starts teaching, and can also become a transition intermediate point; transition point 3 starts to move from transition point 2 A distance is the last point before the partition is placed.

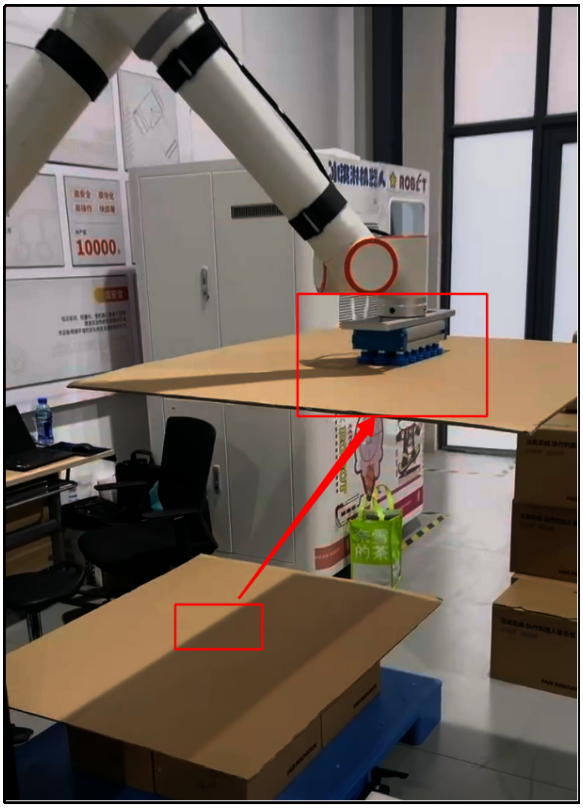


Figure 10-3-27 Partition transition point 1 (take the right station as an example)



Figure 10-3-28 Partition transition point 2 (take the right station as an example)

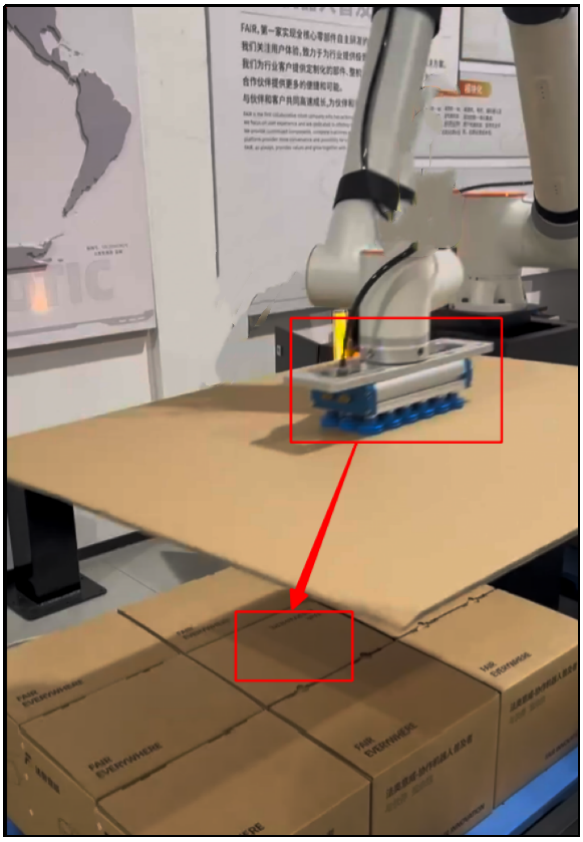


Figure 10-3-29 Partition transition point 3 (take the right station as an example)

Then set the grab point (keep the grab point at the center of the partition, and the bottom of the suction cup will be in a squeeze state when it contacts the partition) and placement point, and click “Confirm” to complete the partition information setting.

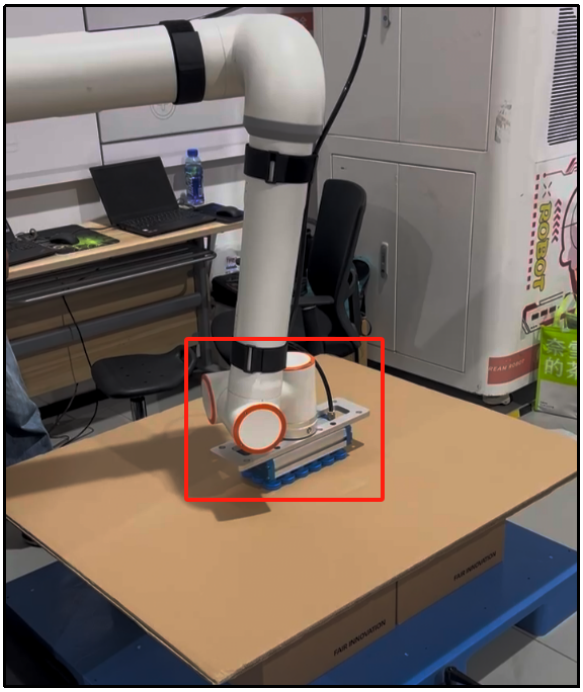


Figure 10-3-30 Partition grab point (take the right station as an example)



Figure 10-3-31 Partition placement point (take the right workstation as an example)

6) Lifting axis: Users can customize the start and stop of the lifting axis, communication parameters (IP address, port number and communication cycle), the floor number to start lifting and select the start and stop of the lifting axis.

Note:

- When the lifting shaft is working, the height it lifts each time is the height of the box.
- When the lifting shaft function is turned on, the home page displays the advanced configuration content and displays the lifting shaft test button. Click the “Test” button to enter the “Lifting” button. Axis Test” pop-up window is used to test the accuracy of loading communication, rising and falling of the lifting axis to avoid problems such as inability to work and large errors when used directly.

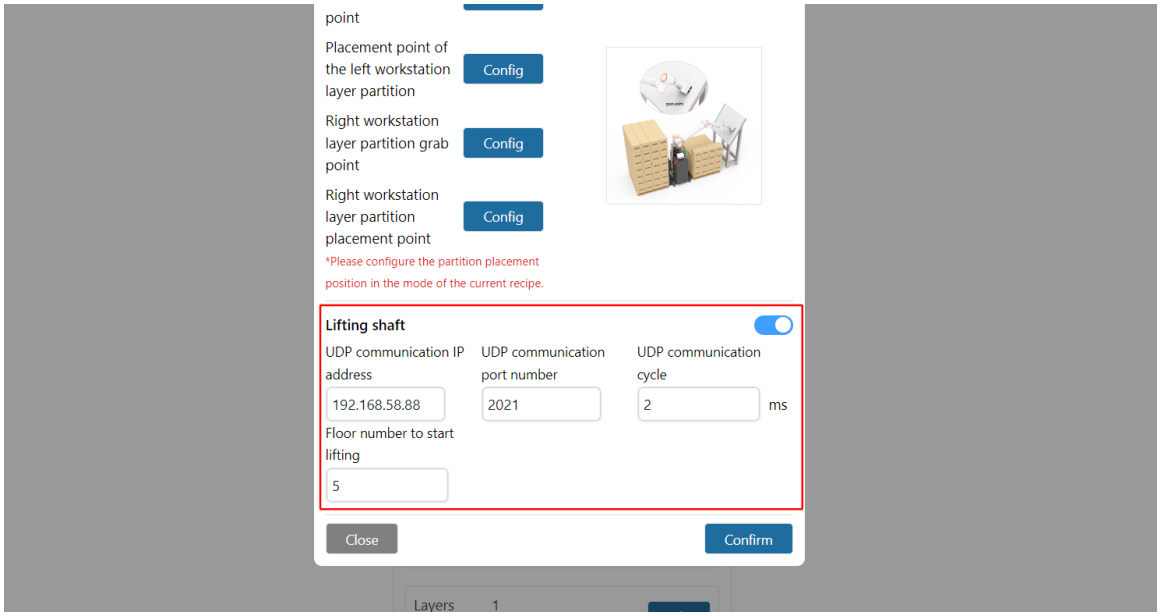


Figure 10-3-32 Lifting shaft configuration

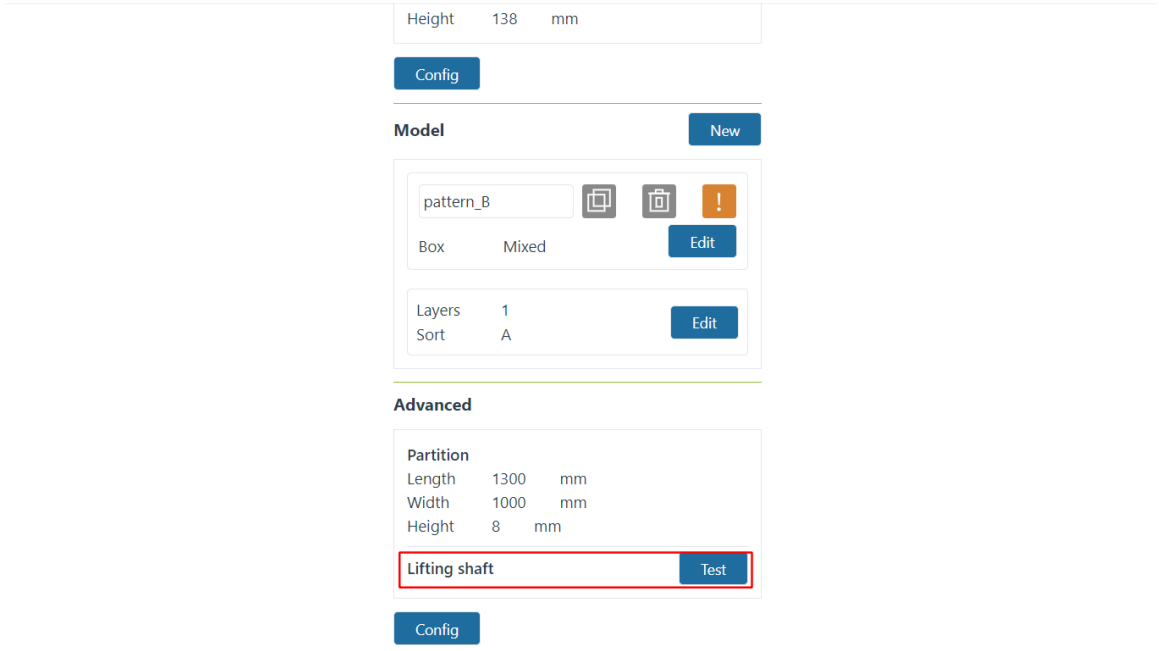


Figure 10-3-33 Recipe Management - Advanced Configuration Display Lifting Axis

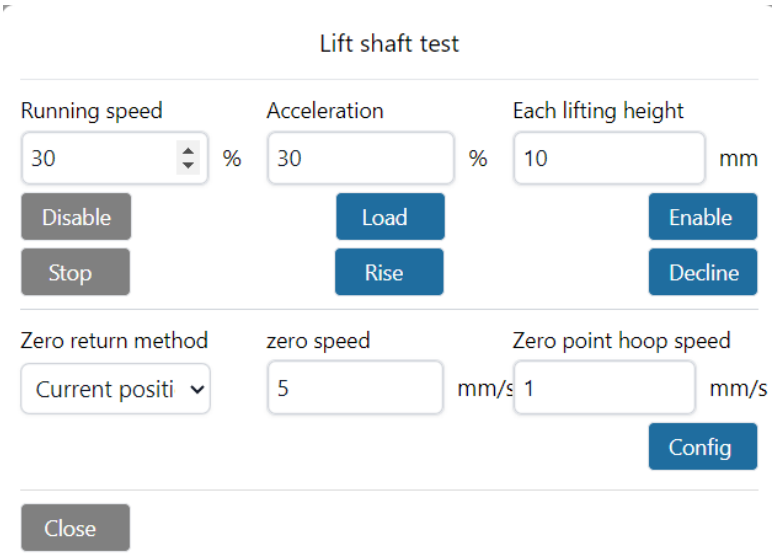


Figure 10-3-34 Lifting shaft test

4.10.4 Program Generation

View “Program Generation” below the recipe display, and select the recipe based on the recipe and requirements. When recipes are selected for both left and right stations, startup priority needs to be selected; when a recipe is selected only for the left or right station, startup priority selection is not required. After entering the program name, click the “Generate” button.

Note: All program names start with “palletizing”. There is no need to input “palletizing”; just input the name after “_”. For example, for “palletizing_program”, input “program”.

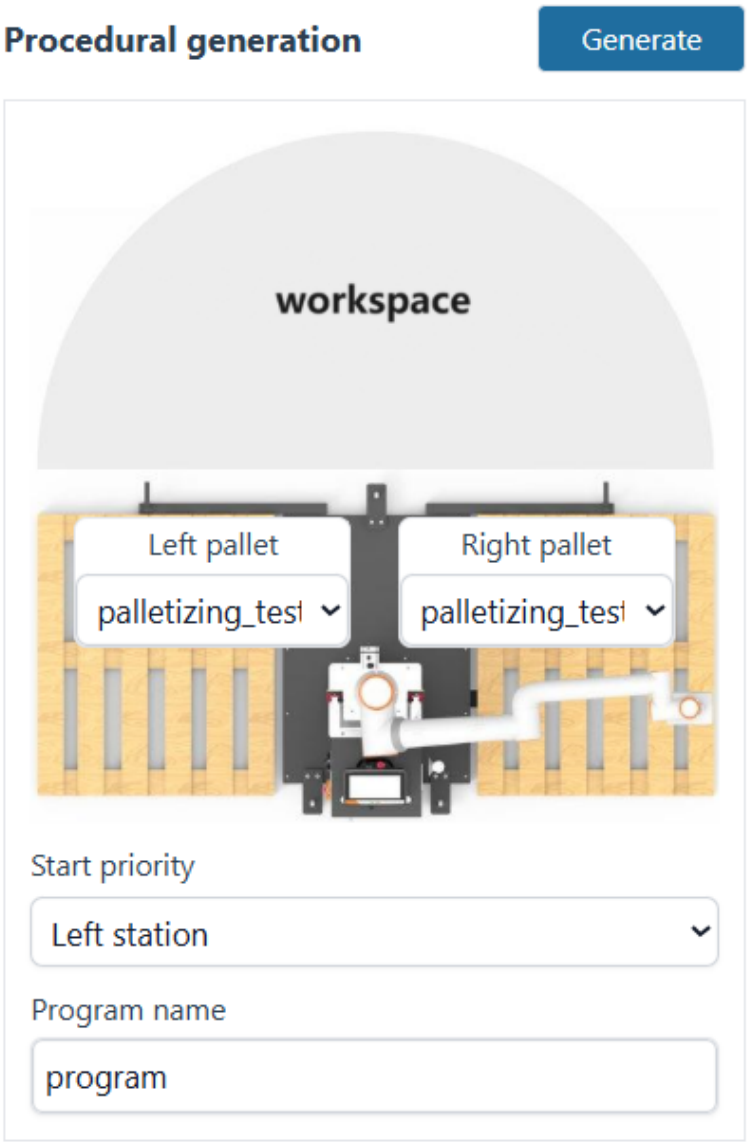


Figure 10-4-1 Program Generation – Selecting Recipes for Both Left and Right Stations

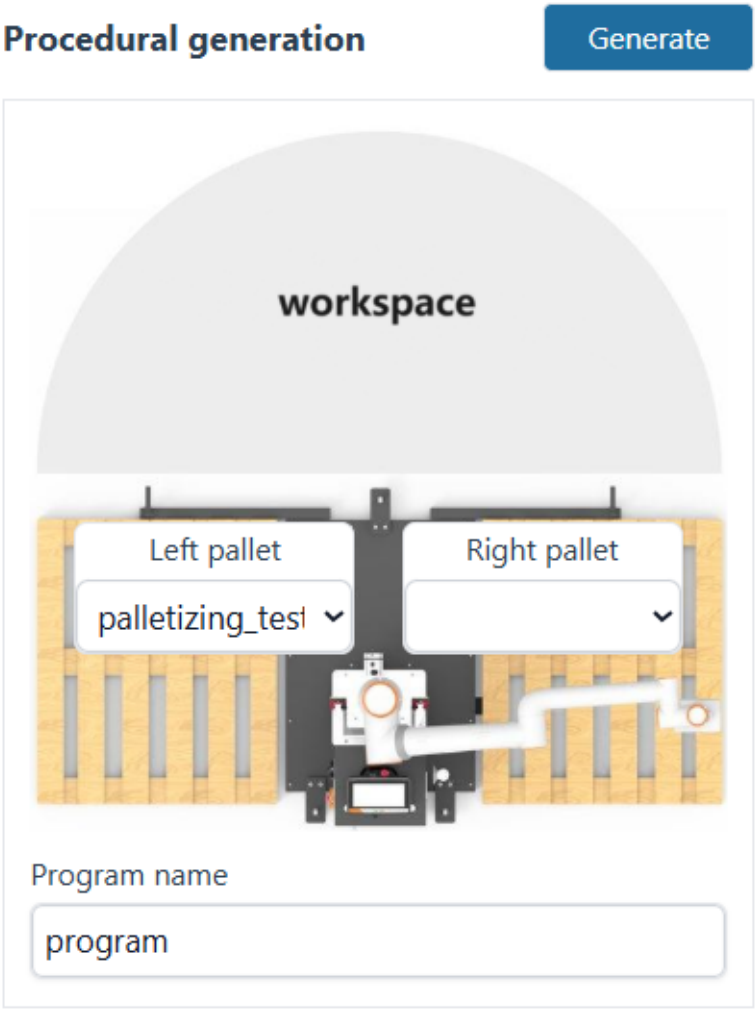


Figure 10-4-2 Program Generation – Selecting Recipe for Left Station, No Recipe for Right Station

Important:

1. If no palletizing recipe is selected for the left or right station, it means that station is not enabled.
2. After successfully generating the program, be sure to manually save all subprograms and the main program in Program Teaching.
3. Depalletizing programs start with “de”. For example, if the palletizing program is “palletizing_program”, the depalletizing program would be “depalletizing_program”.
4. When running a program with recipes configured for both left and right stations, after simultaneously receiving the workpiece-in-position signals for left and right, work proceeds according to the set priority.

4.10.4.1 Single Pick Point Program

There are two scenarios for a single pick point program:

(1) Left and right stations select the same recipe;

(2) Left and right stations select different recipes, but the box pick point pose configured in the recipes is the same.



Figure 10-4-3 Single Pick Point Box Pick Point

4.10.4.2 Dual Pick Point Program

To configure a dual pick point program, the left and right stations must select different recipes, and the box pick point poses configured in the recipes must be different.



Figure 10-4-4 Dual Pick Point Box Pick Point

CHAPTER

FIVE

FR LUA PROGRAMMING SCRIPT

5.1 PDF Download

FRLua programming script user manual

CHAPTER

SIX

ROBOT COMMUNICATION

6.1 Introduction to CNDE

Collaborative robot configurable network data exchange protocol (CNDE) is a way for the client to control the robot and obtain the feedback status of the robot through UDP communication.

Table 1-1 shows all the states of the robot that can be obtained by CNDE. The client can arbitrarily select several required states from the table and make the robot perform state feedback according to the set feedback period.

Similarly, the client can also select the required combination of robot control functions from Table 1-2 for robot control operation. The communication data between the client and the robot CNDE should be in the specified frame format, and the communication port of the robot CNDE is 20006.

There are four main steps to use the CNDE function of robot:

Configuration of input and output data content: the client sends an input and output configuration command to the robot, in which the command content is in the form of a series of control or state function names such as "std_DI_box,cfg_DI_box, motion_queue_len", and the robot records and recognizes these names and then feeds back the corresponding function data types such as "UINT8,UINT8,INT32" to the client, which indicates that the configuration is successful.

Start the CNDE data output of the robot: the client sends a command to start the CNDE data output to the robot, and the robot starts to send the robot state data to the client through UDP in the form of byte array (Little Endian) according to the configured period.

Analyze the robot state data: the client receives the state data fed back by the robot circularly, and analyzes the data according to the data types fed back by the robot during output configuration and the byte length corresponding to each data type in Table 1-3 to obtain the actual value of each state. The output data of robot CNDE can support up to 4096 bytes, The CNDE output period ranges from 1 ms to 200ms.

Sending robot control data: the client groups the control data according to the data types fed back by the robot during input configuration and the byte length corresponding to each data type in Table 1-3, and sends it to the robot through UDP communication. After receiving the control data, the robot performs data analysis and robot control operations. The CNDE input of the robot supports 256 recipes, and the client can configure multiple input recipes as needed. When sending the input data to the robot, it is necessary to specify the recipe number corresponding to the current data.

Table 1-1 Robot Output Configuration Function

Name	Data type	Description
std_DI_box	UINT8	Control box standard DI input (bit0 ~ bit7 indicates DI0 ~ DI7)
cfg_DI_box	UINT8	Control box configurable CI input (bit0 ~ bit7 indicates CI0 ~ CI7)
cfg_DI_tool	UINT8	Configurable tool DI inputs (bit0 ~ bit2 indicates toolDI0 ~ toolDI1)

continues on next page

Table 1 – continued from previous page

std_AI0_box	DOUBLE	Control box analog input AI0(0 ~ 4095)
std_AI1_box	DOUBLE	Control box analog input AI1(0 ~ 4095)
std_AI_tool	DOUBLE	Analog input of end tool tool_AI0(0 ~ 4095)
run_up_time	DOUBLE	Statistics of Robot Boot Time (s)
target_joint_pos	DOUBLE_6	Target position of joint 1-6 (°)
target_joint_vel	DOUBLE_6	Target speed of joints 1-6 (°/s)
target_joint_acc	DOUBLE_6	Target acceleration of joints 1-6 (°/s2)
target_joint_curre	DOUBLE_6	Joint 1-6 target current (A)
target_joint_torqu	DOUBLE_6	Target torque of joints 1-6 (Nm)
actual_joint_pos	DOUBLE_6	Current position of joints 1-6 (°)
actual_joint_vel	DOUBLE_6	Current speed of joints 1-6 (°/s)
actual_joint_curr	DOUBLE_6	Current current of joints 1-6 (A)
actual_joint_torqu	DOUBLE_6	Joint 1-6 target torque (Nm)
actual_TCP_pos	DOUBLE_6	Current position of tool DKR(mm)
actual_TCP_vel	DOUBLE_6	Current tool speed DKR(mm/s)
actual_TCP_force	DOUBLE_6	Tool resultant force DKR(mm/s2)
target_TCP_pos	DOUBLE_6	Tool target position DKR(mm)
target_TCP_vel	DOUBLE_6	Tool target speed DKR(mm/s)
std_DO_box	UINT8	Standard DO output of control box (bit0 ~ bit7 indicates DO0 ~ DO7)
cfg_DO_box	UINT8	Control box configurable with CO output (bit0 ~ bit7 indicates CO0 ~ CO7)
cfg_DO_tool	UINT8	Standard tool DO output (bit0 ~ bit1 indicates toolDO0 ~ toolDO1)
std_AO0_box	DOUBLE	Control box analog AO0 (0.0 ~ 4095.0)
std_AO1_box	DOUBLE	Control box analog AO1 (0.0 ~ 4095.0)
std_AO_tool	DOUBLE	Tool analog AO1 (0.0 ~ 4095.0)
robot_mode	UINT8	Robot mode (0- automatic; 1- Manual)
collision_level	UINT8_6	Joint 1-6 collision grade (1-10)
speed_scaling_	DOUBLE	Manual mode speed percentage (0 ~ 100)
speed_scaling_	DOUBLE	Automatic mode speed percentage (0 ~ 100)
program_state	UINT8	Robot program running state (1- stop; 2- in motion; 3- pause; 4- Drag)
line_number	INT32	Current program running line number
payload	DOUBLE	Load mass (kg)
pay_cog	DOUBLE_3	Load centroid (x,y,z)(mm)

continues on next page

Table 1 – continued from previous page

motion_queue_len	INT32	Current motion queue length
ft_sensor_data	DOUBLE_6	Force sensor raw data
main_code	INT32	Main fault code
sub_code	INT32	Sub fault code
emergency_stop	UINT8	Emergency stop status
motion_done	INT32	Motion completion status
timestamp_us	UINT64	Robot system time (us)
output_BIT_reg_8	UINT8_X	BIT-type robot output registers (8xX indicates the number of registers, if you need 16 BIT-type output registers, the actual name is “output_BIT_reg_8x2”, and the robot can support up to 128 bit-type output registers)
output_INT_reg_X	INT32_X	INT robot output registers (X represents the number of registers. If you need 16 INT output registers, the actual name is “output_INT_reg_16”, and the robot can support up to 64 INT output registers)
output_DOUBLE_	DOUBLE_X	DOUBLE robot output register (X represents the number of registers, if you need 16 DOUBLE output registers, the actual name is “output_DOUBLE_reg_16”, and the robot can support up to 64 DOUBLE output registers)
ft_sensor_data	DOUBLE_6	Force sensor data

Table 1-2 Configuration Functions of Robot Input Control

Name	Data type	Description
speed_mask	UINT8	Global speed setting mask: 0-disable; 1- enable
speed	UINT8	Set the global speed (0-100)
std_DO_mask	UINT8	Control box standard DO output control mask (bit0 ~ bit7 indicates DO0 ~ DO7)
std_DO_box	UINT8	Control box standard DO output (bit0 ~ bit7 indicates DO0 ~ DO7)
cfg_DO_mask	UINT8	Control box configurable CO output mask (bit0 ~ bit7 indicates CO0 ~ CO7)
cfg_DO_box	UINT8	Control box Configurable with CO output (bit0 ~ bit7 indicates CO0 ~ CO7)
cfg_DO_tool_r	UINT8	Control box standard tool DO output control mask (bit0 ~ bit1 indicates toolDO0 ~ toolDO1)
cfg_DO_tool	UINT8	Control box standard tool DO output (bit0 ~ bit1 indicates toolDO0 ~ toolDO1)
std_AO_mask	UINT8	Robot analog output control mask (bit0 ~ bit1 indicates control box AO0 ~ AO1; Bit2 stands for tool AO0)
std_AO0_box	DOUBLE	Control box analog AO0 (0.0 ~ 4095.0)
std_AO1_box	DOUBLE	Control box analog AO1 (0.0 ~ 4095.0)
std_AO0_tool	DOUBLE	Tool analog AO1 (0.0 ~ 4095.0)
input_BIT_reg_8	UINT8_X	BIT-type robot input registers (8xX indicates the number of registers, if you need 16 BIT-type input registers, the actual name is “input_BIT_reg_8x2”, and the robot can support up to 128 bit-type registers)
input_INT_reg_X	INT32_X	INT robot input registers (X represents the number of registers, if you need 16 INT input registers, the actual name is “input_INT_reg_16”, and the robot can support up to 64 INT registers)
input_DOUBLE_	DOUBLE_X	DOUBLE robot input register (X represents the number of registers, if you need 16 DOUBLE input registers, the actual name is “input_DOUBLE_reg_16”, and the robot can support up to 64 DOUBLE registers)

Table 1-3 Correspondence between Data Types and Byte Length

Data type	Byte length
UINT8	1
INT32	4
DOUBLE	8
UINT8_X	1*X
INT32_X	4*X
DOUBLE_X	8*X

6.2 CNDE data frame protocol format

The communication protocol of CNDE for cooperative robots is as follows. The protocol distinguishes data frames with different functions by frame types. See Table 2-2 for the definition of frame types. Different types of frames correspond to different data content. See Table 3-1 ~ Table 3-7 for specific data content definitions.

Table 2-1 Data Frame Format of Robot CNDE

Name	Frame header	Frame count	Frame type	Data length	Content	Frame end
Length(byte)	2	1	1	2	–	2
Content	0x5A5A	0 ~ 255	0 ~ 8	Number of bytes of "content"	Data frame content	0xA5A5

Table 2-2 Data Frame Types of Robot CNDE

Type	Numerical value	Data frame direction
Input Configuration Frame (Control Configuration)	0x00	Client->Robot
Output Configuration Frame (Status Configuration)	0x01	Client->Robot
CNDE output start	0x02	Client->Robot
CNDE output stops	0x03	Client->Robot
Output data frame (status data)	0x04	Robot->Client
Input data frame (control data)	0x05	Client->Robot
Message	0x06	Client->Robot, Robot->Client
Set the version number of CNDE protocol of robot	0x07	Client->Robot
Obtain the robot soft firmware version	0x08	Client->Robot, Robot->Client

6.3 CNDE function operation

6.3.1 Input configuration and input data

The client sends data frames to the robot through CNDE to control the robot's DO, AO output, input registers, etc. Before sending the input data, it is necessary to configure the functional content to be controlled. Table 2-1 shows the format of CNDE input configuration content, including recipe number and a series of input configuration function names (Table 1-2); The corresponding table 3-2 is the content format of input data, including recipe number and input data byte group.

CNDE data input supports up to 8 recipes. When sending input data, the robot will match the recipe number in the received data to the corresponding recipe configuration function name group, and analyze the data to get the input data value of each function name, and then control the robot according to the input data.

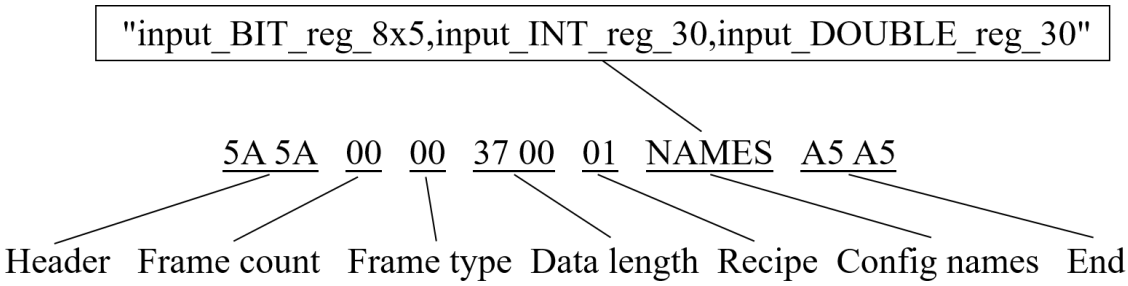
Table 3-1 Input Configuration Content Format

Name	Recipe number	Function name string
Length (byte)	1	–
Content	0 ~ 7	A series of input data function names

Table 3-2 Input Data Content Format

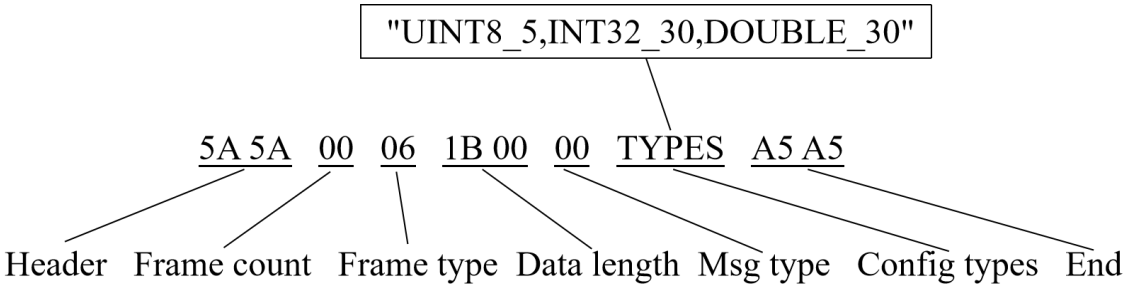
Name	Recipe number	Data byte group
Length(byte)	1	–
Content	0 ~ 7	Input data content

When inputting configuration, the robot controller will check each name after receiving the configuration name group. If the configured function names are correct, the robot will feed back the data type names of all configured functions divided by ","; If the configured function name is wrong, the robot will feed back the corresponding error content. An example of the input configuration data frame (hexadecimal) is as follows:



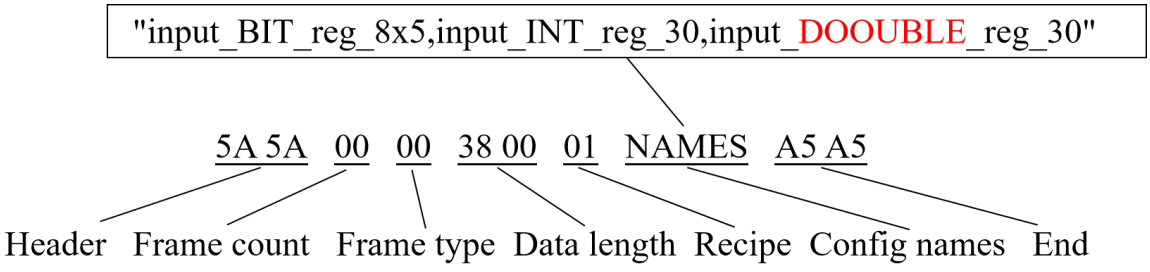
Among them, the total length of the configured input function name group is 54 bytes, plus 1 byte of the input recipe number, which is 55 bytes, which is converted into hexadecimal as 0x0037. In the Little-Endian mode, the data length in the corresponding input data frame is "37 00".

At this time, the robot will feed back a data frame with the type of messages (Message in section 3.3.1 of this article):

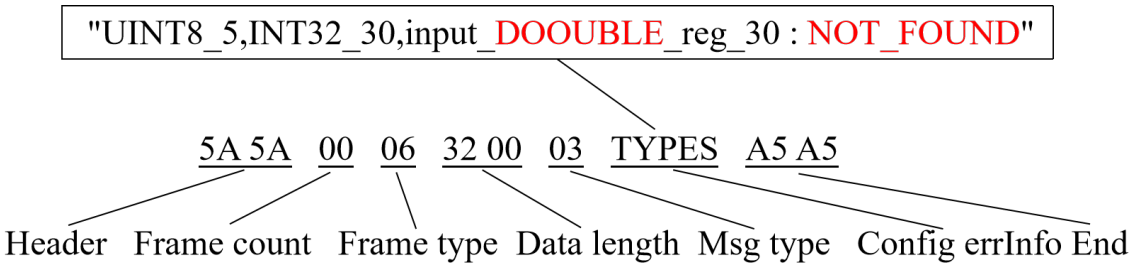


The message type "00" indicates that this is a successful feedback message. The client can extract the "Input Data Configuration Type" and compare it with Table 1-3 to get the byte length of the input configuration. In this example, the total data length is 1*5+4*30+8*30 = 365 bytes.

If the configuration name is entered incorrectly:

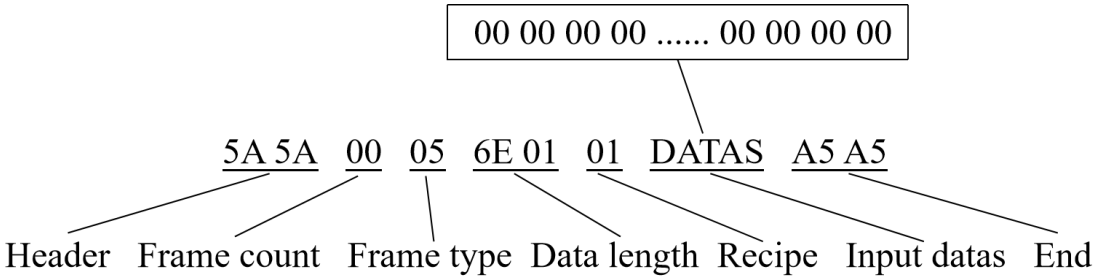


The corresponding feedback information is:



The input data can be input circularly according to a certain period, or only when necessary. The fastest period that the robot can process during cyclic input is 1ms, but the faster input period will bring some resource expenses to the robot system. It is suggested that you set the data input period reasonably according to the actual situation.

In addition, when sending a data frame to the robot, the robot will not have feedback information unless the length of the sent data frame or the data is abnormal. An example of an input data frame is as follows, in which the input data recipe number and the input data byte length should be consistent with the input configuration:



6.3.2 Output configuration and output data

The client can customize the content and period of the state feedback through CNDE. Using the state feedback of the robot CNDE requires the following three steps: output configuration; Start the output; Receiving output data.

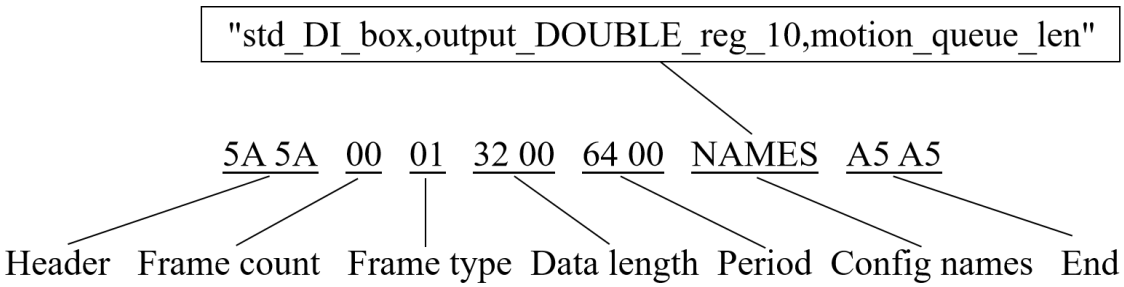
6.3.2.1 Output configuration

The output configuration frame includes the output period and the output function name group (see Table 1-1 for all configurable names). The configurable range of the output period is 1 ~ 200ms, and the maximum number of output data bytes supports 4096byte. The output function name group is a series of output function name strings separated by “,”.After the client sends the output configuration frame, the robot will check the configured function names. If the configured function names are all supported by the current robot CNDE, the robot will feed back a series of data type combinations separated by “,”; Otherwise, if the verification of the output configuration name fails, the corresponding error message is fed back.

Table 3-3 Output Configuration Contents

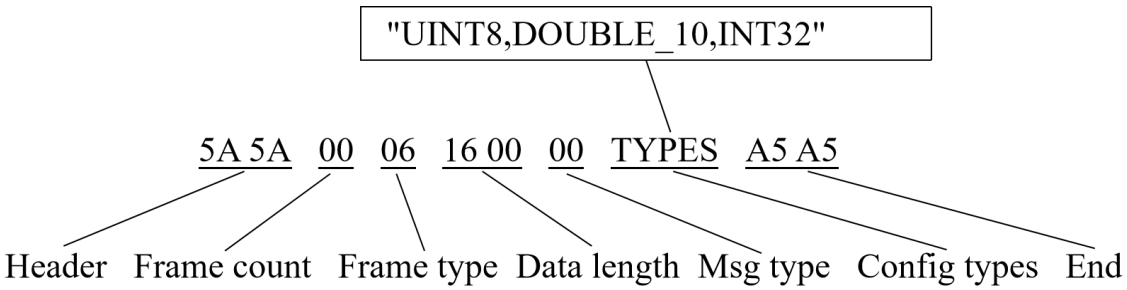
Nmae	Output period (ms)	Function name string
Length (byte)	2	–
	1-200	Output function name group

If the output configuration frame is as follows:



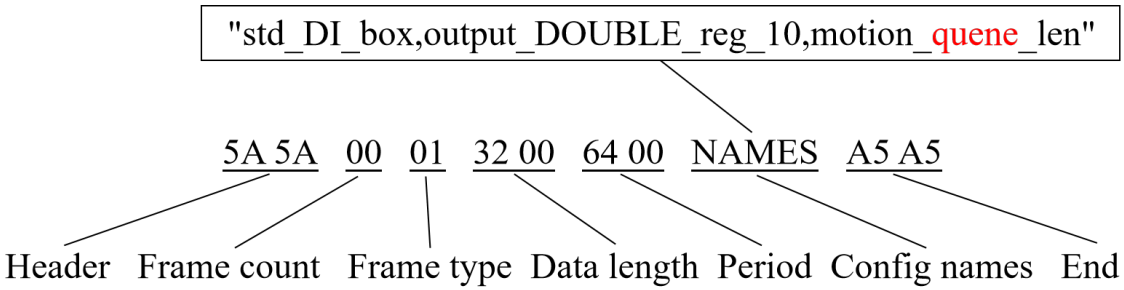
Among them, the total length of the configured output function name group is 48 bytes, plus 2 bytes of the output period, totaling 50 bytes, which is converted into hexadecimal as 0x0032. In the Little-Endian mode, the data length in the corresponding input data frame is “32 00”.

At this time, the robot will feed back a data frame with the type of message (Message in section 3.3.1 of this article):

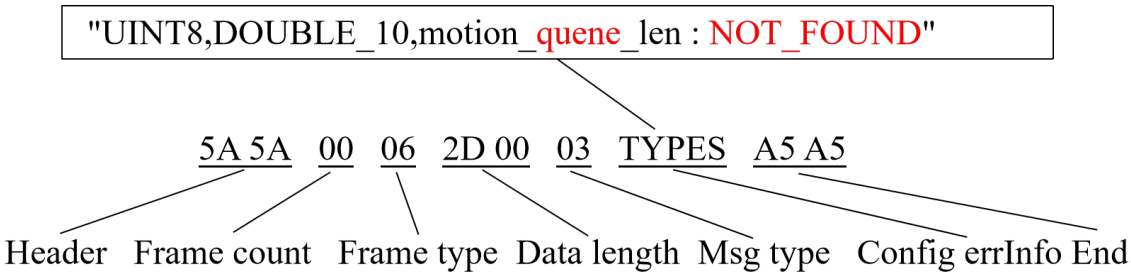


The message type “00” indicates that this is a successful feedback message. The client can extract the “output data configuration type” and compare it with Table 1-3 to get the byte length of the output configuration. In this example, the total data length is 1+8*10+4 = 85 bytes.

If the input configuration name is wrong, such as “queue” wrongly written as “quene”:



The corresponding feedback information is:



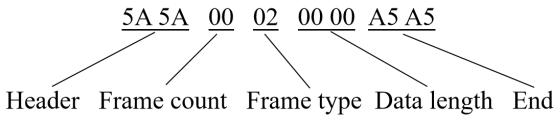
6.3.2.2 Output start and stop

After the CNDE output configuration of the robot is completed, send a start command to start the CNDE output, and the robot will perform state feedback output according to the configured output period and output content, and also send a CNDE stop output command, and the robot will stop the state feedback output. CNDE start and stop commands have no command content, and the corresponding data length is 0.

Table 3-4 Start and Stop Contents of CNDE Output

Name	Data byte group
Length (byte)	0
Content	without

An example of starting the robot CNDE to output data frames is as follows:



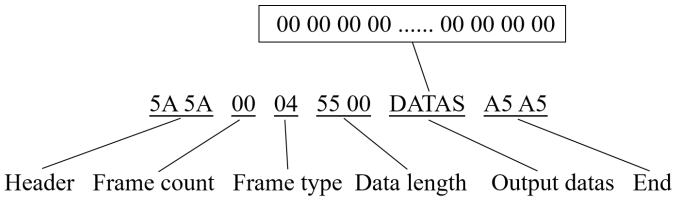
6.3.2.3 The client receives the output data

After the CNDE data output of the robot is started, the client needs a cycle to receive the data information fed back by the robot, and the cycle receiving frequency of the client is higher than the configured output data frequency, otherwise data packet loss may occur. The contents of robot output data are shown in Table 3-5. The length of the byte group of robot output data is the sum of the byte lengths of all the function data configured for output, and the byte array is a combination of all the state data in the order of configured functions with 1 byte alignment.

Table 3-5 CNDE Output Data Content

Name	Data byte group
Length(byte)	–
Content	Output data byte group

Examples of robot output data frames are as follows:



6.3.3 CNDE auxiliary function

6.3.3.1 Messages

The client and the robot can send messages to each other through CNDE, including message types and message strings (Table 3-6), where the message types are defined in Table 3-7. When the CNDE client sends commands such as input configuration, output configuration, output start and output stop to the robot, the robot replies with a message.

If the above commands are executed successfully, the feedback message type of the robot is "successful", and the numerical code of the corresponding message type is 0x00; On the other hand, if the above commands fail to be executed, the message type fed back by the robot is "error", and the corresponding message type value is 0x03. The client can judge the command execution result according to the feedback message type, and if the message type is "error", the error information can be extracted to analyze the cause of the error.

Table 3-6 Content of Message

Name	Message type	Message string
Length(byte)	1	–
Content	0 ~ 4	Message string

Table 3-7 Robot CNDE Message Types

Type	Numerical value
SUCCESS	0x00
INFORMATION	0x01
WARINING	0x02
ERROR	0x03
FAULT	0x04

6.3.3.2 Switch the version number of CNDE protocol of robot

At present, there is only one version of the robot CNDE, and the version number is "FR-CNDE-V0001", so this function is reserved and has not been put into use yet.

6.3.3.3 Acquire that version information of the robot soft firmware

The client sends an command to get the software firmware version information to the robot through CNDE, and the command content is empty. After receiving the request, the robot will feed back a message, including the robot model, robot software version, robot firmware version, robot hardware version and other related information.

6.4 Robot input and output register

The CNDE client and robot can exchange data through input and output registers, which specifically includes two processes:

The input configuration of CNDE client includes the input register, and the input register value is modified when data is input. The input register value modified by CNDE client can be read by adding the command of reading the input register in the robot LUA program.

Add the command of writing output register into the robot LUA program, execute the LUA program to write the numerical value into the output register, and the output configuration of the CNDE client includes the output register to

start the feedback of the CNDE status of the robot, so that the client can read the numerical value of the output register written in the LUA program after receiving the CNDE output data.

6.4.1 Read input register

Open the WebApp, click “Program” and “Coding” in turn to create a new user program “testReg.lua”.

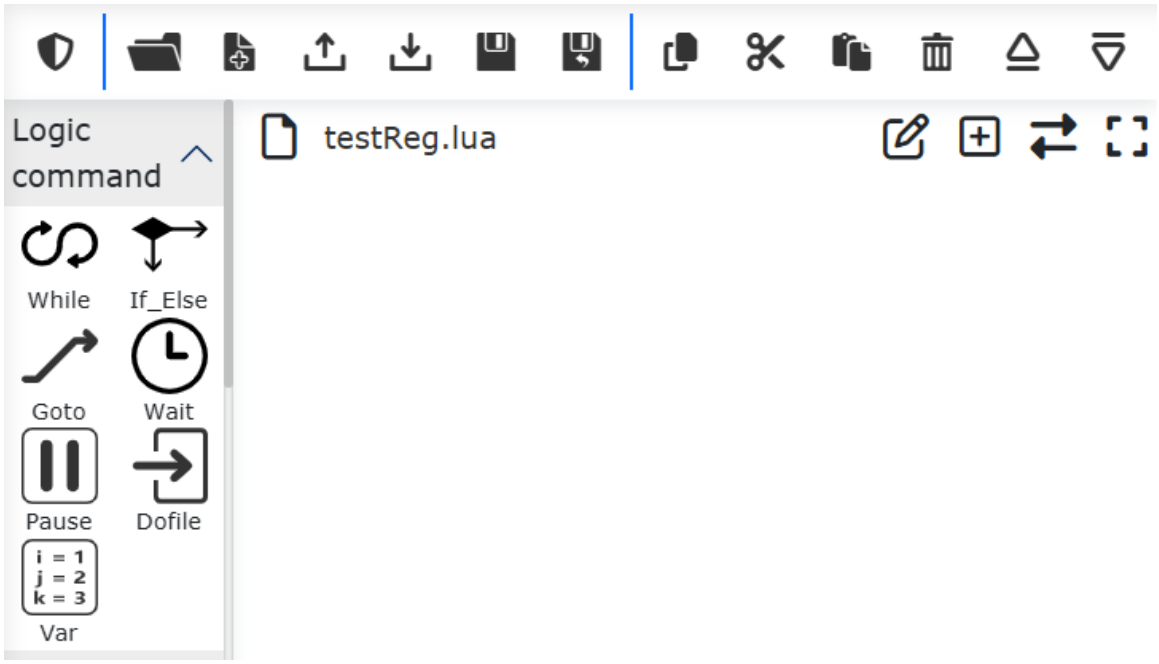


Figure 4-1 New “testReg.lua” program

Click “Var”, select “Input register variable reading” in the command addition box on the right, select variable type as “int”, register start index as 0, and number of register as 3, and click “Add” and “Apply” buttons.

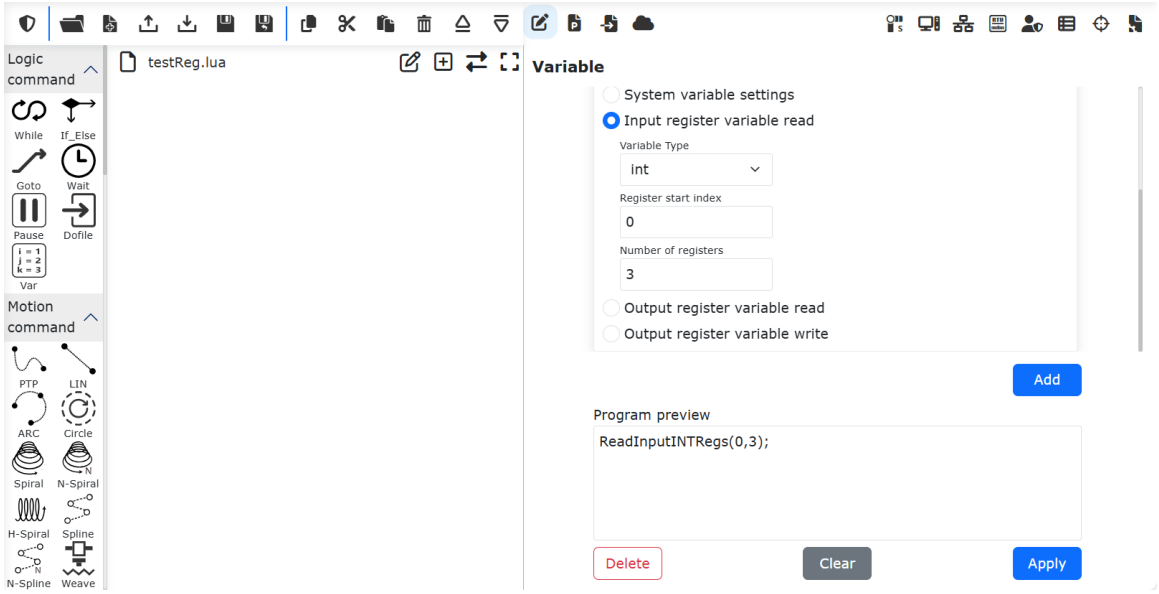


Figure 4-2 Adding a Read Input Register command

At this time, an command to read the “int” input register has been added in “testReg.lua”.

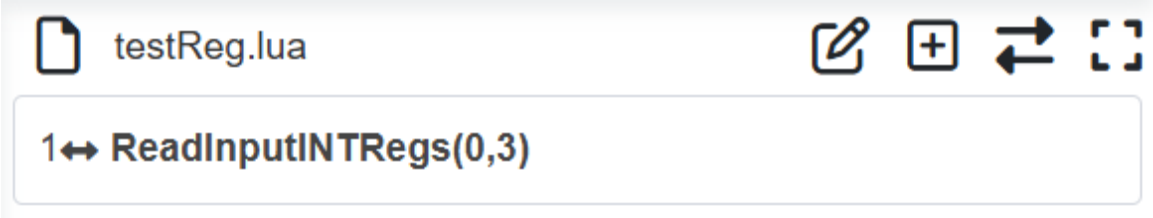


Figure 4-3 Command Addition for Reading “int” Type Input Register

Click the switch mode button to switch to the program editable mode, and add three lua program variables before reading the input register command to receive the read three input register values.

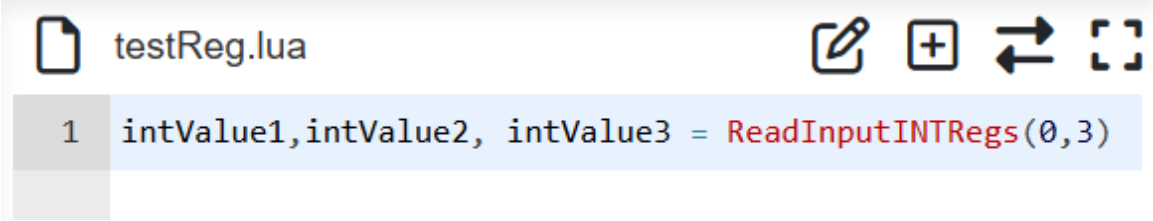


Figure 4-4 Adding Reading Input Register Value

In the same way, you can add “bit” and “double” register data reading respectively.

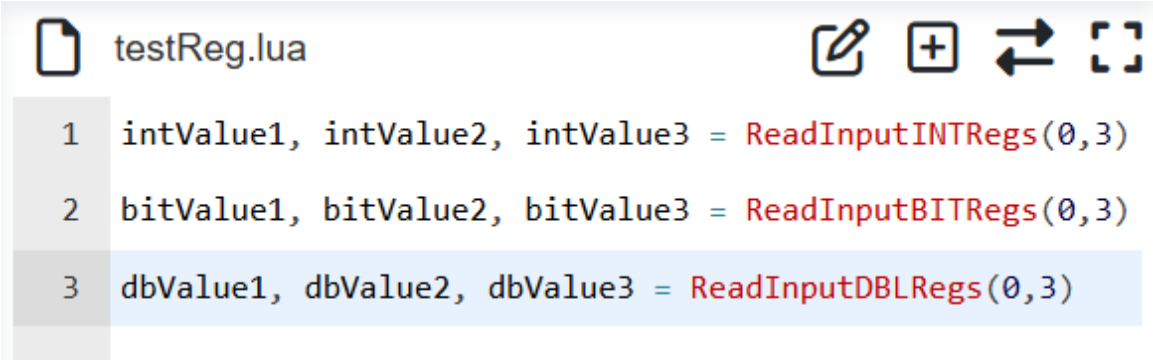


Figure 4-5 Add “bit” and “double” input register to read

Save the above program, switch the robot to automatic mode, execute the program, and the input register value will be read into the lua program variable.

6.4.2 Write output register

Open the WebApp, click “Program” and “Coding” in turn to create a new user program “testReg.lua”.

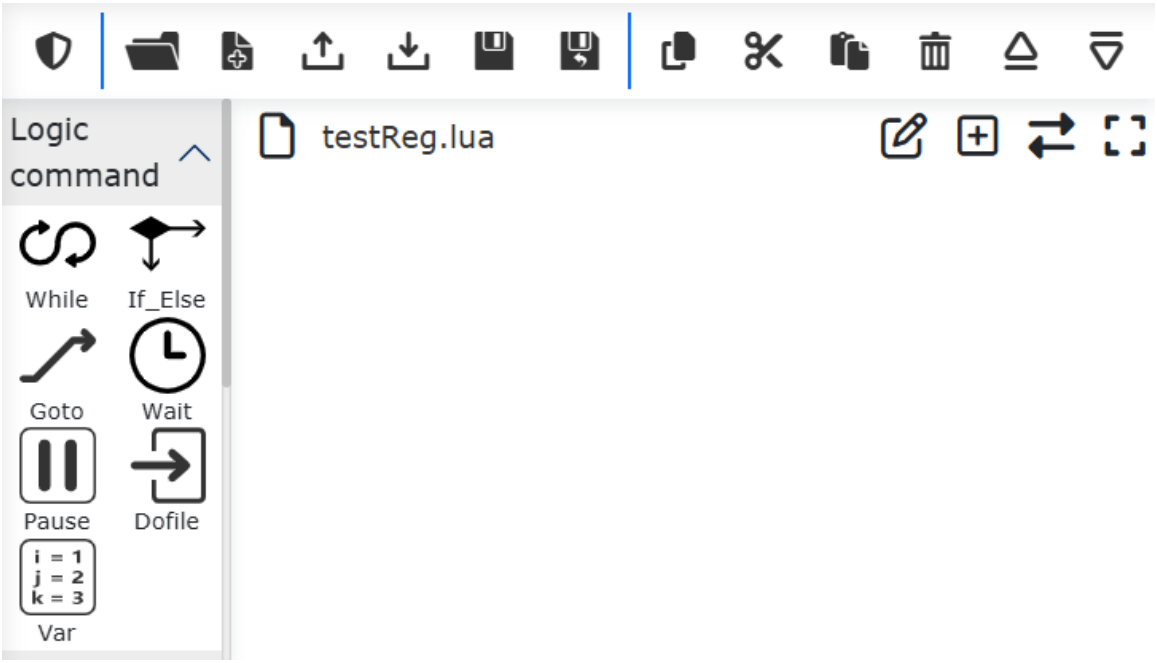


Figure 4-6 New “testReg.lua” program

Click “Var”, select “Output Register Variable write ” in the command add box on the right, select variable type as “int”, register start index as 0, number of registers as 2 and register value as “18,55”, and click “Add” button; Select “Output register variable read” again, select variable type as “int”, register start index as 0 and number of register as 2, and click “Add” and “Apply” buttons.

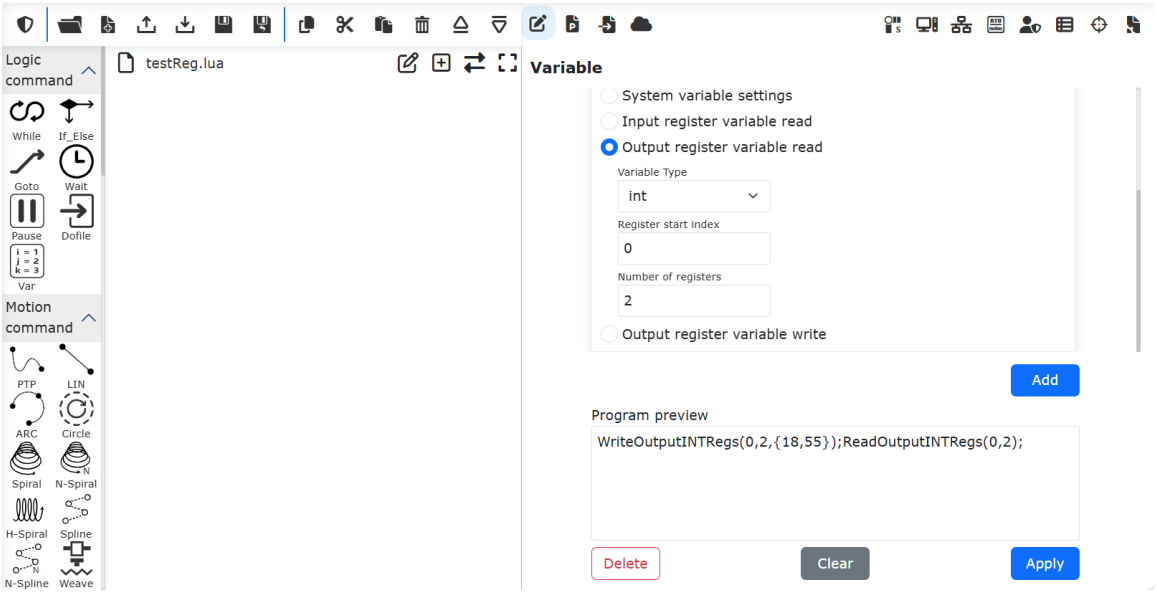


Figure 4-7 Add Read-Write Output Register command

At this time, the “int” output register writing and reading commands have been added in “testReg.lua”.

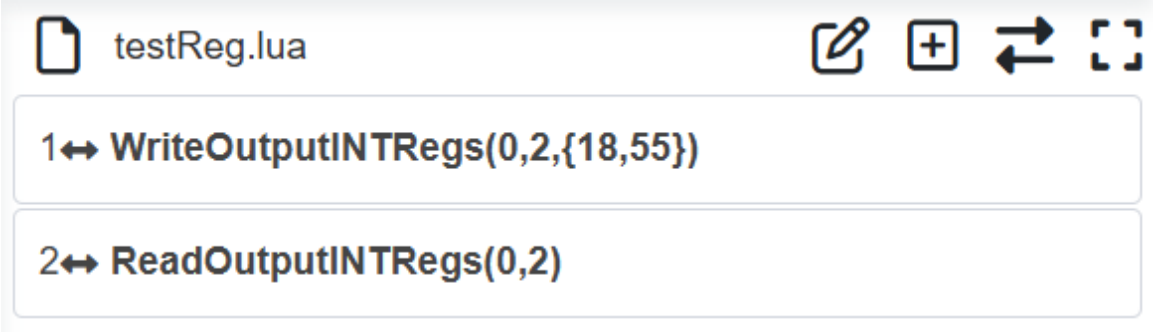


Figure 4-8 Addition of Write and Read commands for “INT” Type Output Register

Click the switch mode button to switch to the program editable mode, and add two lua program variables before reading the output register command to receive the read two output register values.

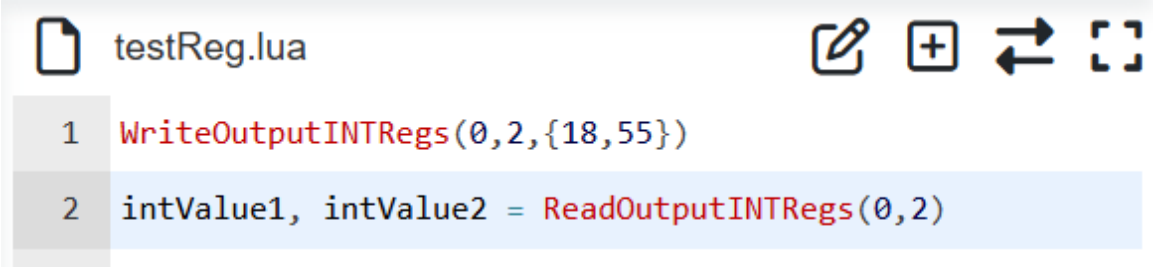


Figure 4-9 Adding Reading Input Register Value

Save the above program, switch the robot to automatic mode and execute the program. At this time, the values of LUA program variables “intValue1” and “intValue2” are 18 and 55 respectively. The operation of “bit” and “double” registers is the same as that of “int” registers.

6.4.3 CNDE input,output register interactive application

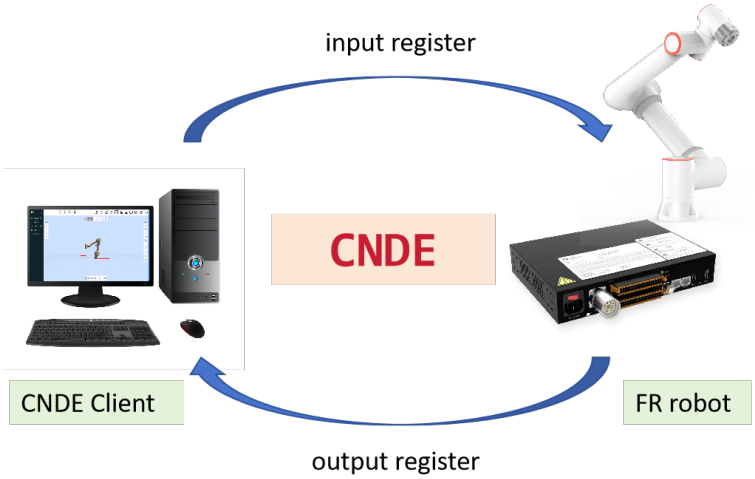


Figure 4-10 Data interaction between input and output registers

Data interaction scenarios between the robot and the CNDE client through input and output registers include but are not limited to the following types:

Input register to control robot movement; The CNDE client plans the robot target position and writes the robot target position into the input register; Read the input register value in the robot LUA program to obtain the robot target position, and then control the robot to move to the target position through PTP, LIN, ServoJ and other motion commands. The LUA example program is as follows:

```
i = 0;
oldFlag = 0
while(1) do
    startFlag = ReadInputINTRegs(0,1)
    if(startFlag ~= oldFlag) then
    oldFlag = startFlag
    x, y, z, a, b, c = ReadInputDBLRegs(0,6)
    ServoJ({x, y, z, a, b, c}, {0, 0, 0, 0}, 10, 10, 0.008, 0, 0)
    end
end
```

The CNDE client writes different values to an input register, and then controls the robot to perform different actions. The robot LUA program cycles to obtain the corresponding input register values, and performs different actions according to the different register values. The example program is as follows:

```
runFlag = ReadInputINTRegs(0,1)
while(runFlag > 0) do
    motion,target = ReadInputINTRegs(1,2)
    if(motion > 0) then
        if(target == 1)then
            Lin(a1,100,-1,0,0)
        else if(target == 2) then
            Lin(a2,100,-1,0,0)
        else
            Lin(safety,100,-1,0,0)
        end
        end
    else
        sleep_ms(100)
    end
end
```

The robot writes the current program status to the output register during operation, and the CNDE client realizes the monitoring of the robot program operation by reading the output register status. The example program is as follows:

```
local weldCount = 0
runFlag = ReadInputINTRegs(0,1)
while(runFlag > 0) do
    Lin(safety,100,-1,0,0)
    Lin(a1,100,-1,0,0)
    ARCStart(0, 0, 3000)
    Lin(a2,100,-1,0,0)
    ARCEnd(0, 0, 3000)
    runFlag = ReadInputINTRegs(0,1)
    weldCount = weldCount + 1
    WriteOutputINTRegs(0,1,{weldCount})
end
```

CHAPTER

SEVEN

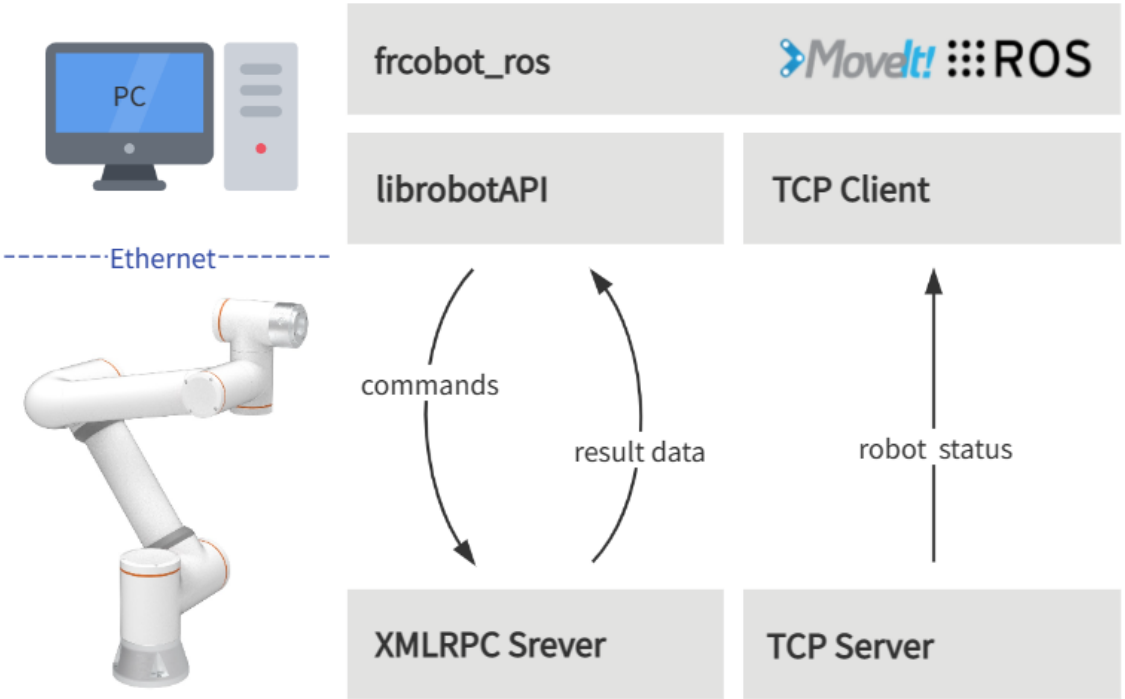
FRCOBOT_ROS

7.1 Overview

The brief architecture of frcobot_ros is shown in the figure below. The collaborative robot side provides an XMLRPC server and a TCP server.

- The XMLRPC server mainly provides the robot command API to complete the robot movement and state value acquisition function, which is mainly based on the C++ SDK.
- The TCP server of the state feedback provides real-time feedback of the state of the robot, and the feedback period is 8ms.

ROS and Moveit! have been installed on the user's PC, and frcobot_ros has been compiled. Each function package in frcobot_ros includes the lib library of the robot API, and establishes a TCP client in frcobot_hw to communicate with the robot status feedback server to obtain robot status feedback data.



7.2 Install

This chapter introduces how to build frcobot_ros and the required installation environment.

7.2.1 Environmental requirements

The recommended environment for frcobot_ros is as follows:

Note:

- Ubuntu 18.04 LTS Bionic Beaver & ROS Melodic Morenia
- Ubuntu 20.04 LTS Focal Fossa & ROS Noetic Ninjemys

The instructions below are for Ubuntu 20.04 LTS systems and ROS Noetic Ninjemys. If you are using Melodic, replace `noetic` in the command line with `melodic`.

7.2.2 ROS installation & requirements

After installing the Ubuntu system, Install and configure the ROS Noetic environment.

After configuring ROS Noetic, install the required environment as follows:

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt-get install -y \
    ros-noetic-rosparam-shortcuts \
    ros-noetic-ros-control \
    ros-noetic-ros-controllers \
    ros-noetic-moveit \
    libxmlrpcpp-dev
```

7.2.3 Compile

After ROS Noetic is properly installed and configured, create a Catkin workspace in a directory of your choice.

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin_init_workspace src
```

Then clone the frcobot_ros library from Github.

```
cd src
git clone https://github.com/FAIR-INNOVATION/frcobot_ros.git
```

Build the frcobot_ros package

```
cd ~/catkin_ws
catkin_make
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

If an error occurs, please check whether the packages in the ROS installation requirements have been installed successfully. After the compilation is complete, copy the lib library to the ROS lib environment (the path is: /opt/ros/noetic/lib), so that the program can run normally .

```
# The default path of catkin_ws here is "~", if it is different, just change "~" to the actual path
sudo cp ~/catkin_ws/src/frcobot_ros/frcobot_hw/lib/* /opt/ros/noetic/lib
```

7.3 Quick start

7.3.1 frcobot_hw

`frcobot_hw` mainly provides basic functions for communicating with collaborative robots.

Note:

- Contains the collaborative robot status feedback msg
- Provide command demos for controlling collaborative robots
- Provide collaborative robot status feedback nodes and topics
- The status node and command demo can be quickly started through the launch file

---

The content of `frcobot_hw.launch` is as follows:

```
<launch>

    <!-- params -->
    <param name="robot_ip" type="string" value="192.168.58.2"/>
    <param name="robot_port" type="int" value="8083"/>

    <!-- frcobot status node -->
    <node pkg="frcobot_hw" type="frcobot_status_node" name="frcobot_status_node" output="screen" />

    <!-- frcobot control demo -->
    <node pkg="frcobot_hw" type="frcobot_cmd_demo" name="frcobot_cmd_demo" output="screen" />

</launch>
```

Important:

- `robot_ip` and `robot_port` need to be consistent with the IP and port of the controlled collaborative robot
- The default IP of the factory robot is 192.168.58.2, and the user status feedback port is 8083

Use the following commands to quickly start the robot status feedback node and command demo functions.

```
roslaunch frcobot_hw frcobot_hw.launch
```

Open a new terminal, and use the following commands to print and view real-time status feedback data.

```
rostopic ehco /frcobot_status
```

CHAPTER

EIGHT

FRCOBOT_ROS2

8.1 Introduction

fairino_hardware is an API interface developed based on ROS2 for FaAO collaborative robot, aiming at making it more convenient for entry-level users to use FaAO SDK. Through the parameter configuration file to configure the default parameters, it can adapt to different customer requirements.

8.2 fairino_hardware

This section explains how to configure the APP environment.

8.2.1 Basic Environment Installation

It is recommended to use Ubuntu22.04LTS(Jammy). Once the system is installed, you can install ROS2. Ros2-humble is recommended:https://docs.ros.org/en/humble/index.htmlBefore compiling fairino_hardware, you need to install the official ros2_control package. For ros2_control installation, see the tutorial:https://control.ros.org/humble/index.htmlThere are two official ros2_control installation modes, namely command installation mode and source code compilation installation mode. Because command installation mode may lead to incomplete installation of function package, it is recommended to use source code compilation installation mode.

The detail progress of installation of ROS2(humble)

1.Open a shell window,set locale information

```
locale  # check for UTF-8

sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale  # verify settings
```

2.Set source

```
sudo apt install software-properties-common
sudo add-apt-repository universe

sudo apt update && sudo apt install curl -y
```

(continues on next page)

(continued from previous page)

```
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/
↪share/keyrings/ros-archive-keyring.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
↪keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_
↪CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

3.Install ROS2

```
sudo apt update
sudo apt upgrade
sudo apt install ros-humble-desktop
```

4.Then install dev-tool

```
sudo apt install ros-dev-tools
```

The detail installation process of ros2_control:

1.First the ROS2 source must be added

```
source /opt/ros/humble/setup.bash
```

2.Create ros2_control workspace, download the source code

```
mkdir -p ~/ros2_control_ws/src
cd ~/ros2_control_ws/
wget https://raw.githubusercontent.com/ros-controls/ros2_control_ci/master/ros_controls.
↪$ROS_DISTRO.repos
vcs import src < ros_controls.$ROS_DISTRO.repos
```

3.Install dependences

```
rosdep update --rosdistro=$ROS_DISTRO
sudo apt-get update
rosdep install --from-paths src --ignore-src -r -y
```

4.Build ros2_control

```
. /opt/ros/${ROS_DISTRO}/setup.sh
colcon build --symlink-install
```

8.2.2 Compile and build fairino_hardware

1. Create the colcon workspace fairino_hardware consists of two packages: fairino_msgs for custom data structures and fairino_hardware for the program body. Once you have the base environment installed, create a colcon workspace, like this:

Rmenber source ROS2&ros2_control first

```
source /opt/ros/humble/setup.bash
source ~/ros2_control_ws/install/setup.bash
```

```
cd ~/
mkdir -p ros2_ws/src
```

2. Compile feature packs Copy the package code to ros2_ws/src and run the following command inside ros2_ws:

```
colcon build --packages-select fairino_msgs
```

Wait for the previous command to finish compiling, then start to build fairino_hardware package:

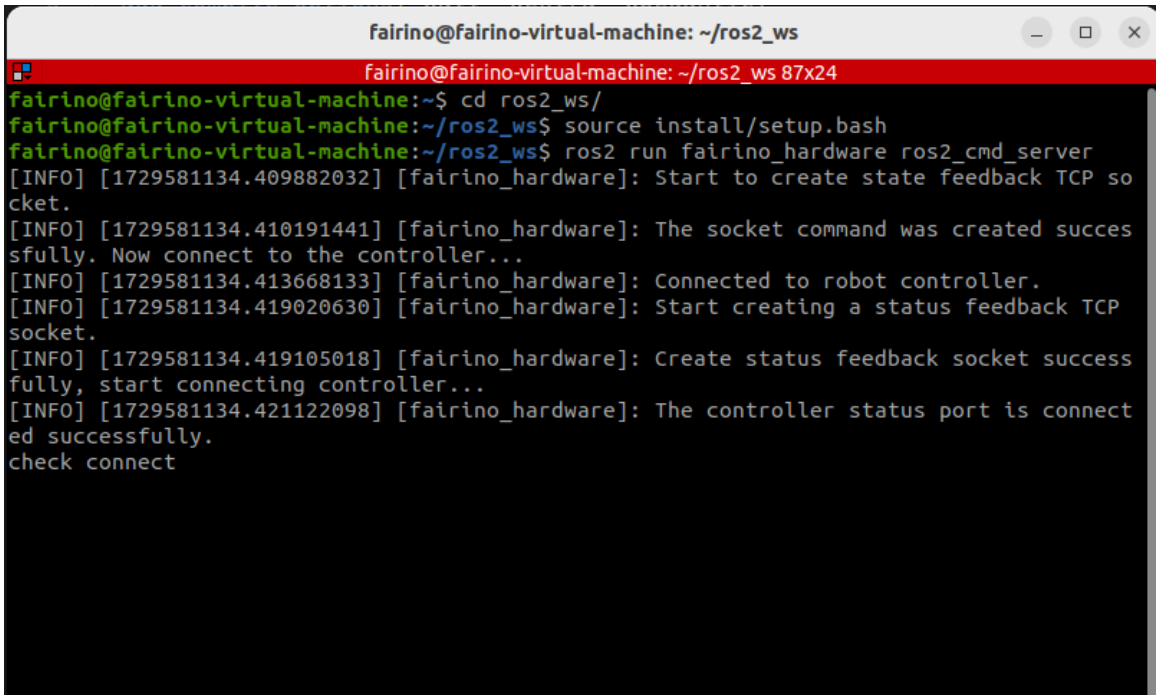
```
colcon build --packages-select fairino_hardware
```

8.3 Quick start

8.3.1 Starting the process

On Ubuntu, open the command line and type:

```
cd ros2_ws
source install/setup.bash
ros2 run fairino_hardware ros2_cmd_server
```



8.3.2 View the manipulator state feedback process

The state feedback of the manipulator is published through the topic. Users can observe the state data refresh through the commands that come with ros2, or write programs to obtain the data.

On ubuntu, open the command line and type:

```
cd ros2_ws
source install/setup.bash
ros2 topic echo /nonrt_state_data
```

Can see the status data being refreshed in the command-line window, as shown in the following screenshot:

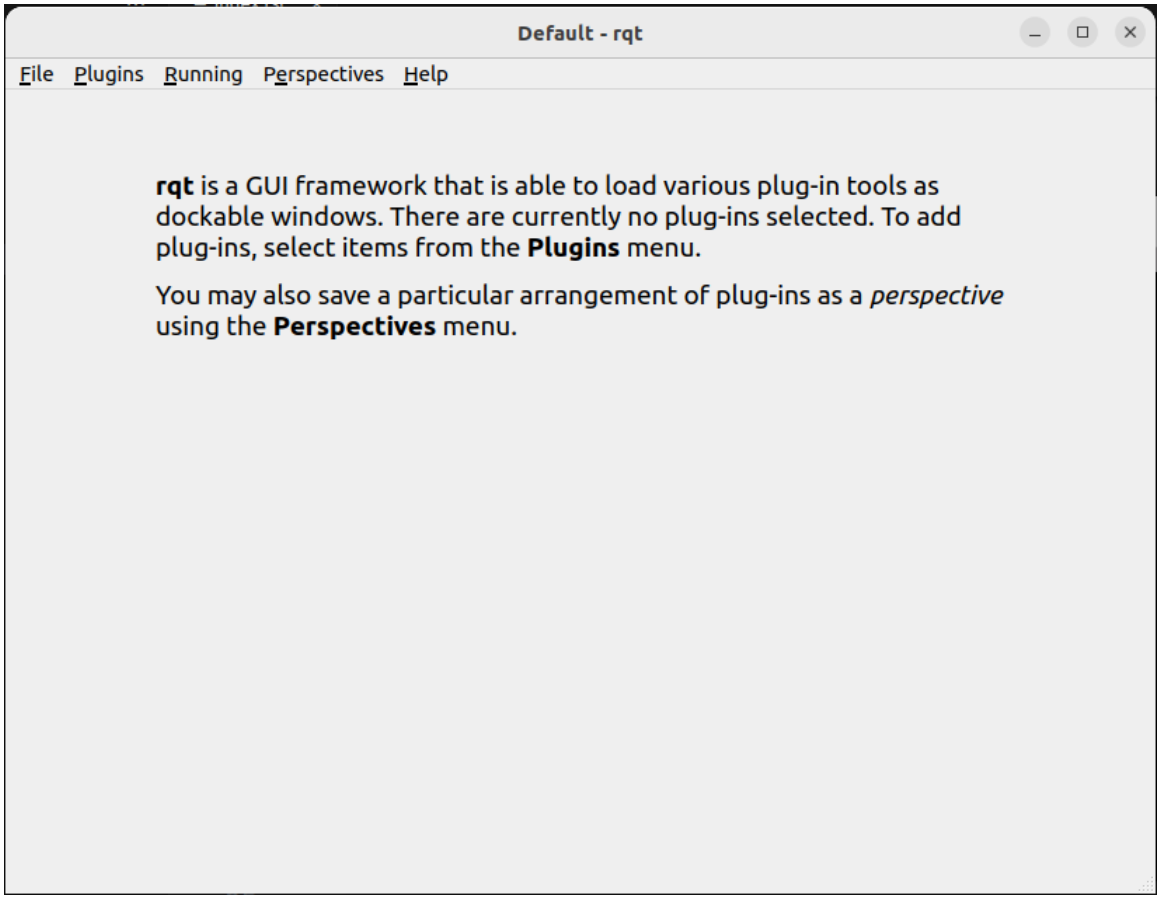


8.3.3 Process for issuing instructions

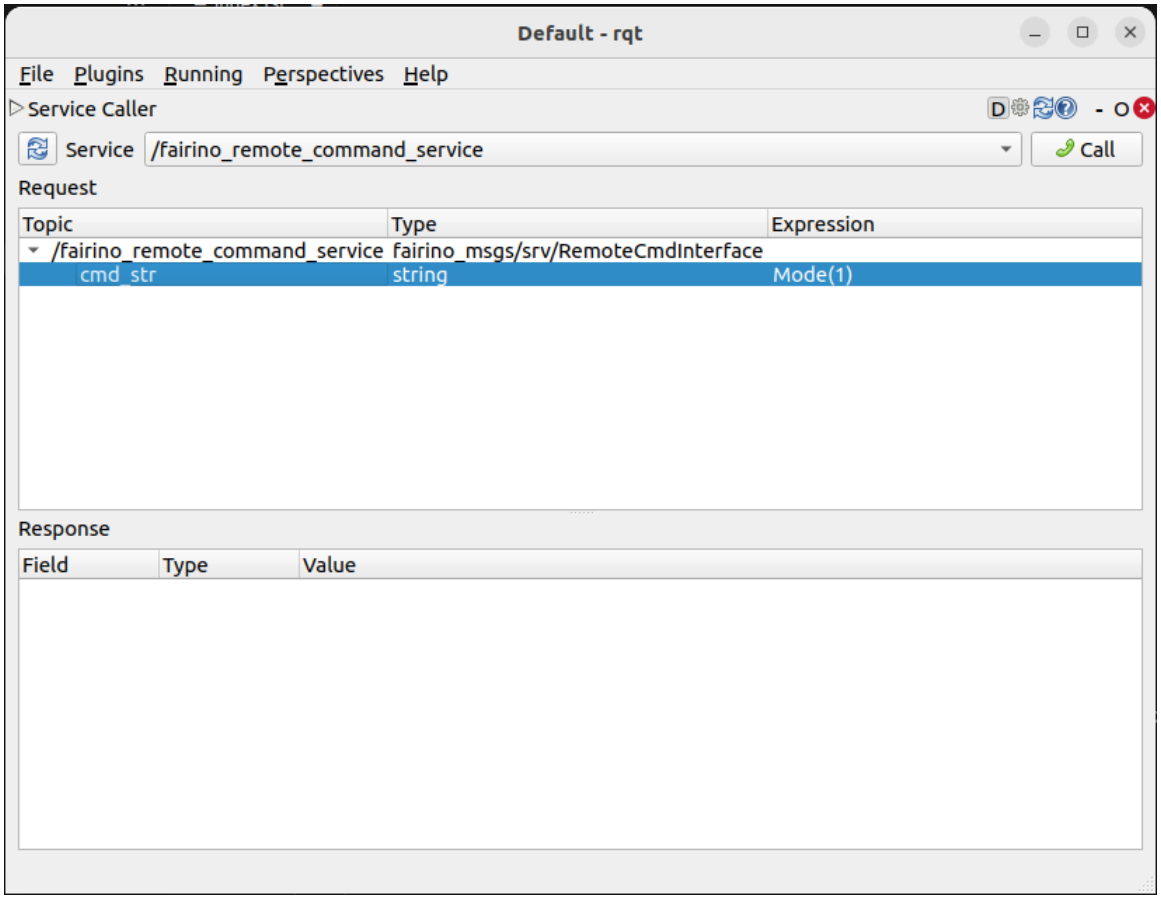
On ubuntu, open the command line and type:

```
cd ros2_ws
source install/setup.bash
rqt
```

After executing the above command, an rqt GUI will be brought up, as shown in the following figure.



In the GUI, select plugins->serivce->serivce caller, bring up the following screen, select /fairino_remote_command_service, Enter the instruction string in the interface expression and click call to see the reply message pop up in the dialog box below.



Important:

- Input string rule description:

The program internally filters the input strings to be of the form [function name](), and the parenthesis argument string must consist of letters, numbers, commas, and minus signs; any other characters or Spaces will throw an error.

- Instruction feedback value description:

Except for the GET command, which returns a string, the rest of the function feedback values are int values, typically 0 for error, 1 for correct execution, and any other values refer to the error code defined in the xmlrpc SDK.

8.3.4 Modifying the parameter flow

Since the simplified SDK is to improve the native SDK interface, it can be simplified because some parameters are given default values. However, in the actual use process, the default parameters can not meet the requirements, in this case, you can modify the value of the corresponding default parameters and then load them into the node.

There is a fairino_remotecmdinterface_para.yaml parameter file in the source code file. The parameters in the file are preset default parameters, which are used to simplify the instruction input parameters. You can modify the parameters according to your specific needs, and then use the command to modify the parameters dynamically: ros2 param load fr_command_server ~/ros2_ws/src/fairino_hardware/fairino_remotecmdinterface_para.yaml

8.4 API Description

```
/*
Function description: Store a joint point position information
id - Stores the point id, starting with 1, independently of the point id of CARTPoint
double j1j6  6 joint positions in degrees
*/
int JNTPoint(int id, double j1, double j2, double j3, double j4, double j5, double j6)
// Examples
JNTPoint(1,10,11,12,13,14,15)

/*
Function description: Store a Cartesian point position information
id - Stores the point id, starting from 1, independently of the point id of JNTPoint
double x,y,z,rx,ry,yz - Cartesian point position information, position in mm, Angle in degrees
*/
int CARTPoint(int id, double x,y,z,rx,ry,rz)//Store a point in Cartesian space
// Examples
CARTPoint(1,100,110,200,0,0,0)

/*
Function description: Get the joint or Cartesian position information of the specified sequence point
string name - 'JNT' or 'CART', where JNT stands for getting information about joint points and 'CART' stands for getting information about Cartesian points
int id - The point id, starting at 1
*/
string GET(string name, int id)//Get the contents of the corresponding id sequence number point; name can be entered as JNT or CART
// Examples
GET(JNT,1)

/*
Function description: Drag mode switch
uint8_t state - 1- Enable drag mode,0- disable drag mode
*/
int DragTeachSwitch(uint8_t state)
// Examples
DragTeachSwitch(0)

/*
Function description: Manipulator enable switch
uint8_t state - 1 -manipulator enabled,0 -manipulator deenabled
*/
int RobotEnable(uint8_t state)
// Examples
RobotEnable(1)

/*
Function description:Mode switching
uint8_t state - 1- Manual mode,0- automatic mode
```

(continues on next page)

(continued from previous page)

```
*/
int Mode(uint8_t state)
// Examples
Mode(1)

/*
Function description:Set the manipulator speed in the current mode
float vel - Percentage of speed, ranging from 1-100
*/
int SetSpeed(float vel)
// Examples
SetSpeed(10)

/*
Function description:Sets and loads the tool coordinate system with the specified␣
↪sequence number
int id - Tool coordinate system number, range 1-15
float x,y,z,rx,ry,rz - Offset information for the tool coordinate system
*/
int SetToolCoord(int id, float x,float y, float z,float rx,float ry,float rz)
// Examples
SetToolCoord(1,0,0,0,0,0,0)

/*
Function description:Set the list of tool coordinate systems
int id - Tool coordinate system number, range 1-15
float x,y,z,rx,ry,rz - Offset information for the tool coordinate system
*/
int SetToolList(int id, float x,float y, float z,float rx,float ry,float rz );
// Examples
SetToolList(1,0,0,0,0,0,0)

/*
Function description:Set the external tool coordinate system
int id - Tool coordinate system number, range 1-15
float x,y,z,rx,ry,rz - Offset information for the external tool coordinate system
*/
int SetExToolCoord(int id, float x,float y, float z,float rx,float ry,float rz);
// Examples
SetExToolCoord(1,0,0,0,0,0,0)

/*
Function description:Sets the list of external tool coordinate systems
int id - Tool coordinate system number, range 1-15
float x,y,z,rx,ry,rz - Offset information for the external tool coordinate system
*/
int SetExToolList(int id, float x,float y, float z,float rx,float ry,float rz);
// Examples
SetExToolList(1,0,0,0,0,0,0)

/*
Function description:Set the workpiece coordinate system
```

(continues on next page)

(continued from previous page)

```
int id - Workpiece coordinate system number, range 1-15
float x,y,z,rx,ry,rz - Offset information of the workpiece coordinate system
*/
int SetWObjCoord(int id, float x,float y, float z,float rx,float ry,float rz);
// Examples
SetWObjCoord(1,0,0,0,0,0,0)

/*
Function description:Set the list of workpiece coordinate systems
int id - Workpiece coordinate system number, range 1-15
float x,y,z,rx,ry,rz - Offset information of the workpiece coordinate system
*/
int SetWObjList(int id, float x,float y, float z,float rx,float ry,float rz);
// Examples
SetWObjList(1,0,0,0,0,0,0)

/*
Function description:Set the end load weight
float weight - Load weight in kg
*/
int SetLoadWeight(float weight);
// Examples
SetLoadWeight(3.5)

/*
Function description:Set the end-load centroid coordinates
float x,y,z - Coordinates of the center of mass in mm
*/
int SetLoadCoord(float x,float y,float z);
// Examples
SetLoadCoord(10,20,30)

/*
Function description:Set the robot installation mode
uint8_t install - Installation mode,0- formal,1- side,2- inverted
*/
int SetRobotInstallPos(uint8_t install);
// Examples
SetRobotInstallPos(0)

/*
Function description:Set the robot installation Angle, free installation
double yangle - Angle of inclination
double zangle - Angle of rotation
*/
int SetRobotInstallAngle(double yangle,double zangle);
// Examples
SetRobotInstallAngle(90,0)


//Security configuration
/*
```

(continues on next page)

(continued from previous page)

```
Function description:Set the robot collision level
float level1-level6 - Collision levels for axes 1-6, ranging from 1-10
*/
int SetAnticollision(float level1, float level2, float level3, float level4, float␣
↪level5, folat level6);
// Examples
SetAnticollision(1,1,1,1,1,1)

/*
     * @brief  Set the post-collision strategy
     * @param  [in] strategy  0-Error and stop, 1-Continue running
     * @param  [in] safeTime  Safe stop time [1000 - 2000] ms
     * @param  [in] safeDistance  Safe stop distance [1-150] mm
     * @param  [in] safeVel  Safe speed [50-250] mm/s
     * @param  [in] safetyMargin  Safety factor for J1-J6 [1-10]
     * @return  Error code
*/
    int SetCollisionStrategy(int strategy, int safeTime, int safeDistance, int safeVel,␣
↪int safetyMargin[]);
// Example
SetCollisionStrategy(1);

/**
* @brief sets the collision detection method of the robot
* @param [in] method Collision detection method: 0- current mode; 1- Dual encoder; 2-␣
↪Current and dual encoder turn on simultaneously
* @param [in] thresholdMode Collision level threshold method 0-Collision level fixed␣
↪threshold mode 1- Customize collision detection thresholds
* @return  error code
*/
int SetCollisionDetectionMethod(int method, int thresholdMode);
// Examples
SetCollisionDetectionMethod(0,0)

/**
* @brief Indicates that collision detection is disabled in static mode
* @param [in] status 0- Off; 1- Open
* @return  error code
*/
int SetStaticCollisionOnOff(int status);
// Examples
SetStaticCollisionOnOff(1)

/**
    * @brief joint torque power detection
    * @param [in] status 0- Off; 1- Open
    * @param [in] power Set maximum power (W);
* @return  error code
*/
    int SetPowerLimit(int status, double power);
//Examples
SetPowerLimit(1,100)
```

(continues on next page)

(continued from previous page)

```
198
199     /**
200 *@brief  Configured force sensor
201 *@param  [in] company  Manufacturer of force sensors, 17-KUNWEI19-CAAA20-ATI21-HKM22-
    ↪GZCX23-NBIT24-XJC26-NSR
202 *@param  [in] device  Device number,  KUNWEI(0-KWR75B)CAAA(0-MCS6A-200-4)ATI(0-AXIA80-
    ↪M8)HKM(0-MST2010)GZCX(0-WHC6L-YB-10A)NBIT(0-XLH93003ACS)XJC(0-XJC-6F-D82)NSR(0-NSR-
    ↪FTSensorA)
203 *@param  [in] softvesion  Software version. The value is not used. The default value is 0
204 *@param  [in] bus The device is attached to the terminal bus and is not in use. The␣
    ↪default value is 0
205 *@return  Error code
206      */
207     int FT_SetConfig(int company, int device, int softvesion, int bus);
208 // Examples
209 FT_SetConfig(0,1,0,0)
210
211     /**
212 *@brief  Get the force sensor configuration
213 *@param  [in] company  Force sensor manufacturer, to be determined
214 *@param  [in] device  Device number, not used yet. The default value is 0
215 *@param  [in] softvesion  Software version. The value is not used. The default value is 0
216 *@param  [in] bus The device is attached to the terminal bus and is not in use. The␣
    ↪default value is 0
217 *@return  Error code
218      */
219     int FT_GetConfig(int *company, int *device, int *softvesion, int *bus);
220 // Examples
221 FT_GetConfig()
222
223     /**
224 *@brief  Force sensor activation
225 *@param  [in] act  0- reset, 1- activate
226 *@return  Error code
227      */
228     int FT_Activate(uint8_t act);
229 // Examples
230 FT_Activate(1)
231
232     /**
233 *@brief  Force sensor calibration
234 *@param  [in] act  0- zero removal, 1- zero correction
235 *@return  Error code
236      */
237     int  FT_SetZero(uint8_t act);
238 // Examples
239 FT_SetZero(1)
240
241     /**
242 *@brief  Collision guard
243 *@param  [in] flag 0- Disable collision guard. 1- Enable collision guard
244 *@param  [in] sensor_id Force sensor number
```

(continues on next page)

(continued from previous page)

```
*@param  [in] select  Select the six degrees of freedom whether to detect collision, 0-no detection, 1- detection
*@param  [in] ft  Impact force/torquefx,fy,fz,tx,ty,tz
*@param  [in] max_threshold Maximum threshold
*@param  [in] min_threshold Minimum threshold
*@note   Force/torque detection range(ft-min_threshold, ft+max_threshold)
*@return  Error code
     */
    int FT_Guard(uint8_t flag, int sensor_id, uint8_t select[6], ForceTorque *ft, float max_threshold[6], float min_threshold[6]);
// Examples
FT_Guard(1,1,0,0,1,0,0,0,0,0,100,0,0,0,0,0,200,0,0,0,0,0,50,0,0,0)


    /**
*@brief  Constant force control
*@param  [in] flag 0- turn off constant force control, 1- turn on constant force control
*@param  [in] sensor_id Force sensor number
*@param  [in] select  Select the six degrees of freedom whether to detect collision, 0-no detection, 1- detection
*@param  [in] ft  Impact force/torquefx,fy,fz,tx,ty,tz
*@param  [in] ft_pid Force pid parameter, torque pid parameter
*@param  [in] adj_sign Adaptive start-stop control, 0- off, 1- on
*@param  [in] ILC_sign ILC start stop control, 0- stop, 1- training, 2- operation
*@param  [in] Maximum Adjustment distance, unit: mm
*@param  [in] Maximum Adjustment Angle, unit: deg
*@return  Error code
     */
    int FT_Control(uint8_t flag, int sensor_id, uint8_t select[6], ForceTorque *ft, float ft_pid[6], uint8_t adj_sign, uint8_t ILC_sign, float max_dis, float max_ang, int filter_Sign = 0, int posAdapt_sign = 0, int isNoBlock = 0);
// Examples
FT_Control(1,1,0,0,1,0,0,0,0,0,-10,0,0,0,0.0005,0,0,0,0,0,0,0,100,10,0,0,0)


    /**
*@brief  Compliant control on
*@param  [in] p Coefficient of position adjustment or compliance
*@param  [in] force Compliant opening force threshold, unit: N
*@return  Error code
     */
    int FT_ComplianceStart(float p, float force);
// Examples
FT_ComplianceStart(0.005,20)

    /**
*@brief  Compliant control off
*@return  Error code
     */
    int  FT_ComplianceStop();
// Examples
FT_ComplianceStop()
```

(continues on next page)

(continued from previous page)

```

/*
Function description:Set the positive limit, note that the set value must be within the hard limit range
float limit1-limit6 - Six joint limit values
*/
int SetLimitPositive(float limit1, float limit2, float limit3, float limit4, float limit5, float limit6);
// Examples
SetLimitPositve(100,90,90,90,90,90)

/*
Function description:Set the negative limit, note that the set value must be within the hard limit range
float limit1-limit6 - Six joint limit values
*/
int SetLimitNegative(float limit1, float limit2, float limit3, float limit4, float limit5, float limit6);
// Examples
SetLimitNegative(-100,-90,-90,-90,-90,-90)

/*
Function description:Error state removal
*/
int ResetAllError();

/*
Function description:Joint friction compensation switch
uint8_t state - 0- off, 1- on
*/
int FrictionCompensationOnOff(uint8_t state);
// Examples
FrictionCompensationOnOff(1)

/*
Function description:Set the joint friction compensation coefficient -formal suit
float coeff1-coeff6 - Six joint compensation coefficients, ranging from 0-1
*/
int SetFrictionValue_level(float coeff1,float coeff1,float coeff3,float coeff4,float coeff5,float coeff6);
// Examples
SetFrictionValue_level(1,1,1,1,1,1)

/*
Function description:Set the joint friction compensation coefficient -side loading
float coeff1-coeff6 - Six joint compensation coefficients, ranging from 0-1
*/
int SetFrictionValue_wall(float coeff1,float coeff1,float coeff3,float coeff4,float coeff5,float coeff6);
// Examples
SetFrictionValue_wall(0.5,0.5,0.5,0.5,0.5,0.5)
```

(continues on next page)

(continued from previous page)

```

/*
Function description:Set the joint friction compensation coefficient -flip
float coeff1-coeff6 - Six joint compensation coefficients, ranging from 0-1
*/
int SetFrictionValue_ceiling(float coeff1,float coeff1,float coeff3,float coeff4,float␣
↪coeff5,float coeff6);
// Examples
SetFrictionValue_ceiling(0.5,0.5,0.5,0.5,0.5,0.5)


//Peripheral device control
/*
Function description:Activated gripper
int index - Clamp claw number
uint8_t act - 0- Reset, 1- Activate
*/
int ActGripper(int index,uint8_t act);
// Examples
ActGripper(1,1)

/*
Function description:Control gripper
int index - Clamp claw number
int pos - Percentage of position, range 0-100
*/
int MoveGripper(int index,int pos);
// Examples
MoveGripper(1,10)


//IO
/*
Function description:Set the control box digital output
int id - io number, range 0-15
uint_t status - 0- off, 1- on
*/
int SetDO(int id,uint8_t status);
// Examples
SetDO(1,1)

/*
Function description:Set tool number output
int id - io number, range 0-1
uint_t status - 0- off, 1- on
*/
int SetToolDO(int id,uint8_t status);
// Examples
SetToolDO(0,1)

/*
Function description:Set the control box analog output
```

(continues on next page)

(continued from previous page)

```
int id - io number, range 0-1
float vlaue - Percentage of current or voltage value, range 0-100
*/
int SetAO(int id,float value);
// Examples
SetAO(1,100)

/*
Function description:Set tool analog output
int id - io number, range 0
float vlaue - Percentage of current or voltage value, range 0-100
*/
int SetToolAO(int id,float value);
// Examples
SetToolAO(0,100)


//Motor command
/*
Function description:Robot JOG Start
uint8_t ref - 0-joint JOG, 2-JOG in base coordinate system, 4-JOG in tool coordinate␣
↪system, 8-JOG in workpiece  coordinate system
uint8_t nb - 1 - Joint 1 (or X-axis), 2 - Joint 2 (or Y-axis), 3 - Joint 3 (or Z-axis),␣
↪4 - Joint 4 (or Rotation around X-axis), 5 - Joint 5 (or Rotation around Y-axis), 6 -␣
↪Joint 6 (or Rotation around Z-axis)
uint8_t dir - 0- negative direction, 1- positive direction
float vel - Percentage of speed, ranging from 0-100
*/
int StartJOG(uint8_t ref, uin8_t nb, uint8_t dir, float vel);
// Examples
StartJOG(1,1,1,10)

/*
Function description:Robot JOG Stop
uint8_t ref - 0 - Joint Jog Stop, 2 - Jog Stop in Base Coordinate System, 4 - Jog Stop␣
↪in Tool Coordinate System, 8 - Jog Stop in Workpiece Coordinate System
*/
int StopJOG(uint8_t ref);
// Examples
StopJOG(1)

/*
Function description:Robot JOG Immediate Stop
*/
int ImmStopJOG();

/*
Function description:Joint space motion
string point_name - For example, JNT1 is the point with the sequence number 1 of the␣
↪node information,CART1 is the point with the sequence number 1 of the Cartesian point␣
↪information,MoveJ instruction supports the input of the node or Cartesian point. It␣
↪should be noted that because the default parameters of the MoveJ instruction specify␣
```

(continues on next page)

(continued from previous page)

```
    ↪the tool coordinate system and the workpiece coordinate system, when the serial number␣
    ↪of the two coordinate systems is inconsistent with the current load, the instruction␣
    ↪will cause an error. It is necessary to modify the coordinate system parameters and␣
    ↪LOAD parameters in the default parameters before running the movement instruction.
434 float vel - Command speed percentage, range 0-100
435 int tool - tool coordinate index
436 int user - reference coordinate index
437 */
438 int MoveJ(string point_name, float vel,int tool, int user);//point_name indicates the␣
    ↪input prestored point information,
439 // Examples
440 MoveJ(JNT1,10,1,1)
441
442 /*
443 Function description:Rectilinear motion in Cartesian space
444 string point_name - For example, JNT1 is the point whose sequence number is 1,CART1 is␣
    ↪the point whose sequence number is 1, and the MoveL instruction supports the input of␣
    ↪the point or Cartesian point. It should be noted that since the default parameters of␣
    ↪the MoveL instruction specify the tool coordinate system and the workpiece coordinate␣
    ↪system, when the serial number of the two coordinate systems is inconsistent with the␣
    ↪current load, the instruction will cause an error. The coordinate system parameters␣
    ↪and load parameters need to be modified in the default parameters before running the␣
    ↪motion instruction.
445 float vel - Command speed percentage, range 0-100
446 */
447 int MoveL(string point_name,float vel);
448 // Examples
449 MoveL(CART1,10)
450
451 /*
452 Function description:Circular motion in Cartesian space
453 string point1_name point2_name - For example, JNT1 is the point whose sequence number is␣
    ↪1,CART1 is the point whose sequence number is 1. MoveC command supports the input of␣
    ↪the point or Cartesian point, but the two points must be of the same type, that is,␣
    ↪the first point does not support the input of the joint space point and the second␣
    ↪point input of the Cartesian point. It should be noted that because the default␣
    ↪parameters of the MoveC instruction specify the tool coordinate system and the␣
    ↪workpiece coordinate system, when the serial number of the two coordinate systems is␣
    ↪inconsistent with the current load, the instruction will cause an error. It is␣
    ↪necessary to modify the coordinate system parameters and LOAD parameters in the␣
    ↪default parameters before running the movement instruction.
454 float vel - Command speed percentage, range 0-100
455 */
456 int MoveC(string point1_name,string point2_name, float vel);
457 // Examples
458 MoveC(JNT1,JNT2,10)
459
460 /*
461 Function description:The spline movement begins
462 */
463 int SplineStart();
464
```

(continues on next page)

(continued from previous page)

```
/*
Function description:Joint space spline movement, this command only supports the input of joint data such as JNT1, the input of Cartesian point will report an error
string point_name - The prestored point name, such as JNT1, is the point whose sequence number is 1.
float vel - Percentage of speed, range 0-100
*/
int SplinePTP(string point_name, float vel);
// Examples
SplinePTP(JNT2,10)

/*
Function description:The spline movement is over
*/
int SplineEnd();

/*
Function description:The Cartesian space spline motion begins
uint8_t ctlpoint - 0- trajectory passes through the path point, 1- trajectory does not pass through the control point, at least 4 points
*/
int NewSplineStart(uint8_t ctlpoint);
// Examples
NewSplineStrart(1)

/*
Function description:For Cartesian space spline movement, only Cartesian space points such as CART1 can be entered, and an error will be reported when entering joint space points
string point_name - Prestored point names, such as CART1, are points in Cartesian space with sequence number 1.
float vel - Percentage of speed, range 0-100
int lastflag - 0- not the last point, 1- the last point
*/
int NewSplinePoint(string point_name, float vel, int lastflag);
// Examples
NewSplinePoint(JNT2,20,0)

/*
Function description:The Cartesian space spline motion ends
*/
int NewSplineEnd();

/*
Function description:Stop motion
*/
int StopMotion();

/*
Function description:Global shift of point position begins
int flag - 0- Offset in base coordinate/workpiece coordinate, 2- offset in tool coordinate
```

(continues on next page)

(continued from previous page)

```
double x,y,z,rx,ry,rz - Offset pose amount
*/
int PointsOffsetEnable(int flag,double x,double y,double z,double rx,double ry,double
↪rz);
// Examples
PointsOffsetEnable(1,10,10,10,0,0,0)

/*
Function description:The point offset is complete
*/
int PointsOffsetDisable();
```

CHAPTER

NINE

MOVEIT2

9.1 Plug-in Introduction

Fairino MoveIt2 plug-in is a plug-in that provides support for motion control and path planning of Fairino robots. With the help of Fairino MoveIt2 plug-in can realize complex robot motion control, path planning, inverse kinematics solving and real-time collision detection, etc. It is suitable for a variety of robotic arm application scenarios, such as industrial, welding, manufacturing, automated loading and unloading, palletizing, medical and other scenarios.

9.2 Rapid Access

This chapter describes how to configure the APP running environment.

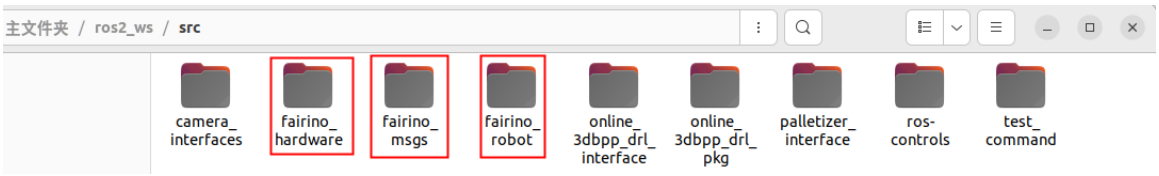
It is recommended to use it on Ubuntu22.04LTS (Jammy). After the system is installed, you can install ROS2. It is recommended to use ros2-humble. For the installation of ROS2, please refer to the tutorial: https://docs.ros.org/en/humble/index.html.

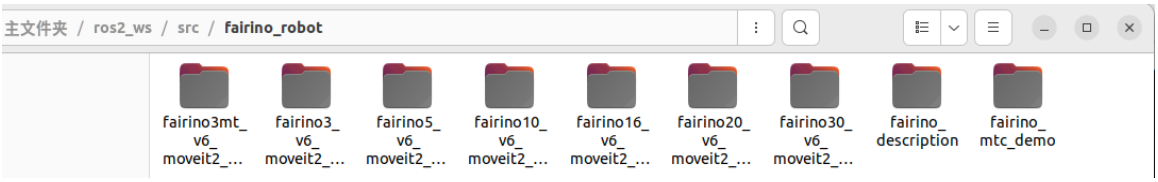
9.2.1 MoveIt2 Installation

9.2.1.1 Clone Fairino MoveIt2 Plugin

Clone the Fairino MoveIt2 plugin locally, and then cd to the target directory, where the main files include fairino_msgs Fairino robot data transfer data type function package; fairino_hardware Fairino robot fairino_hardware plugin function package;

fairino_robot/ fairino_description Fairino robot appearance and urdf file function package; fairino_robot/fairino3mt_v6_moveit2_config, fairino_robot/fairino3_v6_moveit2_config, fairino_robot/fairino5_v6_moveit2_config, fairino_robot/fairino10_v6_moveit2_config, fairino_robot/fairino16_v6_moveit2_config, fairino_robot /fairino20_v6_moveit2_config, fairino_robot/fairino30_v6_moveit2_config Fairino Robotics moveit2 Configuration Package, fairino_robot/fairino_mtc_demo Fairino mtc Sample Code Package.





9.2.1.2 Compile Feature Packs

Compile the fairino_msgs function package.

```
cd ros2_ws
colcon build --packages-select fairino_msgs
source install/setup.bash
```

Compile the fairino_hardware function package.

```
cd ros2_ws
colcon build --packages-select fairino_hardware
source install/setup.bash
```

Compile the fairino_description function package.

```
cd ros2_ws
colcon build --packages-select fairino_description
source install/setup.bash
```

Compile the Fairino robot moveit2 configuration package, using fairino5_v6_moveit2_config as an example.

```
cd ros2_ws
colcon build --packages-select fairino5_v6_moveit2_config
source install/setup.bash
```

Compile Fairino Robotics fairino_mtc_demo sample code package, if the code sample package does not appear in the official ros2_ws workspace, you can contact the after-sales service to get it.

```
cd ros2_ws
colcon build --packages-select fairino_mtc_demo
source install/setup.bash
```

9.2.2 Configuring the Moveit2 model of the Fairino robotic arm

If you do not want to use the official robot moveit2_config package, you can customize the robot moveit2_config package by configuring it with moveit_setup_assistant.

9.2.2.1 Creating A Workspace

Create the workspace and create the feature pack.

```
mkdir -p test_fa_ws/src
cd test_fa_w/src
mkdir fairino5_v6_robot_moveit_config
cd ..
cd ..
```

Compile the functional package and source.

```
colcon build
source install/setup.bash
```

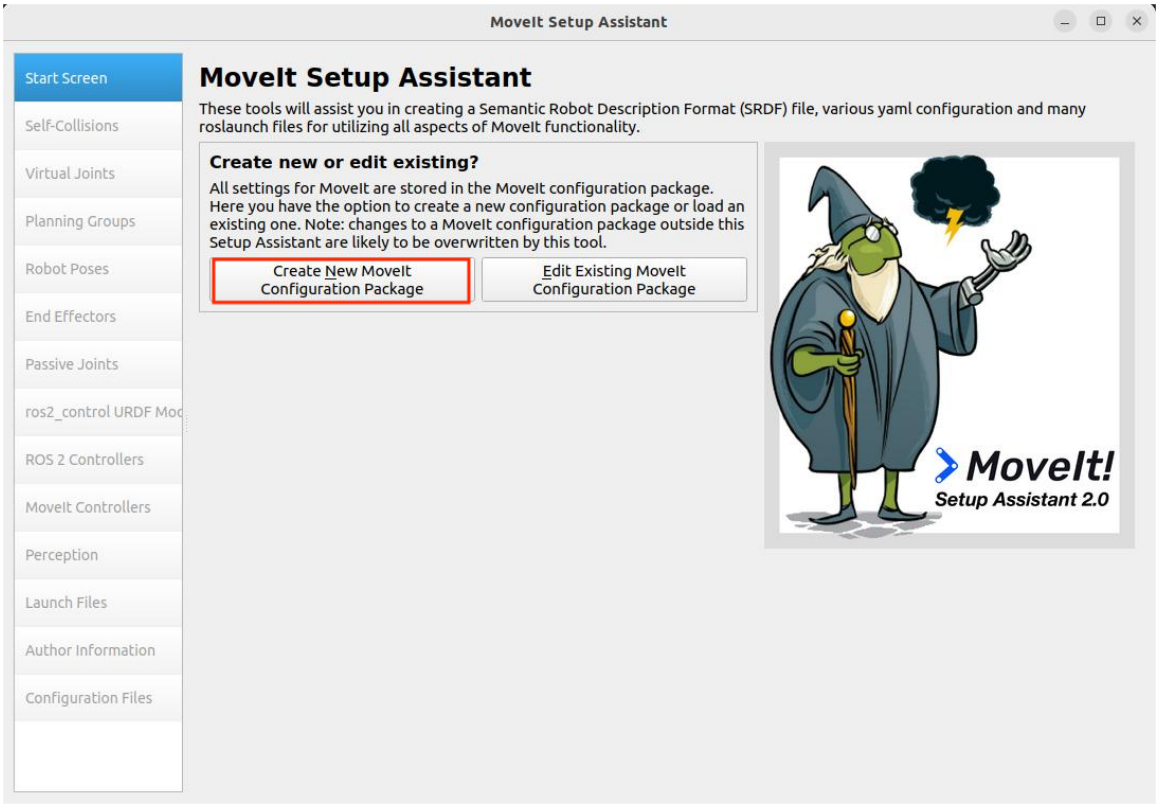
Start moveit_setup_assistant for robot configuration.

```
ros2 launch moveit_setup_assistant setup_assistant.launch.py
```

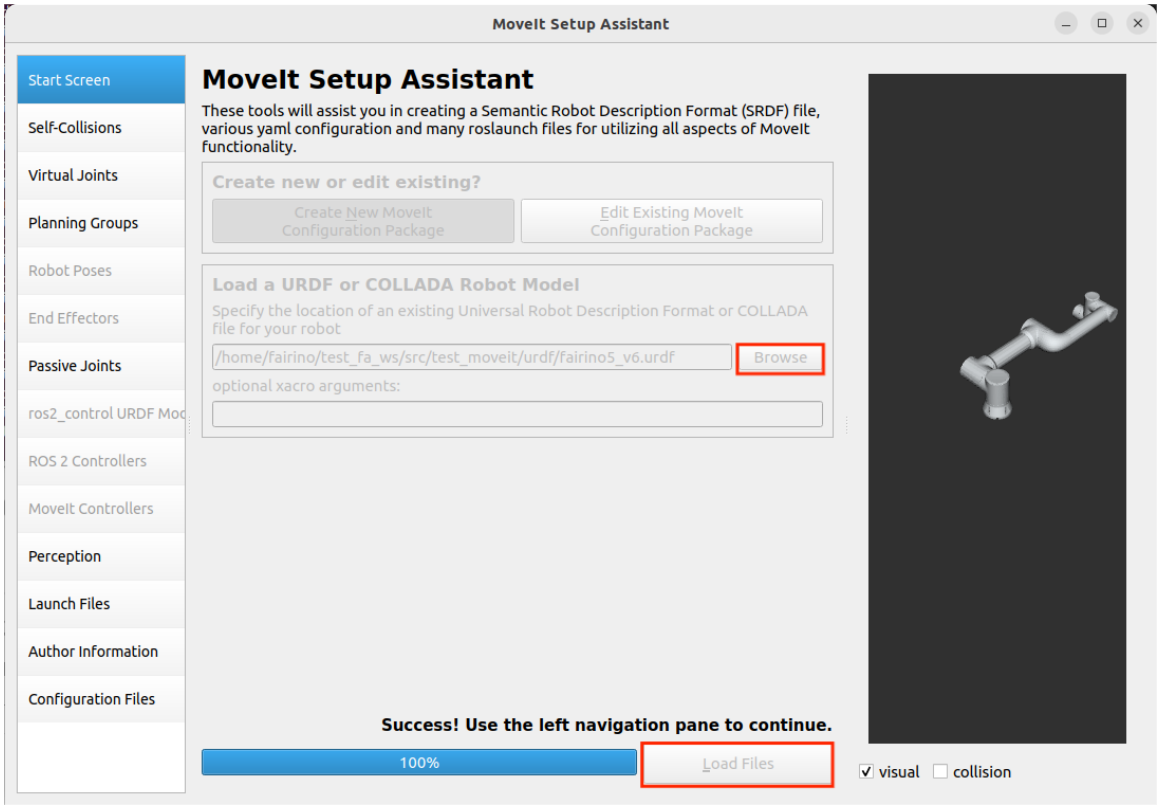
9.2.2.2 Configuration Robot

9.2.2.2.1 Launching the configuration interface

Open the terminal in the test_fa_ws directory and select "Create New Moveit Configuration Package" in the configuration interface to create a new moveit configuration package.

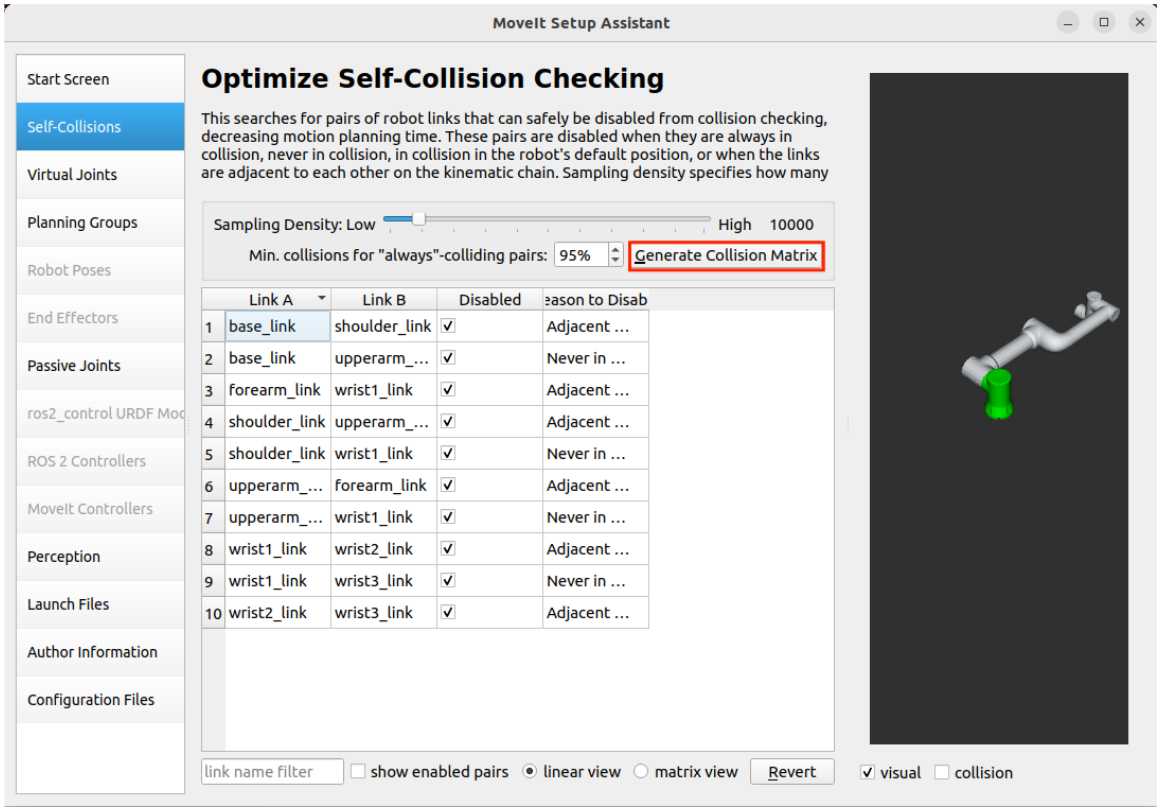


Then select the description file of the robot, that is, the .urdf file, and then select Load Files, load the robot model, you can see the right side of the robot model loaded out.



9.2.2.2.2 Configuring Self-Collisions

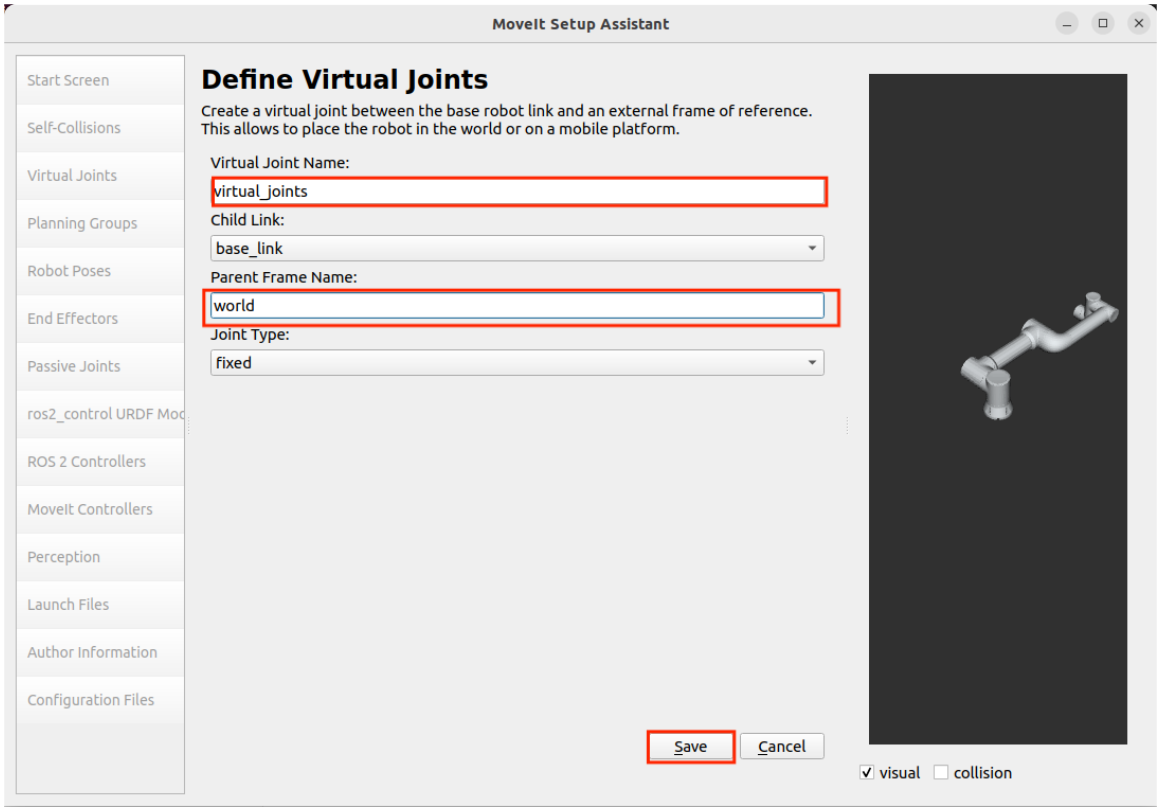
Self-Collisions for the robot collision settings, click Generate Collision Matrix can automatically generate the joints collision matrix, which will cancel the collision between the two contacting connecting rods and the connecting rods that can never be contacted, so as to configure the robot joints collision matrix, thus avoiding the calculation of the collision between the two contacting surfaces, click Generate Collision Matrix is automatically generated by clicking Generate.



9.2.2.2.3 Configuring Virtual Joints

Virtual Joints is the virtual axes of the robot, when the robot is installed on the mobile platform, it is necessary to set the virtual axes for the robot, set the name of the virtual axes, the sub-links, the type of joints, etc.

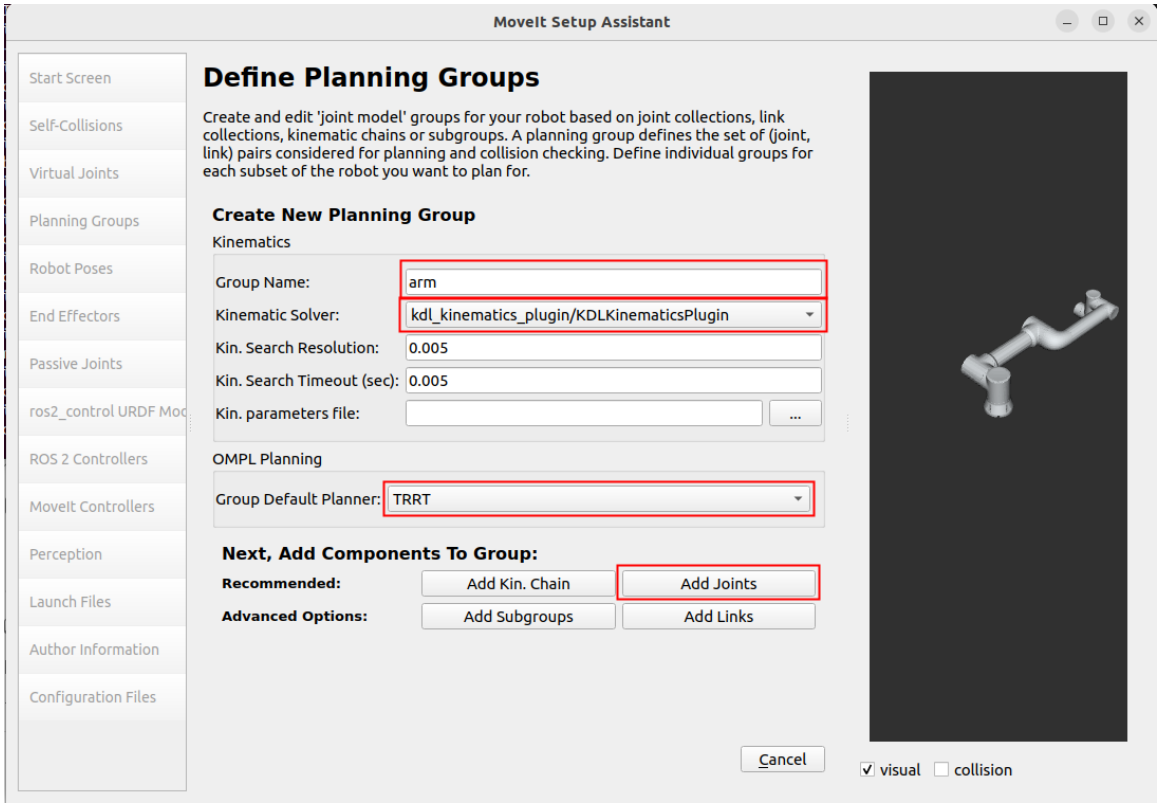
when the mobile platform moves, the virtual axes will also synchronize with the movement of the robot to drive the robot to move, so as to realize the robot with the movement of the mobile platform function this time the robot will be placed directly on the world coordinate system, named virtual_joints.



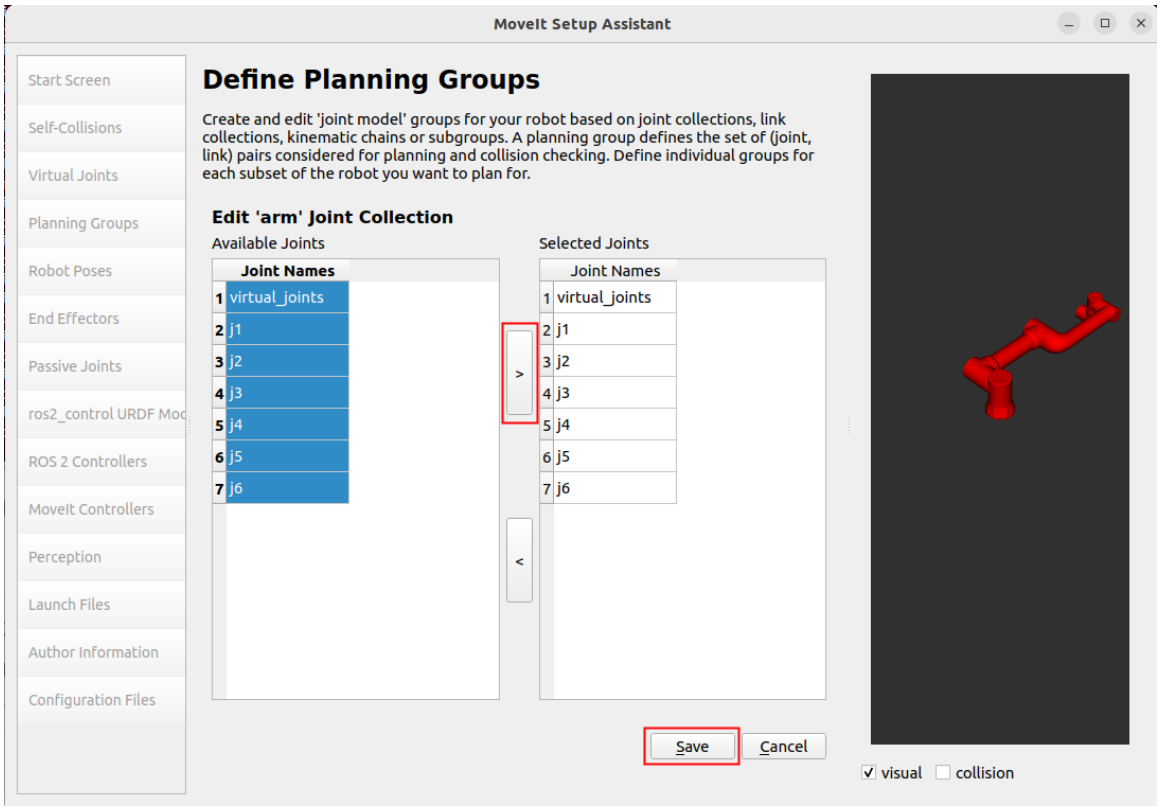
9.2.2.2.4 Configuring Planning Groups

Planning Groups is the planning group of the robot, which will be the joints that need to be considered in the same kinematic calculation in the same planning group for unified forward and reverse kinematic calculations, such as placing a robot on an AGV cart, and then installing a fixture at the end of the robot, the test will be the four joints of the AGV cart in a planning group, the six joints of the robot in a planning group, and one of the joints of the fixture in a planning group for kinematics calculations.

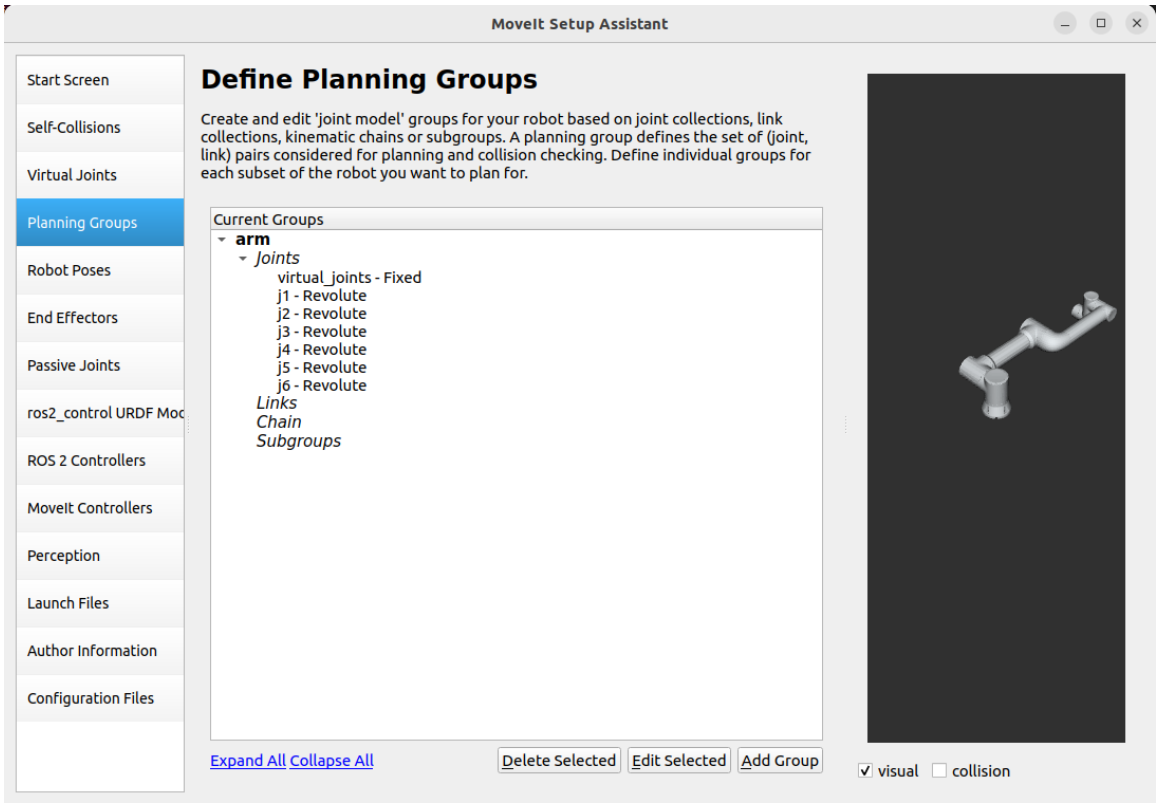
One joint of the fixture is in one planning group for kinematic calculation. Since this does not involve the fixture so only add the robot's joints group, that is, the arm group, first of all add the arm group, the dynamics solver Kinematic Solver select kdl_kinematics_plugin/KDLKinematicsPlugin, and then the default planner Group Default Planner select TRRT, then click Add Joints to add joints for this planning group.



arm's joints hold shift to make multiple selections, click '>' to add, then click save to save.

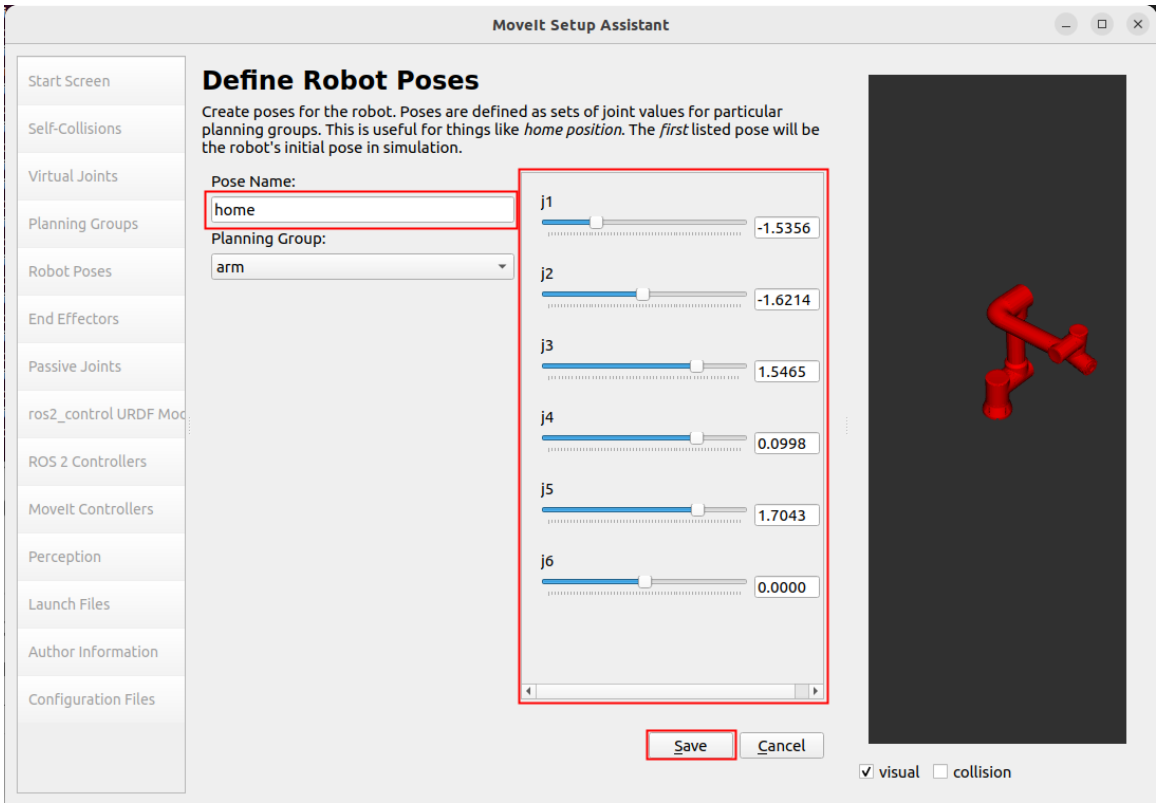


The defined planning group is shown below.



9.2.2.2.5 Configuring Robot Poses

Robot Poses is the robot preset poses, which defines some preset poses for each planning group, and a home pose for arm, which can be chosen at will.



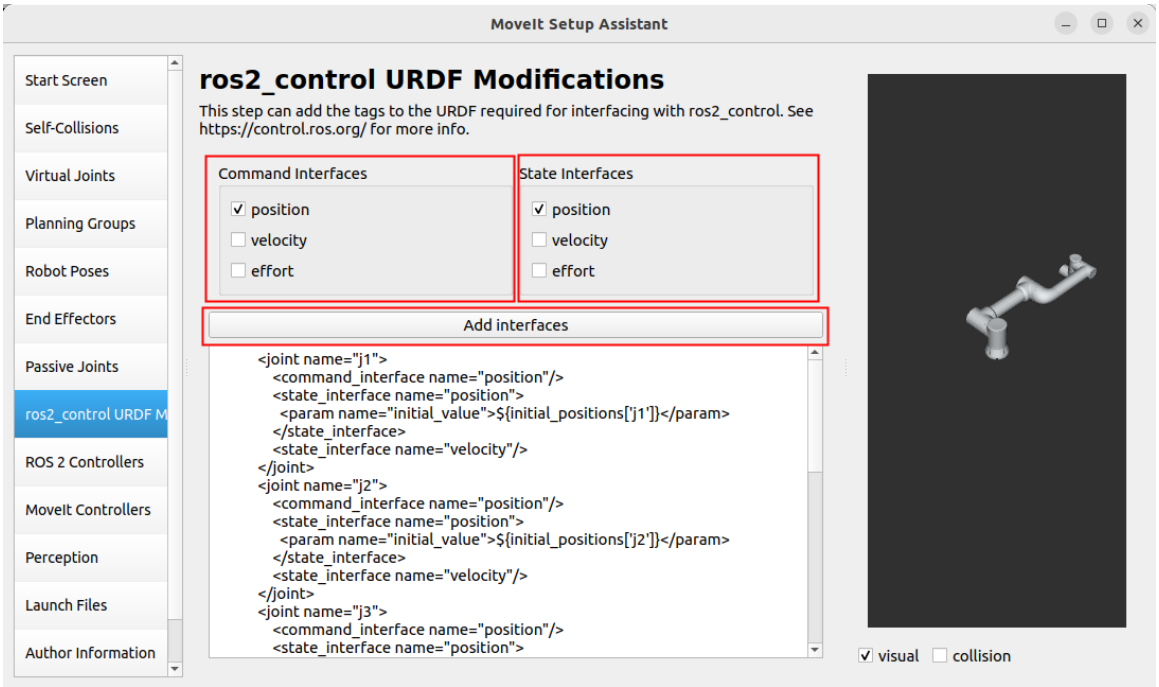
Robot Poses can define preset poses for each planning group. When there are fixtures in the robot, you can add a fixture planning group in the Planning Groups section, and then set the preset poses for the fixtures when setting poses in Robot Poses.

9.2.2.2.6 Configuring End Effectors

End Effectors is the robot end actuator, the planning group of end actuator is hand, and then the default connected parent_link is panda_link8, since there is no end actuator this time, so this step can be skipped.

9.2.2.2.7 ros2_control URDF Modifications

ros2_control URDF Modifications is mainly used to set the joint data type of sending and feedback, you can choose the position, speed, torque three kinds of joint data type of sending and feedback are position control, and then directly Add interfaces to be able to.



Important:

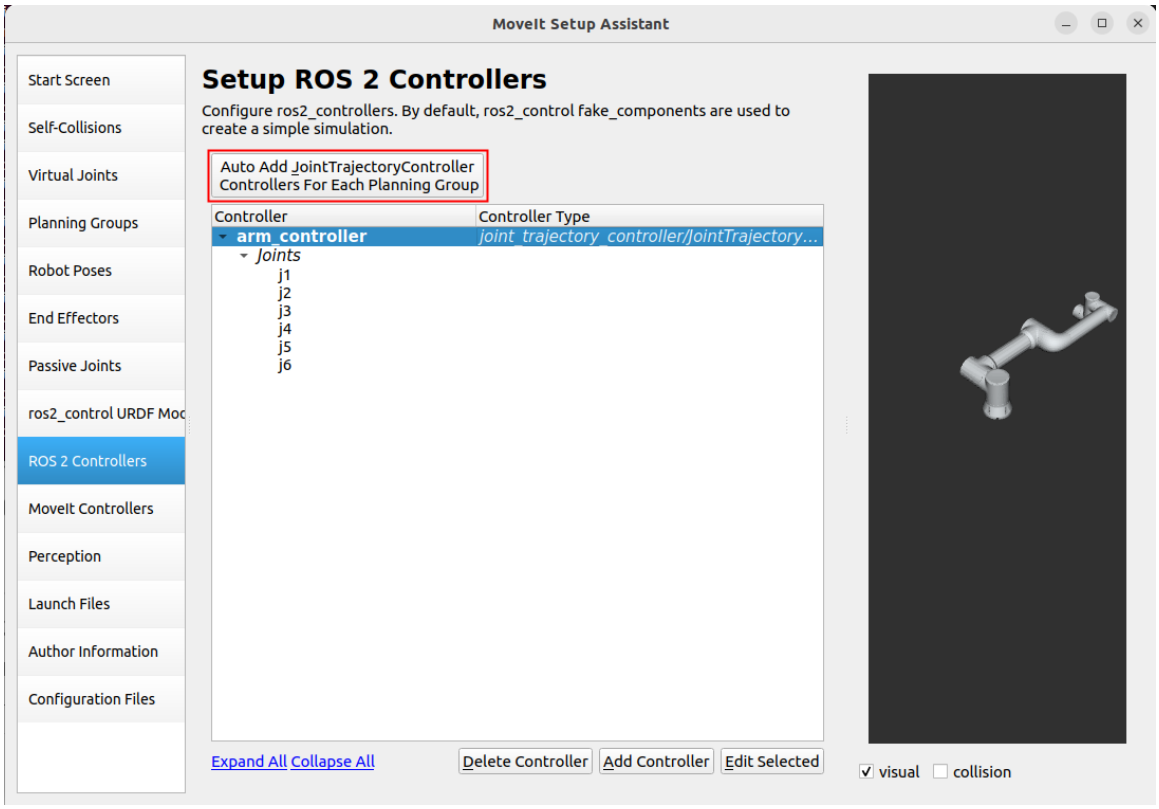
- Note:

  Selecting the joint data type needs to match the subsequent fairino_hardware plug-in, and select the joint data type for sending down and feedback according to the data transmitted by the fairino_hardware plug-in. Since the fairino_hardware plug-in controlling the motion of the actual robot this time uses the position data type, this time the Because the fairino_hardware plug-in that controls the motion of the actual robot uses the position data type, this time the joint data type chosen for sending and feedback are both position control.

9.2.2.2.8 ROS 2 Controllers

ROS 2 Controllers is mainly used to generate the ros2_controllers.yaml file, which sets the release frequency, joint name, controller name, controller type, etc.

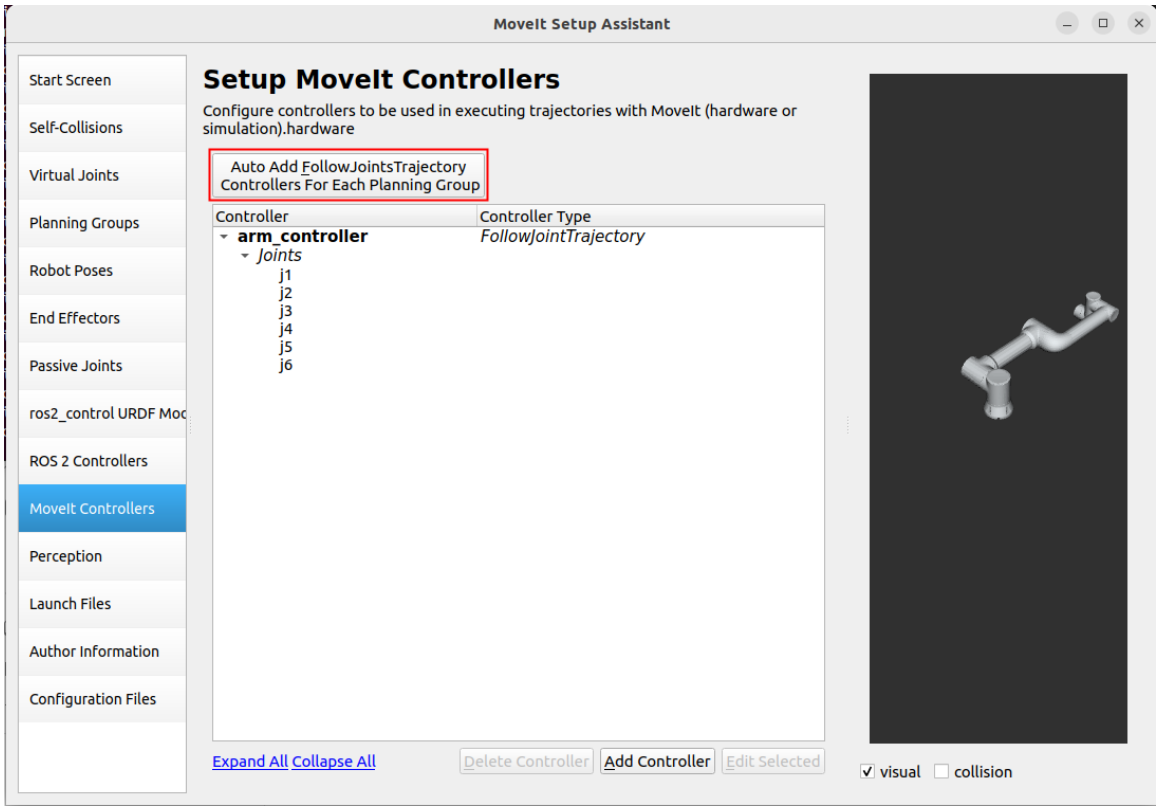
Configure ROS 2 Controllers, configure controllers for each planning group, click Auto Add JointTrajectoryController Controllers For Each Planning Group.



9.2.2.2.9 Moveit Controllers

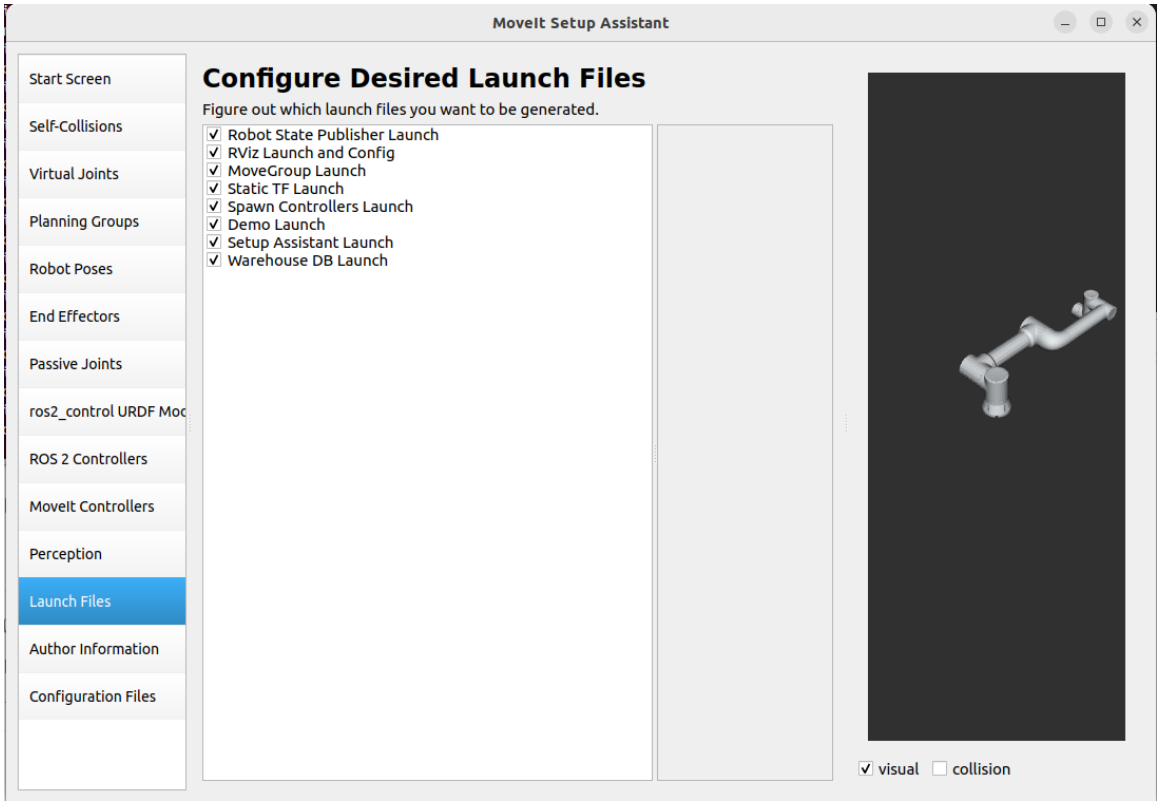
Moveit Controllers is mainly used to generate the moveit_controllers file, which sets the controller name, controller type, etc. It should be noted that the controller name in moveit_controllers needs to be the same as that of ros2_controllers, or it will not run smoothly. It should be noted that the controller name in moveit_controllers should be the same as the controller name in ros2_controllers, otherwise it will not run smoothly.

When the controller name in moveit_controllers is the same as the controller name in ros2_controllers, the controller type in moveit_controllers will be automatically mapped to the controller type in ros2_controllers, realizing that the control data sent down through the moveit_controllers is sent to ros2_controllers, and then the actual robot motion is driven by the plug-in in ros2_controllers.

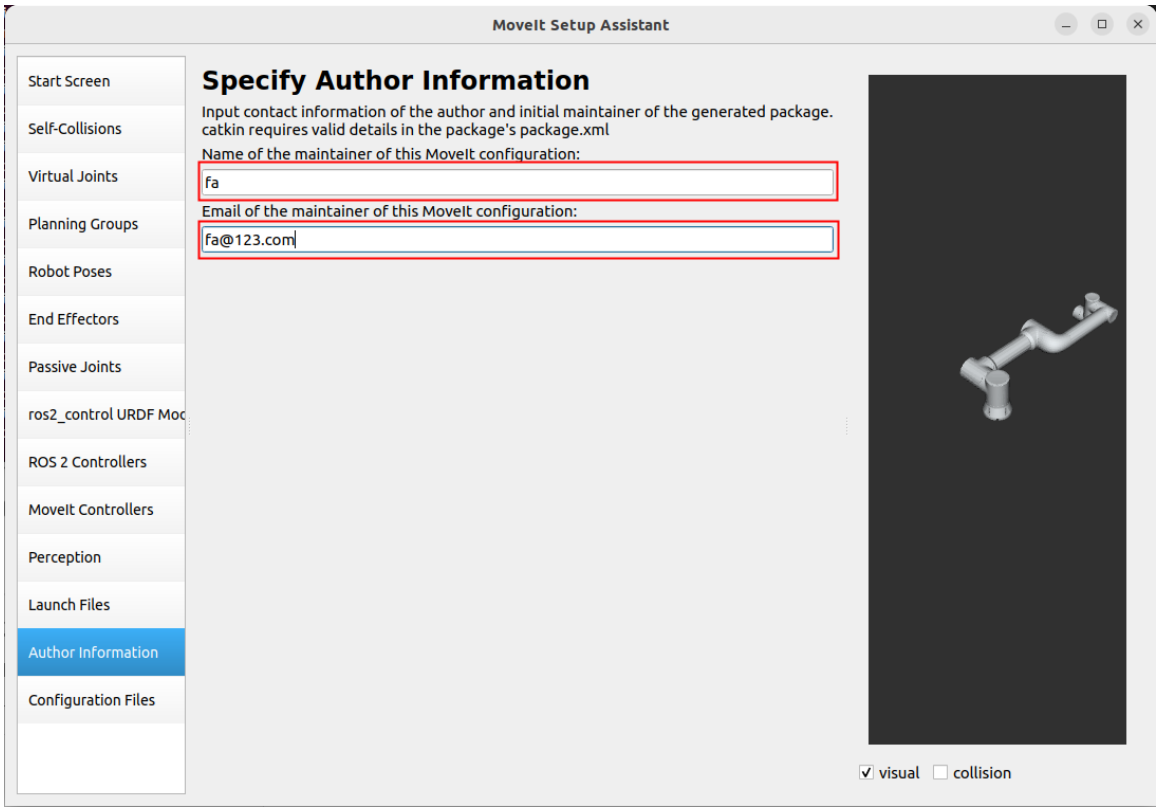


9.2.2.2.10 Launch Files

Configure Launch Files, just use the default configuration.

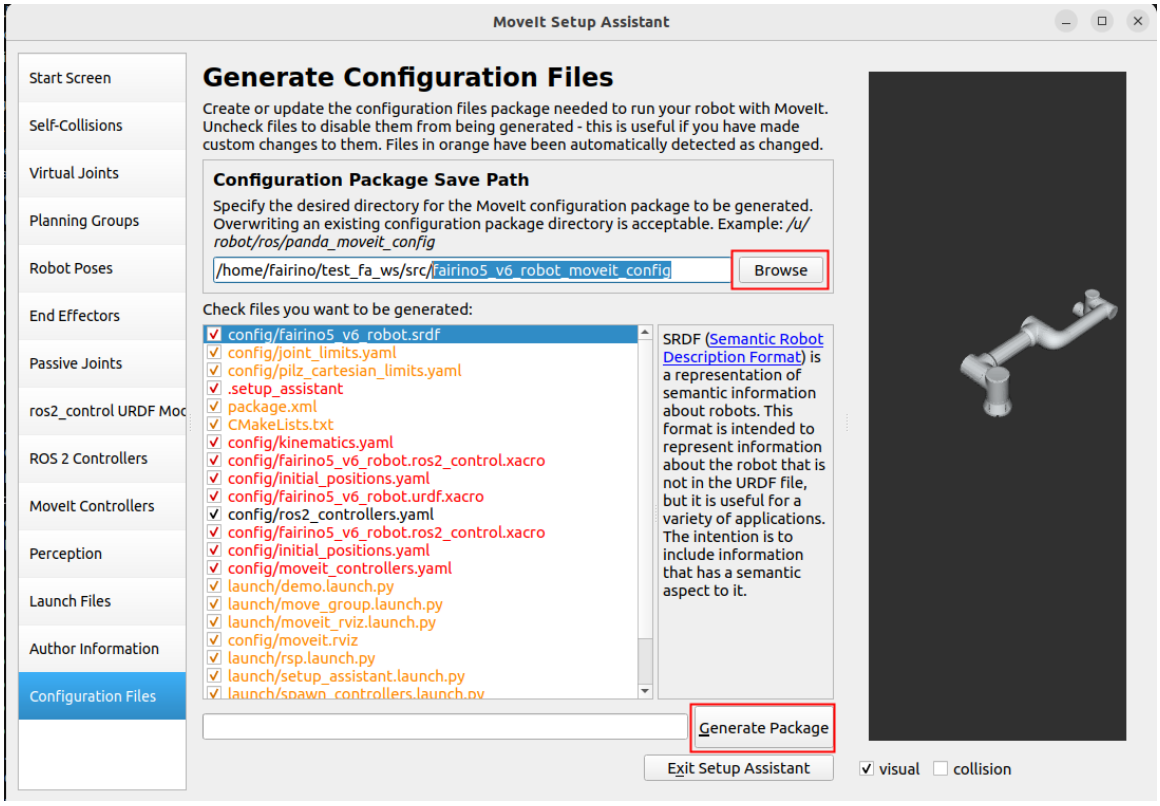

MoveIt Setup Assistant
Start Screen
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
ros2_control URDF Moc
ROS 2 Controllers
MoveIt Controllers
Perception
Launch Files
Author Information
Configuration Files
Configure Desired Launch Files
Figure out which launch files you want to be generated.
Robot State Publisher Launch
RViz Launch and Config
MoveGroup Launch
Static TF Launch
Spawn Controllers Launch
Demo Launch
Setup Assistant Launch
Warehouse DB Launch
visual
collision

9.2.2.2.11 Author Information



9.2.2.2.12 Generate Launch File

Generate the Launch file, select the location, this time in the test_fa_ws/src file path to create a folder fairino5_v6_robot_moveit_config used to store the configuration file, and then select the Generate.



As this has been configured once before, if the initial configuration Check files you want to be generated part of the contents of the black, indicating that you can generate Launch files.

9.2.2.3 Launch

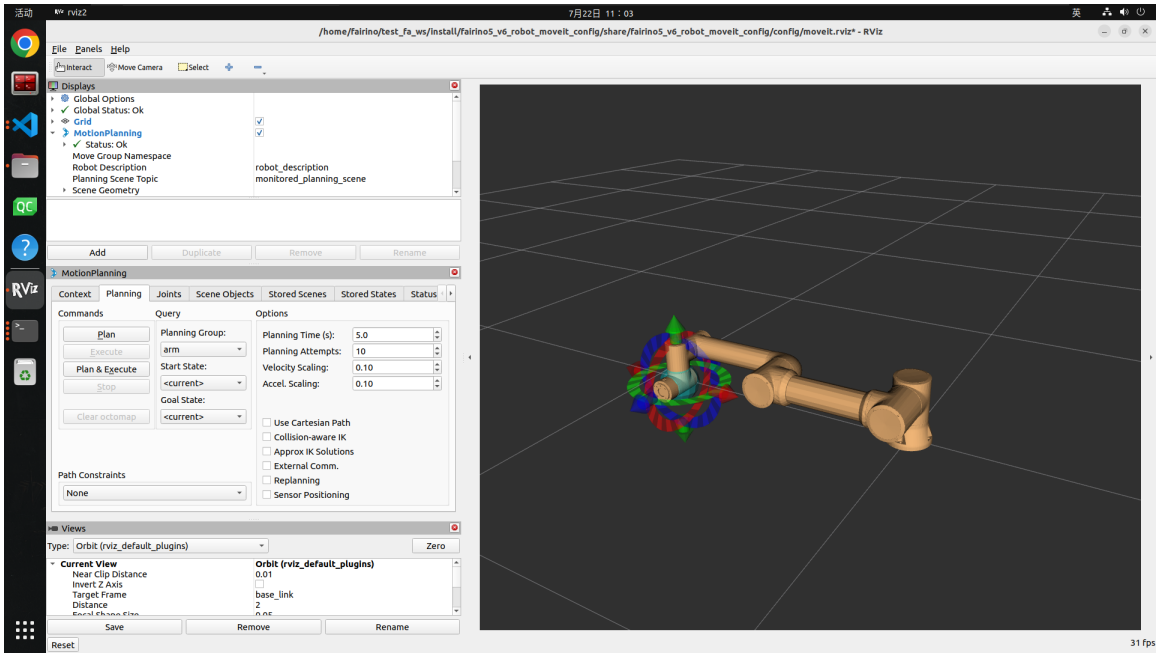
After the configuration is complete, you can compile the functionality packages, you can use the custom robot moveit2 configuration package to replace the FAIRINO Robot moveit2 configuration package, to achieve plug-in compatibility for the use of user-defined robots.

```
colcon build --packages-select fairino5_v6_robot_moveit_config
source install/setup.bash
```

Then just run the Launch file you just configured.

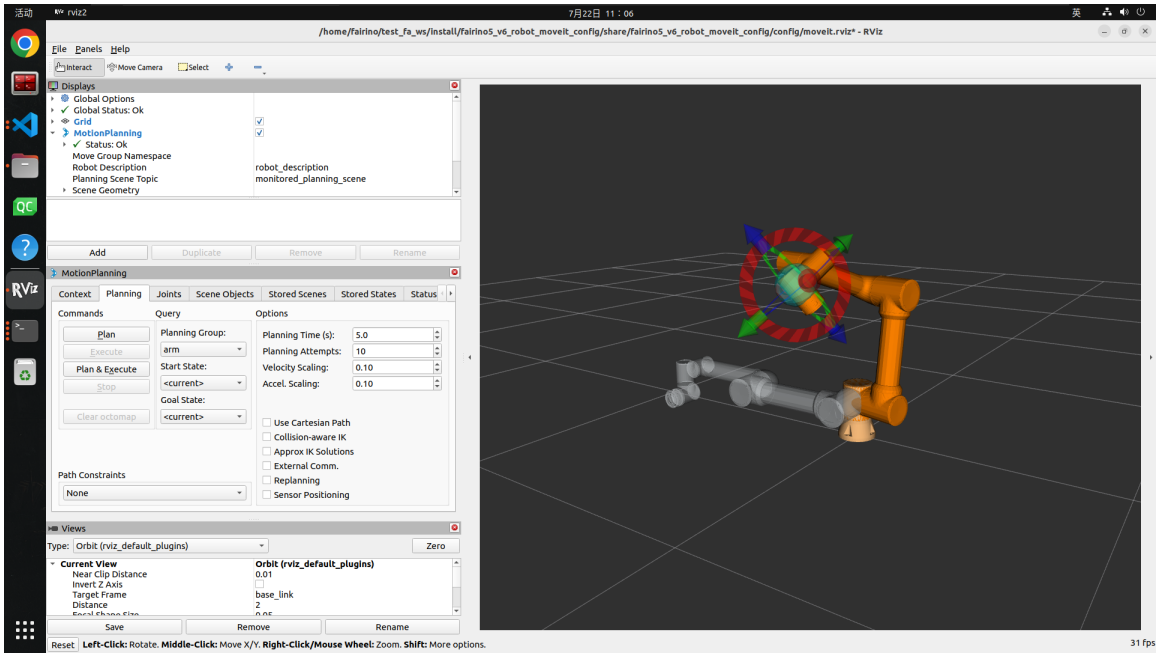
```
ros2 launch fairino5_v6_robot_moveit_config demo.launch.py
```

After that you can see the rviz2 interface with the configuration completed.

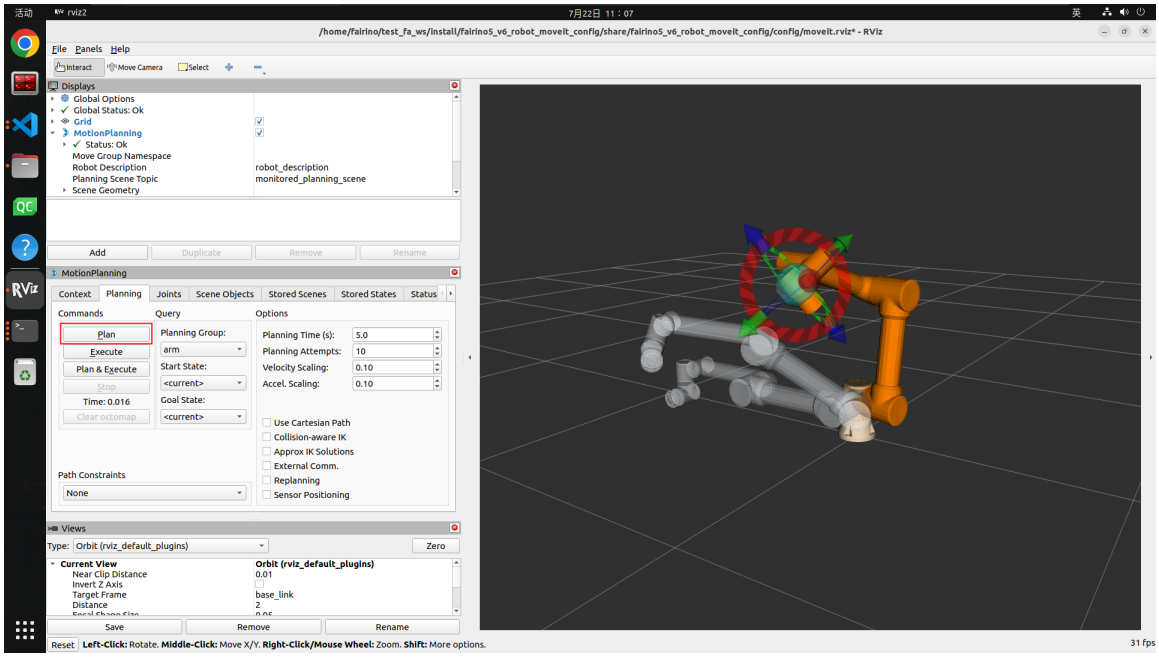


9.2.2.4 Moveit2 Usage

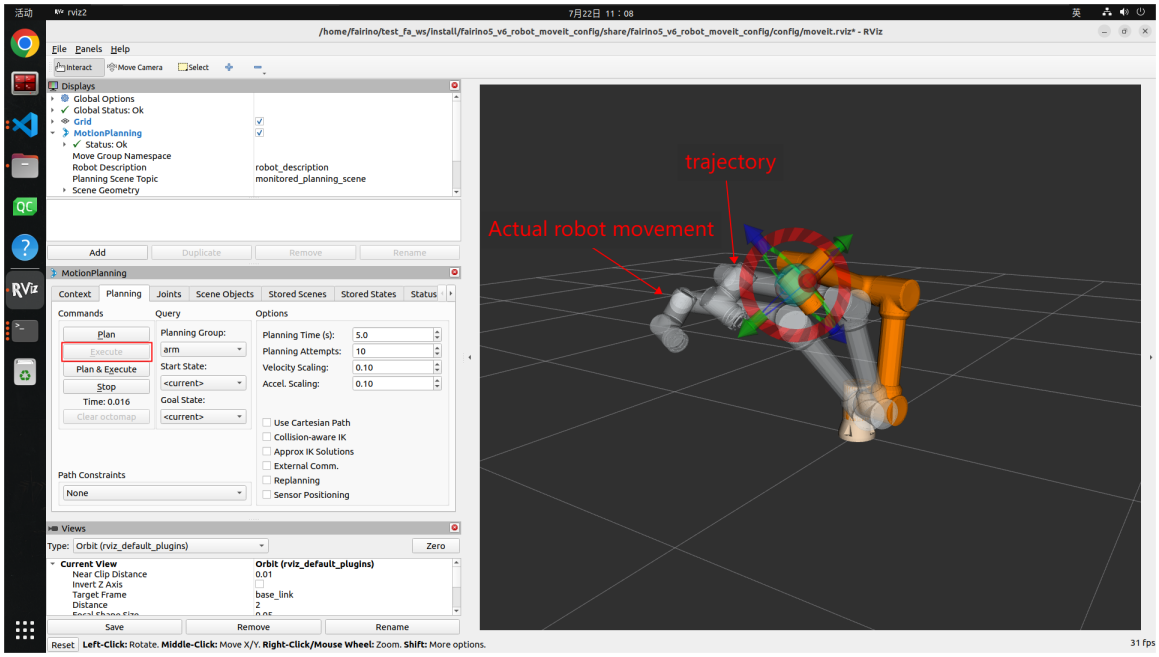
After opening the configured package, you can set the robot target position by dragging and dropping the blue sphere at the end of the robot in the 3D interface on the right side, and then change the robot end attitude by using the red, green, and blue rings at the end of the robot.



Click on the Plan button on the left to plan the robot trajectory.

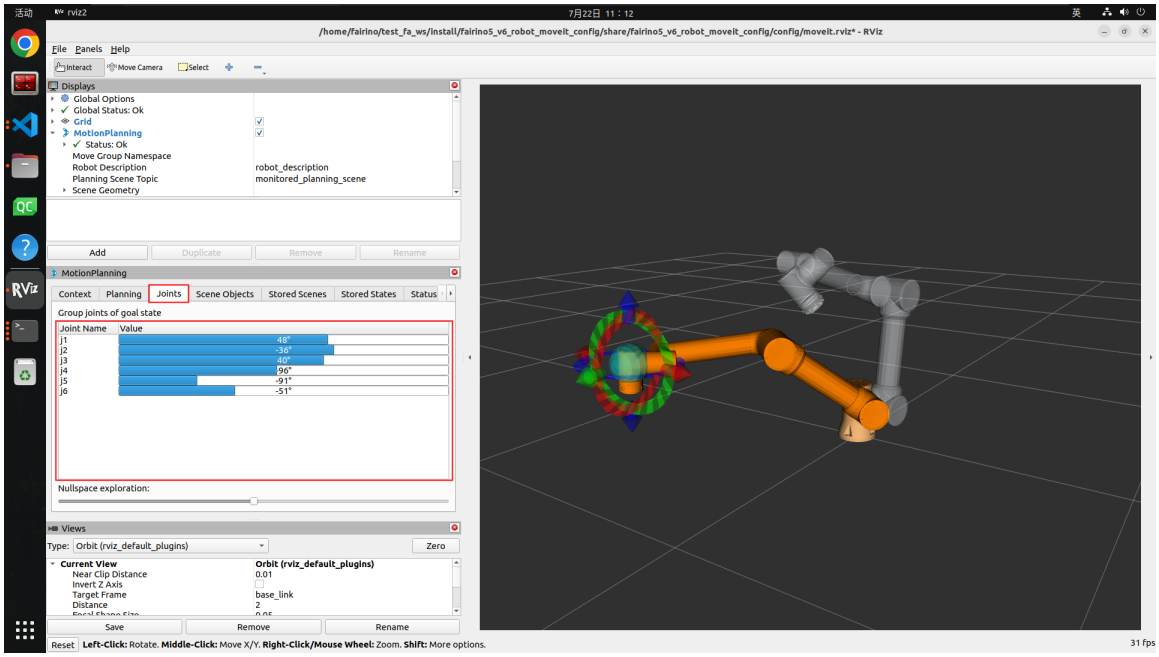


Click the Execute button on the left to drive the robot to the target position according to the planned trajectory.



Plan&Execute buttons are used to automatically control the robot's motion after planning a trajectory.

Then click on the Joints tab to change the robot's target position by changing the angles of the joints, and then use the Plan, Execute, Plan&Execute buttons to drive the robot.



9.2.3 Fairino_hardware Plugin (custom robot moveit configuration package)

Fairino_hardware plugin for the connection between moveit and the robot's intermediate layer, through the fairino_hardware plugin move_group will be sent to moveit_control, and then forwarded to ros2_control, ros2_control and then through the fairino_hardware plugin to drive the actual robot movement, and fairino_hardware plugin will also accept the actual robot feedback data, so as to realize the rviz2 simulation interface robot model and the actual robot synchronization.

Hardware plug-in drives the actual robot motion, and fairino_hardware plug-in also accepts the feedback data from the actual robot, thus realizing the synchronization between the robot model of the rviz2 simulation interface and the actual robot, and thus realizing the function of the user driving the actual robot motion through the rviz2 interface.

And thanks to the implementation of the fairino_hardware plug-in, the Fairino robot can access the ros2_control control framework, making the Fairino robot compatible with third-party packages based on ros2_control.

9.2.3.1 Fairino_hardware Plugin Compilation

Compile the fairino_hardware plugin package in the official ros2_ws package, compile the fairino_hardware plugin package through the previous section, then you will see the .so file libfairino_hardware.so generated by the plugin under "ros2_ws/install/fairino_hardware/lib/", which means the plugin is compiled successfully.

Then you will see the .so file libfairino_hardware.so generated by the plug-in under "ros2_ws/install/fairino_hardware/lib/", which indicates that the plug-in has been compiled.

It should be noted that you need to make the naming of each joint of the robot by the fairino_hardware plug-in the same as the naming of each joint of the robot configured by moveit2.

The naming of the six joints of the robot by this fairino_hardware plug-in from the base coordinate position to the end of the robot are j1, j2, j3, j4, j5 and j6 respectively, so you need to name the joints of the robot as j1, j2, j3, j4, j5 and j6 when the robot is configured by moveit2. moveit2 configuration of the robot need to robot joints named j1, j2, j3, j4, j5, j6.

9.2.3.2 Fairino_hardware Plugin Usage

Compile the fairino_hardware plugin package in the official ros2_ws package, compile the fairino_hardware plugin package through the previous section, then you will see the .so file libfairino_hardware.so generated by the plugin under "ros2_ws/install/fairino_hardware/lib/", which means the plugin is compiled successfully.

Then you will see the .so file libfairino_hardware.so generated by the plug-in under "ros2_ws/install/fairino_hardware/lib/", which indicates that the plug-in has been compiled successfully.




```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
    <xacro:macro name="fairino5_v6_robot_ros2_control" params="name initial_positions_file">
        <xacro:property name="initial_positions" value="${load_yaml(initial_positions_file)['initial_positions']}"/>

        <ros2_control name="${name}" type="system">
            <hardware>
                <!-- By default, set up controllers for simulation. This won't work on real hardware -->
                <!-- <plugin>mock_components/GenericSystem</plugin> -->
                <plugin>fairino_hardware/FairinoHardwareInterface</plugin>
            </hardware>
            <joint name="j1">
                <command_interface name="position"/>
                <state_interface name="position">
                  <param name="initial_value">${initial_positions['j1']}</param>
                </state_interface>
            </joint>
            <joint name="j2">
                <command_interface name="position"/>
                <state_interface name="position">
                  <param name="initial_value">${initial_positions['j2']}</param>
                </state_interface>
            </joint>
            <joint name="j3">
                <command_interface name="position"/>
                <state_interface name="position">
                  <param name="initial_value">${initial_positions['j3']}</param>
                </state_interface>
            </joint>
            <joint name="j4">
                <command_interface name="position"/>
                <state_interface name="position">
                  <param name="initial_value">${initial_positions['j4']}</param>
                </state_interface>
            </joint>
```

where "fairino_hardware/FairinoHardwareInterface" is the name of the plug-in for the hardware settings, which can be found in the "fairino_hardware.xml" file in the "ros2_ws/src /fairino_hardware" directory of the 'fairino_hardware.xml' file to see.




```
<library path="fairino_hardware">
    <class name="fairino_hardware/FairinoHardwareInterface"
           type="fairino_hardware::FairinoHardwareInterface"
           base_class_type="hardware_interface::SystemInterface">
        <description>
            Hardware interface for Fairino Robots
        </description>
    </class>
</library>
```

9.2.3.3 Running Plug-ins

Open the terminal, then go to the ros2_ws workspace and source workspace, the purpose is to add the fairino_hardware plugin, you can also load the path to the "~/.bashrc" file, but it is not recommended.

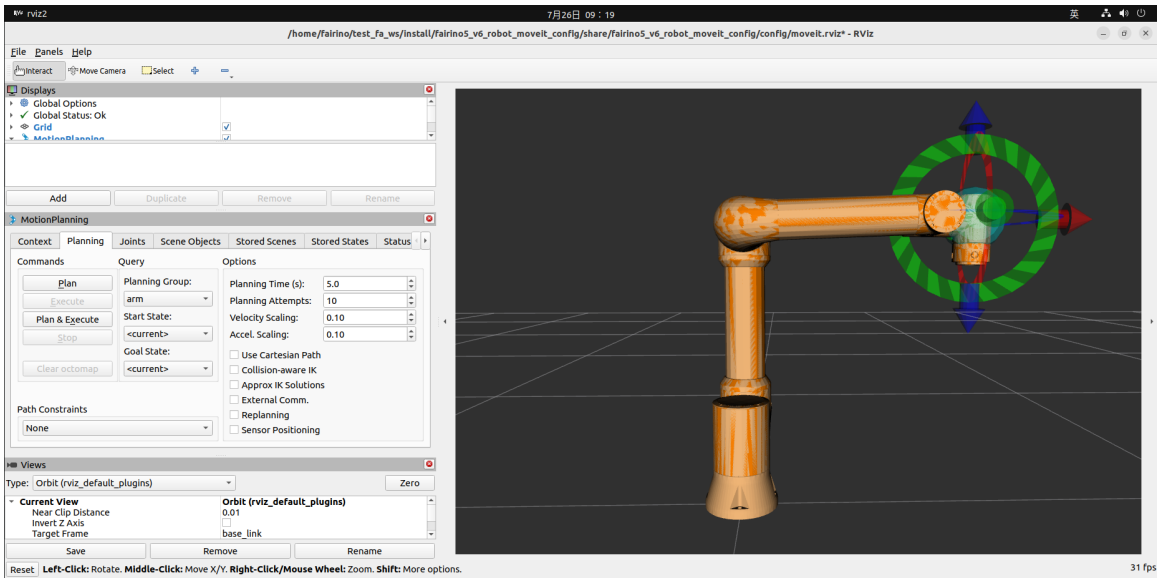
```
cd ros2_ws
source install/setup.bash
```

Then go back to the home directory and go to the test_fa_ws workspace and the source workspace and run the demo.launch.py file.

```
cd ..
cd test_fa_ws
source install/setup.bash
ros2 launch fairino5_v6_robot_moveit_config demo.launch.py
```

9.2.3.4 Running Result

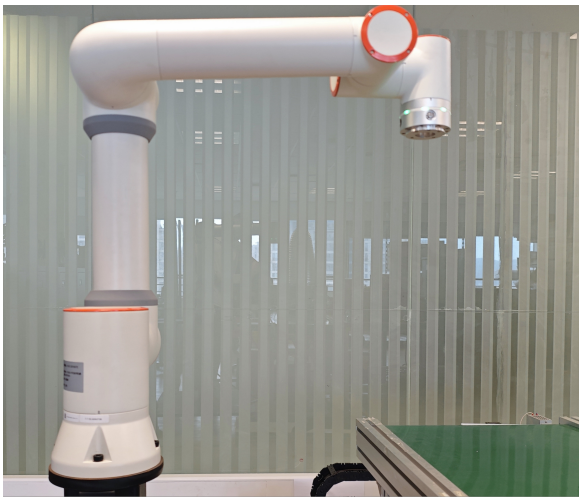
After the demo.launch.py file is launched, the rviz2 interface is shown below.



Compile the fairino_hardware plugin package in the official ros2_ws package, compile the fairino_hardware plugin package through the previous section, then you will see the .so file libfairino_hardware.so generated by the plugin under "ros2_ws/install/fairino_hardware/lib/", which means the plugin is compiled successfully.

Then you will see the .so file libfairino_hardware.so generated by the plug-in under "ros2_ws/install/fairino_hardware/lib/", which indicates that the plug-in has been compiled successfully.

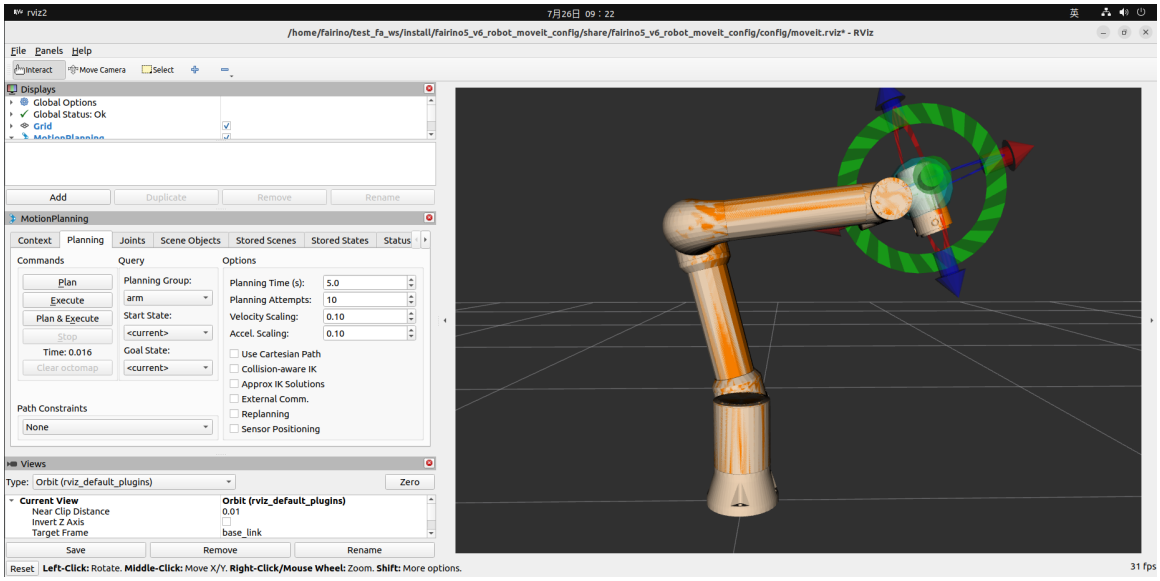
The actual robot position at this point is as follows.



At this point, you can drive the actual robot through the rviz2 interface, drag the blue sphere at the end of the robot in the rviz2 interface to move the end of the robot to the target position, then drag the red, green, and blue rings at the end of the robot to change the robot's end attitude, and then click on the left-hand side of the Planning&Execute button to carry out the motion trajectory planning and drive the robot motion.

You will find that the actual robot moves synchronously with the simulated robot on the rviz2 interface and stops at the target position.

The following figure shows the motion of the actual robot and the simulated robot through the rviz2 interface to the target position.





At this point it is possible to control the synchronized motion of the actual robot and the simulated robot on the rviz2 interface via moveit2.

9.2.4 Fairino_hardware Plugin (official robot moveit configuration package)

In the ros2_ws directory, compile the functionality packages for your corresponding robot model, using the fairino5 robot as an example. In plugin corresponding to robotsofeware V3.8.3, torque command mode is added, which means robot can receive torque command.

```
cd ros2_ws
colcon build --packages-select fairino5_v6_moveit2_config
source install/setup.bash
```

Then you need to add the fairino_hardware plugin to synchronize the movement with the actual robot, go to "ros2_ws/install/fairino5_v6_moveit2_config/share/fairino5_v6_moveit2_config/" directory.

Locate the fairino5_v6_robot.ros2_control.xacro file, focus on the 3rd line of xacro file:

```
use_fake_hardware:=false
```

replace it to:

```
use_fake_hardware:=true
```

Be carful, the parameter "robot_control_mode" in line3 decides which operation mode robot start with when plugin loaded, 0 means position mode, which plugin will export position command interface, 1 means troque mode, which plugin will export effort command interface. The demo of torque mode operation will release in the fairino_hardware package which corresponding to robot softwareV3.8.4, please don't set this parameter to 1 before that.

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
    <xacro:macro name="fairino5_v6_robot_ros2_control" params="name initial_positions_file use_fake_hardware:=true robot_control_mode:=0">
        <xacro:property name="initial_positions" value="${load_yaml(initial_positions_file)['initial_positions']}"/>

        <ros2_control name="${name}" type="system">
            <hardware>
                <!-- By default, set up controllers for simulation. This won't work on real hardware -->
                <xacro:if value="${use_fake_hardware == true}">
                  <plugin>mock_components/GenericSystem</plugin>
                </xacro:if>
                <xacro:unless value="${use_fake_hardware == false}">
                  <plugin>fairino_hardware/FairinoHardwareInterface</plugin>
                  <param name="robot_control_mode">${robot_control_mode}</param>
                </xacro:unless>
            </hardware>
            <joint name="j1">
                <xacro:if value="${robot_control_mode == 0}">
                        <command_interface name="effort"/>
                </xacro:if>
```

where "fairino_hardware/FairinoHardwareInterface" is the name of the plug-in for the hardware settings, which can be found in the 'fairino_hardware.xml' file in the "/ros2_ws/src /fairino_hardware" directory of the 'fairino_hardware.xml' file to see the.



At this point to run the plugin you need to go to the ros2_ws workspace, then source the environment and run the demo.launch.py file.

```
cd ros2_ws
source install/setup.bash
ros2 launch fairino5_v6_moveit2_config demo.launch.py
```

9.3 MTC Sample Code Package

9.3.1 MTC Sample Code Package Introduction

The mtc sample code package provides a refactored rviz2 interface using the moveit2 and fairino_hardware plugins. The original MotionPlanning tab is replaced with the Motion Planning Tasks tab, which is used to display the various phases of the robot's motion.

Rviz2 interface can be accessed via the The rviz2 interface can be edited from the file "mtc.rviz" under the path "ros2_ws/install/fairino_mtc_demo/share/fairino_mtc_demo/launch". "You can edit the 'mtc.rviz' file to customize

the rviz2 interface to meet your functional requirements.

And mtc sample code package also provides a moveit2 and fairino_hardware plugin to drive the robot cycle to grab the target example, through the example you can understand how to make use of the form of code to better use the moveit2 and fairino_hardware plugin and the actual robot interaction, based on which you can make Customization to fit your needs.

9.3.2 MTC Sample Code Package Compilation

9.3.2.1 MTC Sample Code Package Cloning

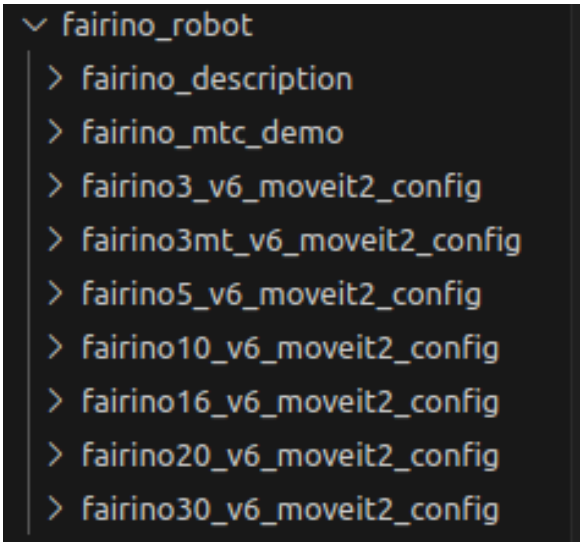
Clone the official mtc sample code package 'fairino_robot' into the src directory of the 'ros2_ws' workspace.

9.3.2.2 Robot Model Selection

Select the robot model in the mtc_demo_env.launch.py file in the 'ros2_ws/src/fairino_robot/fairino_mtc_demo_launch' directory of the official mtc sample code package. Select the robot model in the file, and modify lines 9, 10, and 11 of the file to match your robot.

```
 9    robotname = "fairino5_v6_robot"
10    packagename = "fairino5_v6_moveit2_config"
11    ros2controllername = "fairino5_controller"
```

You can refer to the function packs for each robot model in the "ros2_ws/src/fairino_robot/" directory for the robot model names.



9.3.2.3 MTC Sample Code Package Compilation

9.3.2.3.1 Compiling the fairino_description function package

Open a terminal, go to the ros2_ws directory, compile the fairino_description feature package, and then do a source.

```
cd ros2_ws
colcon build --packages-select fairino_description
source install/setup.bash
```

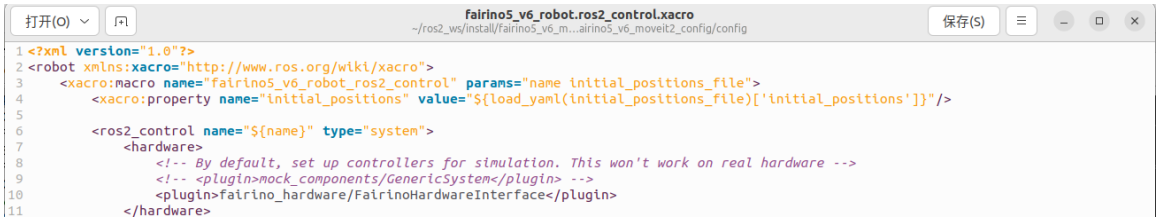
9.3.2.3.2 Compile robot function packs

In the ros2_ws directory, compile the functionality packages for your corresponding robot model, using the fairino5 robot as an example.

```
colcon build --packages-select fairino5_v6_moveit2_config
source install/setup.bash
```

Then add the fairino_hardware plugin to synchronize the movement with the actual robot, go to "ros2_ws/install/fairino5_v6_moveit2_config/share/fairino5_v6_moveit2_config/ config" directory, find fairino5_v6_moveit2_config.

Find fairino5_v6_robot.ros2_control.xacro, replace "<plugin>mock_components/GenericSystem</plugin>" with "<plugin>fairino_hardware/FairinoHardwareInterface</plugin>" in line 9 of the file, save and exit.



9.3.2.3.3 Compile fairino_mtc_demo function package

Compile fairino_mtc_demo function package and source.

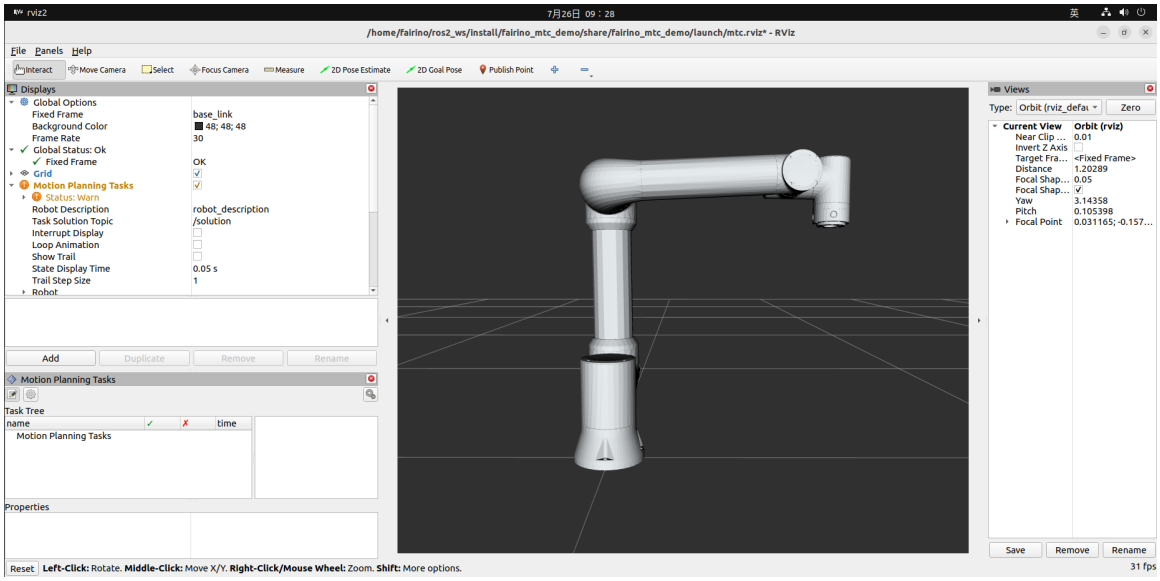
```
colcon build --packages-select fairino_mtc_demo
source install/setup.bash
```

9.3.3 The MTC Sample Code Package Runs

9.3.3.1 Rviz2 Interface

Run the mtc_demo_env.launch.py file to open the customized rviz2 interface. The Motion Planning Tasks tab on the left side of the interface is used to display the customized processes for each motion of the robot.

```
cd ros2_ws
source install/setup.bash
ros2 launch fairino_mtc_demo mtc_demo_env.launch.py
```



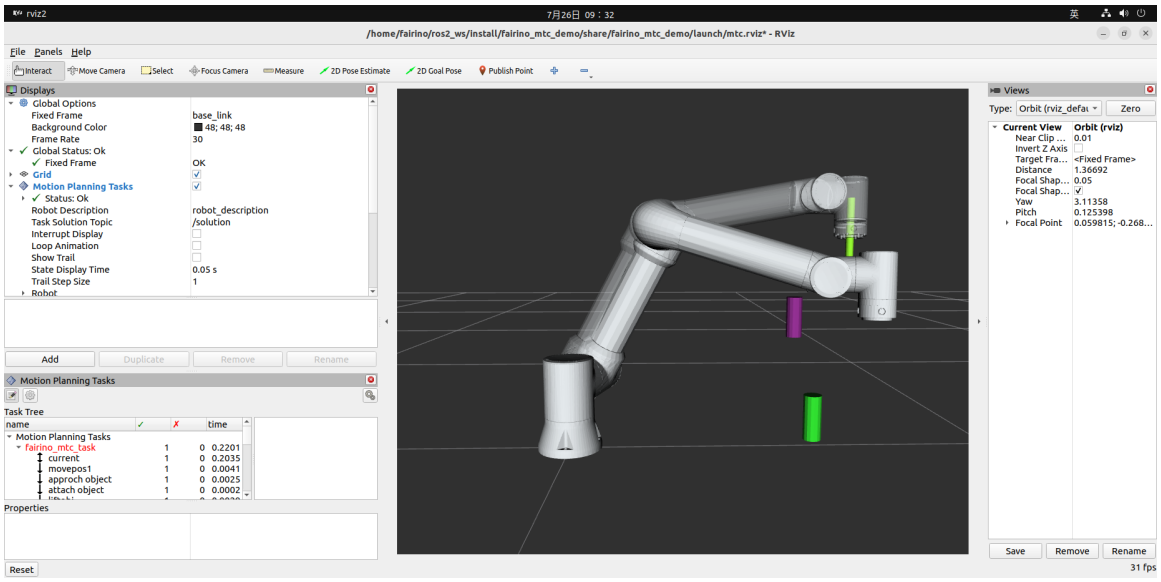


9.3.3.2 Robot Motion

Reopen a new terminal, go to the ros2_ws directory and source file and run the mtc_demo_app.launch.py file to execute the robot motion.

```
cd ros2_ws
source install/setup.bash
ros2 launch fairino_mtc_demo mtc_demo_app.launch.py
```

Then the Motion Planning Tasks tab in the rviz2 interface will show the robot's various motion processes, and the actual robot and the simulated robot in the rviz2 interface will be synchronized.





9.4 Caveat

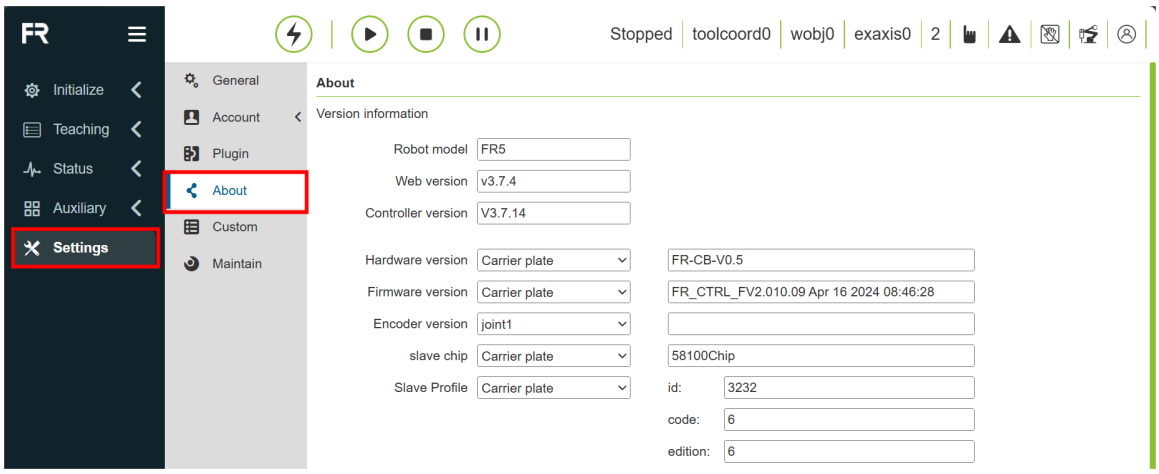
9.4.1 Fairino_hardware Plugin Version Synchronization

To use the fairino_hardware plug-in, the version of the fairino_hardware plug-in needs to be the same as the version of the Valeo robot. robot motion data sent by ros2_control is converted to a data type acceptable to Fairino robot.

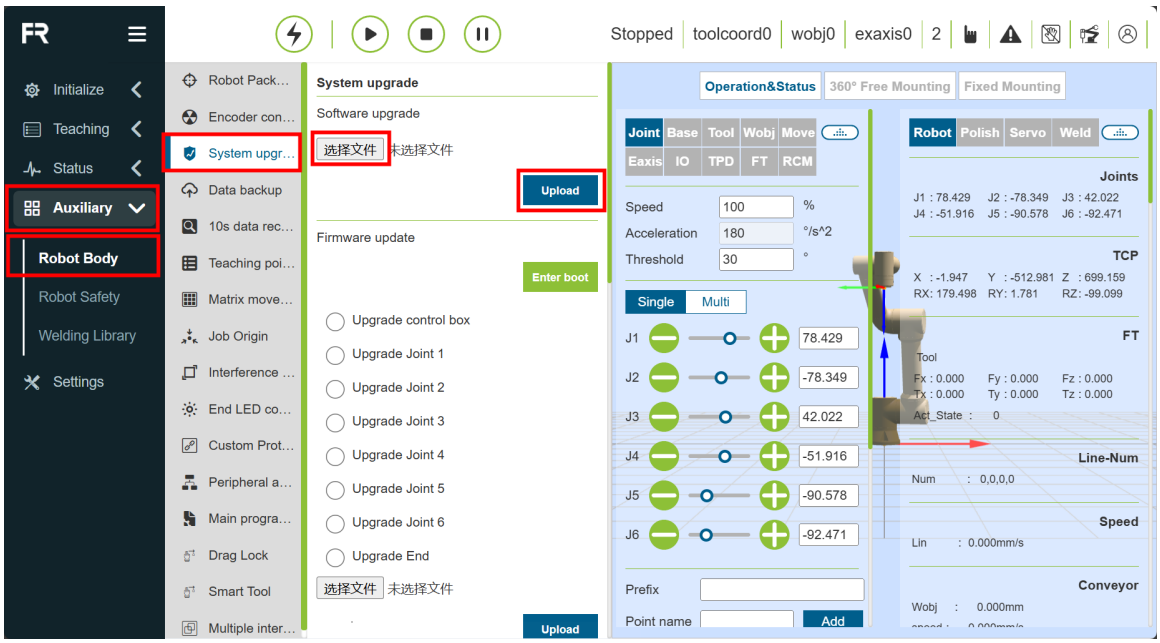
So it is critical that the fairino_hardware plug-in's data type is consistent with the Fairino robot's data type, and the different versions of the plug-in and the robot may result in different data types, so before officially debugging the fairino_hardware plug-in, you need to confirm that the version of Fairino robot is consistent with the version of fairino_hardware plug-in and the version of the robot.

Before debugging the fairino_hardware plugin, you need to make sure that the version of the Fairino robot is the same as the version of the fairino_hardware plugin, and if it is not, you need to upgrade the Fairino robot.

- First of all, you can check the current version of the robot in the "WebAPP Interface->System Settings->About" interface of Fairino Robotics.



- Then you need to prepare the official robot software package, then enter the “WebAPP interface->Auxiliary applications->Robot body->System upgrade” interface of the Fairino Robot, and then click the “Select file” button to select the prepared software package corresponding to the version of the fairino_hardware plug-in, and then select “Upload package” and wait for the software upgrade to complete. Click the “Select File” button, select the robot software upgrade package corresponding to the version of the fairino_hardware plug-in, select “Upload Package”, and wait for the software upgrade to complete.
- After the upgrade is completed, the system will prompt the need to restart the robot, the robot control box switch to the off position, wait for about 25 seconds, and then start the robot, so the robot software version upgrade is complete, you can carry out the subsequent fairino_hardware plug-in compilation and use.



9.4.2 Problems Likely To Be Encountered

9.4.2.1 The robot model may not be loaded on the right side of the Configure Robot Feature Pack.

Solution: This error may be due to the path in the .urdf file is not written correctly, can be solved by modifying the path in the .urdf file and copying the meshes file into the workspace under install/test_moveit/share/test_moveit.

9.4.2.2 After generating the package, there is an error running it.

Solution: Removing line 203 in the launches.py file "default_value=moveit_config.move_group_capabilities["capabilities"]," in ["capabilities"] will fix it.

9.5 Summarize

This tutorial describes the installation, configuration and use of the MoveIt2 plug-in; the installation and use of the fairino_hardware plug-in to realize the synchronous motion of the rviz2 simulation robot and the actual robot; and the compilation and running of the mtc sample code package to realize the customization function with the help of the moveit2 and fairino_hardware plug-ins.

It is hoped that the elaboration of this tutorial will enable users to have a more comprehensive understanding of MoveIt2 and the fairino_hardware plugin, and hopefully help them to better personalize the FAIRINO Robot service features.

CHAPTER

TEN

BROCHURE

FAIRINO Product Catalogue

FAIRINO Accessories Ecosystem

CHAPTER

ELEVEN

QUALIFICATION CERTIFICATION

FAIRINO ISO9001 Quality Certification

FAIRINO ISO14001 Environmental Management System Certification

FAIRINO ISO45001 Occupational Health and Safety Management System Certification

Robot Certification:

Certificate/Model	FR3	FR3C	FR3WM	FR3WM	FR5	FR10	FR10	FR20	FR30	Download
High Protection IP65 Test Certificate										download
NSF Certificate										download
Crash Force Test Certificate										download
RoHS Test Certification Certificate										download
Explosion-proof Certificate										download
CE-EMC Certificate										download
CE-MD Certificate										download
Functional Safety Certificate										download
KCs Certificate										download
SEMI Certificate										download
NRTL Industrial Robot Certificate										download
NRTL Collaborative Robot Certificate										download
Cleaning Robot CLASS3 Certificate								–	–	download

Control box certification:

Certification	Product Model	Download
Control box CE(MD+EMC) certification	FRC100-AC-2000-A, FRC100-AC-5000-A	download

Component certification:

Certification	Product Model	Download
Motor certification	FR60-100, FR60-200, FR80-470, FR130-960, FR130-1500	download
Driver certification	FR01-SD6-14, FR01-SD6-17, FR01-SD8-25, FR01-SD13-32, FR01-SD13-40	download
Power supply certification	FRP-800-4824	download
Core board certification	PETREL	download

CHAPTER

TWELVE

SECONDARY DEVELOPMENT

FAIRINO ROS1

FAIRINO ROS2

CHAPTER

THIRTEEN

ONTOLOGY & DIMENSIONAL DRAWINGS

FR3 Drawings

FR5 Drawings

FR10 Drawings

FR16 Drawings

FR20 Drawings

FR30 Drawings

CHAPTER

FOURTEEN

2D CAD FILES

FRCobots-V6.0 DWG Format

Mini Control Box 2kW (DC+AC) DWG Format

Control Box 5kW (DC+AC) DWG Format

Teach Pendant DWG Format

Button Box DWG Format

CHAPTER

FIFTEEN

3D MODELS

FRCobots-V5.0 STEP Models

FRCobots-V6.0 STEP Models

Mini Control Box 2kw (DC + AC) STEP Models

Control Box 5kw (DC+AC) STEP Models

Teaching Pendant STEP Models

Button Box STEP Models

Palletizing Station STEP Models

Sorting Workstation STEP Models

Spark Project Education Work Platform STEP Models

Intelligent Welding Workstation STEP Models

Envelope Diagram STEP Model

CHAPTER

SIXTEEN

FAIRINO SIMMACHINE

FAIRINO SimMachine VMware

FAIRINO SimMachine Docker

FAIRINO_SimMachine_Software

CHAPTER

SEVENTEEN

PALLETIZER FRCAP

Palletizer-v2.2.0

CHAPTER

EIGHTEEN

CPP SDK

Latest version: `FAIRINO CPP SDK-v2`

Historical Version

CHAPTER

NINETEEN

C# SDK

Latest version: FAIRINO C# SDK-v1

Historical Version

CHAPTER

TWENTY

PYTHON SDK

Latest version: `FAIRINO Python SDK-v2`

Historical Version

CHAPTER

TWENTYONE

JAVA SDK

Latest version: `FAIRINO Java SDK-v1`

Historical Version

CHAPTER

TWENTYTWO

STATUS FEEDBACK PROTOCOL

Robot 8083 Port Status Feedback Protocol

CHAPTER

TWENTYTHREE

COMMUNICATION COMMAND PROTOCOL

Robot Controller Communication Command Protocol

CHAPTER

TWENTYFOUR

ROBOT SOFTWARE

FAIRINO-CobotSoftware-QX-V3.9.3.1-20260304.zip

FAIRINO-CobotSoftware-LA-V3.9.3.1-20260304.zip