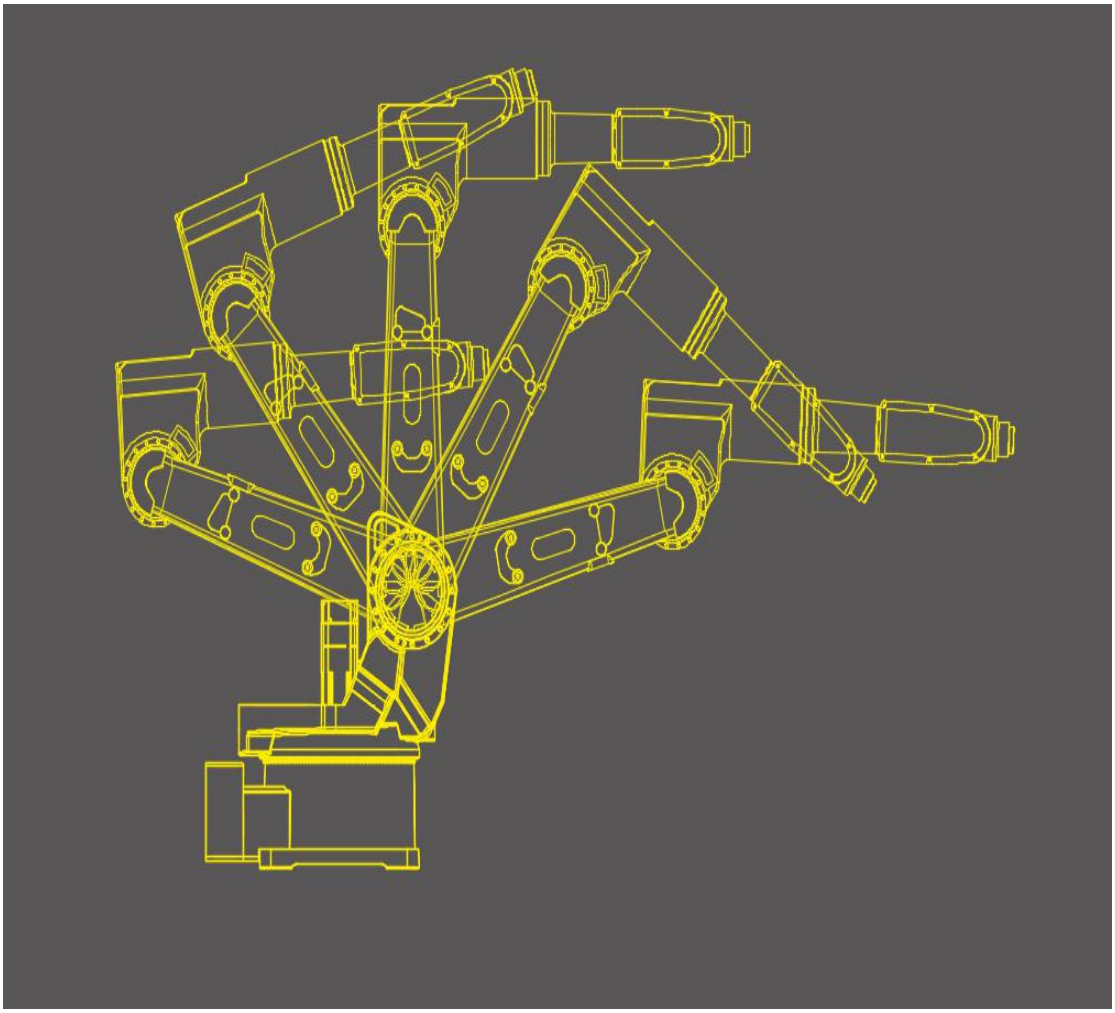
TURIN

Industrial robot Debugging manual



SOT TECH CO.,LTD

www.sotrobot.com

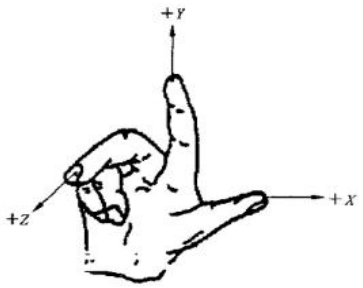
V3.0

Content

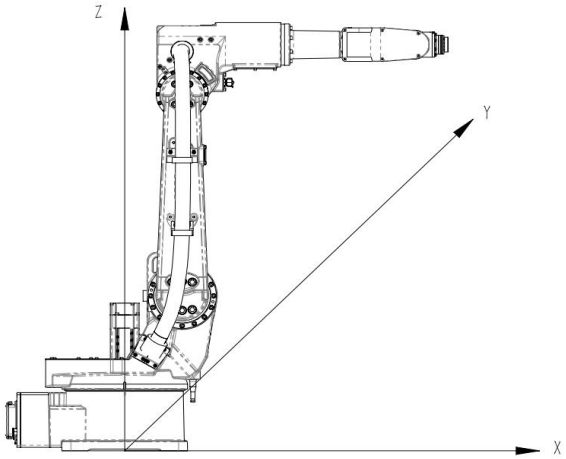
I. Basic concepts

1.1 Coordinate system

1.1.1 Right hand coordinate system



1.1.2 Frame coordinate system

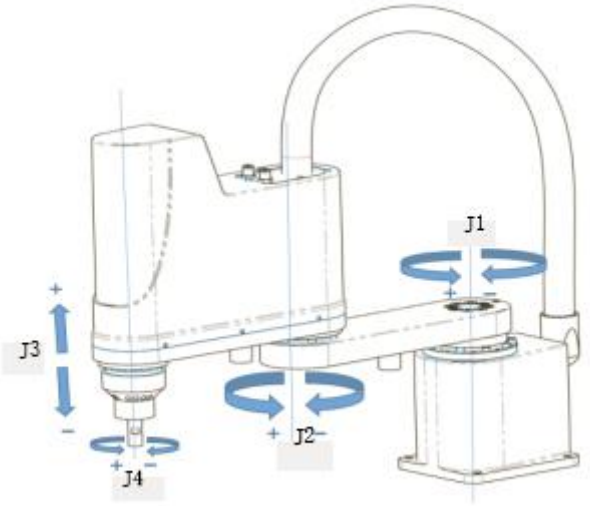


This coordinate system is the reference system for the manipulator to perform linear motion.

The position of the end flange of the manipulator is the coordinate system of the mechanical interface. The rectangular coordinate in the teaching interface describes the translation and rotation between the mechanical interface coordinate system and the frame coordinate system. The x-axis direction of the frame coordinate system is from the origin point to the center of the working area of the manipulator, the z-axis is

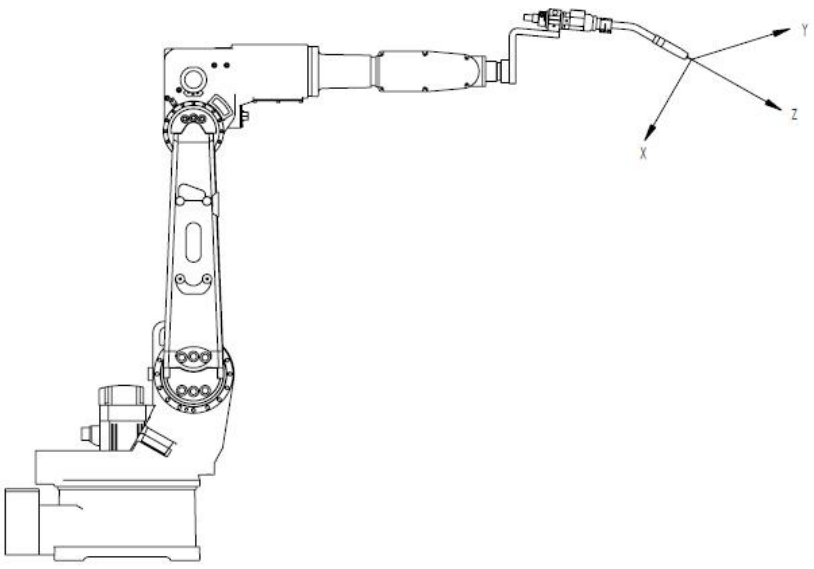
vertical to the frame, and the y-direction of the robot body is established according to the right-hand coordinate system.

1.1.3 Joint coordinate system



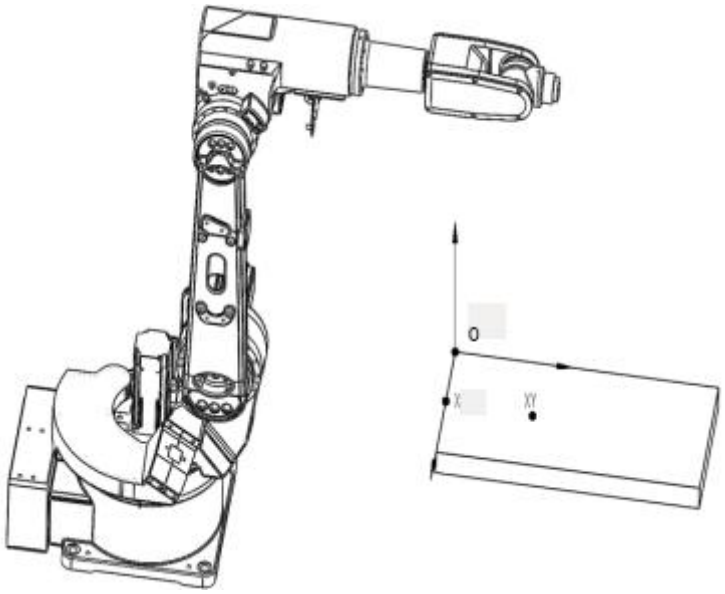
Joint coordinate system is also known as column coordinate system.
Z axis of joint coordinate system is the straight line of rotation axis of corresponding joint. The positive and negative rotation of each joint is determined by the right hand rule, that is, when the joint angle is 0, the palm holds the Z axis of the target joint coordinate system, the thumb points to the positive direction of the parallel axis in the rectangular coordinate system, and the four finger direction is the positive direction of joint rotation.

1.1.4 Tool coordinate system



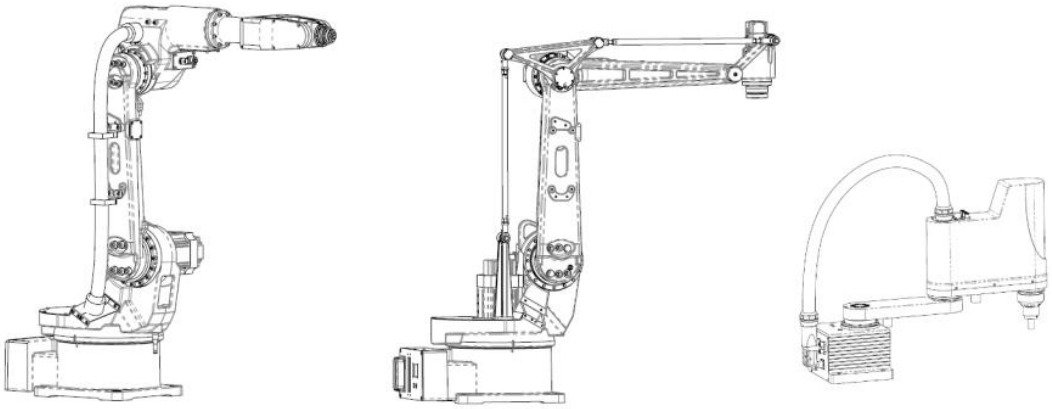
Tool coordinate system is the coordinate system with the end actuator installed on the mechanical interface as the reference system.
The origin o is the reference point (TCP) of the tool, and the Z axis is related to the tool, usually the direction of the tool.

1.1.5 User coordinate system



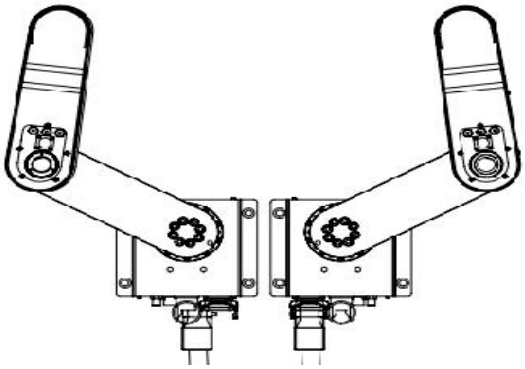
User coordinate system is also known as workpiece coordinate system.
When the user wants to change the reference system (frame coordinate system) of the end actuator, the user coordinate system can be established. At this time, the attitude and orientation of the end actuator are mutually referenced with the user coordinate system.

1.2 Zero position



The above image shows a robot in zero position. You can imagine the state of soldiers standing in the army. In the zero position state, each joint angle of the robot is zero.

1.3 Arm posture of SCARA manipulator

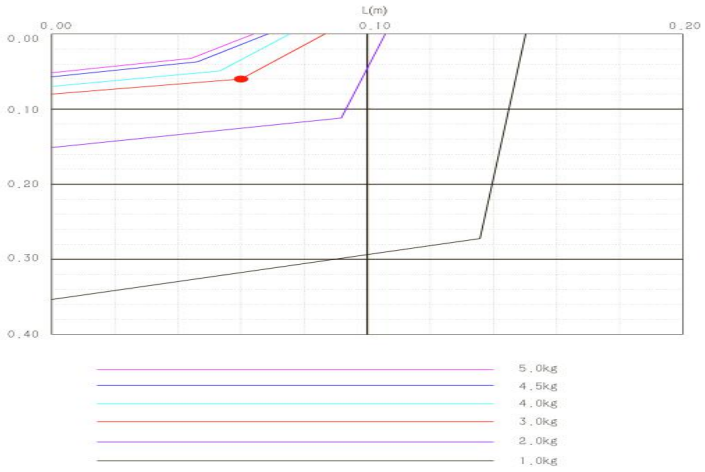


1.3 left hand posture and right hand posture

Manipulator can be divided into left hand posture and right hand posture.

During the operation, the posture of a certain position should be consistent with that of teaching as much as possible. Otherwise, there will be a slight shift in position or unexpected path movement.

1.4 Manipulator load



1.4 load curve of manipulator

The load loaded by the robot must conform to the robot load chart. Overload will overload the motor and reducer, shorten the service life of the robot, and damage the robot in serious cases.

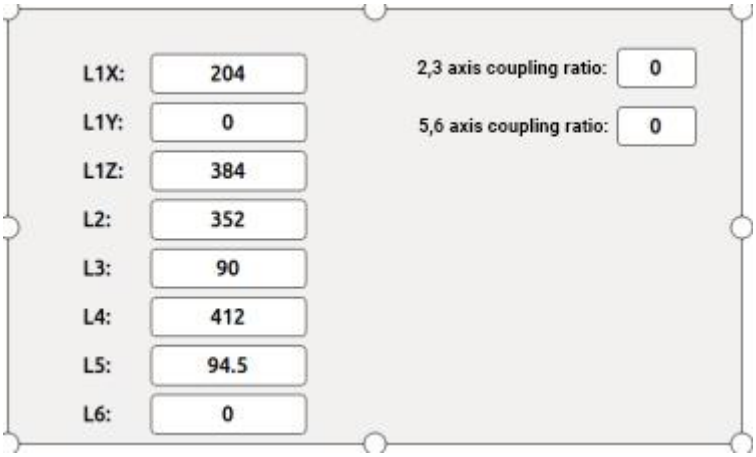
II.Basic parameters of manipulator

2.1 Zero point

The position recording of the manipulator is based on the code value fed back by the absolute value encoder. Zero calibration is to calibrate the position of the motor shaft and the space angle of the manipulator when it is 0.

Zero position is very important for the absolute accuracy of manipulator. During the delivery test, the "20 point calibration method" shall be used for calibration, and the user shall not change it without permission.

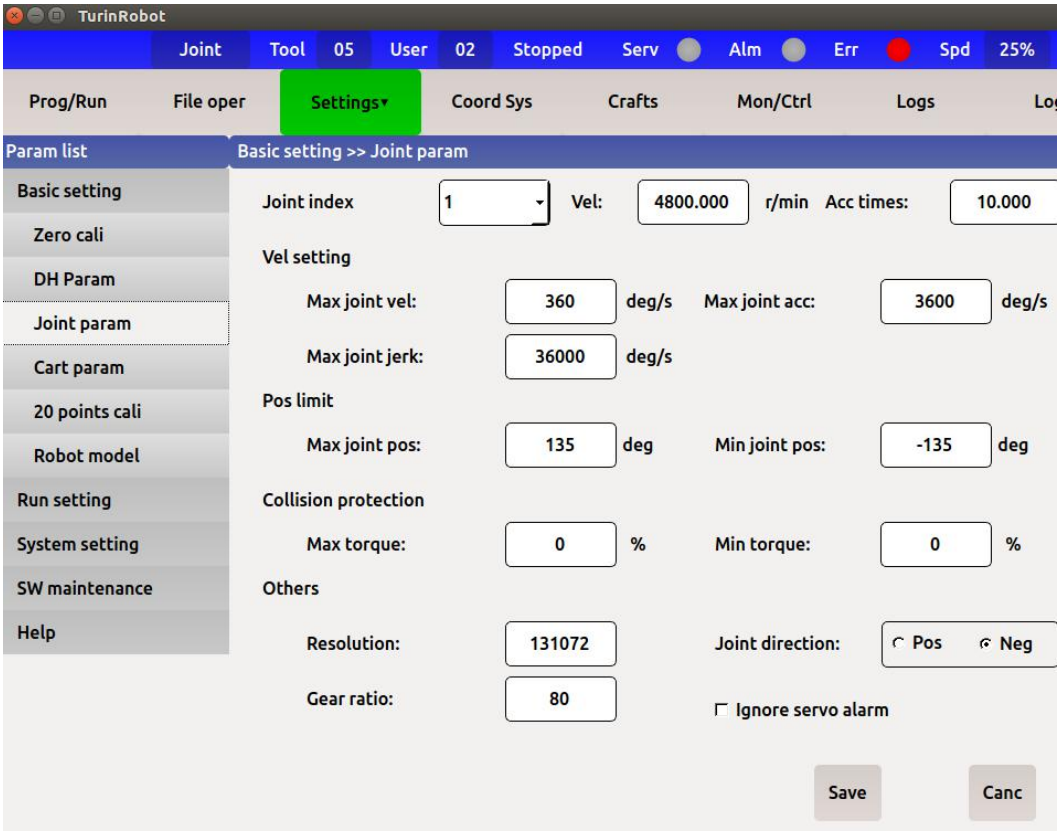
2.2 DH Param



The definition of connecting rod parameters in robotics is no longer represented here. The accuracy of connecting rod parameters is very important for the absolute accuracy of the robot. The manipulator will use the "20 point calibration method" to calibrate before leaving the factory.
Due to the coupling requirements of some joint mechanical transmission modes, that is, the 5 axis rotates independently, and the 6 axis will also rotate at the same time. The correct coupling ratio should be filled in the parameter interface and decoupled by the controller. This allows the target shaft to operate independently.

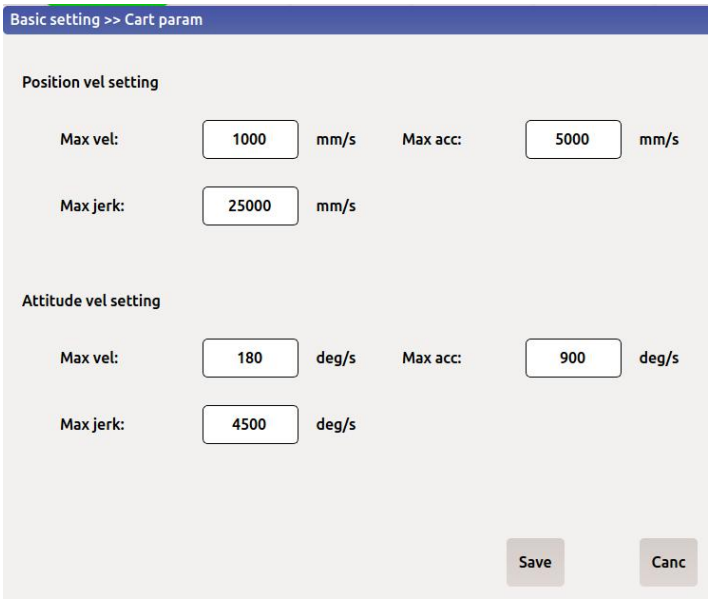
2.3 Joint param

The joint parameters mainly set the movement range and maximum speed of each joint when the arm is running.
The parameters are determined by the actual operating range of the arm in space, the driving performance of the motor and the requirements for the joint coordination in the trajectory planning.
Other parameters in the interface, such as pulse number, deceleration ratio and joint steering, shall not be modified.



2.4 Cart param

Space parameter is the speed limit of manipulator when it is running in Cartesian coordinate system.
In Cartesian coordinate system, there is not only translation, but also attitude transformation, that is, rotation in Rx / ry / RZ direction. The joint linkage is needed during the movement, so the factory setting value of spatial parameters is conservative to meet the safe operation under various working conditions. It is necessary to adjust it properly in actual operation. In the motion of attitude transformation, the controller also limits and protects the actual motion speed of each joint based on the setting of joint parameters.



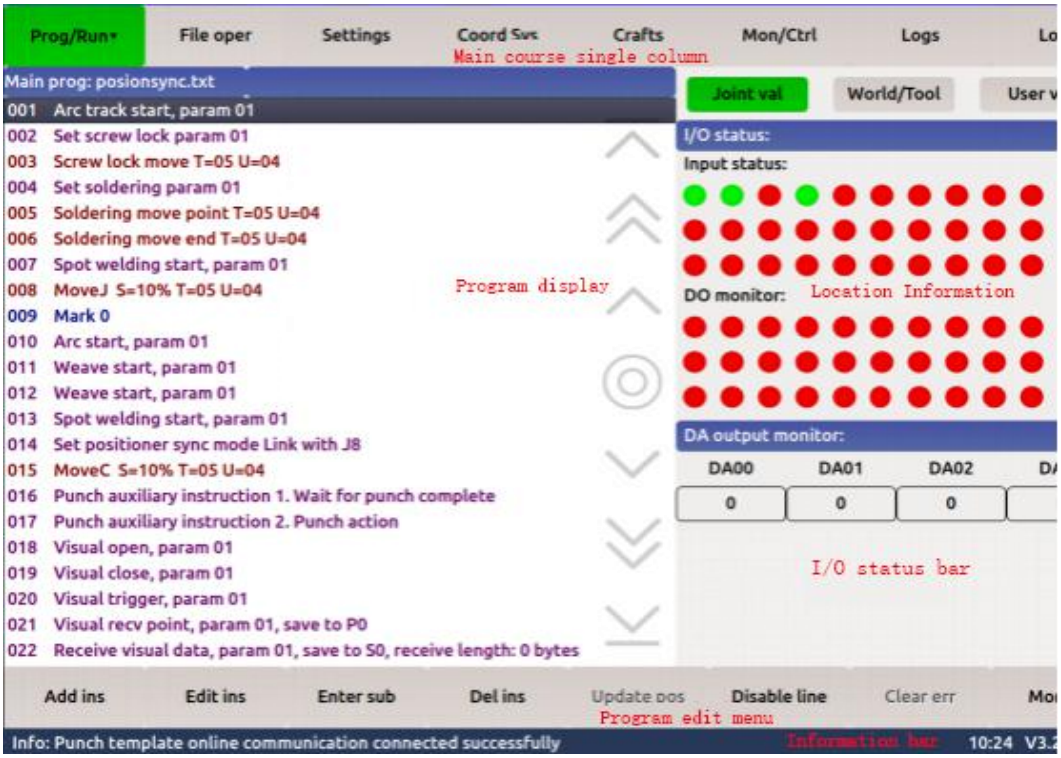
III. Manual manipulator

3.1 Appearance of teaching device

The front of the teaching device is a touch screen, a manual automatic switch key, an emergency stop button and a key. The side is the manual pressure enable switch.

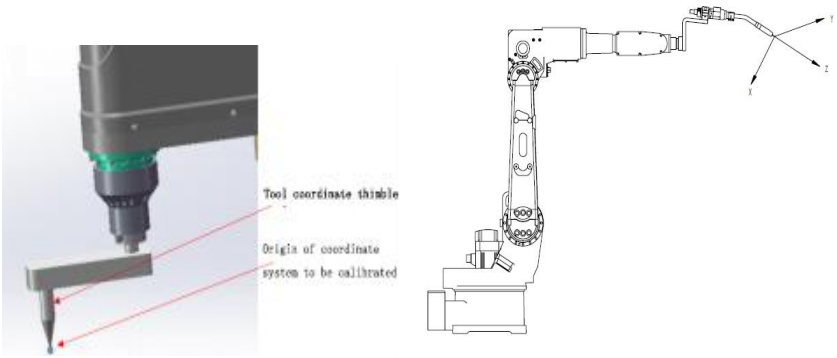
F1 ~ F4 are user-defined keys, which can be defined as IO output control keys. In the joint motion mode, j1j6 controls 1-6 axis motion separately, and in the linear motion mode, it controls the manipulator to move along the X, y, Z, Rx, ry and RZ directions.

3.2 Teaching device user interface



3.3 Establish tool coordinates

The coordinate system is defined on the tool by the user. The origin of the tool coordinate system is also called TCP. Generally, the effective direction of the tool is defined as the z-axis direction of the tool coordinate system, the x-direction is defined by the user, and the y-direction is defined by the right-hand coordinate system.



3.3 schematic diagram of tools

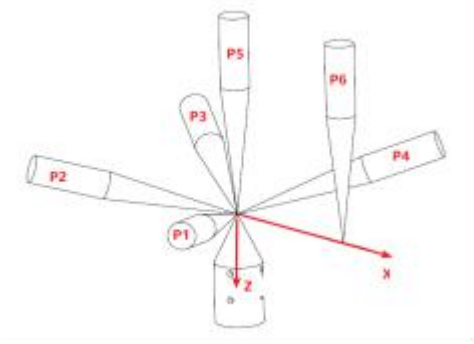
SCARA manipulator TCP calibration mode:

1. Check the zero position, DH and deceleration ratio of the robot. The accuracy of these parameters directly affects the accuracy of TCP.
2. Move out the zero position of axis 2, avoid the singular point, adjust RZ to 0 radian, and then align the tool tip with the tip of thimble (since then, the thimble can not be moved), and record the robot rectangular coordinates (x1, Y1).
3. In the right angle motion mode, adjust RZ to 3.14 radians, align the tip of the thimble again, and record the right angle coordinates (X2, Y2) of the robot at this time.
4. Calculation

$$x = \frac{\mathrm{X2} - \mathrm{X1}}{2} y = \frac{\mathrm{Y2} - \mathrm{Y1}}{2}$$

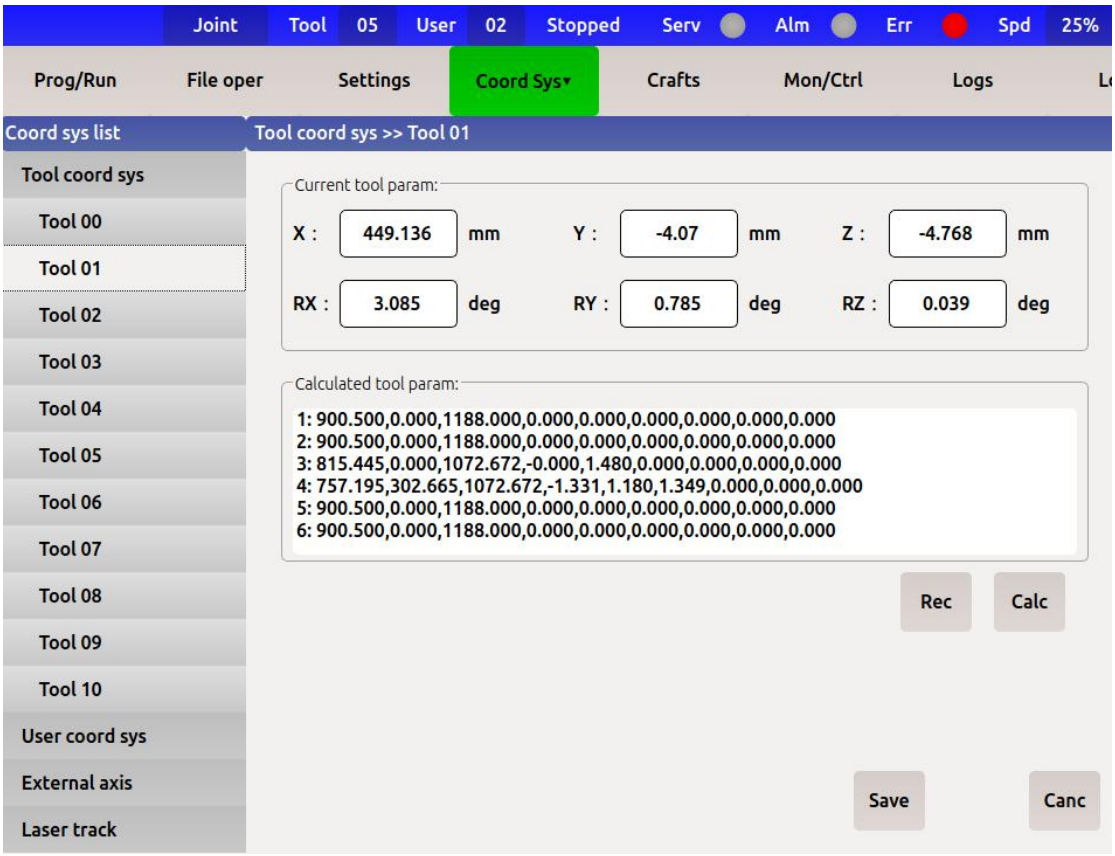
X. Y is the tool coordinate parameter. Fill in the tool coordinate system.

TCP calibration mode of six joint manipulator:

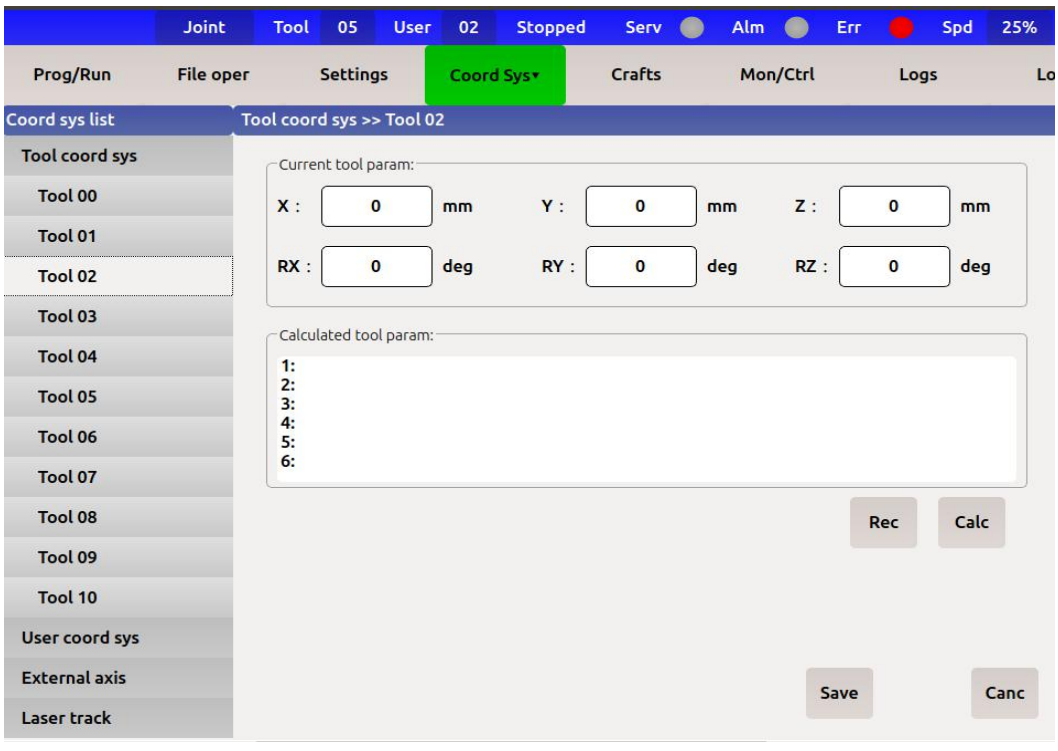


1. The tip of the tool is aligned with a relatively stationary reference point for attitude swing.
2. The first four points should swing as much as possible, the fifth point should calibrate the Z direction, and the sixth point should calibrate the X direction.

3. After the point position is recorded, calculate and save.



3.4 Establish user coordinates



Setting of user coordinate system (3-point method)

1. Determine the required user coordinate system.
2. Move the robot with operation tools to the origin of coordinate system and record the point.
3. move the robot with operation tools to any point on the x-axis, record the point, and then calibrate the x-direction vector.
4. Move the robot with operation tools to any point in the first quadrant in the XY plane, and record the point.
5. Save the calculation results.

IV. Instructions for manipulator

4.1 Motion command

Generally, motion instructions record position data, joint types and motion speed.
Joints specify the trajectory of the motion between teach points at execution time. Robots generally support three types of motion: joint motion (moveJ), linear motion (moveL) and circular motion (MOVC).
To add a motion command, press the enable switch to enable the driver.

4.1.1 Joint joints

Joint motion can also be understood as free motion, that is, the user only gives the end point, and the way of arrival is planned by the trajectory planner. It is characterized by the fastest speed and unknowable path. Therefore, this type of motion is generally applied to space points. Before the automatic operation of the program, it is necessary to check it at a low speed to observe whether the actual motion track of the robot interferes with the surrounding equipment.

4.1.2 Straight joints

When the robot needs to move to the current teaching point through the straight-line path, the straight-line motion type is adopted.

4.1.3 Arc joints

When the robot needs to move to the current teaching point through the arc path, the arc motion type is adopted. The motion command corresponding to arc joint is [arc motion] (English command is MOVC).

Three motion points are required for arc motion. The first arc motion starting point is a non arc command, passing through the first point (auxiliary point) to reach the second point (end point). **If multiple arc motions are connected, the end point of the previous arc motion is the start point of the next arc motion, and the start point of the second arc is the arc command.**

```
001   Joint Movement S=100% A=100%
002   Circular Motion S=10% A=10% B=100
003   Circular Motion S=10% A=10% B=100
```

4.2 Variable operation

4.2.1 Variable specification

1. Variable type

There are three types of variables that can be used in the control system, namely, digital variable (floating-point number), character variable (1024 bytes), and coordinate point variable (space and joint are saved respectively, that is, there can be no kinematic relationship between them).

2. Variable scope

To be exact, there are three scope variables in the system, which are system variable, global variable and file variable.

System variable: the system variable life cycle is created and destroyed as the system is started / stopped. System variable is a common variable in the whole system, that is, the variable that can be used by the motion script and background script program at the same time. System variables are not initialized by the system because of which program is started or stopped. That is, unless the script or network (such as TCP server) is deliberately modified, the system variables will never be changed by the system.

Global variables: the life cycle of global variables is created and destroyed with the start and stop of this script. Global variables are shared within the script started this time. That is, the sub file and the main file are shared. Other places (such as between the front and back office, network, etc.) cannot access this variable.

File variable： It is also called local variable. Its life cycle is the same as the global variable. The difference is that it cannot be shared between the main file and each sub file.

3. Definition of variables

Predefined system variables:

1）V0 ~ v999: 1000 system digital variables. It should be noted that V0, v00 and v000 represent the same variable for writing convenience. For example, V011 and V11 are the same variable. All system preset variables are the same. However, in order to look comfortable, please use the format of v000 when programming manually.

2）S0~S999：1000 system character variables.

3）P0~P999：1000 system coordinate point variables.

Predefined file variables：

1）FV0～FV999：1000 file numeric variables。

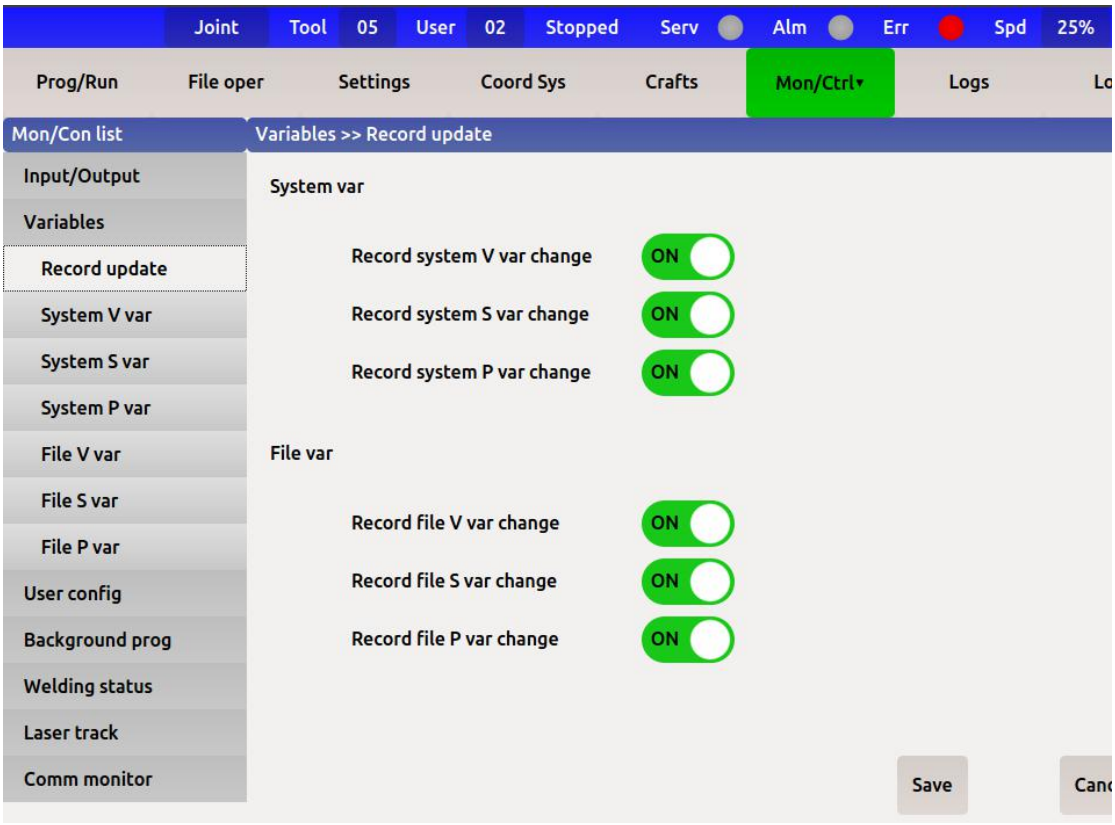
2）FS0～FS999：1000 file character variables.

3）FP0～FP999：1000 file coordinate point variables.

Note: as mentioned before, the scope of file variables is a single file. Even if the name is the same, i.e. each main file and each sub file have 1000 FV, FS, FP and so on (a total of 3000 variables), they cannot be shared. The same variables use their own memory.

4. Variable monitoring

It can monitor the change of variable content in the system in real time



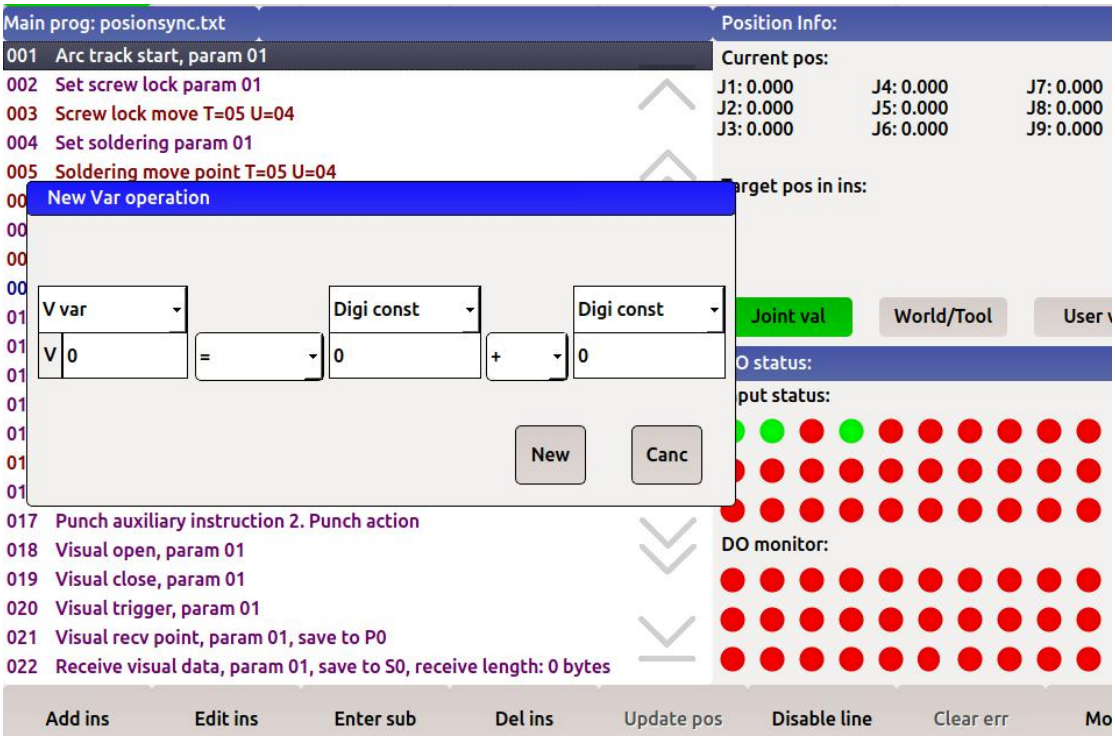
When the corresponding variable is automatically stored as on, the system will monitor that the variable has been used in the script at any time. When its value changes, it will be immediately stored in the file. So that you can use the changed value the next time you start the script.

Note: because the current hardware system does not have power-off protection measures, when the program is running, the abnormal power-off behavior (such as turning on the power supply) may appear that the value just changed is not even stored in the file. This needs special attention.

4.2.2 On the digital V variable and its use

The V variable stores floating-point numbers. Through variable operation, add, subtract, multiply, divide and take remainder.

In practice, V variable is often used to count and store coordinates.



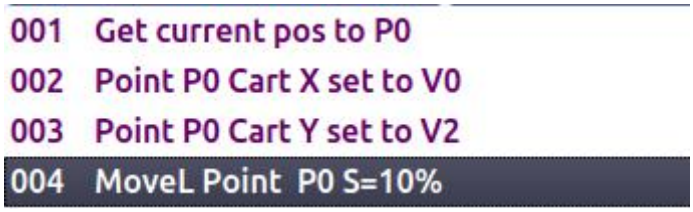
4.2.3 About the use of coordinate point P variable

P variable stores the position information (joint coordinate system, spatial coordinate system) of a coordinate point in the cylindrical coordinate system and Cartesian coordinate system.

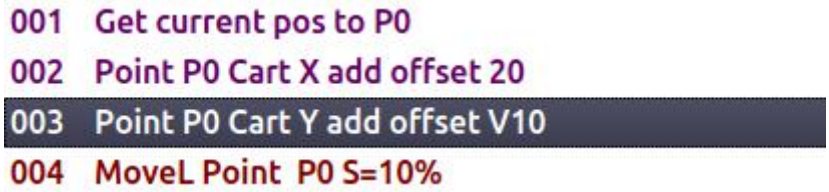
There are generally two ways to use the program.

The first is to directly call the stored P-point to perform the movement, so that it is convenient to modify and debug when the program runs at the same location for many times.

The second is to save the current position to P point, then modify the coordinates of P variable, and then call. In practice, it can be used to perform relative offset or obtain absolute coordinates through communication. Refer to the figure below for the case.



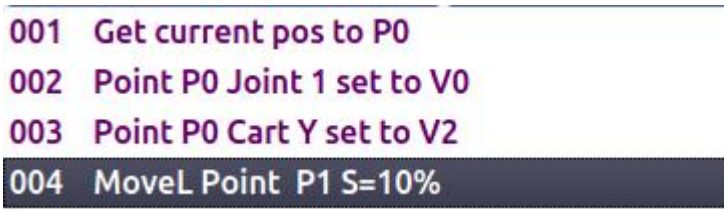
4.2.3-1 Coordinate point modification



4.2.3-2 Coordinate point plus offset

4.2.3-3 Coordinate point copy

Counter example



Note:

1. The variable names of the modified P variable, the stored P variable and the P variable executing the motion should be consistent.
2. The rectangular coordinate is modified to use the straight line command to execute the motion, and vice versa.

Add: Several background instructions corresponding to the foreground instructions of the P variable

1. Copy Coordinate point

需要添加照片

2. Read Coordinate point

需要添加照片

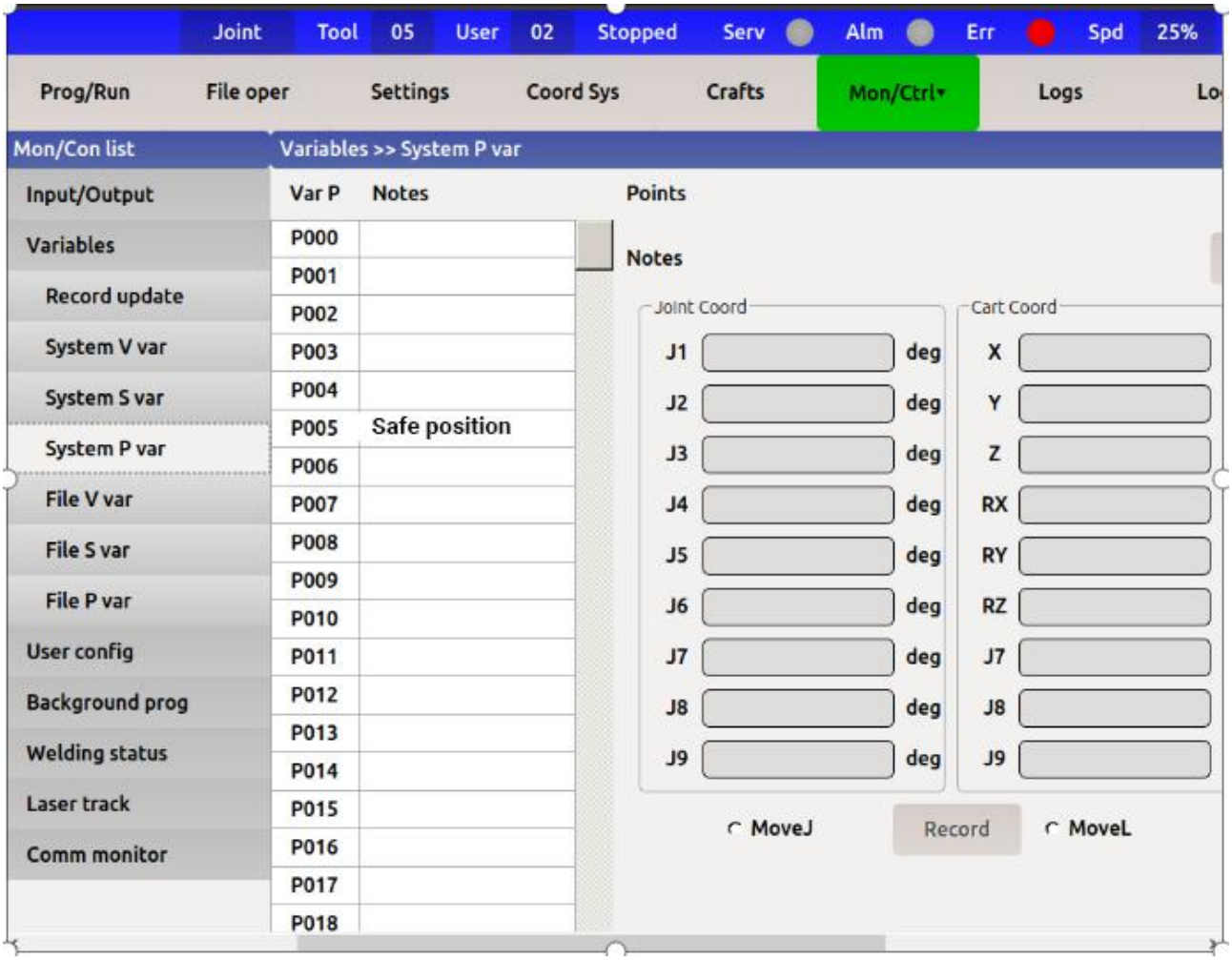
3. Modify the coordinate point

需要添加照片

4.Return data for each joint or each axis of the current point and the target point

需要添加照片

4.2.4 Understanding of P variable

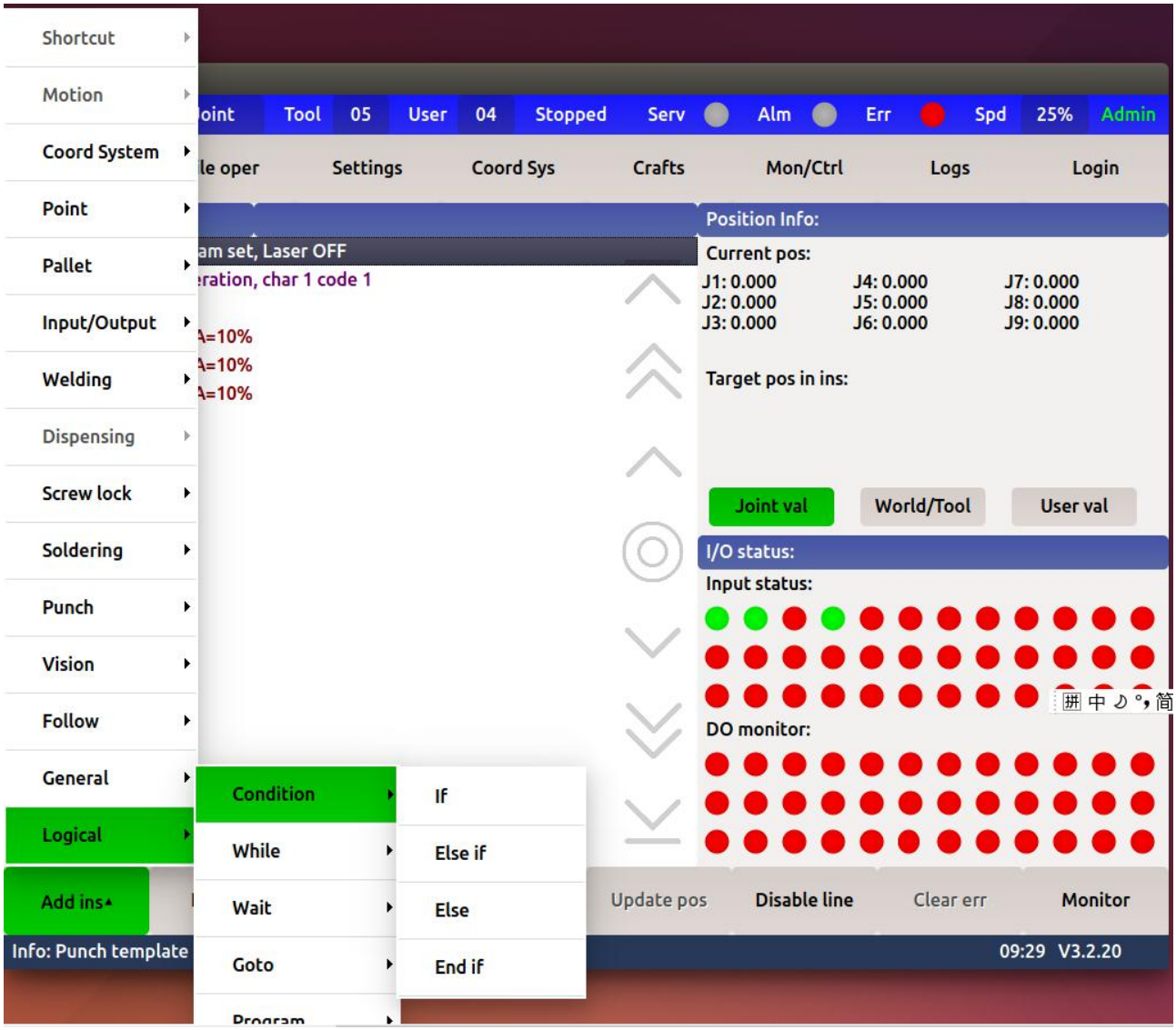


The movement of manipulator is the movement from coordinate point to coordinate point. The coordinate point is stored in the P variable of the Turin controller.

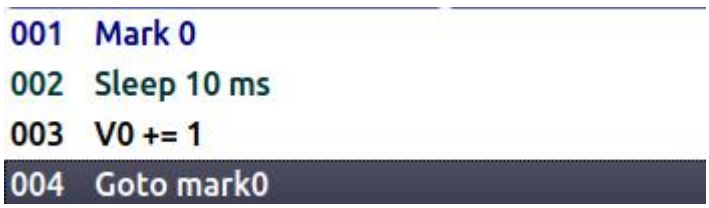
When you need to, you can write a comment on the variable address so that it is clearer when debugging the program. First, control the manipulator to the target position, select the corresponding P variable in the up enable state, and then click the record point. In this way, all the location information of this point will be stored in the target variable.

When the manipulator is manually controlled to the corresponding P point, there are two operation modes. You can select the movement mode to reach the target position by selecting "teaching space" or "teaching joint". One is to follow the straight-line track in the Cartesian coordinate system, and the other is to run more freely based on the joint ratio in the cylindrical coordinate system. In the upper enabling state, select the desired point on the graphic interface, then press the start button, and the manipulator can run to the target position.

4.3 Logical instruction



4.3.1 Jump



Jump unconditionally. When the program executes "jump to mark *", it will jump to the corresponding mark. To jump to "tag 0", there must be and only one "tag 0" in the program corresponding to it

4.3.2 Conditional statement

```
001  If V0 == 8
002  Sleep 500 ms
003  I/O output 04=ON ( no comment)
004  Else if V0 == 5
005  Goto mark0
006  Endif
```

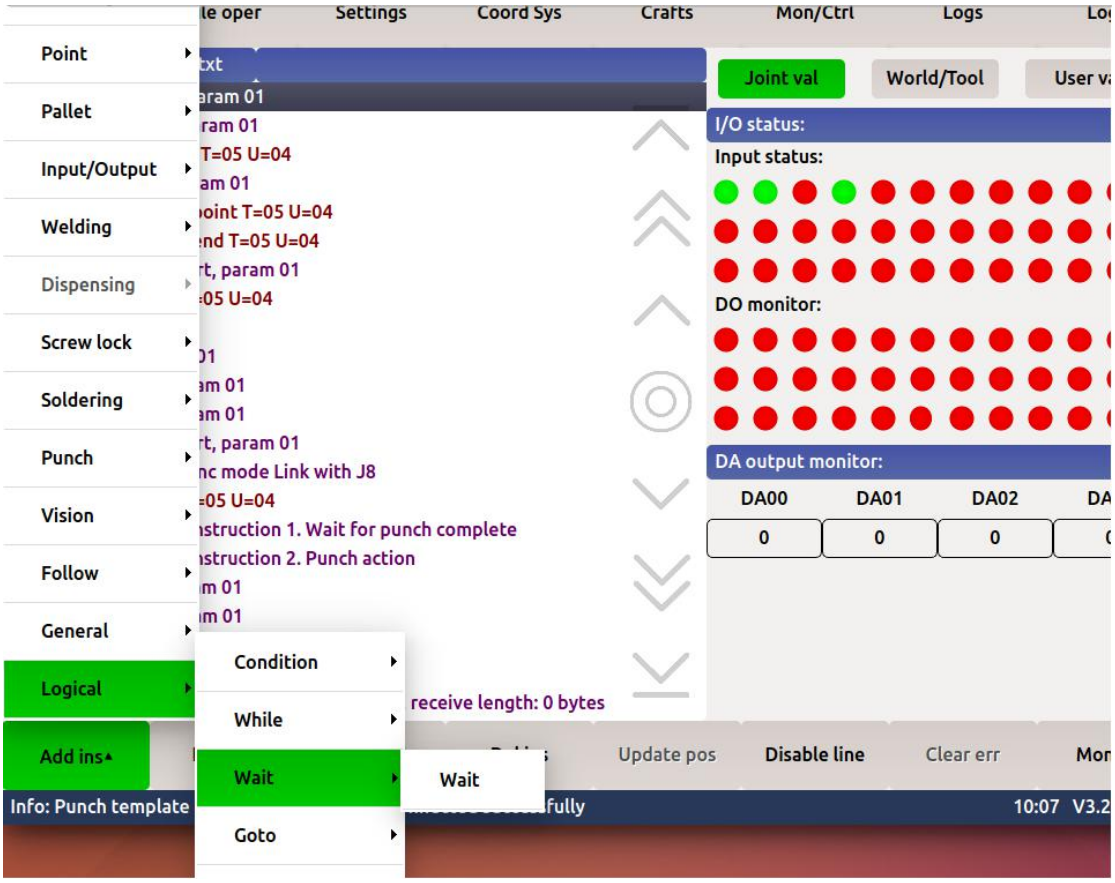
Condition statements are nested. When a conditional statement is used, there must be a conditional end. Otherwise, it cannot be used alone.

4.3.3 WHILE Loop statement

```
001  V0 = 0
002  While V0 < 9
003  Sleep 50 ms
004  MoveJ  S=10%
005  MoveJ  S=10%
006  End while
```

In this case (V0 < 9) is the loop condition expression, and the statement between while and endhide is the loop body. As long as V0 < 9 is met, the content in the loop body is repeated.

4.3.4 WAIT Sentence



Waiting means waiting for a certain state to meet the preset conditions before the program will continue to execute. For example: wait for the level change of IO signal, wait for the value change of digital variable。

4.3.5 SWITCH Sentence

图片

In this case , SWITCH(V1), when the value of V1 corresponds to 1, 2, 3, 4, 5, the instructions in the corresponding case will be executed, and not met, the instructions in default will be executed.

图片

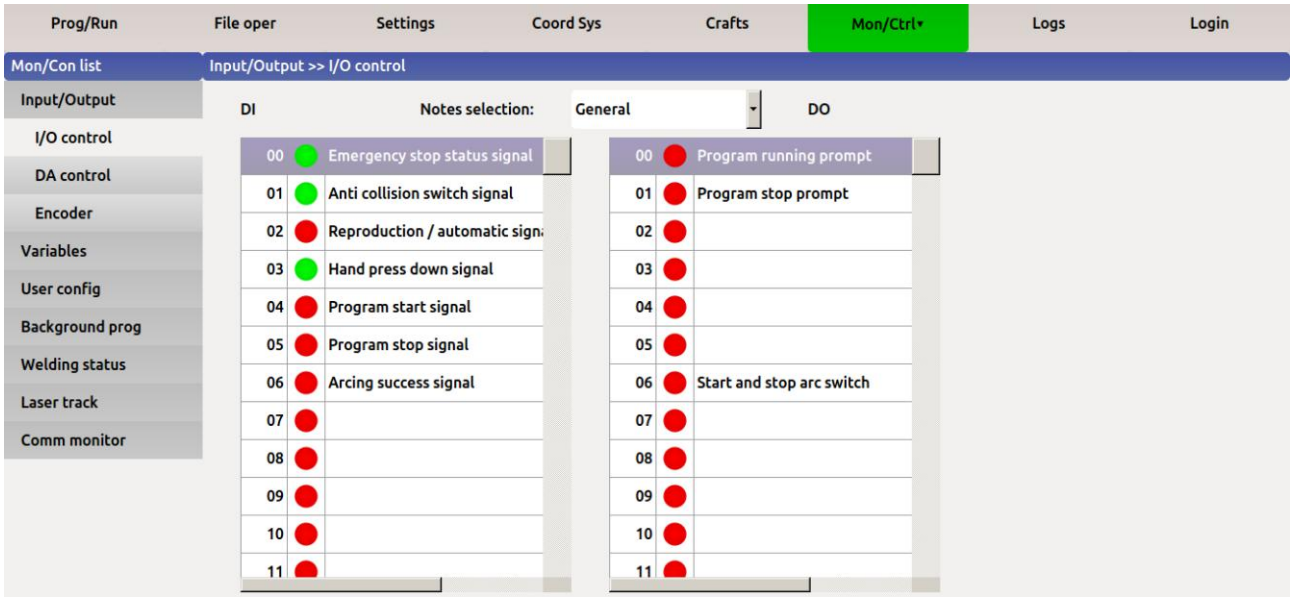
Similar to SWITCH, the S variable can also be used. In this case, when S1 is the corresponding character, the corresponding command is executed. If not met, the command in default is executed.

4.3.6 FOR Sentence

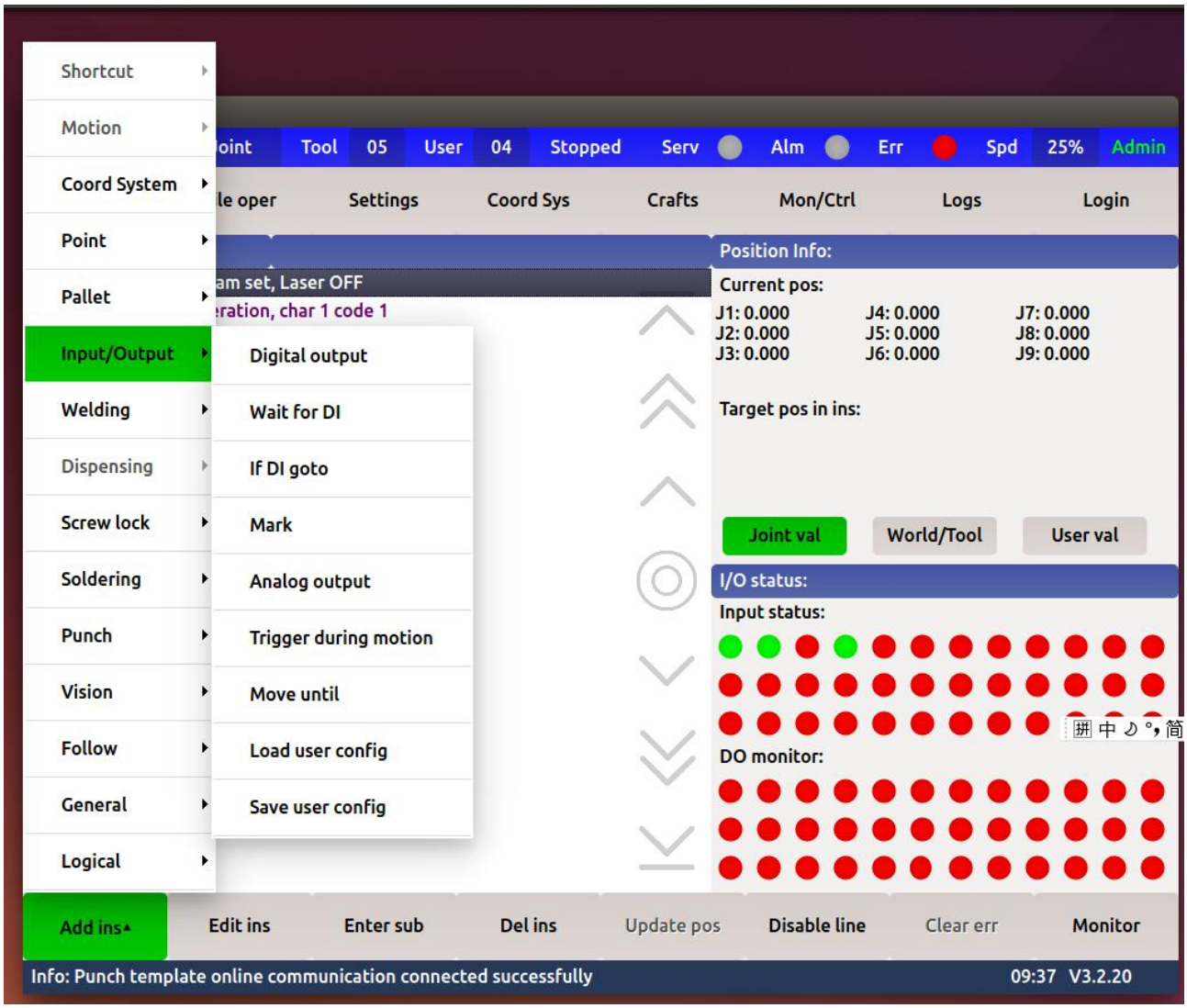
```
001  DIGITAL I = 0
002  FOR(I = 0;I < 12;I++)
003  DO[I] = 0
004  ENDFOR
```

In this case, DO0~DO11 are all set to 0 in the FOR loop statement.

4.3.7 About IO operation

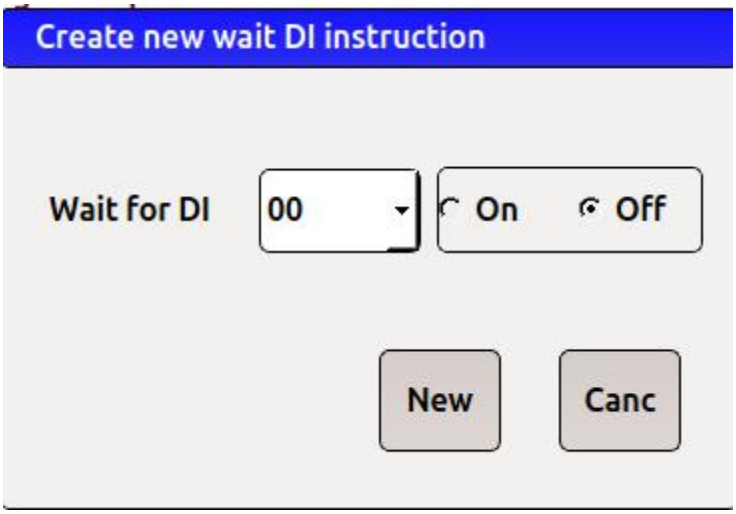


The IO monitoring interface can be used to force on the output or monitor the input of IO

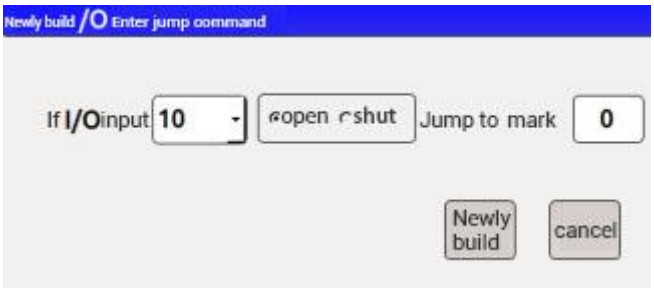


1.Control IO output

Control target IO output port status。

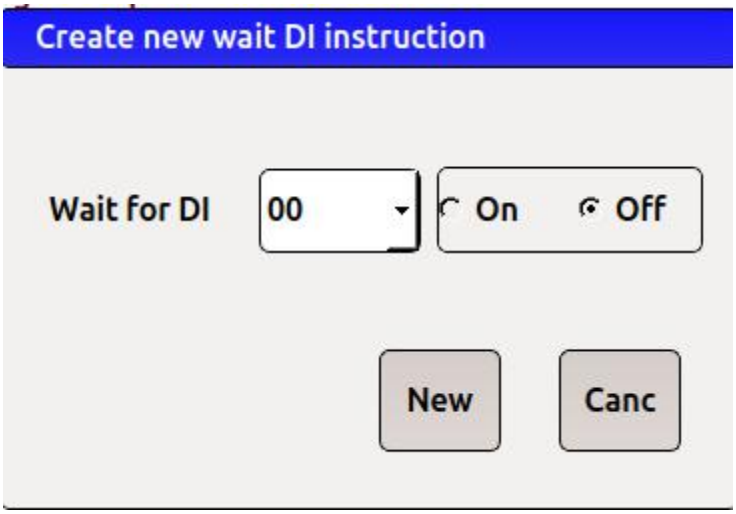


If the reverse function is set, it can be automatically reversed after a certain time delay.



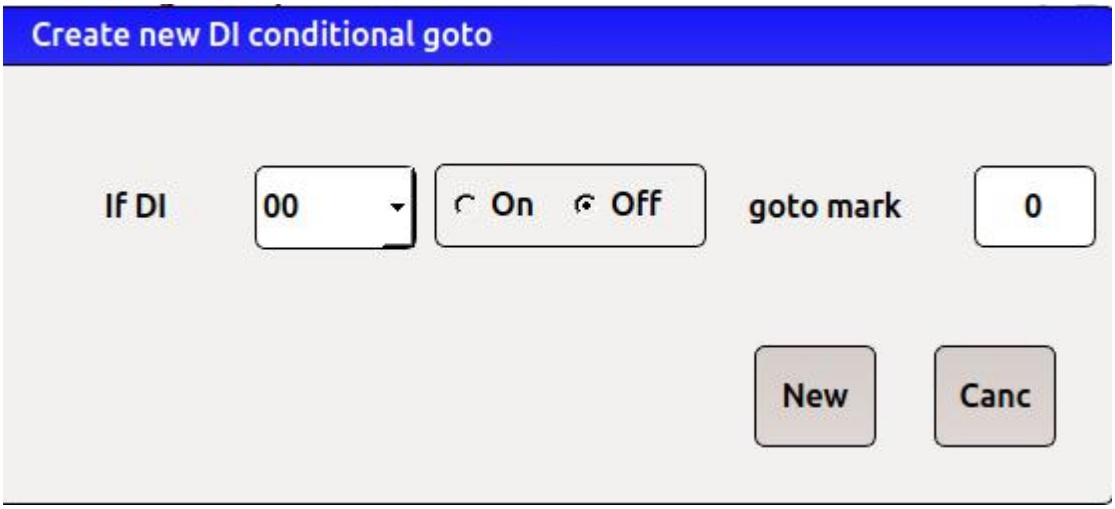
2. Waiting for IO input

The program is blocked here. When the IO status meets the setting conditions, the program continues to run。



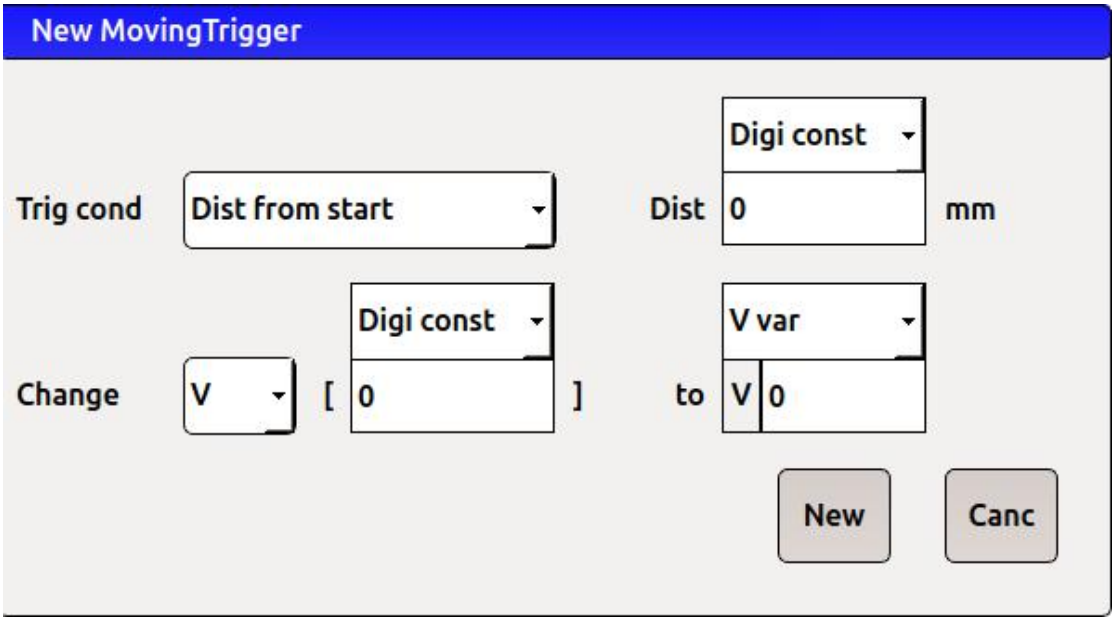
3.IO Input jump

When the target input port meets the setting conditions, the program will automatically jump to the corresponding mark.



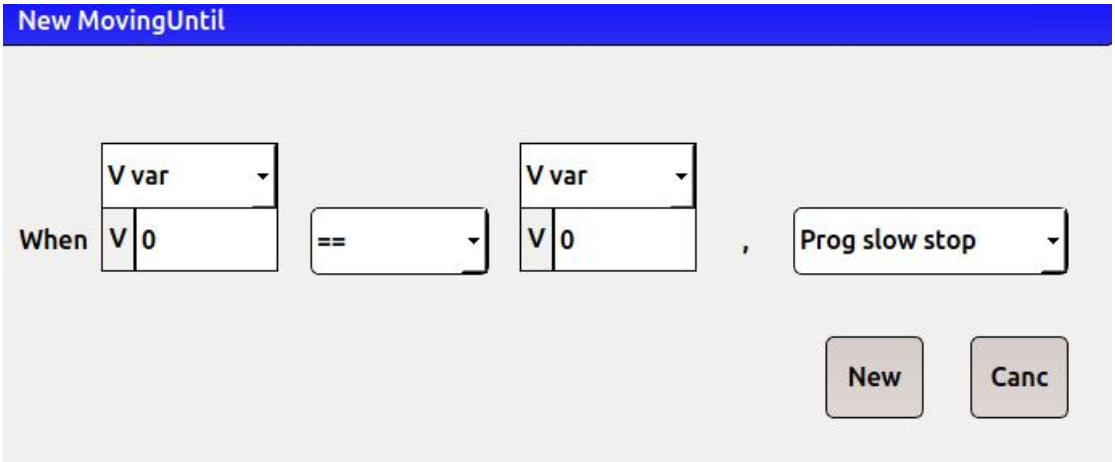
4. Trigger action in motion

The command is added before the motion command. Corresponding actions can be performed during the operation of motion instructions, such as IO output, variable change, etc。



5. Trigger stop in motion

During the execution of motion instructions, once the conditions set in the "trigger stop" instruction are met, the robot can stop in a set way.



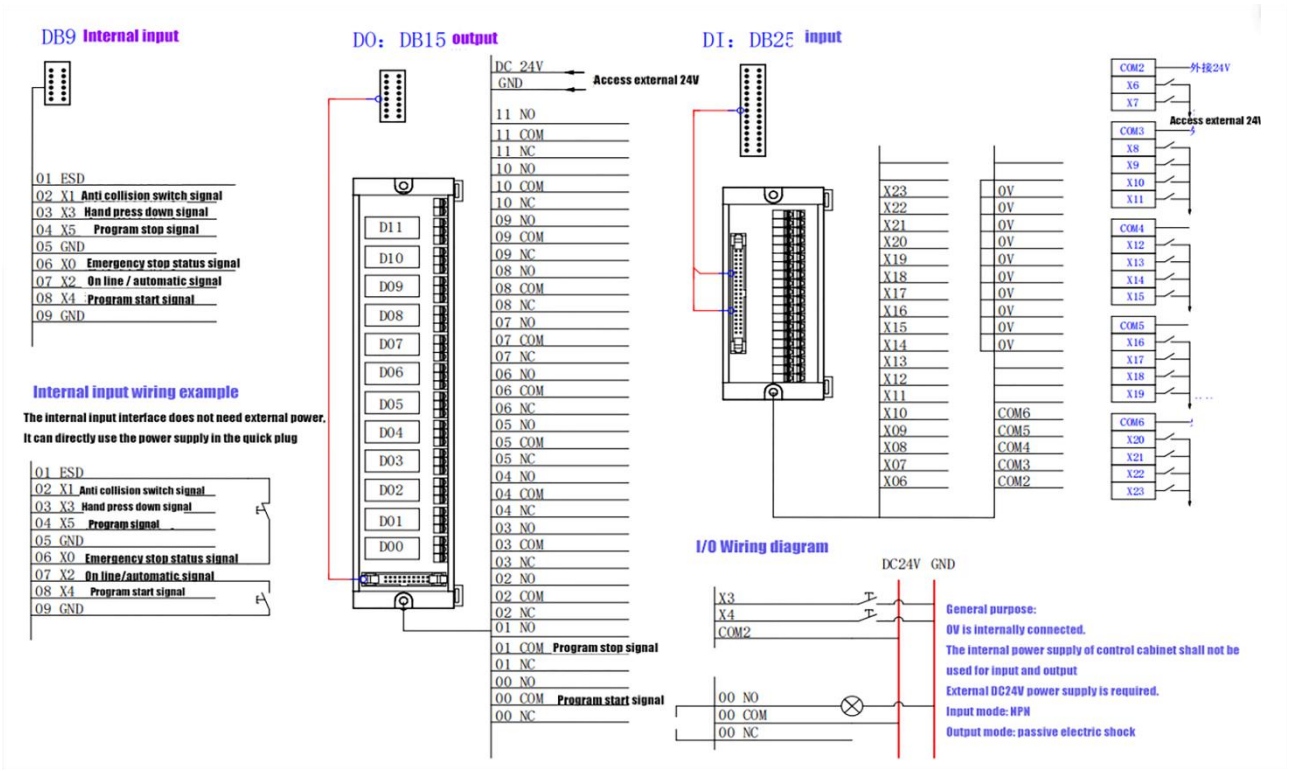
When the program executes the "world incremental motion" command, when the value of the numerical

variable v995 changes to 1, the manipulator will gradually slow down and stop moving.

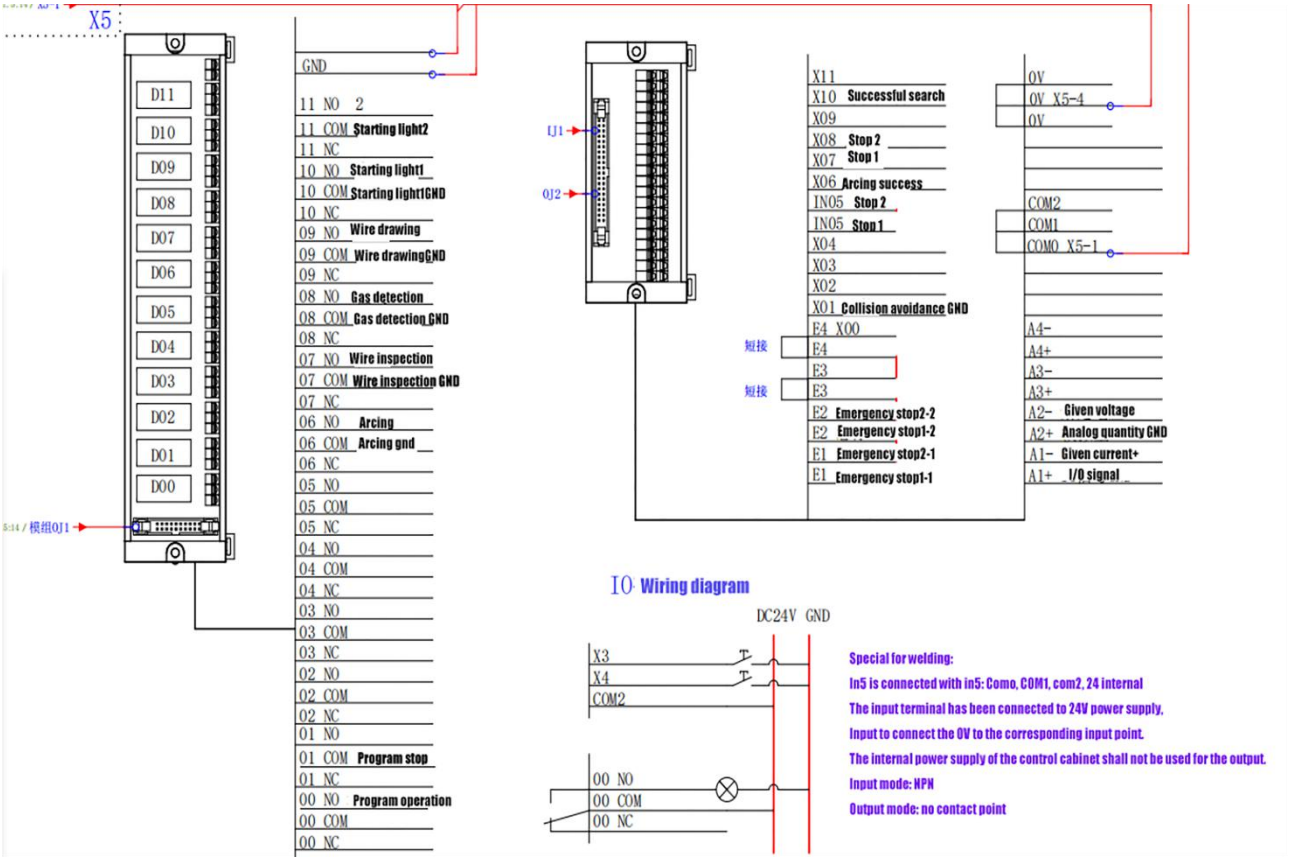
```
001    V995=0
002    Joint Movement S=100% B=0 T=00 U=00
003    World Incremental Movement X=300 Y=0 Z=0 S=2% A=2% B=0 T=00 U=00
004    Trigger In Motion，When v995 = = 1, command deceleration stop
```

6. Basic IO signal wiring

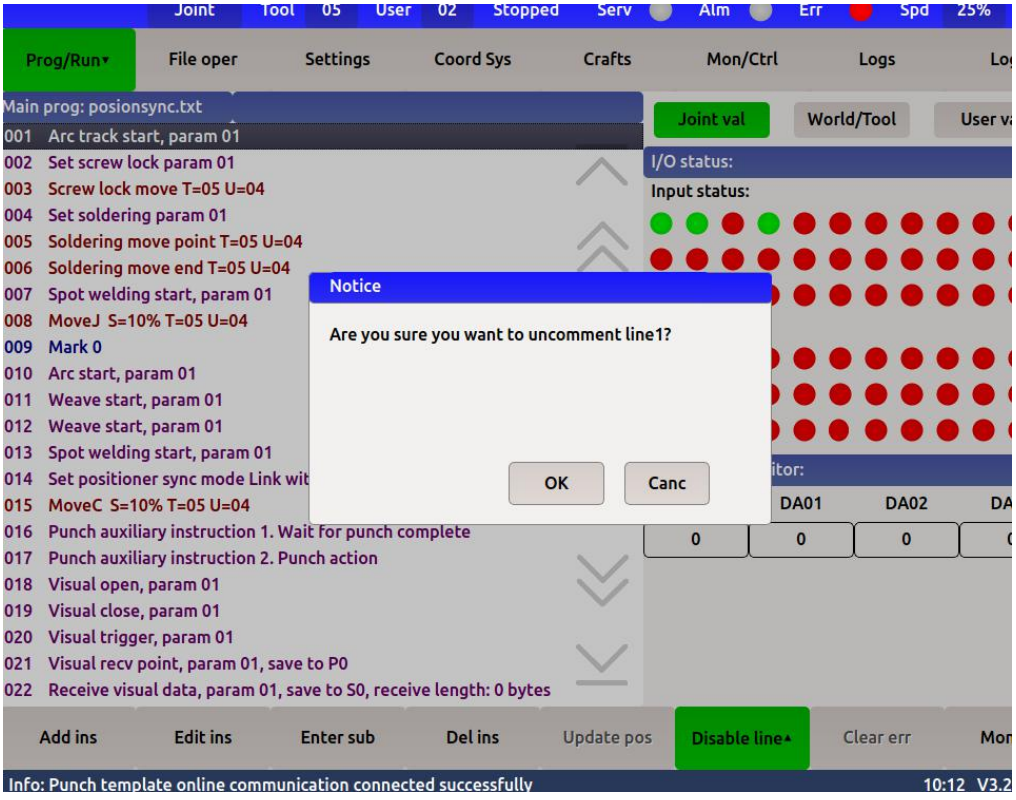
A04 control cabinet wiring (SCARA control cabinet)



B06 control cabinet wiring (standard welding version)



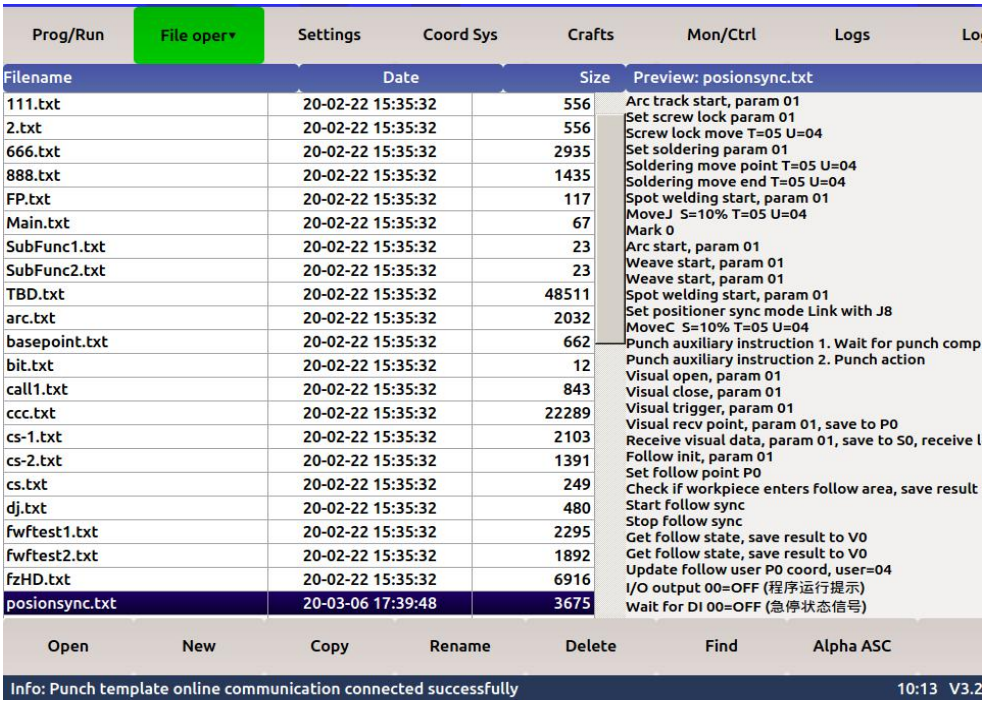
4.3.8 Mask and hide instructions



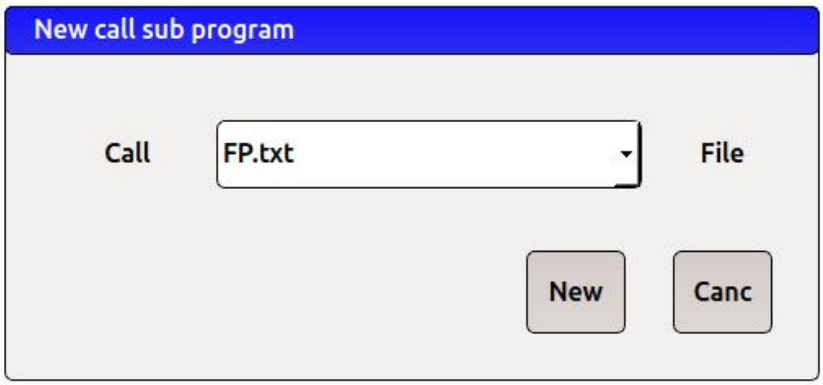
The second coordinate point in the figure is hidden, and is automatically ignored when the program is running.

4.4 Program instruction

4.4.1 subroutine



The controller program is based on the TXT text. Generally, the priority of each program is the same.



Only when the "call subroutine" statement appears in the program, the system says that the current program is defined as the main program, and the called program is divided into primary and secondary, which is distributed in a tree.

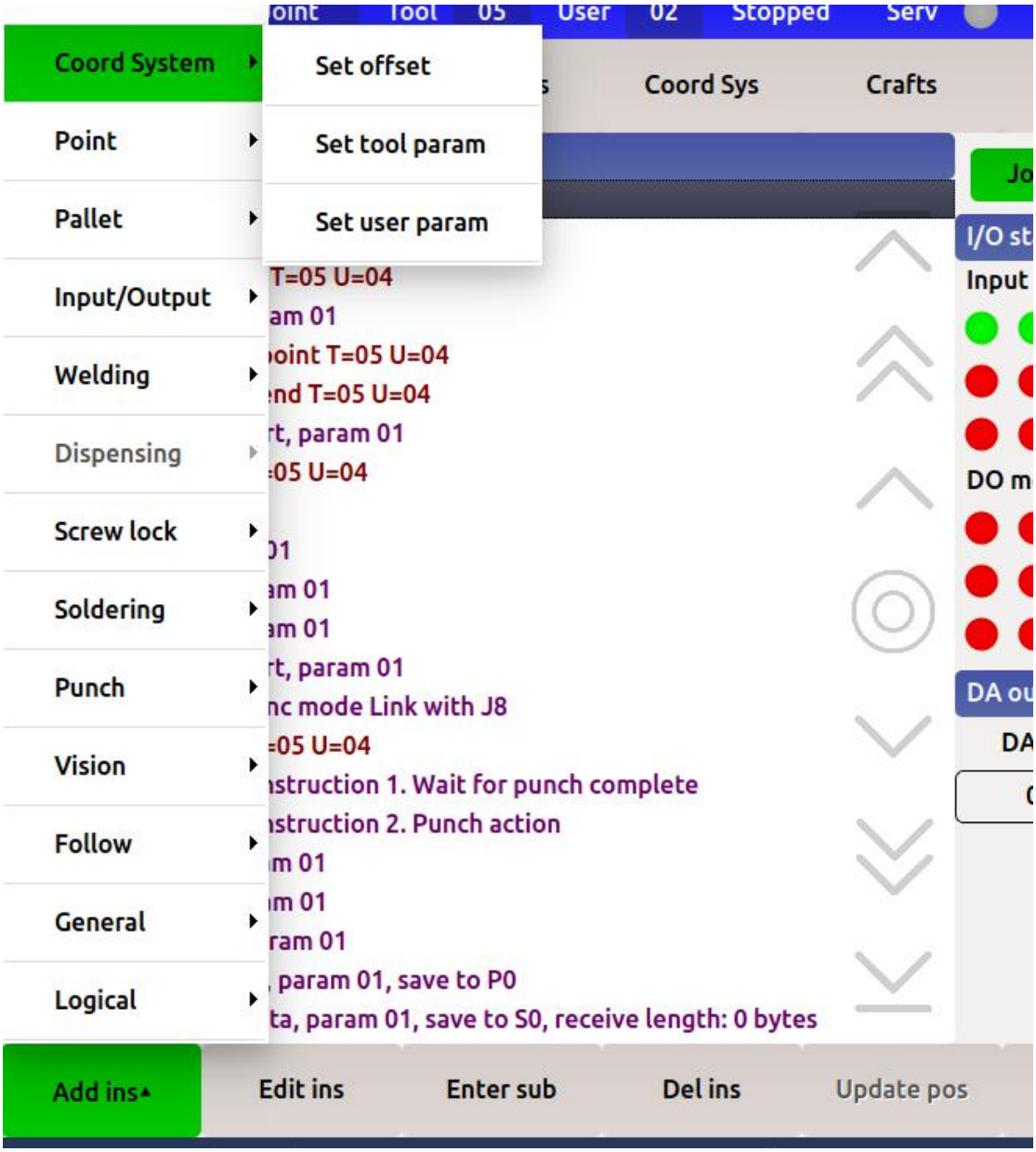
In use, only the subprogram can be understood as a hidden paragraph. This can increase the readability of the program。

005 Return

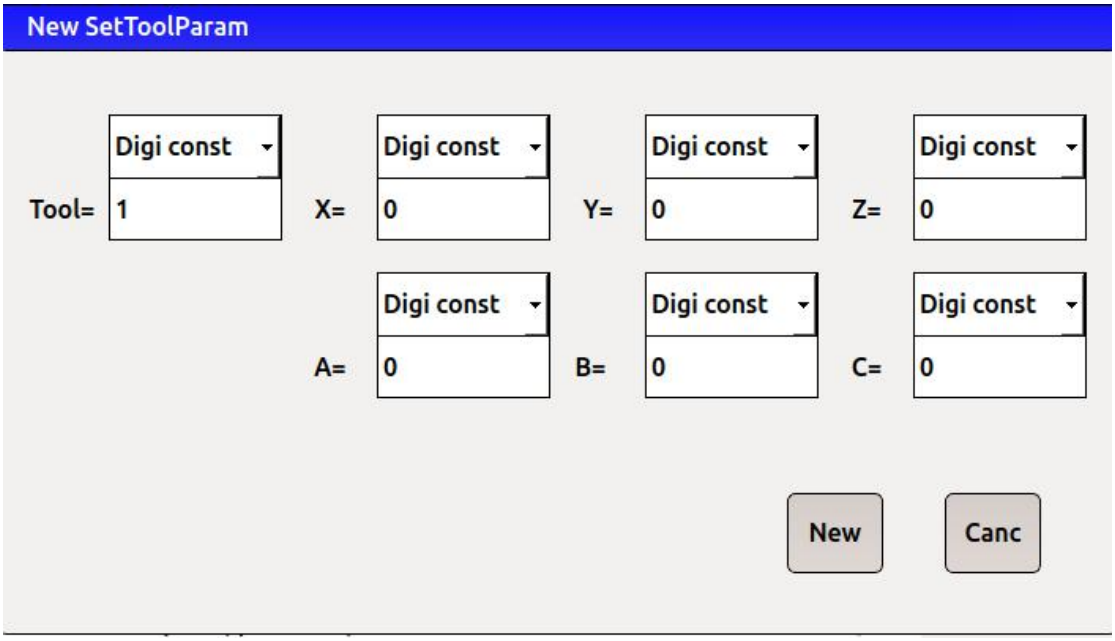
At the end of subroutine operation, add "return" instruction and return to main program.

4.5 Coordinate system command

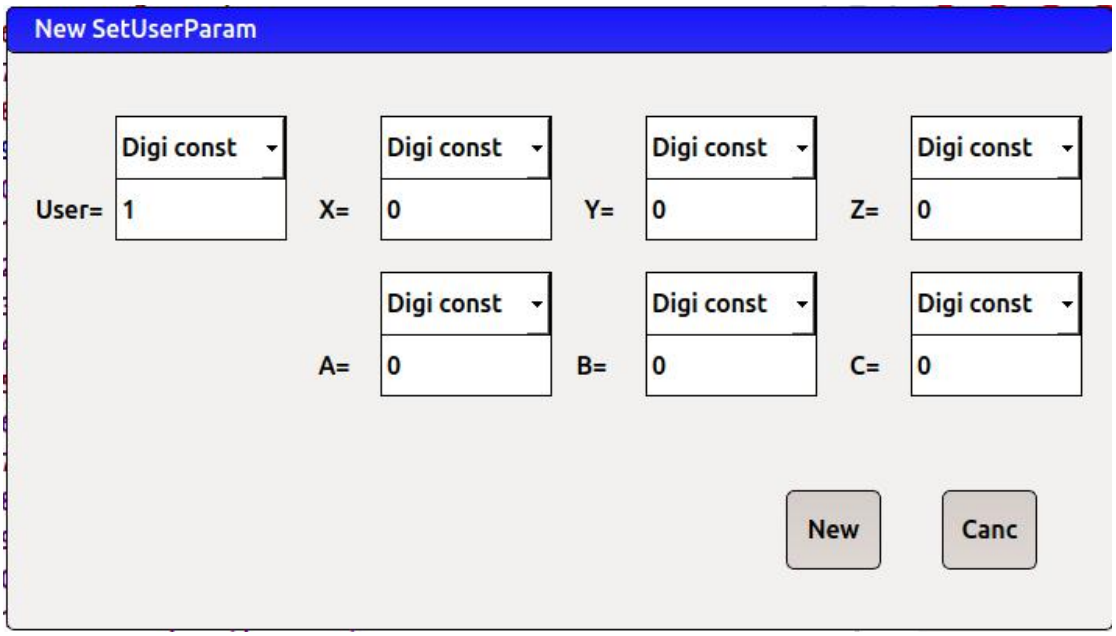
The coordinate system command can automatically modify coordinate system information in the program. The whole offset coordinate system can make all the motion instructions in the program based on the coordinate system offset.
Offsets can use either constants or variables.



Tool coordinate system add offset:

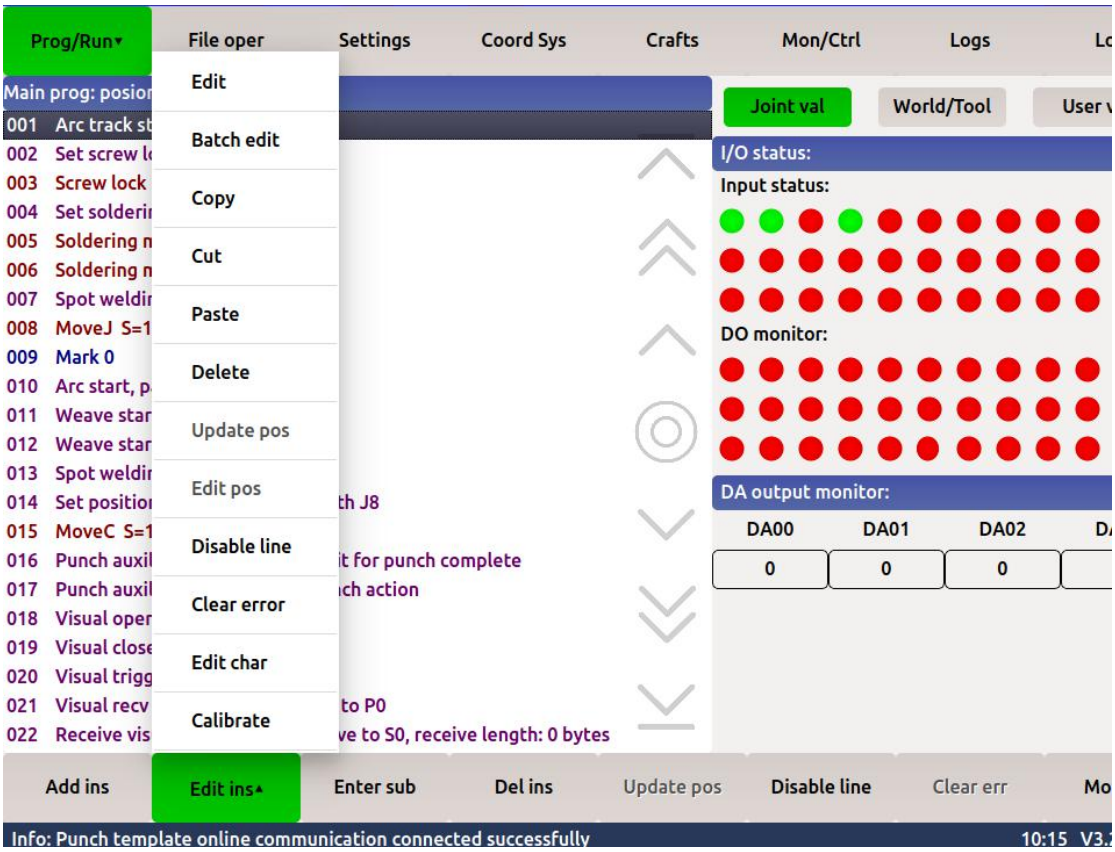


Add offset to user coordinate system:



4.6 Foreign article

4.6.1 English under Chinese command



When the compiler compiles instructions, it also recognizes the English under the instructions. Chinese coverage is just for the convenience of Chinese users to read and use, add notes.
Therefore, it is possible to change the Chinese annotation manually to increase the readability of the program. But for the English of the lower layer of the command, the user cannot easily change it.

4.6.2 And or in conditional instruction

Use character editing, "& &" for and, "|" for or

```
001   If( V0 == 0 && V1 == 0)
002   If( DI9 == 1 && V0 == 2)
```

6.2.1 and

```
001   If( V0 == 0 || V1 == 0)
002   If( DI9 == 1 || V0 == 2)
```

6.2.2 or

4.6.3 Where the conditional statement goes

```
001   If DI8 == 1
002   Call sub program TBD
003   Else if DI9 == 1
004   Call sub program cs-1
005   Else if DI10 == 1
006   Call sub program cs-2
007   Else
008   Sleep 5 ms
009   Endif
```

```
001   If DI8 == 1
002   Call sub program TBD
003   Goto mark0
004   Else if DI9 == 1
005   Call sub program cs-1
006   Goto mark0
007   Else if DI10 == 1
008   Call sub program cs-2
009   Goto mark0
010   Else
011   Sleep 5 ms
012   Endif
013   Mark 0
```

Readers can observe the difference between the two writing methods.
Where to go for conditional statements. Once a condition in the judgment statement is met, the program will jump to the end of the condition after the corresponding action is executed. You don't have to worry about multiple conditions being met at the same time. The program executes in turn. Therefore, the writing method of the right figure is very cumbersome, which increases the difficulty of program reading.

4.6.4 Implementation of selective branch structure

```
001  Mark 0
002  If V0 == 1
003  Call sub program TBD
004  Else if V0 == 2
005  Call sub program cs-1
006  Else if V0 == 3
007  Call sub program cs-2
008  Endif
009  Sleep 5 ms
010  Goto mark0
```

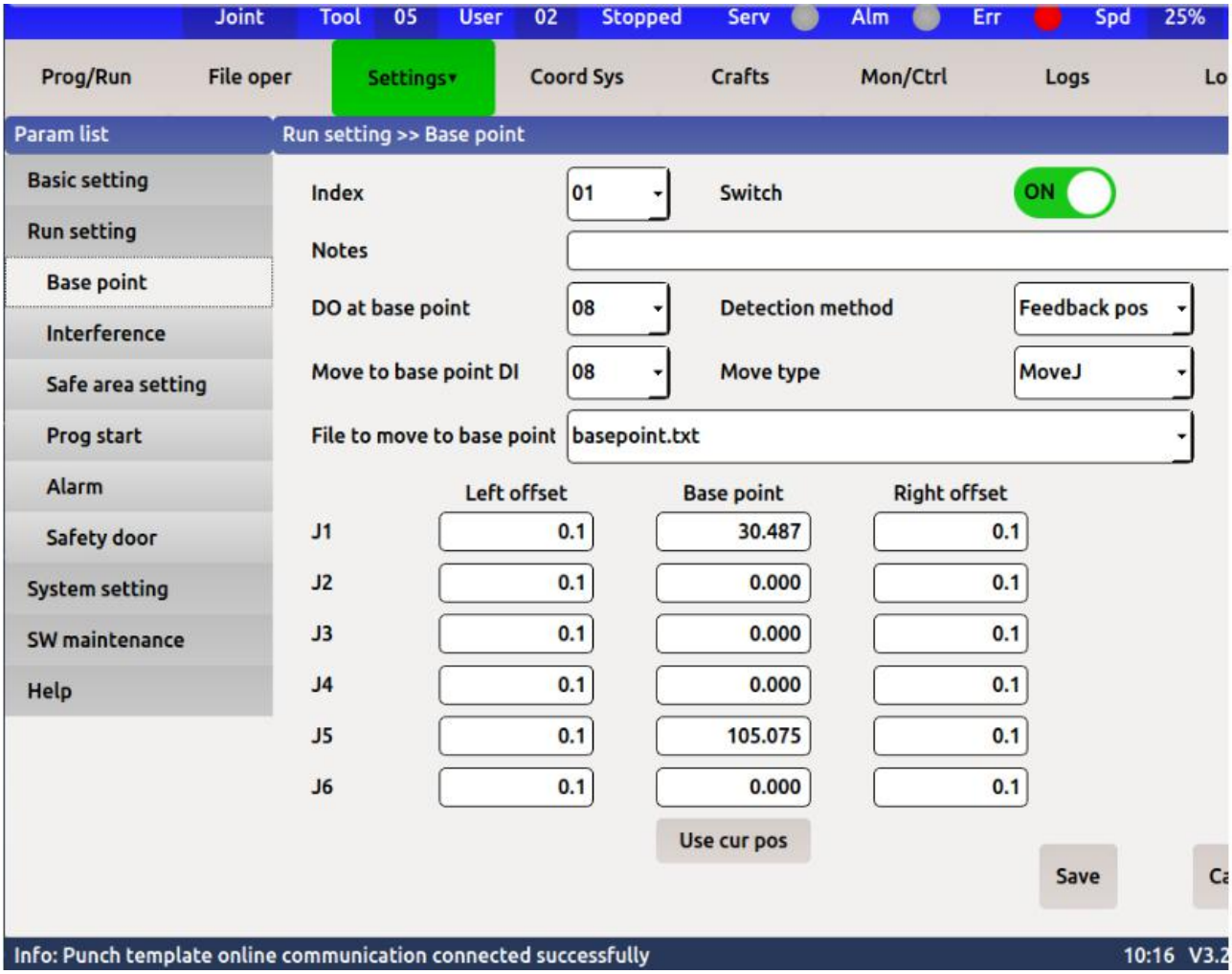
```
001  Mark 0
002  If V0 == 1
003  Call sub program TBD
004  Else if V0 == 2
005  Call sub program cs-1
006  Else if V0 == 3
007  Call sub program cs-2
008  Else
009  Sleep 5 ms
010  Goto mark0
011  Endif
```

6.4.1

Two different ways of writing. The first is to cycle outside the condition, and wait for the state v0 to change after the action is executed. The second method is to judge within the condition. After the condition is satisfied and any action is executed, the judgment is not continued and the program is executed downward.

V. Auxiliary function

5.1 datum point



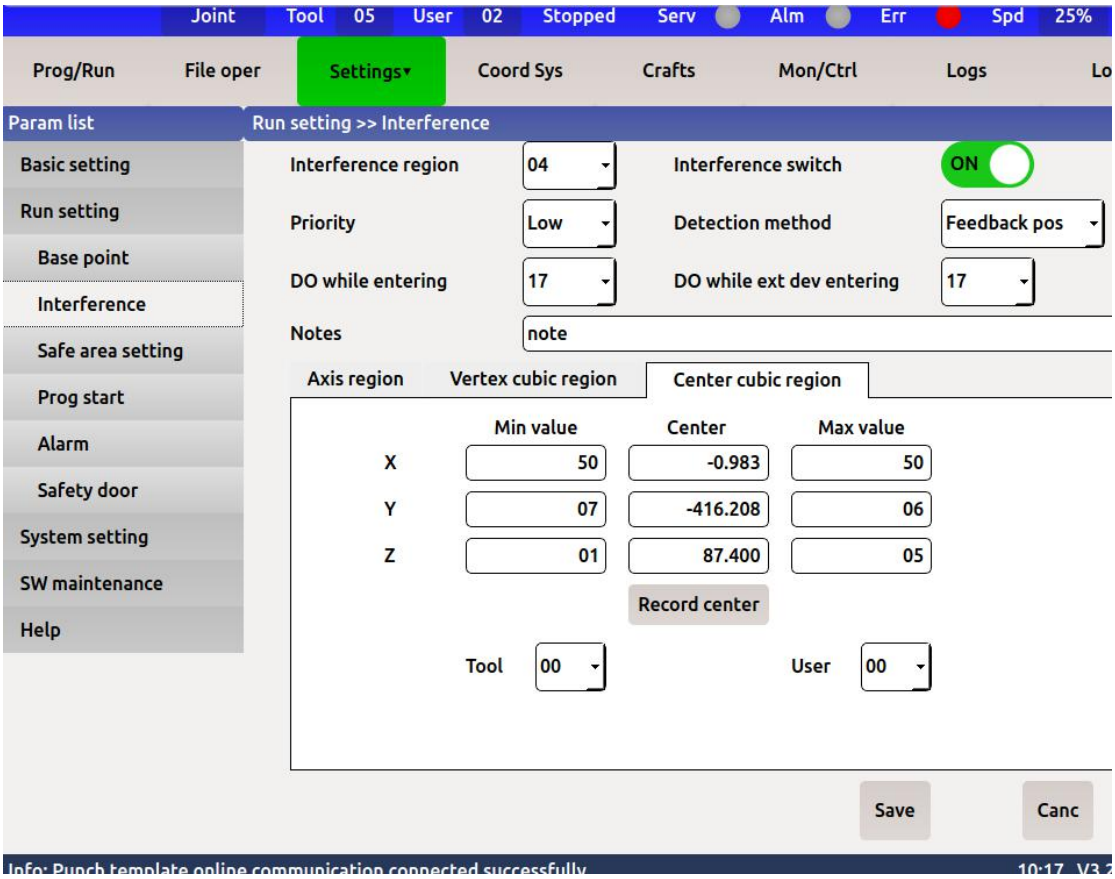
1. Index: 1-10
2. Switch: on / off
3. Notes: remarks for the current datum point
4. Do at base point: when the robot position reaches the reference point, the IO output is set to high level.
5. Detection method: the method of detecting whether the robot is at the reference point. Respectively, command position or current position.
6. Move to base point DI: when input IO is the rising edge, trigger the robot to return to the reference point.
7. Move type: there are three modes of return to reference point: joint motion, linear motion and command file
8. Return to datum file: set the command file to return to datum
9. Left offset: the floating (negative) left value to detect whether the robot is at the reference point
10.Base point: the accurate value of the robot's reference point
11. Right offset: the floating (positive) right value to detect whether the robot is at the reference point

Explain:

a) In this page, you can use the SA key joint to run to the reference point, provided that the reference point must be turned on

b) The reference point can be used in the motion command interface. The precondition is that the reference point must be on. When using a straight line to move to a datum point, the user and the tool must select 0.

5.2 Interference zone



1. Interference Zone No.: 1 ~ 10

2. Interference switch: on / off

3. Priority: high / low. When setting the high priority, the interference area will become larger (joint angle increases by 0.5 degrees, and the axis direction of the space position increases by 10 mm). This function can ensure that the high priority robot sends the signal to enter the interference area first.

4. Detection mode: the reference mode for detecting whether the robot is at the reference point. There are command positions or current positions.

5. Do while entering: output low level when entering the area, otherwise output high level.

6. Do while ext dev entering: when the external equipment enters low level, the movement stops; when the external equipment enters high level, the movement continues (Note: this function is not applicable to the positioner).

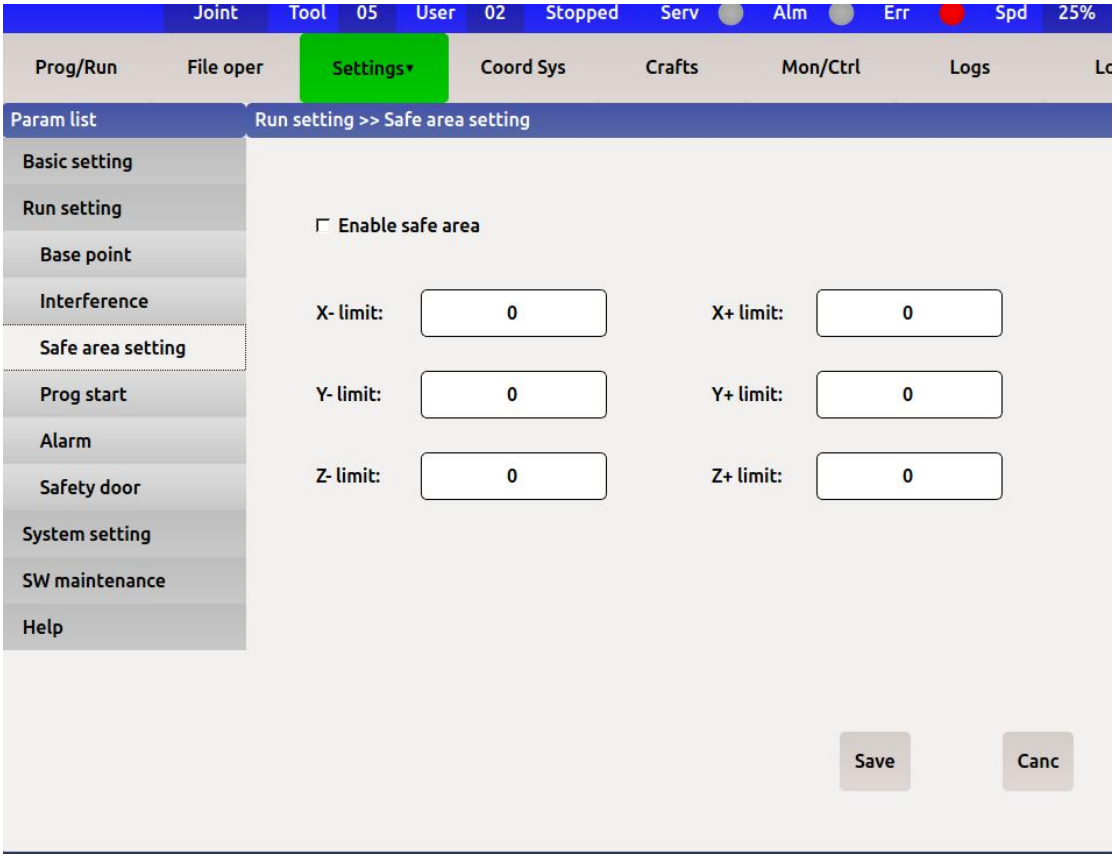
7. Notes: remarks for the current interference area

8. Axis region: set the joint values of 6 start points and 6 end points.

9. Vertex cubic region: set two vertices to form a cube space (the ridge line is parallel to the world coordinate XYZ axis).

10. Center cubic region: set a central point and offset with the front and back, left and right, up and down directions (world coordinate XYZ three directions) of the central point to form a cube interference area.

5.3 Safe area



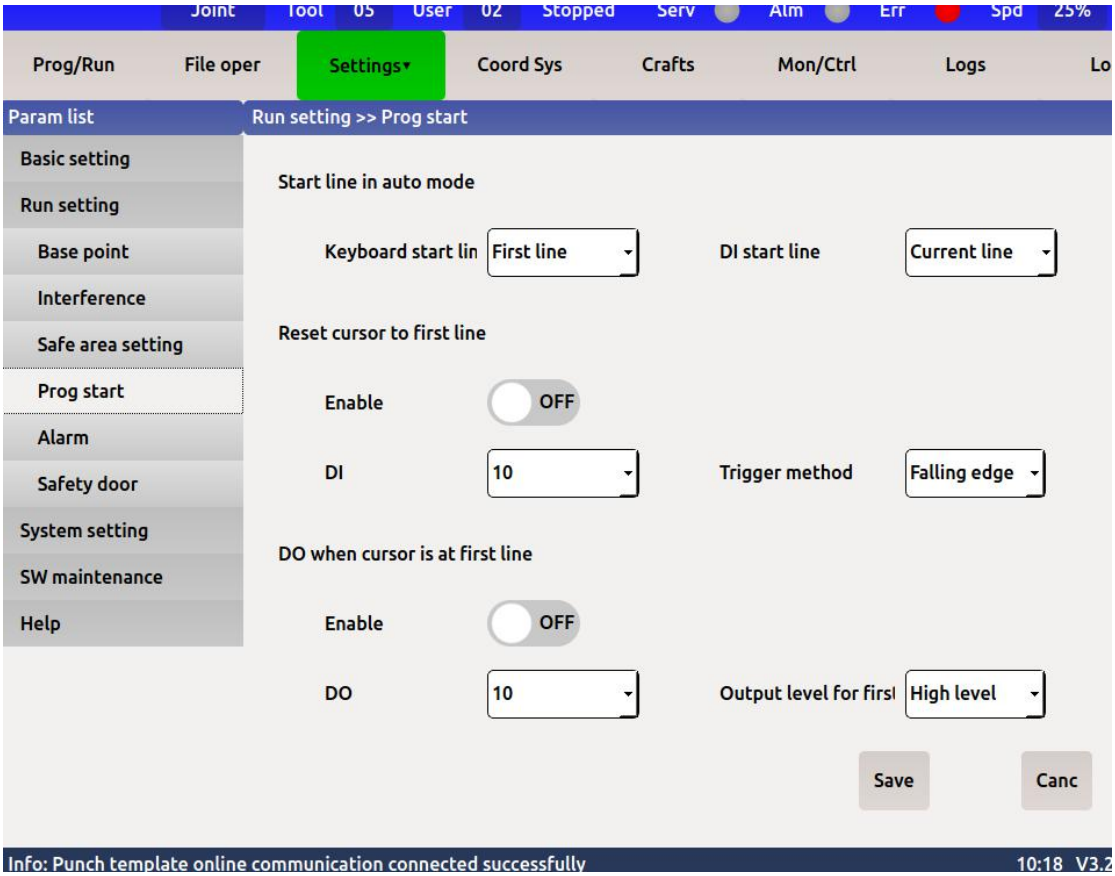
The motion range of the robot is constrained by rectangular coordinates. After the function is turned on, the robot can only move in the limited area, once it exceeds the limit, it will alarm the abnormal movement.

5.4 Safety Door

图片

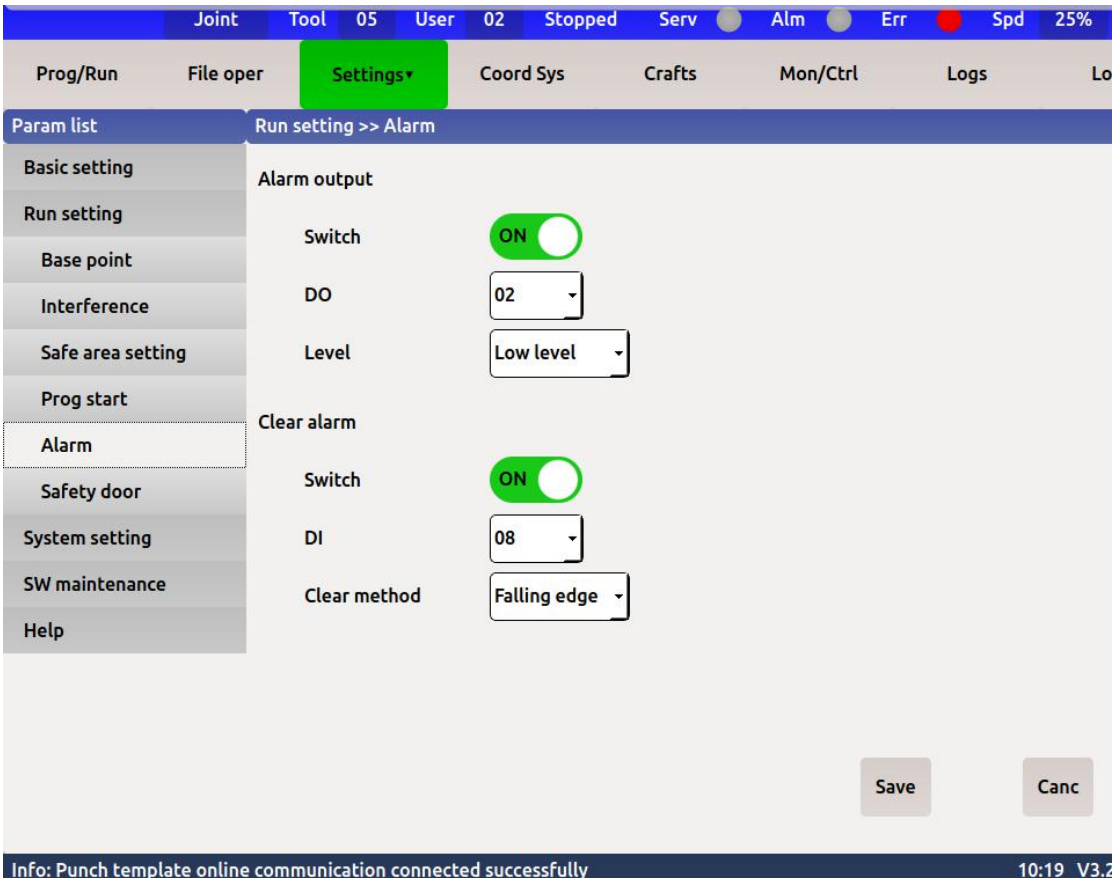
The safety door refers to the robot running in a confined space. The safety door is associated with the robot io signal. When the safety door signal disappears, the machine will alarm and stop running.

5.5 External IO initiator



As shown in the figure, when the robot is in automatic state but the program is not running, the controller captures the rising edge of di10 and the cursor jumps to the first line. The cursor is in the first line, and the robot 7 output status is on.

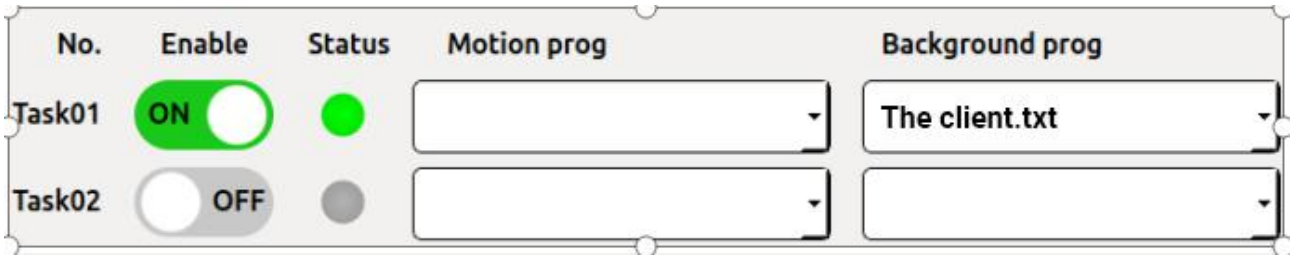
5.6 ALARM

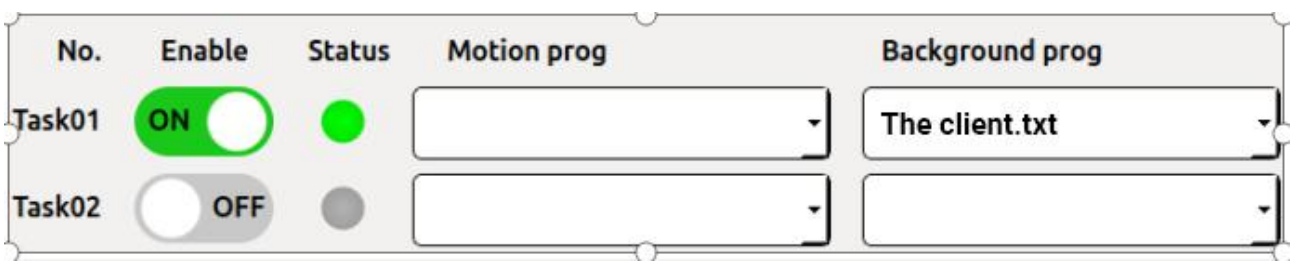


As shown in the figure, once the manipulator alarms, its output 8 is set to high level. External input robot 11 will clear the reset table alarm。

5.7 Background function

5.7.1 two states of background program





Notes:

Switch:

Switch is the main switch, that is, whether to enable this background program thread.

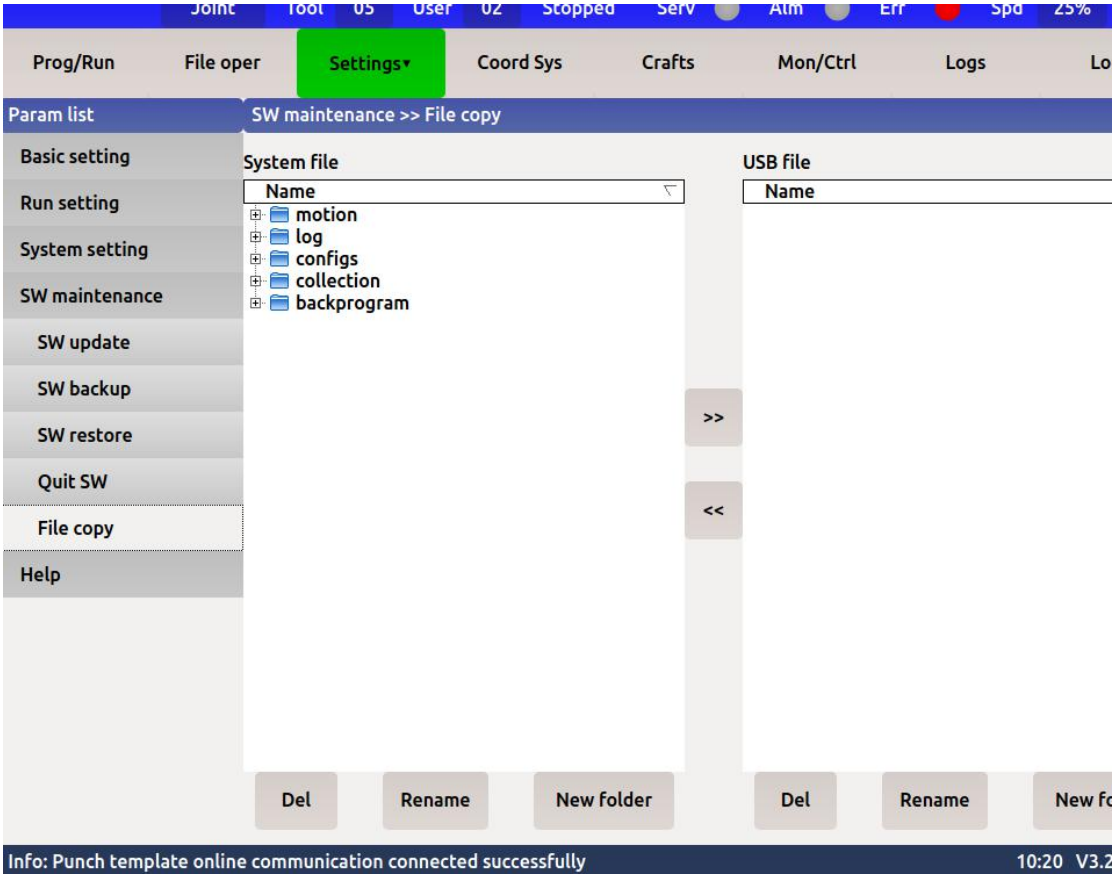
Exercise program:

The motion program can control the start and stop state of the background program. When the background program thread has set up the motion program, the system will start the background program when the motion program runs. **When the motion program stops, the background program stops, that is, the background program starts and stops synchronously with the motion program. When the running program is not set, the background program will start and stop with the system start and stop.** It is suggested that during the test, no specific motion program should be set, i.e. the background should always be running, which is conducive to the early debugging.

Background program:

Indicates the main daemons that this daemons thread runs.

5.7.2background program storage location



Note: the foreground motion program is placed in motion. The background program is placed in the back program.

In general, a background program does not need to be written with a special editor. That is, use the foreground program editor to edit the background content. The system has its own virtual u disk, and uses the virtual u disk path as a bridge to store the program compiled in motion into the back program folder. You can turn on the background function in the monitoring and monitoring interface。

5.7.3 Background basic application case

The biggest difference between the foreground program and the background program is that the foreground program can control the movement of the manipulator, and the background program cannot execute the movement instructions. But the backstage for input and output control, logical judgment, the use of communication programs, are very powerful functions.

The front and back stage are used together to realize multithreading on the program. It is necessary to deal with complex situations. Here are some examples to introduce the background functions.

1. Capture the edge trigger of IO signal

Sometimes it is necessary to record the change of some IO status at any time. Waiting in the foreground program will delay the execution of other actions.

So the edge trigger is put into the background for execution.

```
DEFINE TRIGGER_IO DI14        # 14号 IO输入_上升沿触发
DIGITAL IO_CURR = 0         # 这个周期的DI值
DIGITAL IO_LAST = 0         # 上个周期的DI值

WHILE(1)
   SLEEP(1)
   IO_CURR = TRIGGER_IO
   IF(IO_CURR == IO_LAST):CONTINUE:ENDIF  #如果DI值没有变化，继续下一次循环
   IO_LAST = IO_CURR
   IF(0 == IO_CURR):CONTINUE:ENDIF      #如果DI值为0，继续下一次循环

   #到这里时，DI从0变成了1

#判断托盘位置看哪个托盘准备好
#A托盘在原点，A盘有料，A准备好
If( DI24 == 1 && DI26 == 1 )
   S5 = "A盘准备好"
ENDIF

#B托盘在原点，B盘有料，B准备好
If( DI27 == 1 && DI29 == 1 )
   S6 = "B盘准备好"
ENDIF

ENDWHILE
```

2. Distinguish between program operation and manipulator movement

Normally, do0 and do1 indicate that the program is stopped and running. But program operation does not represent manipulator action. Sometimes the workstation needs to take the manipulator movement as the judgment standard.

```
WHILE(1)
V0 = X
V1 = Y
V2 = Z
   SLEEP(100)
V3 = X
V4 = Y
V5 = Z
#每100ms读取一次当前位置
V6 = V3 - V0
V7 = V4 - V1
V8 = V5 - V2

#8号输出表示动作中
If( V6 > 0.1 && V7 > 0.1 && V8 > 0.1)
IOOut(08,1)
Else
IOOut(08,0)
ENDIF

ENDWHILE
```

5.8 Appointment function

图片

The reservation function first needs to create the corresponding motion program in the teach pendant operation interface. As shown in the picture, the joint motions written in 1.txt and 2.txt.

图片

Reservation function: Associate the programs named 1.txt and 2.txt, and switch between start and stop through the two button boxes in the figure below. The two button boxes respectively control 1.txt and 2.txt. Press the first button box to start, and then 1.txt will start running; according to the scheduled rules, press the start button corresponding to 1.txt, and the start button of 2.txt will start to execute on the way to the end. When you press it, 2.txt needs to enter the appointment waiting, and wait for the execution of 1.txt to complete, then 2.txt will start execution.

When 1.txt is executed, the start button of 1.txt will no longer be effective, and the execution process of 1.txt cannot be interrupted by pressing it multiple times; similarly, when 2.txt is executed, the start button of 2.txt is also Unable to interrupt the execution process of 2.txt.

In the above two cases, only the stop button can interrupt the process, otherwise, you need to wait for the process to complete before you can continue to execute the next task.

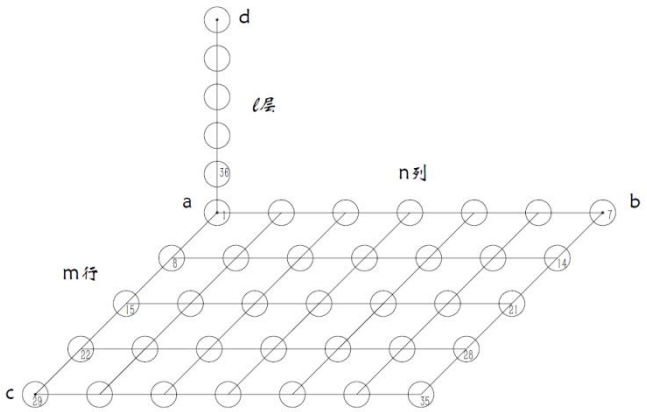
The target number of workpieces needs to be entered manually. Enter the number of workpieces you want to complete, and the file will be executed as much as possible. It can be used in conjunction with multiple files; for example, after file 1 completes 10 pieces, start file 2 and complete 5 pieces.

图片

VI. Process introduction

6.1 stacking process

6.1.1 process setting



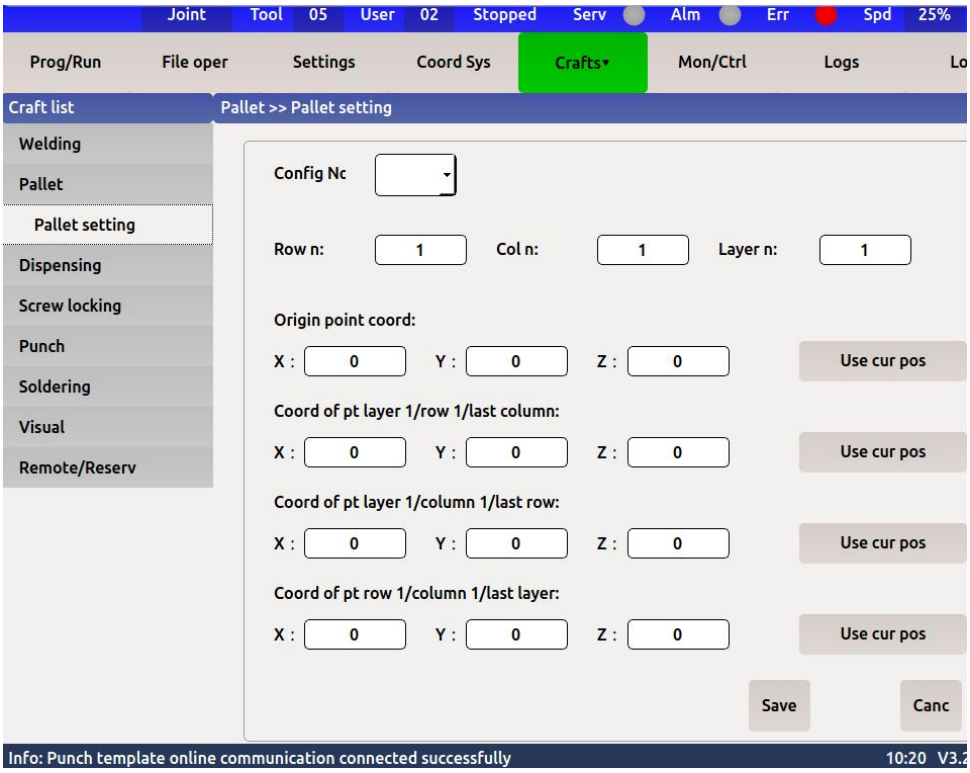
Set the Craft in process - > pallet - > pallet setting.

a. Set the correct number of rows, columns and layers first

b. Set the placement origin coordinate, teach it to point a above, and then click "copy from current position" to record the placement origin coordinate

c. Record the coordinates of points B, C and D in the above figure in turn (as the case may be, not all the

coordinates of the three points need to be recorded, for example, if there is only one line, point C does not need to be recorded; if there is only one layer, point d does not need to be recorded)

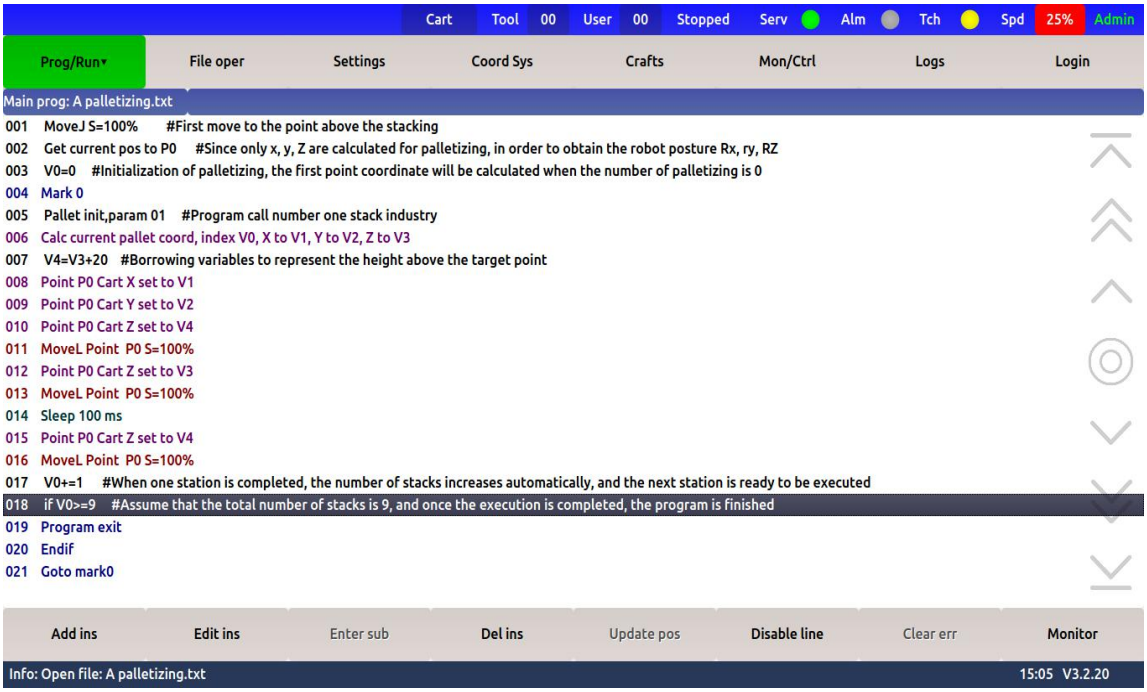


6.1.2 Programming

Understanding of palletizing:

The palletizing program seems to be complex. In fact, only two instructions need to be understood, one is "palletizing initialization parameter * *", the other is "calculation of palletizing coordinate assignment". The first instruction determines how many number stacks to call, which is the preparation for the second instruction. To the second command, the corresponding coordinates will be calculated according to the corresponding number of stacks.

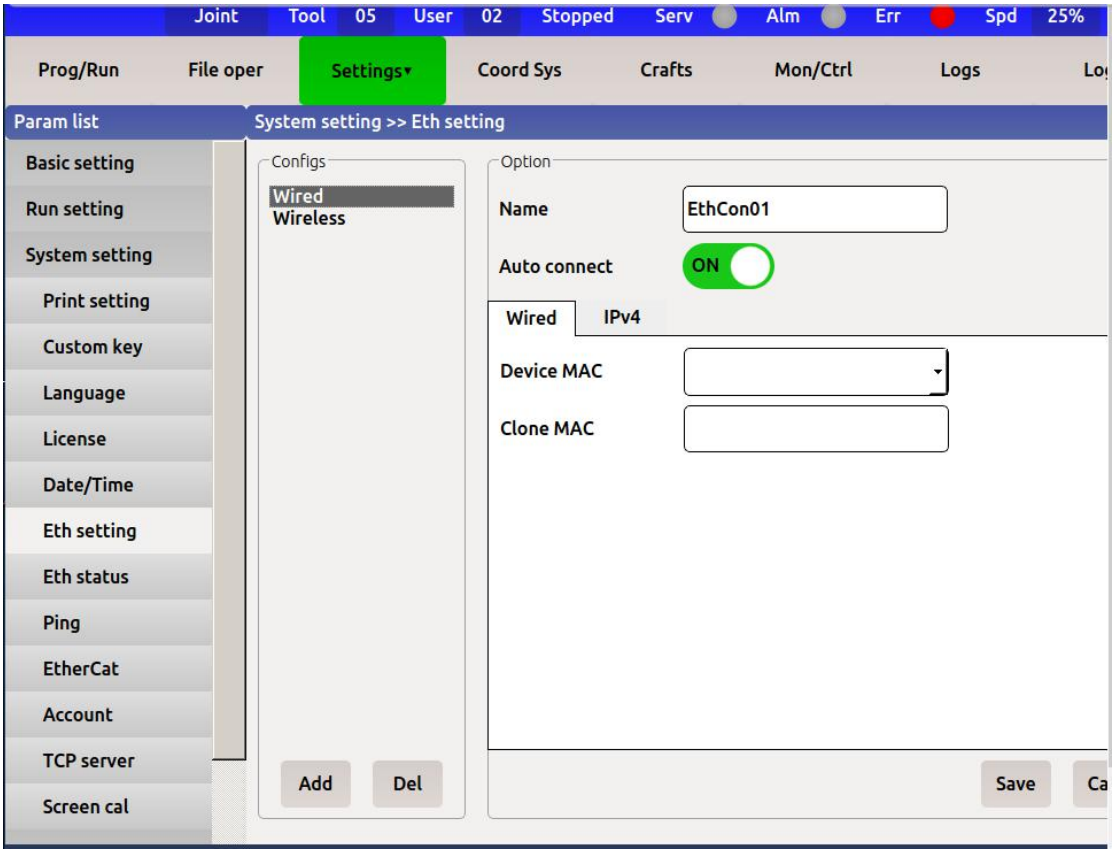
When writing the program, we only need to use these V variables to store the number of stacks and coordinates flexibly, and then we can write all kinds of stacks with flexible functions. Do not be bound by the template program, stacking is both regular and free.

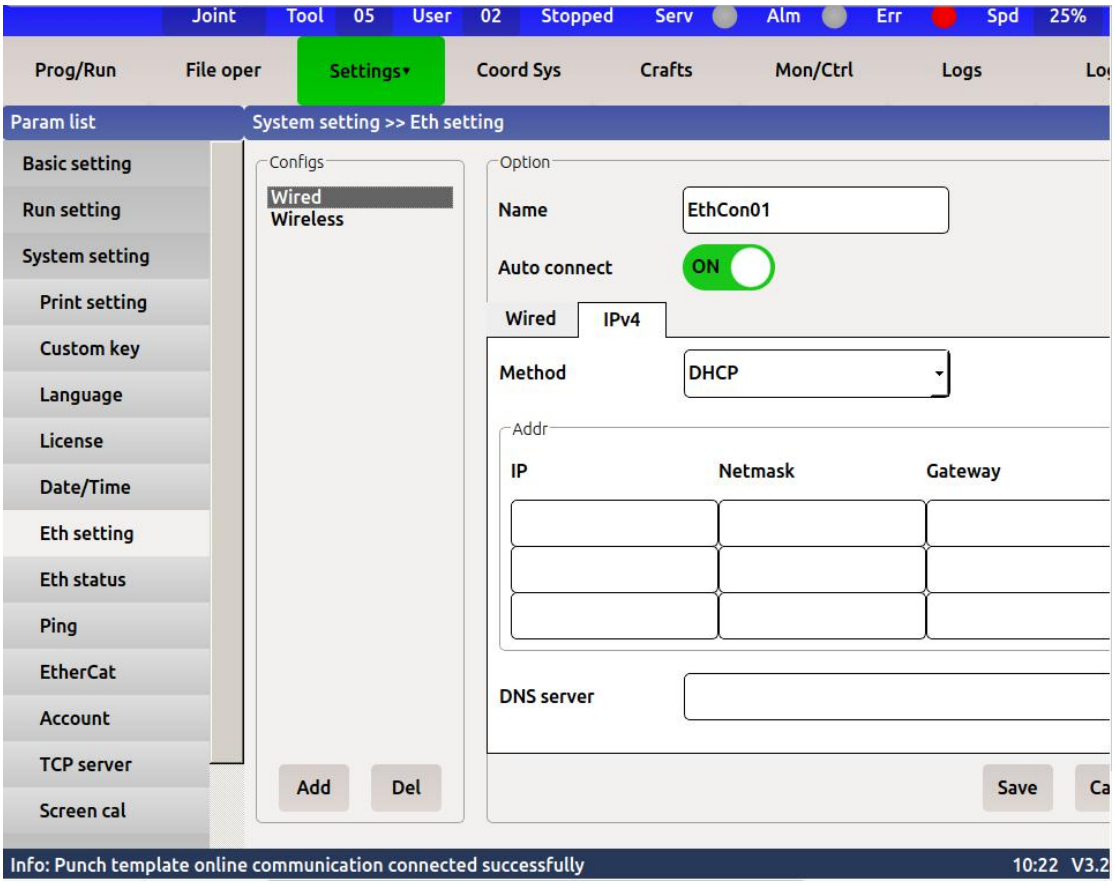

Cart Tool 00 User 00 Stopped Serv Alm Tch Spd 25% Admin
Prog/Run File oper Settings Coord Sys Crafts Mon/Ctrl Logs Login
Main prog: A palletizing.txt
001 MoveJ S=100% #First move to the point above the stacking
002 Get current pos to P0 #Since only x, y, Z are calculated for palletizing, in order to obtain the robot posture Rx, ry, RZ
003 V0=0 #Initialization of palletizing, the first point coordinate will be calculated when the number of palletizing is 0
004 Mark 0
005 Pallet init,param 01 #Program call number one stack industry
006 Calc current pallet coord, index V0, X to V1, Y to V2, Z to V3
007 V4=V3+20 #Borrowing variables to represent the height above the target point
008 Point P0 Cart X set to V1
009 Point P0 Cart Y set to V2
010 Point P0 Cart Z set to V4
011 MoveL Point P0 S=100%
012 Point P0 Cart Z set to V3
013 MoveL Point P0 S=100%
014 Sleep 100 ms
015 Point P0 Cart Z set to V4
016 MoveL Point P0 S=100%
017 V0+=1 #When one station is completed, the number of stacks increases automatically, and the next station is ready to be executed
018 if V0>=9 #Assume that the total number of stacks is 9, and once the execution is completed, the program is finished
019 Program exit
020 Endif
021 Goto mark0
Add ins Edit ins Enter sub Del ins Update pos Disable line Clear err Monitor
Info: Open file: A palletizing.txt
15:05 V3.2.20

6.2 Visual technology

6.2.1 robot IP address configuration

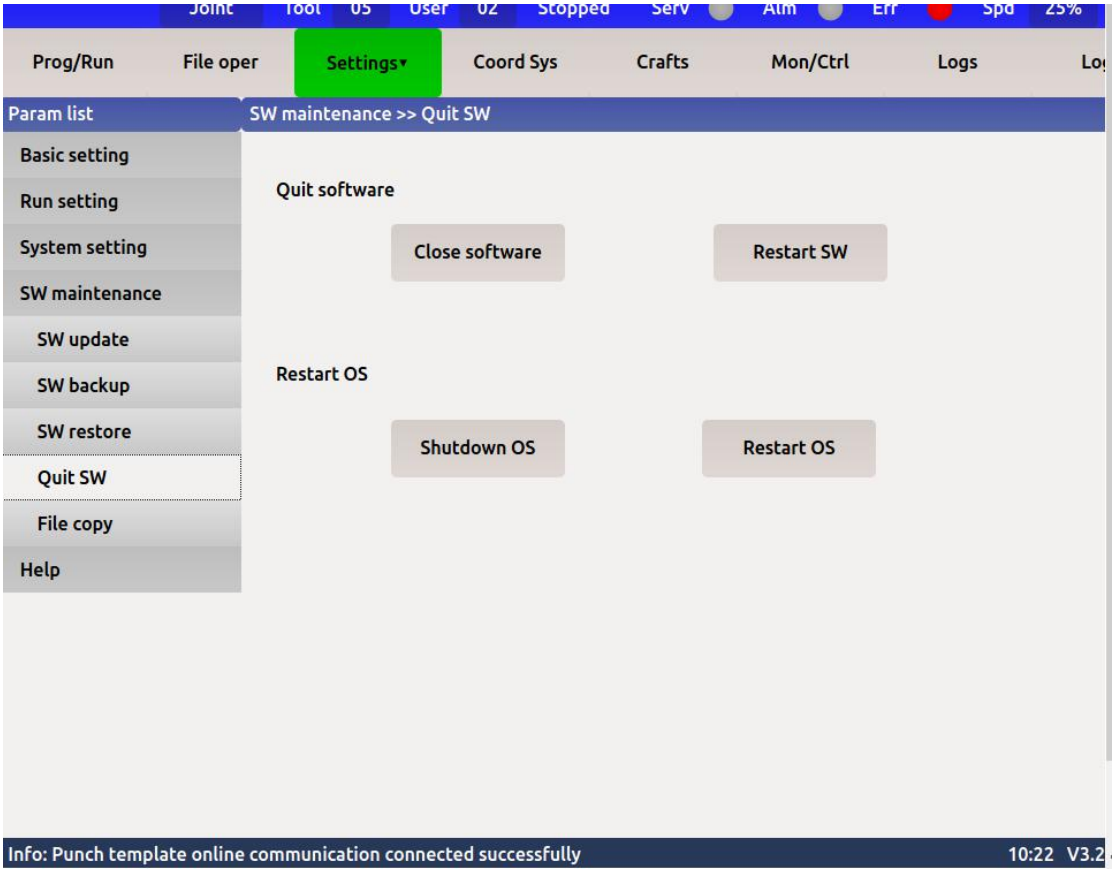
Method 1: use the controller to change the Ubuntu system IP address



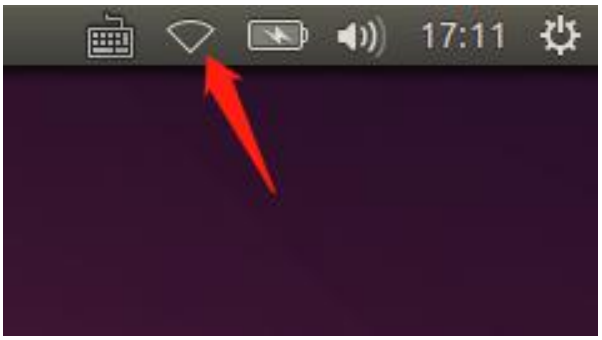


Method 2: enter the Ubuntu interface to directly set the IP address

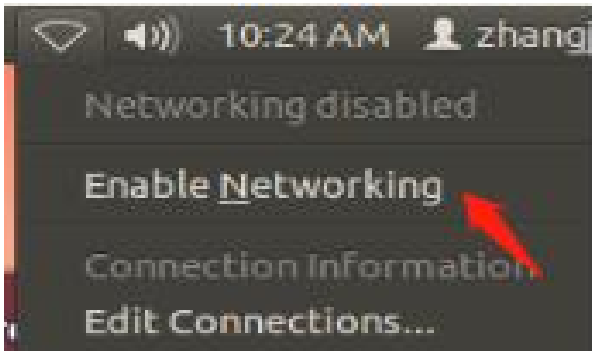
Select **parameter settings - > system maintenance - > exit the system - > exit the controller system.** The interface is as follows:



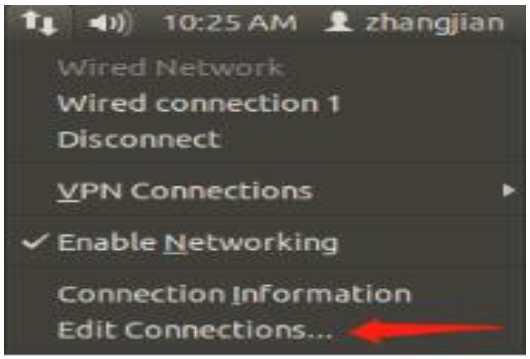
Exit the controller system and enter the Ubuntu desktop. The network icon in the upper right corner of the stand-alone interface is as follows:



If the network is not enabled, enable the network first

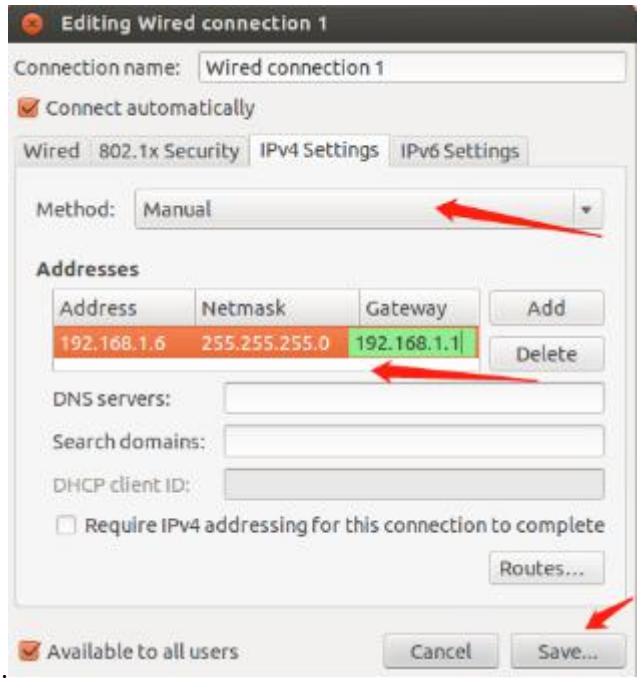


Select edit connection.



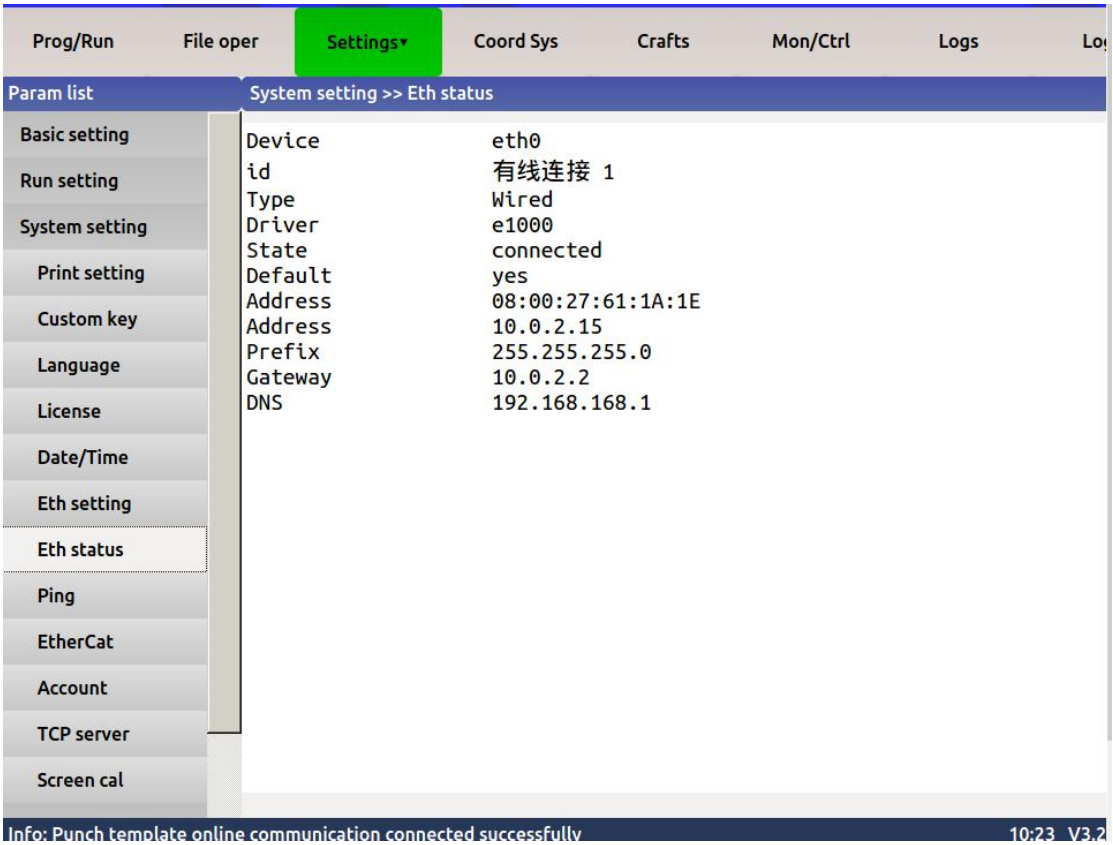
Editor IP:

Select Manual IP, configure IP as shown in the figure below, save and restart the system

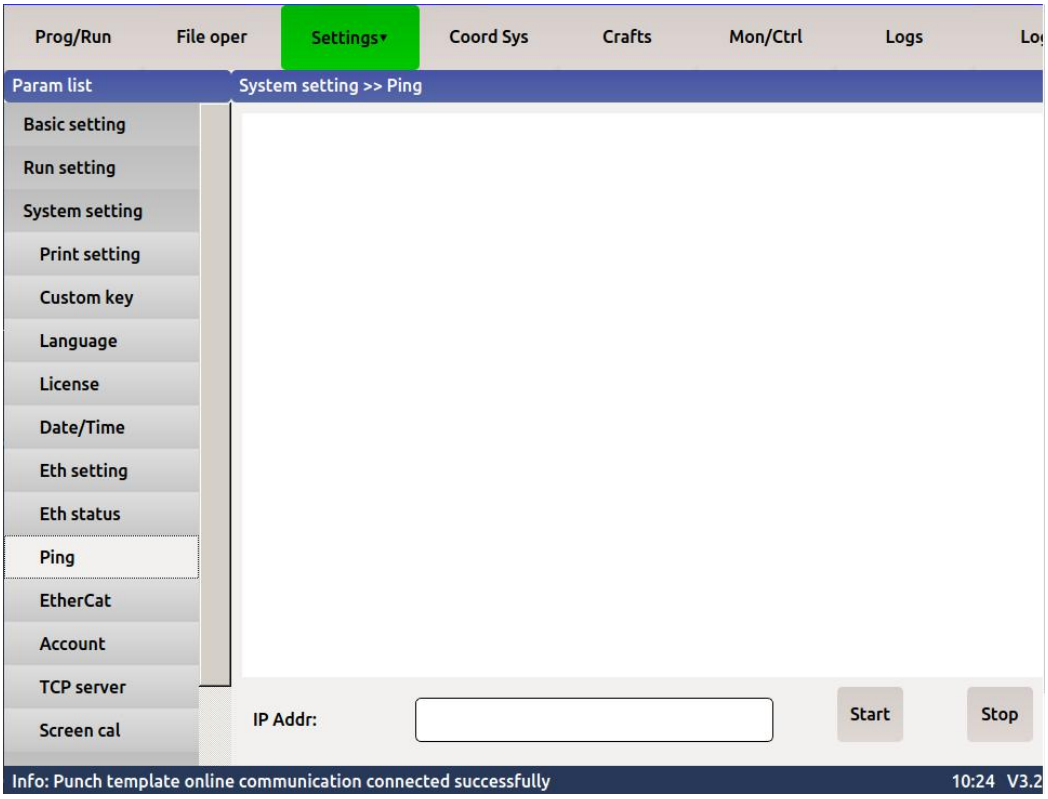


6.2.2 Robot IP address verification

After the IP address is set successfully, connect with the external device, and the Ethernet status interface will display the local IP address.



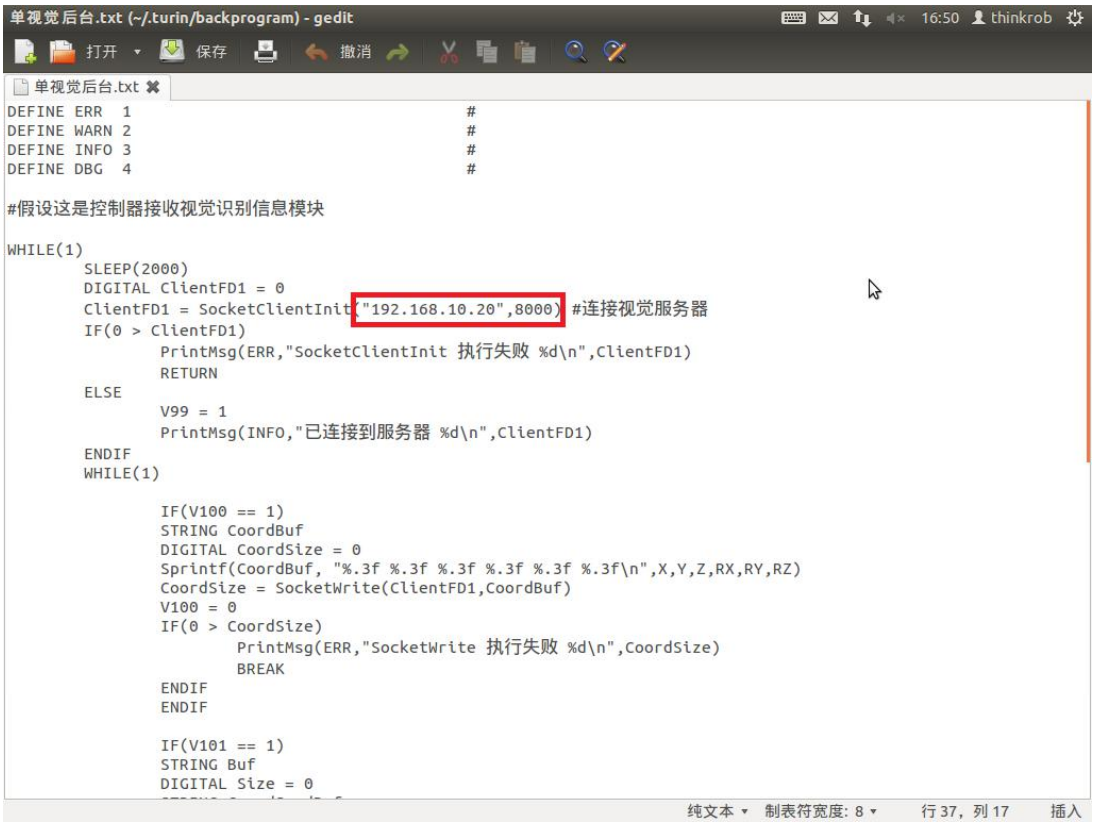
Use the controller ping command to check the physical connection.



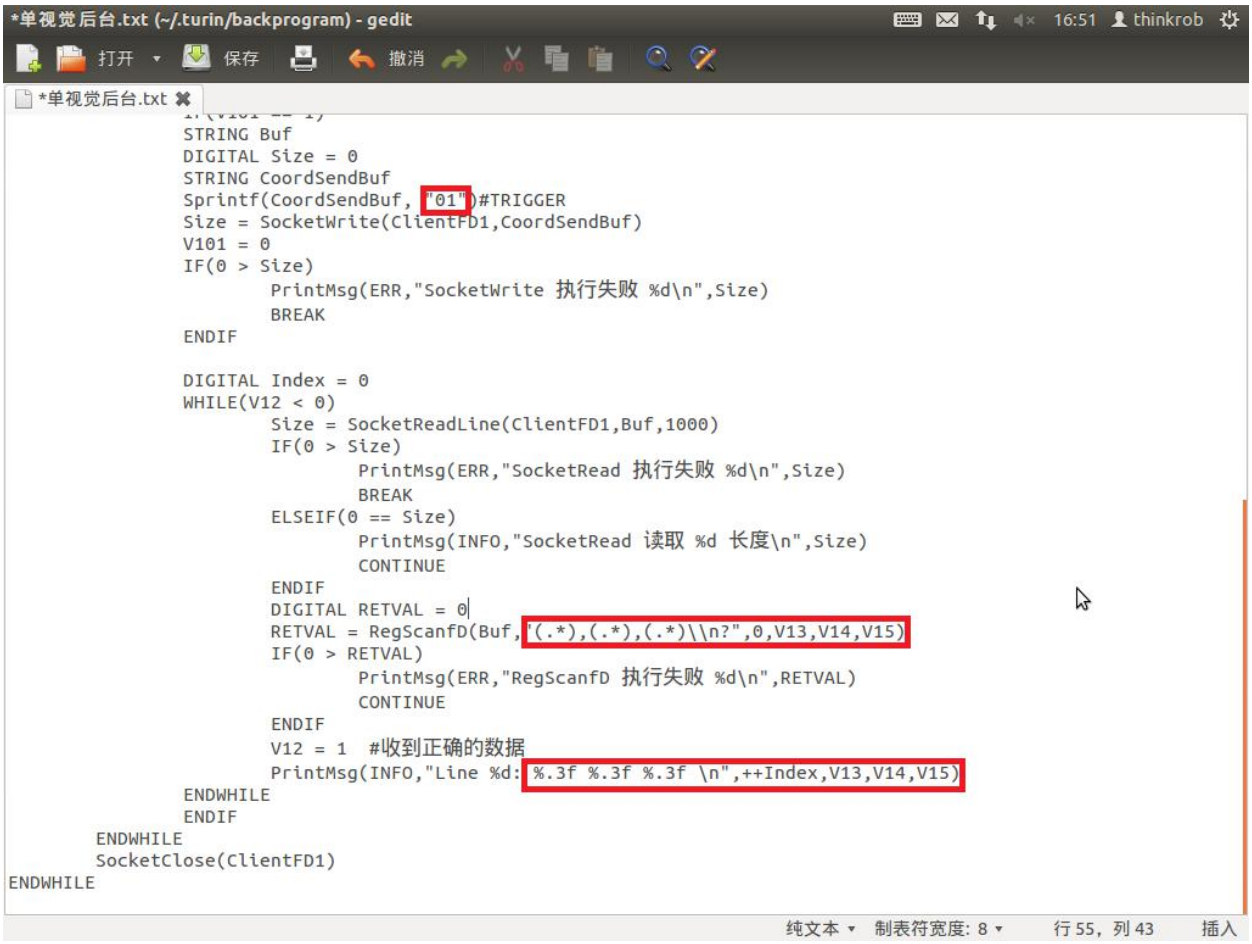
6.2.3 Two ways to perform TCP / IP communication

Method 1: use background scripts

1. Background program content analysis



Note: the IP address and port number of the server to which the manipulator is connected when it is the client in the red box. That is, the address and corresponding port number of the visual server.
Socketclientinit function to connect to the server.
Function return value < 0 means connection failure, and the background will print the contents of printmsg in the form of error.

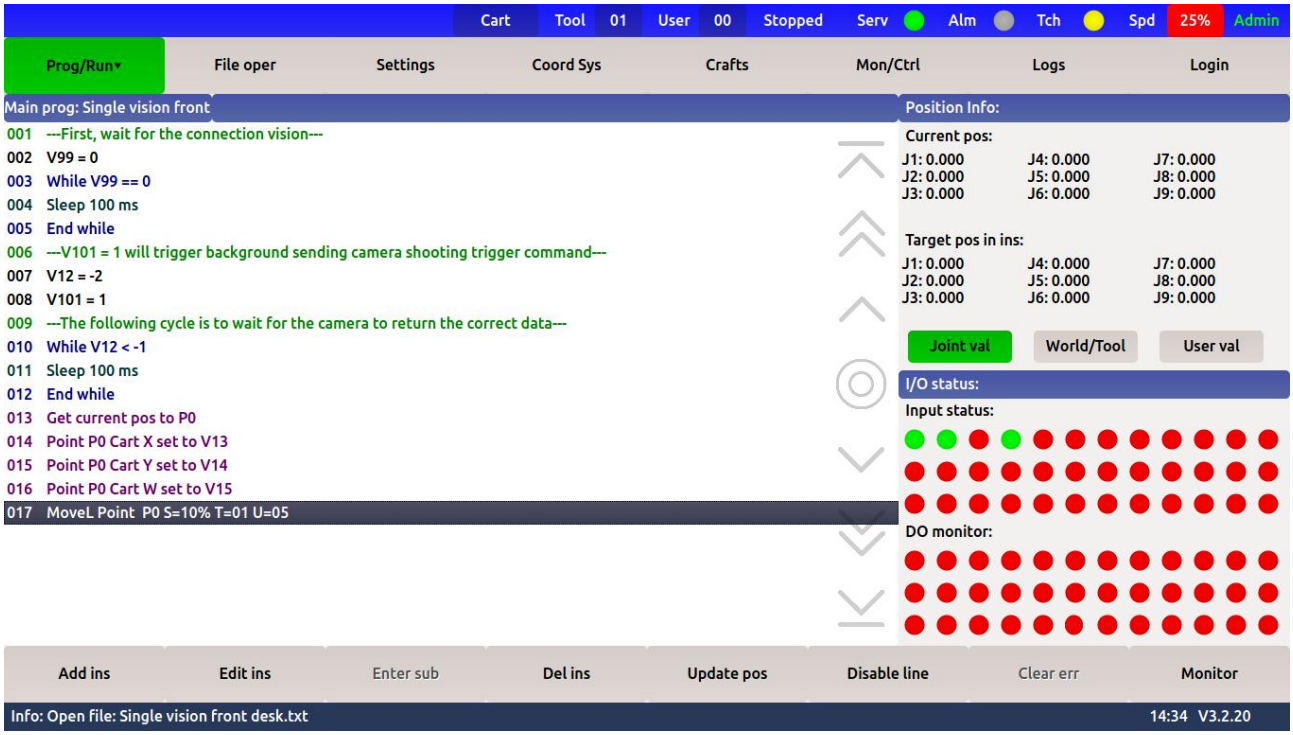


Notes:
The **socketwrite** function sends data to the corresponding port in the server. In the case shown in the figure, send the string coordsendbuf to the server defined in clientfd1, and sprint assigns a value to the string coordsendbuf. The value "01" in the figure can be understood as "01" in the soft trigger photographing command.
The function of **RegScanfD** is to parse the string in the Buf cache area with English comma as the separator and extract it to the digital variable. The storage address is V13/V14/V15.
Retval is the return value of the function. Resolution fails when the return value is less than 0. In general, incorrect data format will cause parsing failure.
In a word, background communication scripts use logic statements such as while and if to restrict each other with the V variables controlled by the foreground. Call the function related to communication function in proper time sequence。

Network communication instructions
1.int SocketServerInit(int)
Function: Initialize the web server,

Parameter 1: Port number.
Return value: Successful execution returns the server listening socket id, which is the socket. Return -1 on failure.
2.int SocketServerAccept(int)
Function: monitor client connection
Parameter 1: socket id
Return value: The client socket id will be returned when the execution is successful, and -1 will be returned on failure
3.int SocketClientInit(string,int)
Function: Connect to the server
Parameter 1: ip address
Parameter 2: Port number
Return value: Return client socket id if successful, return -1 if failed
4.void SocketClose(int)
Function: close the socket
Parameter 1: socket id
Return value: None
5.int SocketWrite(int,string)
Function: Send data
Parameter 1: socket id
Parameter 2: String, the length of the sending is the length of the string, please note that there is no "\0"
Return value: The sending length is returned on success, and a negative number is returned on failure.
6.int SocketWrite(int,string,int)
Unlike 5, you can specify the length of the transmission, in parameter 3
7.int SocketRead(int,string,int)
Function: Receive data
Parameter 1: socket id
Parameter 2: storage buffer (string variable)
Parameter 3: Read length, currently the maximum is 1024, it will not help to give a larger value
Return value: Return the read length on success, 0 or a negative number on failure.
Note that this is a blocking function, unless there is an error, it will wait forever. Suitable for receiving handshake messages....
8.int SocketReadLine(int,string,int)
Function: Read a line of character string.
Parameter 1: socket id
Parameter 2: storage buffer (string variable)
Parameter 3: Timeout
Return value: Returns the read length if successful, returns 0 if it fails, and returns a negative number if network exceptions
Some design details:
1. All network interfaces are implemented using non-blocking sockets, such as SocketRead and SocketReadLine. All blocking phenomena are simulated by scripts. Therefore, it will not affect the execution of other scripts.
2. The network transceiver buffer is set to 4M in size.

2. Foreground script case



Note: to be used with background script

V99 will jump out of the loop once the connection with the server is successful

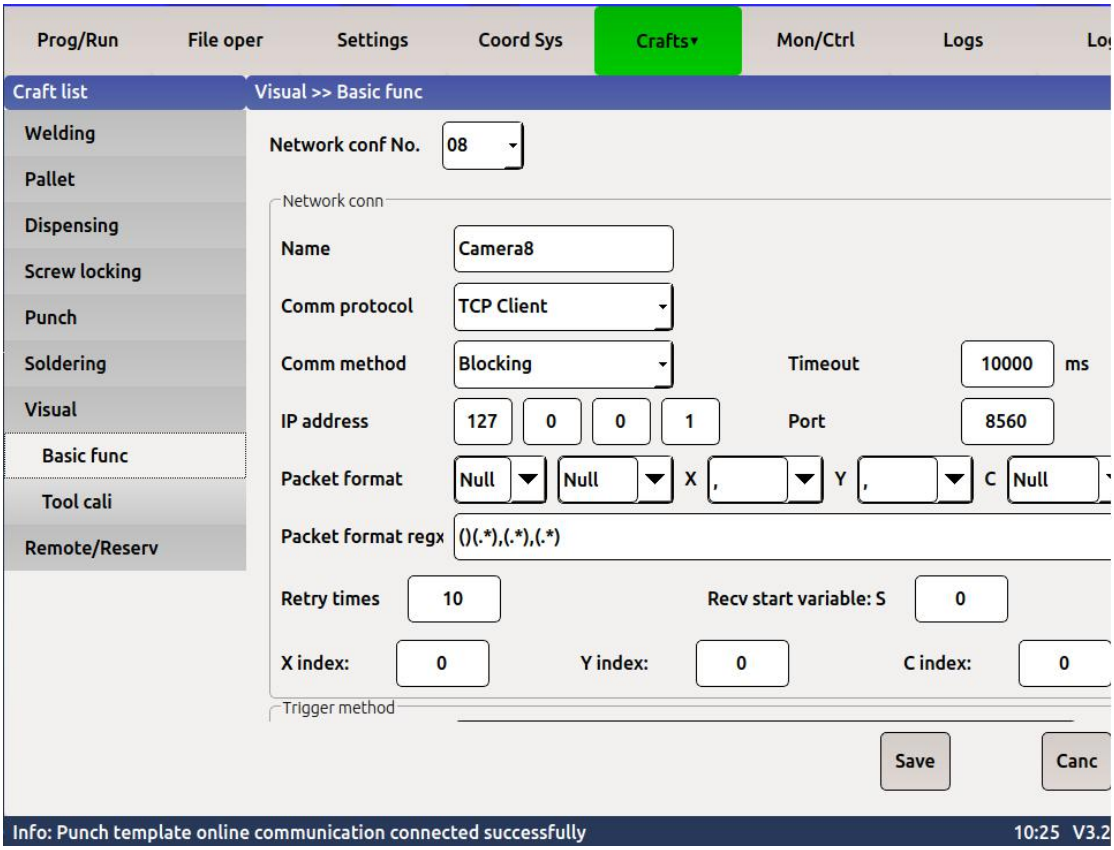
V101 = 1 will send the photographing instruction, which can be modified in the background file to the string to be sent

V12 = - 2: the foreground program is set to - 2 first, and after receiving the data, the background will set to 1, and then jump out of the loop.

V100 = 1 current position of transmitting robot

Method 2: use visual technology package

IP address and other basic configurations are the same as the above. The difference with method 1 is that the function realized in the background in the previous article is encapsulated in the process interface here。



Parameter resolution:

1. Network configuration

Select a network configuration sequence number, ranging from 1 to 10.

2. Network name

Define the name of this connection for easy differentiation.

3. Comm protocol

There are TCP client, TCP server, UDP client and UDP server. Four modes are available.

4. Comm method

Blocking and non blocking are optional. The difference is whether to wait for the visual data and use it with the front desk instruction.

5. Timeout

Set the timeout in blocking mode.

6. IP address

Sets the IP address of the connected object.

7. port

Set the port number of the connection.

8. packet format

Select the format sent by vision. Only partial formats are supported.

9. packet format regx

Fill in the received packet format regular expression by yourself, and finally the packet format regular shall prevail.

10. Retry times

Number of retries.

11. Recv start varible

We will store the stored data in the s variable, and analyze it according to the filled regular expression. Finally, we will store the data in the s variable. Here, we set the start area of the storage. We support the analysis of up to 10 fields.

12.X index

Sets the index of X data in the parsing field of a regular expression.

13.Y index

Set the index of Y data in the parsing field of the regular expression.

14.C index

Sets the index of the C data in the parsing field of the regular expression.

6.2.4 Communication using Modbus Protocol

1.download modbus poll。

图片

2. open modbus pollsoftware

图片

3. choose connection->connect…(F3)

图片

Directly chooose Register later

图片

Click ok

4. Set the connected IP and port number

图片

IP address, choose to enter the IP address of the robot, choose 502 for port, the port number is fixed and cannot be modified. Select TCP/IP for Connection and click the ok button after setting.

5.Setting of the robot's IP address

Note: The IP of the robot and the IP of the modbus poll host must be in the same network segment. The IP of the robot must be set to 192.168.2.100. The following is the setting method for reference

After setting the IP, you can check the Ethernet

图片

Finally, you can test IP, parameter setting->system setting->Ping

图片

Set the IP address as the destination IP address, click Start, the following interface appears, indicating that both parties are connected.

图片

6. Test the connection data

If there is no error message as shown below, it means that the data is being communicated.

图片

6.1 Read IO data

Select Setup->Read/Write Definition... at the top of the application, or click the arrow below:

图片

图片
The configuration is shown in the picture:
Function: Select 01 Read Coils (0X).
Address: Set the starting address of monitoring, we set 0 here
Quantity: The total number of addresses to be monitored, we set 100 here.
Click ok, the setting is complete,
As shown in the picture below, 100 io states are displayed

图片

Set the status of a certain IO, such as setting the IO at address 39, the specific IO mapping, each version is different, check the IO mapping table of the corresponding version, here is an example to set the status of 39IO.
图片
Double click address 39
图片
If you want to set the IO of address 39 to OFF, then set the value option to OFF, otherwise, set the value option to ON, and then select Send. The robot will also respond to the data. If the answer is OK, as shown in the picture below, the result option bar will have a Response ok prompt:
图片
Then check whether the IO status of the robot is correct.

7. Read input register address (only supported by new version)
The robot's reading and writing data area is V900-V999, and the address of the holding register corresponding to modbusTCP is 0-199.
Select function code 03 to read robot coordinates
图片

The V variable of a robot occupies 2 registers, the high-order data contains the high-order address, and the low-order data contains the low-order data. Example Address 0 stores the low 16-bit address of the variable, and address 1 stores the high 16-bit address of the variable
图片

The coordinates of the tool currently executed by the robot are displayed here.
The steps for analyzing the coordinate values are given below. Here, address 4 and address 5 are used as a group for example:

Address 4, decimal is 30671 converted to hexadecimal 0X77CF, address 5 is decimal 17142, converted to hexadecimal 0X42F6, so the stored data is 0X42F677CF, this data corresponds to the floating point number 123.234.， View the data of the robot V902
图片

The actual data is 123.234001, and the calculated 123.234 is due to the float precision relationship. The other coordinate data is the same operation.

8. Write register address (only supported by new version)
The robot's reading and writing data area is V900-V999, and the address of the holding register corresponding to

modbusTCP is 0-199.

If you want to write data to the robot variable through modbus, you need to use modbus to write register operation. Use function code 16 Write Multiple Registers to write data to registers

图片

图片

Double-click the address to write data into it. As shown in the picture, address 0 and address 1 correspond to 1 group of data.

图片

The data filling content is similar to the reading above. For example, if you want to store -1444.614 into the robot's V900 variable, first convert the decimal -1444.614 to hexadecimal 0XC4B493A5. Convert hexadecimal 0XC4B4 to decimal 50356 and write address 1, and convert hexadecimal 0X93A5 to decimal 37797 and write address 0

图片

During the writing process, there may be situations where it cannot be written, such as the following warning

图片

Need to set the type of input data shown in the picture below

图片

Modify the final Display to Unsigned.

Robot V Variable

图片

The gap between V900=-1444.613892 and expected write -1444.614 is a float accuracy problem

9.When using Modbus protocol, the robot can only be used as the master station, and the default port number is 502.

The address and function code are shown below.

Register		
Modbus input register address ID=1;F=04	**Data definition**	**Explain**
1	Feedback real-time user coordinate value position X high 16 bits	
2	Feedback real-time user coordinate value position X low 16 bits	
3	Feedback real-time user coordinate value position Y high 16 bits	
4	Feedback real-time user coordinate value position Y low 16 bits	
5	Feedback real-time user coordinate value position Y high 16 bits	

6	Feedback real-time user coordinate value position Z low 16 bits	
7	Feedback real-time user coordinate value position RX high 16 bits	
8	Feedback real-time user coordinate value position RX low 16 bits	
9	Feedback real-time user coordinate value position RX high 16 bits	
10	Feedback real-time user coordinate value position RY low 16 bits	
11	Feedback real-time user coordinate value position RZ high 16 bits	
12	Feedback real-time user coordinate value position RZ low 16 bits	
Modbus holding register address ID=1;F=03,06, 16	**Data definition**	**Explain**
1	Write target user coordinate position X high 16 bits	Target value * 1000
2	Write target user coordinate position X low 16 bits	Target value * 1000
3	Write target user coordinate position Y high 16 bits	Target value * 1000
4	Write target user coordinate position Y low 16 bits	Target value * 1000
5	Write target user coordinate position Z high 16 bits	Target value * 1000
6	Write target user coordinate position Z low 16 bits	Target value * 1000
7	Write target user coordinate position RX high 16 bits	Target value * 1000
8	Write target user coordinate position RX low 16 bits	Target value * 1000
9	Write target user coordinate position RY high 16 bits	Target value * 1000
10	Write target user coordinate position RY low 16 bits	Target value * 1000
11	Write target user coordinate position RZ high 16 bits	Target value * 1000
12	Write target user coordinate position RZ low 16 bits	Target value * 1000

Bit operation(IO_36)		
Modbus Switch input address ID=1;F=02	**Robot DI/DO PIN**	**Explain**
0	Turing robot I / O module input 0 DI_0	IO board emergency stop signal (normally closed)
1	Turing robot I / 1 module input 0 DI_1	IO board collision detection signal
2	Turing robot I / 2 module input 0 DI_2	Manual automatic switching

3	Turing robot I / 3 module input 0 DI_3	Indicator hand pressure enable signal
4	Turing robot I / 4 module input 0 DI_4	Script start
5	Turing robot I / 5 module input 0 DI_5	Script stop
6	Turing robot I / 6 module input 0 DI_6	
7	Turing robot I / 7 module input 0 DI_7	
8	Turing robot I / 8 module input 0 DI_8	
9	Turing robot I / 9 module input 0 DI_9	
10	Turing robot I / 10 module input 0 DI_10	
11	Turing robot I / 11 module input 0 DI_11	
12	Turing robot I / 12 module input 0 DI_12	
13	Turing robot I / 13 module input 0 DI_13	
14	Turing robot I / 14 module input 0 DI_14	
15	Turing robot I / 15 module input 0 DI_15	
16	Turing robot I / 16 module input 0 DI_16	
17	Turing robot I / 17 module input 0 DI_17	
18	Turing robot I / 18 module input 0 DI_18	
19	Turing robot I / 19 module input 0 DI_19	
20	Turing robot I / 20 module input 0 DI_20	
21	Turing robot I / 21 module input 0 DI_21	
22	Turing robot I / 22 module input 0 DI_22	
23	Turing robot I / 23 module input 0 DI_23	
24	Turing robot I / O module output0 DO_0	Automatic mode, program running
25	Turing robot I / O module output1 DO_1	Auto mode, program stopped
26	Turing robot I / O module output2 DO_2	
27	Turing robot I / O module output3 DO_3	
28	Turing robot I / O module output4 DO_4	
29	Turing robot I / O module output5 DO_5	
30	Turing robot I / O module output6 DO_6	
31	Turing robot I / O module output7 DO_7	
32	Turing robot I / O module output8 DO_8	
33	Turing robot I / O module output9 DO_9	
34	Turing robot I / O module output10 DO_10	
35	Turing robot I / O module output11 DO_11	
36	DO_36	
37	DO_37	
38	DO_38	
39	DO_39	
40	DO_40	
41	DO_41	
42	DO_42	

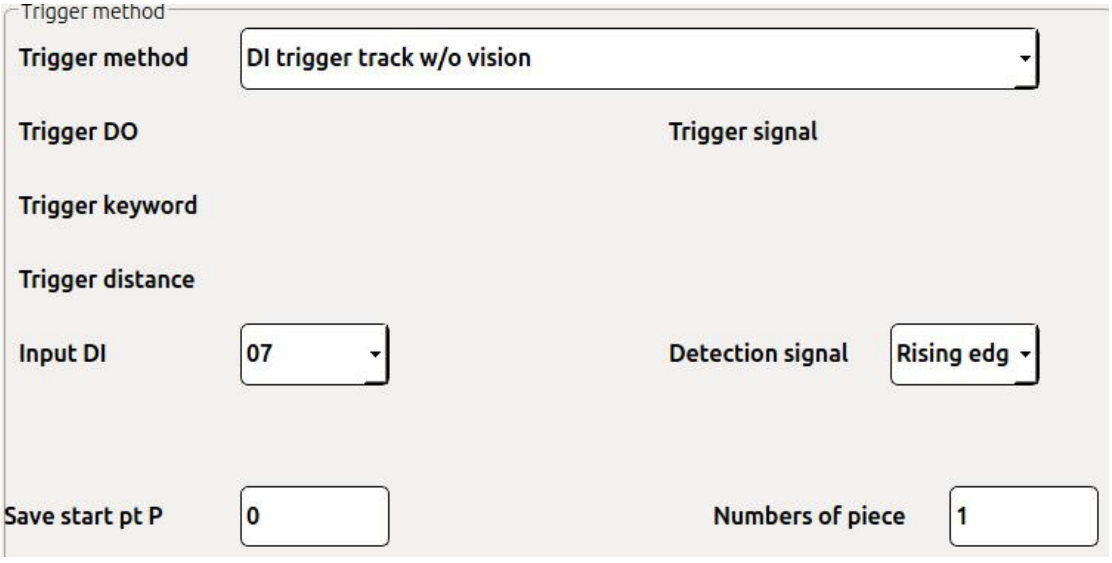
43	DO_43	
44	DO_44	
45	DO_45	
46	DO_46	
47	DO_47	
48	DO_48	
49	DO_49	
50	DO_50	
51	DO_51	
52	DO_52	
53	DO_53	
54	DO_54	
55	DO_55	
56	DO_56	
57	DO_57	
58	DO_58	
59	DO_59	
60	DO_60	
61	DO_61	
62	DO_62	
63	DO_63	
Modbus Coil address ID=1;F=05,15	**Robot DI/DO PIN**	**Explanation**
24	DI_24	
25	DI_25	
26	DI_26	
27	DI_27	
28	DI_28	
29	DI_29	
30	DI_30	
31	DI_31	
32	DI_32	
33	DI_33	
34	DI_34	
35	DI_35	
36	DI_36	
37	DI_37	
38	DI_38	
39	DI_39	
40	DI_40	
41	DI_41	
42	DI_42	

43	DI_43	
44	DI_44	
45	DI_45	
46	DI_46	
47	DI_47	
48	DI_48	
49	DI_49	
50	DI_50	
51	DI_51	
52	DI_52	
53	DI_53	
54	DI_54	
55	DI_55	
56	DI_56	
57	DI_57	
58	DI_58	
59	DI_59	
60	DI_60	
61	DI_61	
62	DI_62	
63	DI_63	

6.3 Tracking technology

6.3.1Process settings

1.Trigger mode



Trigger method：

This parameter sets the mode that triggers the start of the trace. For example, receiving external IO signal to start tracking, receiving network command to start tracking, etc. Different triggering methods are used, and relevant triggering parameters need to be set separately according to requirements.

Trigger (DO) I/O

Set the output trigger IO.

Trigger signal

Set the trigger signal method.

Trigger keyword

Set the trigger keyword for network trigger.

Trigger distance

Set how far the conveyor belt moves for equal interval triggering.

Input (DI) I/O

Set input IO.

Heartbeat

Set the input signal mode.

Storage start point P：

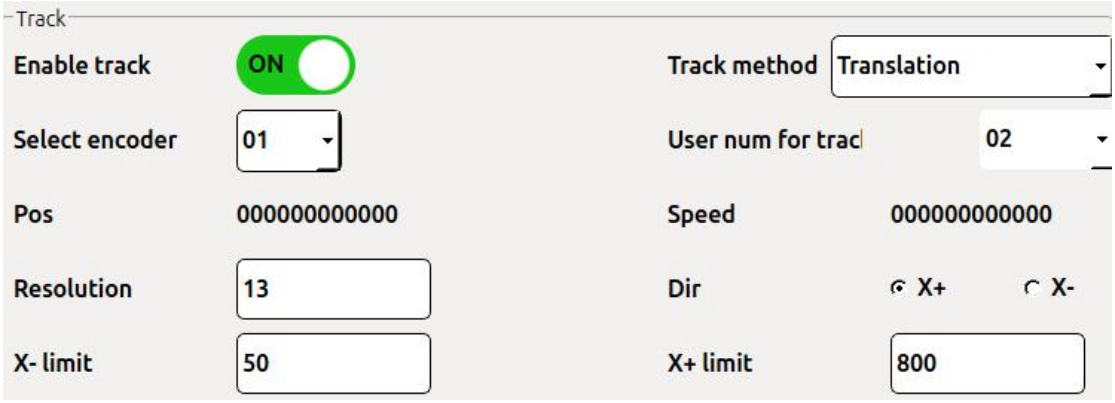
This setting is the first coordinate of each trace, from which P point to read the trace point.

Number of workpieces：

For the number of workpieces added to a disc, that is, how many tracking positions there are in a circle are set. The program automatically reads tracking points from point P.

Note: the use of the storage point here is limited in scope. It cannot be repeated with the storage point address in the network configuration serial number of other groups. Otherwise, it will cause two times of coverage. After the coordinates of the post-p point are set in the monitoring / control, do not change the coordinates of the post-p point at will, otherwise, the location of the tracking point will be lost, leading to the occurrence of unexpected situations such as tracking failure.

2.Track



Enable track：

Sets whether tracing is enabled.

Track method：

There are four ways to set the tracking mode: translation, disc (pure position), disc (clamp follow) and disc (point follow).

Pan: used in the case of a straight conveyor.

Disk (pure position): it means that only the X and Y coordinates of the robot user are changed, and the

attitude change is not adjusted.
Disk (clamp following): it means that when tracking the disk, the robot adjusts the user's x, y and RZ according to the rotation angle of the disk.
Disc (pointing to follow): it means that when tracking the disc, the robot adjusts the X and y of the user according to the rotation angle of the disc, and also adjusts the attitude according to the RZ of the tool.

Select encoder：

Sets the encoder serial number of the connection. Dual encoder is supported.

User num for trac:

The tracking process is based on this user coordinate system. Whether the user coordinates are established correctly or not determines whether the tracking can be realized. The user coordinate X direction and the conveyor belt flow direction should be parallel as the direction vector of the tracking direction.

Pos：

The real-time display reads the encoded value.

Speed:

Real time display of encoder speed read.

Resolution:

Resolution of encoder, unit: pulse / mm. Actual measurement is required when using, and the ratio of code value to actual displacement of conveyor belt.

Dir：

Set the direction of the encoder.

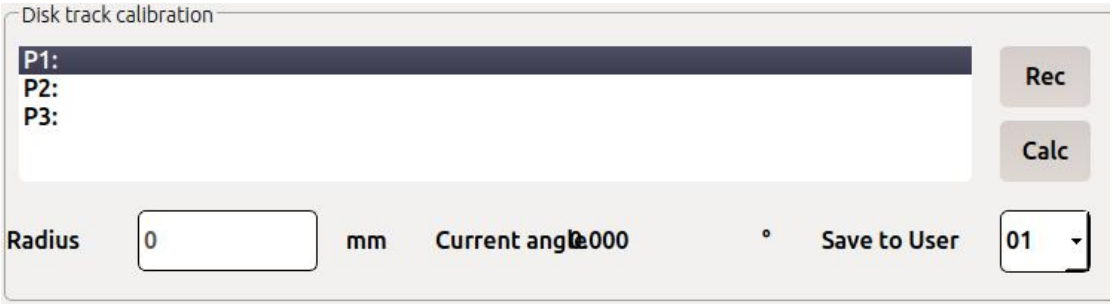
X-limit, X+limit：

Sets the interval of the tracking range. The tracking algorithm is based on user coordinates. The user's X direction is the direction of belt movement. The value entered is the x value of the user coordinate.
If the arc track is tracked, the setting here is the angle, and the setting range of the angle is 0-720. According to the tracking range set by the user, track part of the disc or the whole disc.

Tracking factor:
Set the speed of tracking.
Sampling period (common):
The code value fluctuates during the acquisition period.
Sampling difference (common):
Code value fluctuation display within the period

3.Disk tracking calibration

In the process of disk tracking, we need to calibrate the arc trajectory, and the simple translation tracking mode does not need to set disk tracking calibration.



Record：

When using disc tracking, the position of the disc needs to be calibrated. First, select a reference point on the path where the encoder is installed and record it in P1. Then rotate the disk, align the robot with the disk reference point again, and record it in P2. The third point is the same. Repeat the above operation and record to P3.

Note: when using disk recording, first calibrate the robot tools。

Calculation：

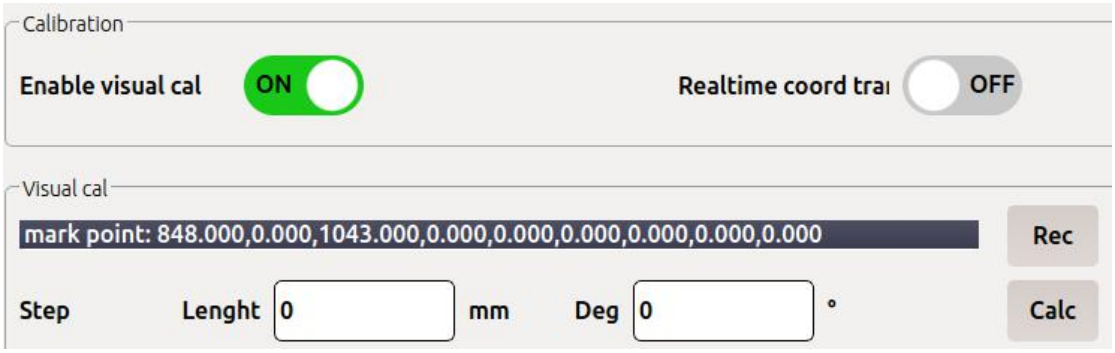
After three records, click calculate. At this time, the accountant calculates the radius of the disc and the center point. The actual position of the robot to the center of the disk can be compared with the radius of the disk, and the same result indicates that the calibration result is correct.

Note: after you click calculate and save, you need to check to make sure that the disk tracking calibration made by the robot is correct. Select the calibrated user and the tool number when calibrating. Then in the robot teaching interface, move the user coordinate to the position where x is 0 and Y is 0 to see if the tool point of the robot is in the center of the circle. If it is in the center of the circle, it means that the calibration is successful, half The diameter must be the radius of the encoder acquisition position.

Store to user：

Select a user number, and the calculated result will exist in the corresponding user coordinate system.

4.Automatic calibration with vision



Enable visual cal:

Whether the visual calibration function is on.

Realtime coord tral:

Whether to convert the coordinates sent by vision.

Re:

Record the center point of the robot calibration range.

Step length:

Set the interval distance for the horizontal movement of the machine during the automatic calibration.

Deg:

Set the rotation angle of the robot when it is automatically calibrated.

Calculation:

After calibration, click calculate to calculate the relationship between pixel coordinates and robot coordinates.

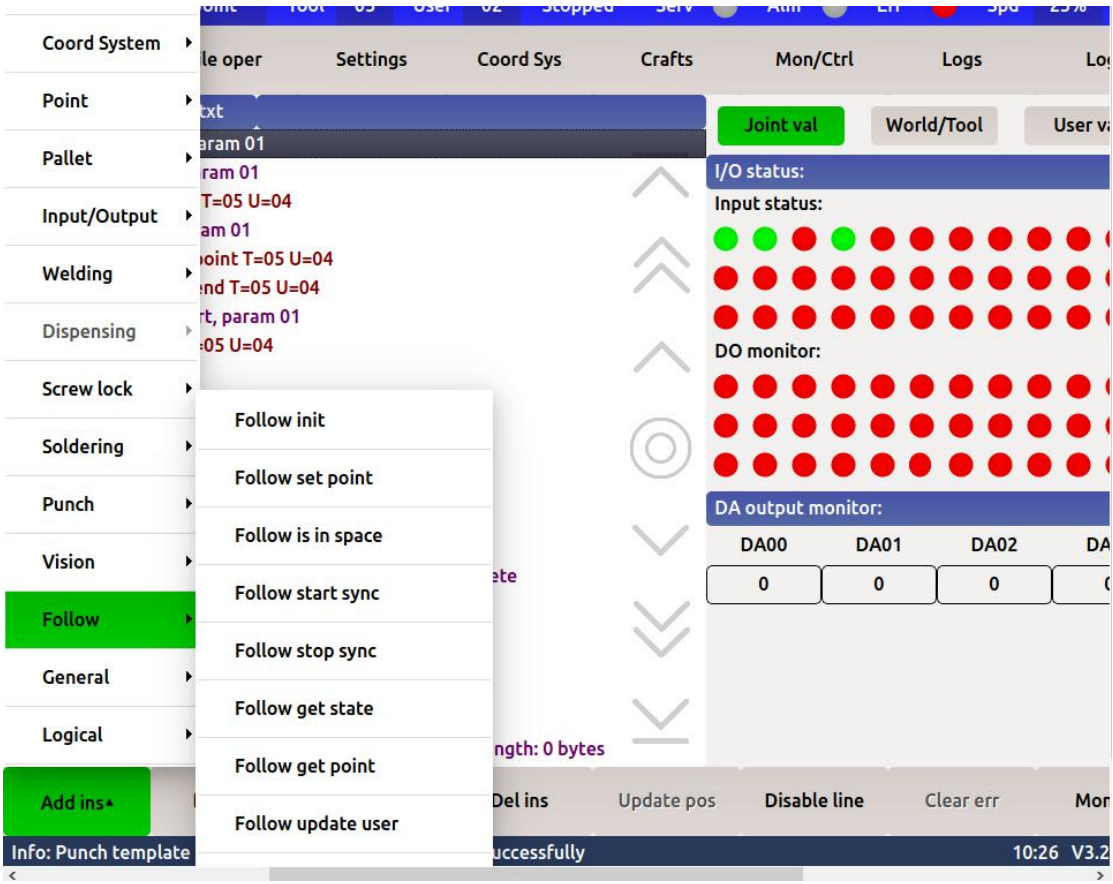
How to calibrate with vision?

When users want to directly use the pixel coordinates sent by vision, they need to calibrate the vision. First, turn off the coordinate real-time conversion, then place the feature points of the robot fixture in the center of vision, select the record, and the mark point will be saved. Set the step length and angle, then select tool 0 and user 0, and the foreground runs the visual automatic calibration program.

After the operation of the foreground, if there is no alarm, calculate and save at this time. Turn on the real-time coordinate conversion function, and the pixel coordinates sent by the next vision will be directly

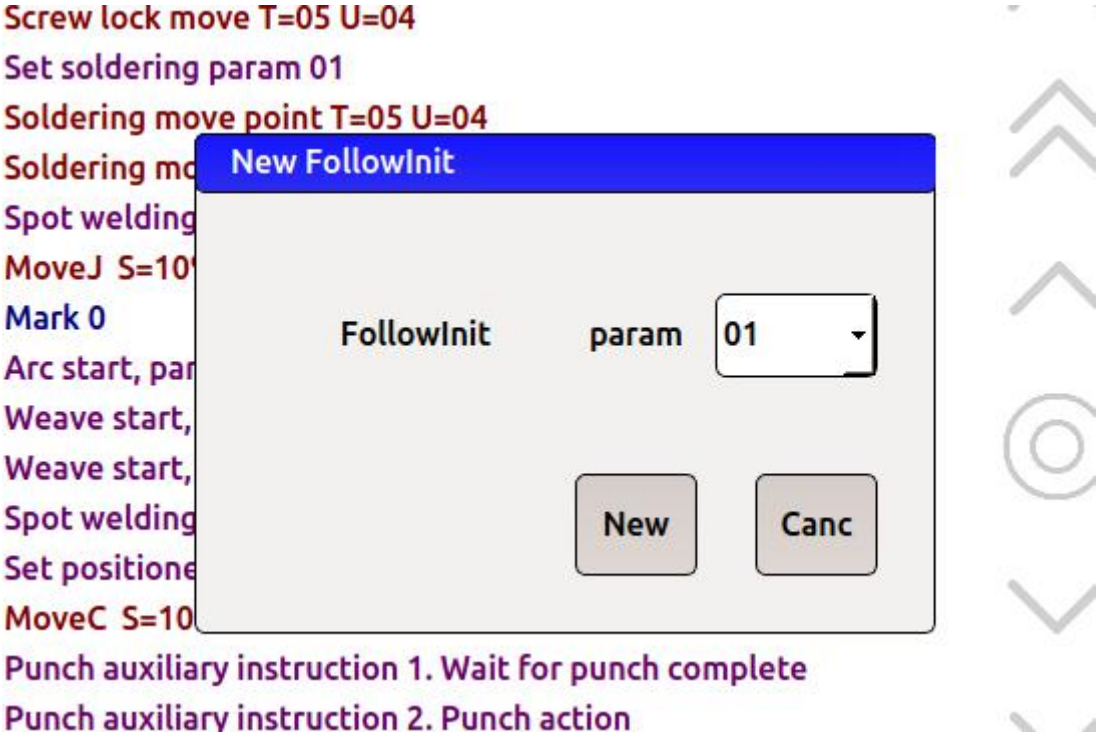
converted into the world coordinates of the robot。

6.3.2Instruction Explanation



1.Initialize trace:

Determines the ordinal number of the tracing process invoked in the program.



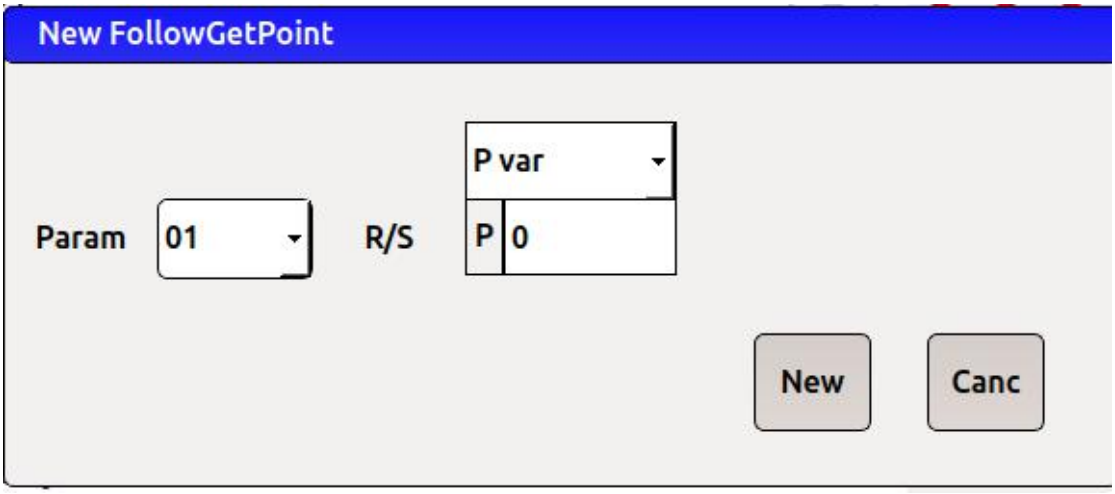
Initialize the tracking parameters.

Parameter

Parameter: Indicates to use the network configuration serial number set above.

2.Get trace points:

Read from the "storage start point" set in the tracking process to the tracking point.



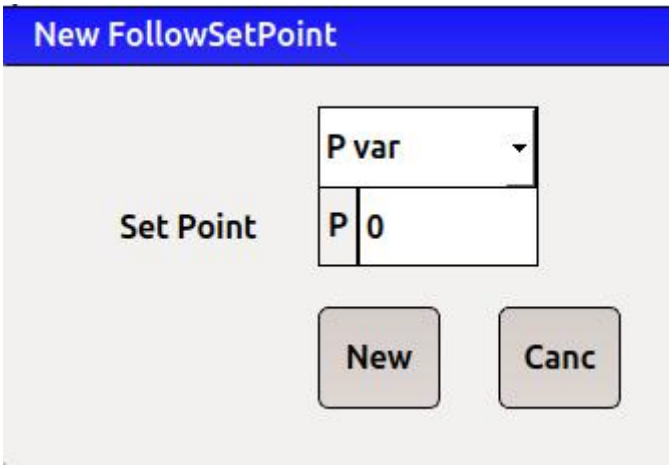
Set the tracking point.

parameter:

Parameter: Indicates to use the network configuration serial number set above. Store the trace point in the P variable so that the program can call it. The tracking point determines the position and posture of the robot when the tracking starts in the program.

3.Set trace point:

Store the trace point in the P variable so that the program can call it. The tracking point determines the position and attitude of the robot at the beginning of tracking in the program.



Get whether the workpiece is within the tracking range, and store the result in the V variable.

Parameters:

V variable: The result of whether the workpiece is within the tracking range will be obtained and stored in the specified V variable.

There are 3 possibilities for the obtained value, 0, 1, 2.

0: Indicates that the workpiece has not entered the tracking range.
1: Indicates that the workpiece is within the tracking range.
2: Indicates that the workpiece exceeds the tracking range.

4.Start tracking

图片

Start tracking. The robot executes this instruction before the robot starts tracking.

5. Finish tracking,

图片

Finish tracking, the robot executes this instruction, the robot starts to stop tracking

6. Get tracking status

图片

Get the tracking status and store it in the specified V variable.
Parameters:
V variable: The result of whether the workpiece is within the tracking range will be obtained and stored in the specified V variable.
There are 5 possible values to get, -1, 0, 1, 2, 3.
-1: Indicates that the robot tracking is in an error state.
0: Indicates that the robot tracking is in an empty state.
1: Indicates that the robot tracking is in acceleration state.
2: Indicates the robot tracking is in synchronization with the conveyor belt.
3: Indicates that the robot tracking is decelerating.

7. Get tracking points

图片

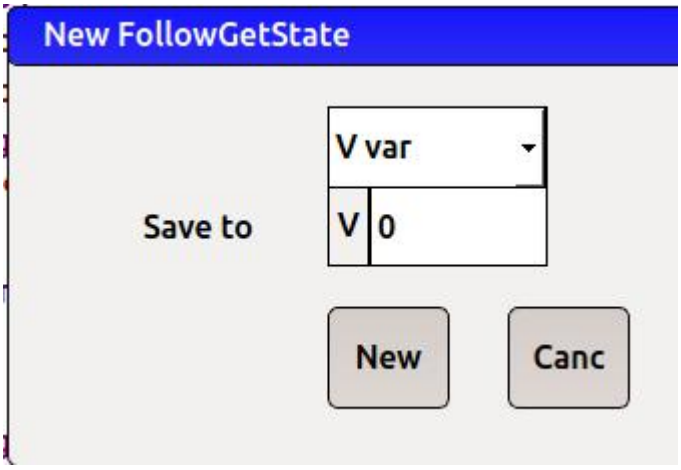
Obtain the tracking point from the tracking queue and store it in the specified P variable.
Parameters:
Parameter: Indicates to use the network configuration serial number set above.
P variable: The point obtained from the tracking queue is stored in the space coordinate system of the specified P variable.
The main application is when the trigger mode is selected to input I/O to trigger fixed-point coordinates.

8. Update user coordinates

图片

Dynamically update the user coordinates, only modify the user coordinate values in the memory, not the text user coordinates, so the modified values cannot be seen in the user coordinates of the coordinate system interface. Obtain the spatial coordinate value of P, update the position of the current tracking point, convert the coordinate of the P point to the user coordinate tracked by the conveyor belt into a dynamic user number, and there is a designated user number.
Parameters:
P variable: Change the original P point of user coordinates
User: Select the updated user number.

9. Update the location of the tracking point



Store the real-time coordinates of the tracking point in the V variable. If it is the translation tracking mode, the V variable stores the tracking X user coordinate value; if it is the disc tracking mode, the V variable stores the tracking angle value.

Parameters:

V variable: The obtained result value is stored in the V variable.

10. Set the starting position of disc tracking

图片

The tracking start point corresponding to the disc tracking parameter number is offset.

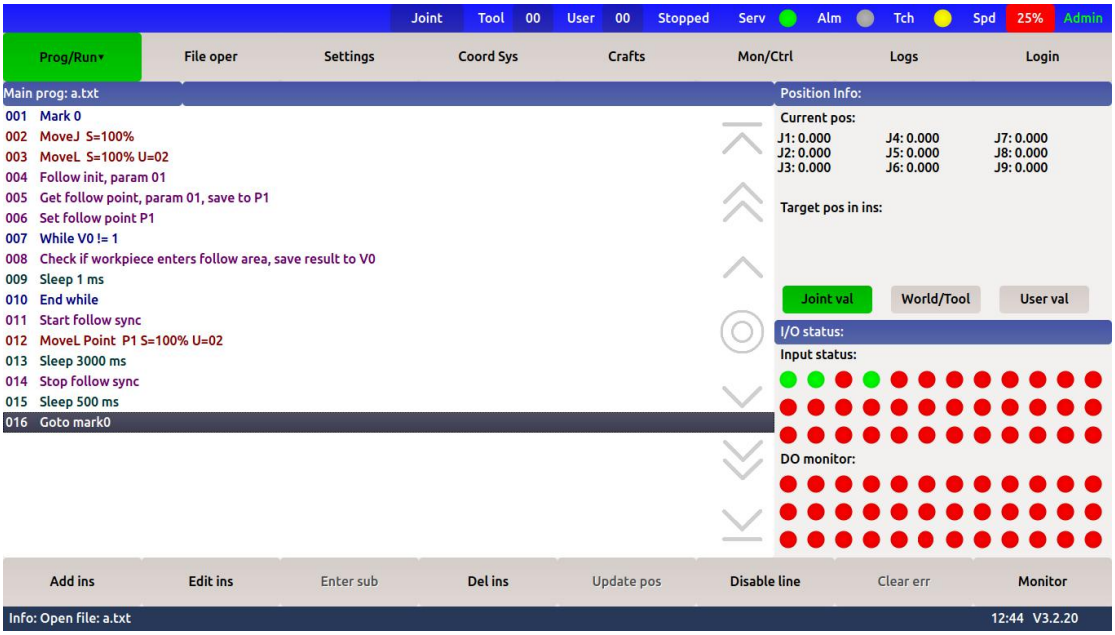
parameter:

Parameter: Indicates to use the network configuration serial number set above.

Offset: The offset of the tracking start point.

6.3.3.Case procedure:

The robot will move over the workpiece and follow the workpiece for 3 seconds.



6.4 Remote operation

6.4.1Remote access robot

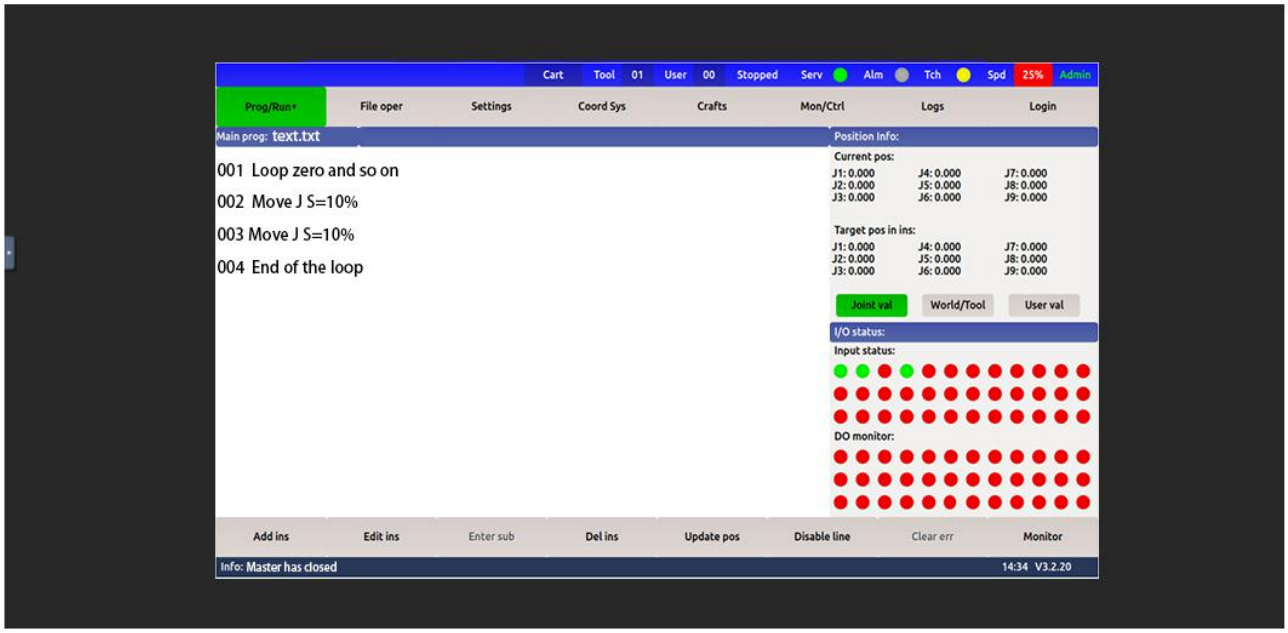
First, configure the IP address of the robot and the computer in the same network segment. See Section 2 of Chapter 6 for IP address configuration.

There are many ways to connect robots:

1. Connect the robot and computer directly through the network.

2. Connect the network port of the robot to the router, and then connect the computer, iPad and other devices to the robot through the router.

Then we just need to open the browser of the device, input the website http://ip address: 6080/vnc.html (input the IP address you set, such as http://192.168.1.7:6080/vnc.html), and then we can see the interface of the robot, as shown in the following figure:



If you need to control the robot remotely, just press the corresponding button on the keyboard, simulate the key on the teaching device, and you can operate the robot just like the teaching device. The following figure shows the mapping relationship between keyboard keys and teaching device keys.

```
#ifdef __USE_VNC_KEY__
#define KEY1A   Qt::Key_A
#define KEY1S   Qt::Key_Z
#define KEY2A   Qt::Key_S
#define KEY2S   Qt::Key_X
#define KEY3A   Qt::Key_D
#define KEY3S   Qt::Key_C
#define KEY4A   Qt::Key_F
#define KEY4S   Qt::Key_V
#define KEY5A   Qt::Key_G
#define KEY5S   Qt::Key_B
#define KEY6A   Qt::Key_H
#define KEY6S   Qt::Key_N

#define KEYSA   Qt::Key_1
#define KEYFW   Qt::Key_2
#define KEYBK   Qt::Key_3
#define KEYST   Qt::Key_4

#define KEYF1   Qt::Key_F1
#define KEYF2   Qt::Key_F2
#define KEYF3   Qt::Key_F3
#define KEYF4   Qt::Key_F4

//#ifdef __USE_VNC_KEY__
```

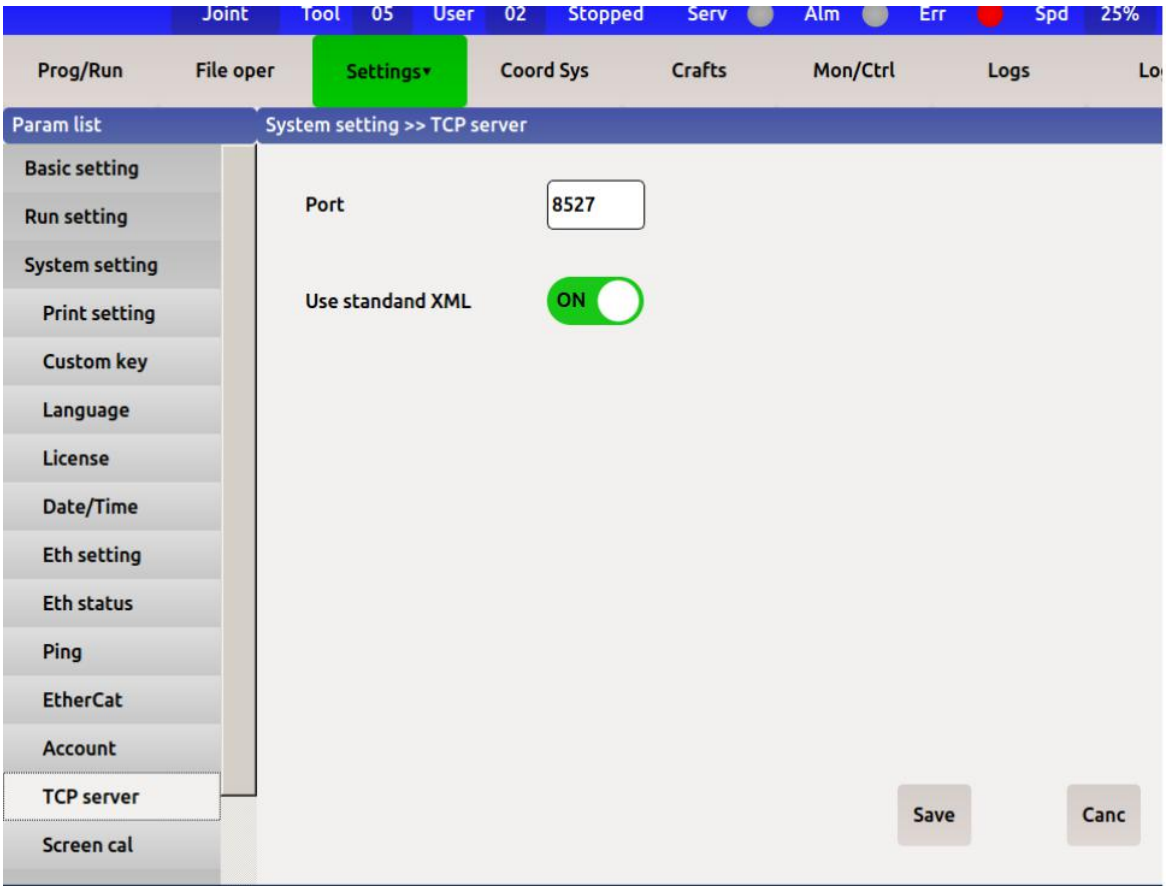
Up enable, emergency stop, switching teaching / playback and other functions still need to be operated by buttons on the teaching device or control cabinet.

6.4.2Network command to operate robot

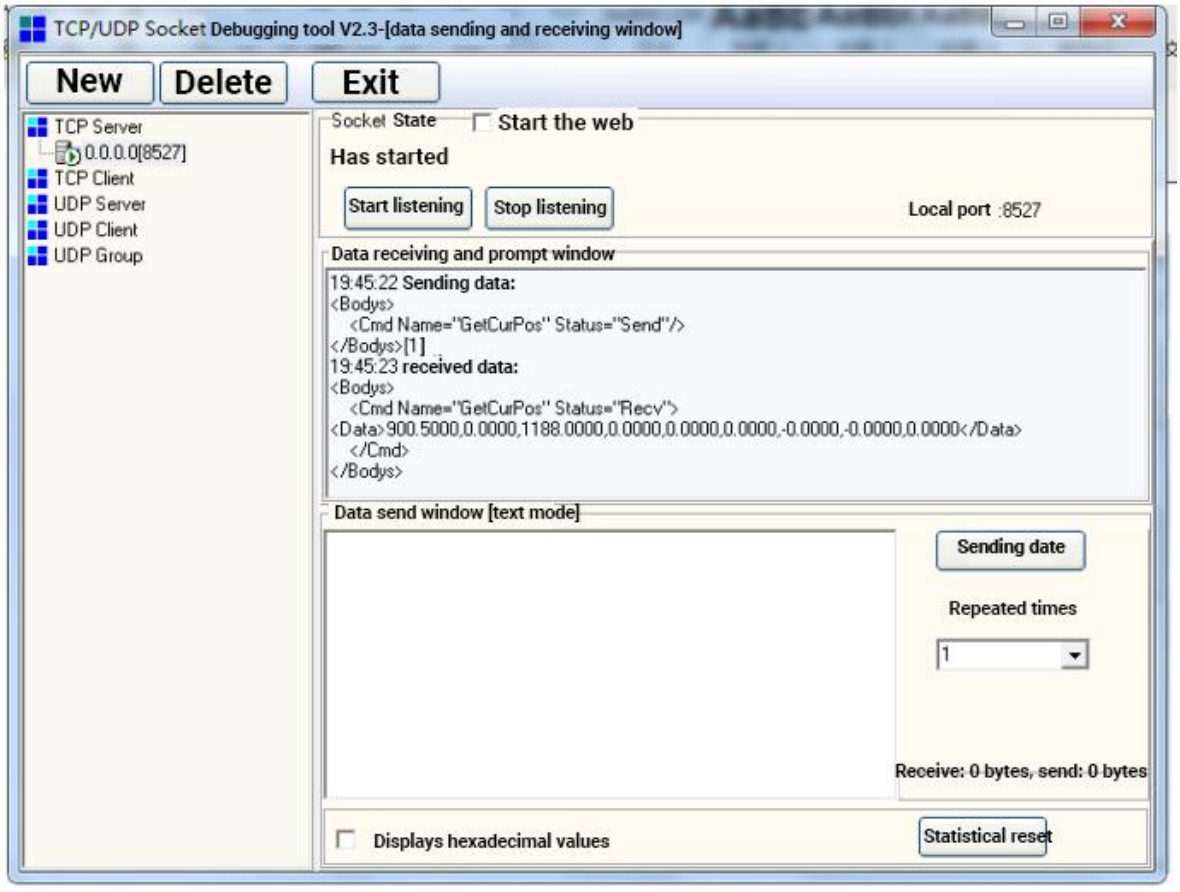
If the user needs to control the robot through the upper computer, remote command is a good choice. Remote command can control robot movement, read and write programs, read and write variables, etc. See Turing robot network command for command package.

First, set the IP of the robot, set the IP of the upper computer to the same network segment as the robot, and set the connection port number to the port number configured by the robot (default 8527),Port cannot be set to 502 (this port is set to the communication port of Modbus TCP by default).

Then the standard XML format function will be used to open the interface, as shown in the figure below.


Joint
Tool 05
User 02
Stopped
Serv
Alm
Err
Spd 25%
Prog/Run
File oper
Settings
Coord Sys
Crafts
Mon/Ctrl
Logs
Param list
System setting >> TCP server
Basic setting
Run setting
System setting
Print setting
Custom key
Language
License
Date/Time
Eth setting
Eth status
Ping
EtherCat
Account
TCP server
Screen cal
Port
8527
Use standand XML
ON
Save
Canc

test：


TCP/UDP Socket Debugging tool V2.3-[data sending and receiving window]
New
Delete
Exit
TCP Server
0.0.0.0[8527]
TCP Client
UDP Server
UDP Client
UDP Group
Socket State
Start the web
Has started
Start listening
Stop listening
Local port :8527
Data receiving and prompt window
19:45:22 Sending data:
<Bodys>
<Cmd Name="GetCurPos" Status="Send"/>
</Bodys>[1]
19:45:23 received data:
<Bodys>
<Cmd Name="GetCurPos" Status="Recv">
<Data>900.5000,0.0000,1188.0000,0.0000,0.0000,0.0000,-0.0000,-0.0000,0.0000</Data>
</Cmd>
</Bodys>
Data send window [text mode]
Sending date
Repeated times
1
Receive: 0 bytes, send: 0 bytes
Displays hexadecimal values
Statistical reset

VII. Case procedure and analysis

7.1 Machine tool double station loading and unloading

The project environment is shown in the figure. There are two machining positions in the machine tool. The end of the manipulator shall be equipped with a claw for picking and placing materials. The coding plate is double material plate, which is convenient for workers to change materials.

Action description:

1. There are two processing stations in the machine tool. For the first time, the machine tool is started to load materials unconditionally twice.
2. When the material tray is full, the completion signal will be given, and the manual intervention will be carried out for the refueling operation.
3. In the process of operation, the sampling signal is received, and the blanking is required to be put into the material sampling port.

Program resolution:

1. The program means initialization. Turn off the interactive signal with the machine tool and upper computer, and initialize the count variable. Judge the material status in the standby position, and discard if there is material.

```
001  Wait for DI 08=ON ( no comment)
002  Wait for DI 09=OFF ( no comment)
003  Wait for DI 10=ON ( no comment)
004  Wait for DI 11=OFF ( no comment)
005  Wait for DI 24=ON ( no comment)
006  Goto mark2
007  Mark 1
008  MoveJ Point  P9 S=100% B=100
009  I/O output 43=ON ( no comment)
010  I/O output 36=ON ( no comment)
011  Sleep 50 ms
012  I/O output 36=OFF ( no comment)
013  Mark 2
014  ---Scrap with materials---
015  If DI 06==OFF goto mark 55 (起弧成功信号)
016  If DI 07==OFF goto mark 56 ( no comment)
017  I/O output 02=OFF ( no comment)
018  I/O output 03=ON ( no comment)
019  I/O output 07=OFF ( no comment)
020  I/O output 08=ON ( no comment)
021  Wait for DI 12=ON ( no comment)
022  Wait for DI 13=ON ( no comment)
```

2. Choose the writing method of branch structure. Decide which tray the work starts from.

```
001  ---Mode selection---
002  Mark 3
003  MoveJ Point  P9 S=60%
004  ---Clear mode---
005  If DI 35==ON goto mark 60 ( no comment)
006  ---A code tray reclaiming---
007  If DI 26==ON goto mark 20 ( no comment)
008  I/O output 49=OFF ( no comment)
009  I/O output 39=OFF ( no comment)
010  I/O output 37=ON ( no comment)
011  Goto mark20
012  Endif
013  ---B code tray reclaiming---
014  If DI 27==ON goto mark 30 ( no comment)
015  I/O output 49=OFF ( no comment)
016  I/O output 40=OFF ( no comment)
017  I/O output 38=ON ( no comment)
018  Goto mark30
019  Endif
020  Sleep 50 ms
021  Goto mark3
```

3. It is assumed that the action starts from the a material tray. In the program, the string variable S5 is used as the flag bit of disk a. In this way, the next action can be continued from the a material tray. Let the robot work "from one end". Record the number of pallets with digital variables. Take out and discharge materials shall be counted separately, without repetition or leakage.

```
001  ---A code disk reclaiming mode---
002  Mark 20
003  I/O output 49=OFF ( no comment)
004  I/O output 39=OFF ( no comment)
005  I/O output 37=ON ( no comment)
006  Wait for DI 08=ON ( no comment)
007  Wait for DI 09=OFF ( no comment)
008  Wait for DI 10=ON ( no comment)
009  Wait for DI 11=OFF ( no comment)
010  Wait for DI 13=ON ( no comment)
011  If DI 25==ON goto mark 1 ( no comment)
012  Wait for DI 26=ON ( no comment)
013  Call sub program A code disk reclaiming sun wheel
014  If V20 < V2
015  S5 = "Pan A is picking"
016  Else if V20 == V2
017  V20 = 0
018  S5 = "Disk a is empty"
019  I/O output 49=ON ( no comment)
020  Endif
021  I/O output 09=ON ( no comment)
022  Goto mark4
```

4. After picking up the code disk, go to the machine tool and wait for the machine position. Select the branch and wait for the machine interaction signal.

```
001  ---Selection of machine reclaiming and discharging---
002  Mark 4
003  MoveJ Point  P20 S=80% B=100
004  ---Machine reclaiming mode---
005  If DI 30==ON goto mark 40 ( no comment)
006  ---Machine discharge mode---
007  If DI 33==ON goto mark 45 ( no comment)
008  Sleep 50 ms
009  Goto mark4
```

5. After reaching the standby position, load the machine tool and perform the loading action. Finish loading, don't be confused. Check the grab status and the material status in the machine tool. Decide whether to continue feeding the second machining position in the machine tool, or whether the blanking has been completed, and put it into the material tray.

```
001  ---Machine discharge---
002  Mark 45
003  I/O output 43=OFF ( no comment)
004  MoveJ Point  P25 S=100% B=100
005  MoveL Point  P26 S=60% B=100
006  Mark 46
007  MoveJ Point  P26 S=100% B=100
008  I/O output 46=OFF ( no comment)
009  Call sub program Machine tool discharge sun wheel
010  V0 += 1
011  MoveL Point  P25 S=70% B=50
012  MoveJ  S=80% B=100
013  I/O output 43=ON ( no comment)
014  If V0 <= 1
015  Goto mark47
016  Else
017  Goto mark5
018  Endif
```

6. The program enters the blanking process. There are three cutting options. Continue to place tray a, B and ab. If none of the three conditions is satisfied, continue to wait.

```
001  ---Here,the cutting process starts after the machine exchanges materials---
002  ---Material discharge selection of A and B code disk---
003  Mark 5
004  MoveJ Point  P9 S=100%
005  If V29 < V2
006  ---If disc a can be palletized and disc B is not in palletizing---
007  Goto mark25
008  Else if V29 == V2
009  I/O output 37=OFF ( no comment)
010  I/O output 39=ON ( no comment)
011  V39 = 0
012  Endif
013  ------
014  Mark 6
015  ---If disc B can be palletized and disc a is not in palletizing---
016  If V39 < V3
017  Goto mark36
018  Else if V39 == V3
019  I/O output 38=OFF ( no comment)
020  I/O output 40=ON ( no comment)
021  V29 = 0
022  Endif
```

```
001  ---AB code disk conversion judgment---
002  ---If the AB disk is in the palletizing and can be palletized---
003  If V29 > V39
004  Goto mark25
005  Else if V29 < V39
006  Goto mark36
007  Else
008  Sleep 50 ms
009  Goto mark5
010  Endif
```

7. It is assumed to feed pan a. Check the discharging result at any time. Full filling represents the completion of a disk, and the completion signal of a disk shall be given.

```
001  ---A code disc discharging mode ---
002  Mark 25
003  If DI 28==ON goto mark 50 ( no comment)
004   If V29 <= V2 && DO37 == 1
005  Call sub program A code plate discharging sun wheel
006  Mark 28
007  If V29 >= V2
008  V29 = 0
009  I/O output 37=OFF ( no comment)
010  I/O output 39=ON ( no comment)
011  Endif
012  I/O output 10=OFF ( no comment)
013  Goto mark47
014  Else if V29 == V2
015  I/O output 37=OFF ( no comment)
016  I/O output 39=ON ( no comment)
017  Sleep 1500 ms
018  Goto mark5
019  Endif
```

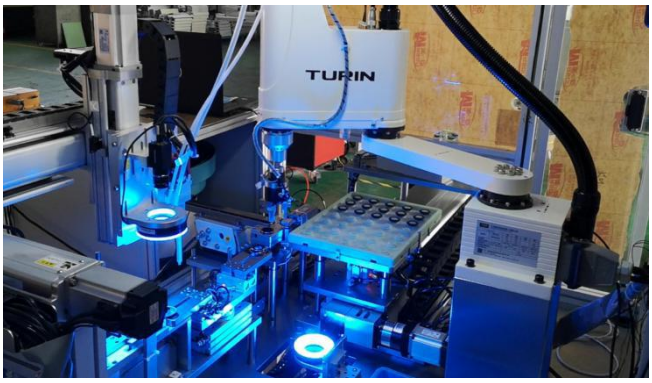
8. The cutting operation is completed, and the grab has no material, so it is necessary to take material from the code tray.
You should continue to pick up the materials from the same tray. However, because there are two machining positions in the machine tool, the material picking and discharging on the code disk should be separated by two positions. So the first disk is empty, but not full. At this time, it is necessary to make a judgment and take materials from the next tray when the next tray can be taken.
This switch is to set its own working state, the S5 string register used by the program. This kind of monitoring for the change of working state enables the manipulator to fill the first plate and take materials from the next plate.
The required functions of the project can be realized by forming a complete cycle.

```
001   Mark 47
002    If DO37 == 1&&DO38 == 0
003   If S5 == "Pan A is picking"
004   Goto mark20
005    Elseif S5 == "Disk a is empty"&&DI27 == 1
006   Wait for DI27 == 1
007   Goto mark30
008   Endif
009    Elseif DO38 == 1&&DO37 == 0
010   If S6 == "Pan B in picking"
011   Goto mark30
012    Elseif S6 == "Disk a is empty"&&DI26 == 1
013   Wait for DI26 == 1
014   Goto mark20
015   Endif
016    Elseif DO38 == 1&&DO37 == 1
017   ---If a is empty and B is fetching---
018   Wait for DI27 == 1
019   Goto mark30
020   ---Elseif B is empty and a is fetching---
021   Wait for DI27 == 1
022   Goto mark20
```

9. For the sampling inspection and discarding in the process, only the corresponding action modules need to be left. When this action is used, it can be called.

```
001   ---Abandoning mode---
002   Mark 55
003   MoveJ Point  P9 S=70% B=100
004   Wait for DI 08=ON ( no comment)
005   Wait for DI 10=ON ( no comment)
006   Wait for DI 29=OFF ( no comment)
007   Call sub program Inspection sun wheel
008   Goto mark2
009   Mark 56
010   MoveJ Point  P9 S=70% B=100
011   Wait for DI 10=ON ( no comment)
012   Wait for DI 08=ON ( no comment)
013   Wait for DI 29=OFF ( no comment)
014   Call sub program Waste sun wheel
015   Goto mark2
016   ---Clearing mode---
017   Mark 60
018   If V6 == 6
019   Goto mark0
020   Else
021   Endif
```

7.2 Robot assembling spring with vision



Action description:

The manipulator takes photos from the material tray to the top of the lower camera. Correct the position deviation and press in the spring coil.

Background script resolution:

```
001  DEFINE ERR  1                                    #
002  DEFINE WARN 2                                    #
003  DEFINE INFO 3                                    #
004  DEFINE DBG  4                                    #
005
006
007  WHILE(1)
008     SLEEP(1000)
009  DIGITAL ServerFD = 0
010  ServerFD = SocketServerInit(V900)
011     IF(0 > ServerFD)
012             PrintMsg(ERR,"SocketServerInit failure %d\n",ServerFD)
013             RETURN
014     ENDIF
015     PrintMsg(INFO,"服务器建立成功,ServerFD %d\n",ServerFD)
016
017     DIGITAL ClientFD = 0
018     ClientFD = SocketServerAccept(ServerFD)
019     IF(0 > ClientFD)
020             PrintMsg(ERR,"SocketServerAccept serverFD %d\n",ServerFD)
021             RETURN
022     ENDIF
023     PrintMsg(INFO,"客户端连接成功 %d\n",ClientFD)
024     V99 = 1
025     WHILE(1)
026             SLEEP(1)
027             STRING Buffer
028             DIGITAL Size = 0
029       while(1)
030             Buffer = ""
031             Size = SocketReadLine(ClientFD,Buffer,1000) #等待客户端发送任意长度命令
032             IF(0 > Size) #小于0一般表示连接断了
033                     PrintMsg(ERR,"Server: SocketRead 执行失败 %d\n",Size)
034                     SLEEP(1000)
035                     Goto L1
036      elseif(0 == Size)
037        continue
038      ELSE
039        BREAK
040             ENDIF
041      endwhile
042             PrintMsg(INFO,"收到数据 %s\n",Buffer)
043       DIGITAL RETVAL = 0
044             S1 = ""
045      RETVAL = RegScanfS(Buffer,"(.*)\\n?",0,S1)
```

```
046       IF(0 > RETVAL)
047         PrintMsg(ERR,"RegScanfS 执行失败 %d\n",RETVAL)
048         CONTINUE
049       ENDIF
050
051         IF(V101 == 1)
052                     SLEEP(1)
053                     STRING Buf
054         STRING RecvBuf
055                     DIGITAL Size = 0
056         DIGITAL RETVAL = 0
057         DIGITAL Index = 0
058                     Buf = ""
059               IF(V100 == 1)
060                         Sprintf(Buf, "1,%.3f %.3f %.3f\n",X,Y,RZ)
061                     ELSEIF(V100 == 2)
062                         Sprintf(Buf, "2,%.3f %.3f %.3f\n",X,Y,RZ)
063                     ELSEIF(V100 == 3)
064                         Sprintf(Buf, "3,%.3f %.3f %.3f\n",X,Y,RZ)
065                     ENDIF
066                     V101 = 0
067
068                     Size = SocketWrite(ClientFD,Buf)
069                     IF(0 > Size)
070                         PrintMsg(ERR,"SocketWrite 发送失败 %d\n",Size)
071                         BREAK
072                     ENDIF
073
074         RecvBuf = ""
075                     Size = SocketRead(ClientFD,RecvBuf,0)
076                     IF(0 > Size)
077                         PrintMsg(ERR,"Server: SocketRead size %d\n",Size)
078                         BREAK
079                     ENDIF
080                     PrintMsg(INFO,"RecvBuf %s\n",RecvBuf)
081
082                     RETVAL = RegScanfD(RecvBuf,"(.*),(.*),(.*),(.*)\\n?",0,V13,V14,V15,V16)
083                     IF(0 > RETVAL)
084                         PrintMsg(ERR,"RegScanfD failure %d\n",RETVAL)
085                         CONTINUE
086                     ENDIF
087                     V12 = 1
088                     PrintMsg(INFO,"Line %d: %.3f %.3f %.3f %.3f \n",++Index,V13,V14,V15,V16)
089         ENDIF
090     ENDWHILE
091     L1
092     SocketClose(ClientFD)
093    SocketClose(ServerFD)
094  ENDWHILE
```

Robot as server, vision as client. The overall structure of the communication script is as follows：

WHIIE(1)

Set up the server to wait for the client to connect. The connection successfully enters the next level of circulation. Block the function. If it fails, wait all the time.

WHILE(1)

while(1)

Wait for receiving the visual data unconditionally, the receiving successfully jumps out of the cycle, and the data is stored in S1。

endwhile

Send data to the client through the setting of variable V101. The value of V100 determines what data to send.

ENDWHILE

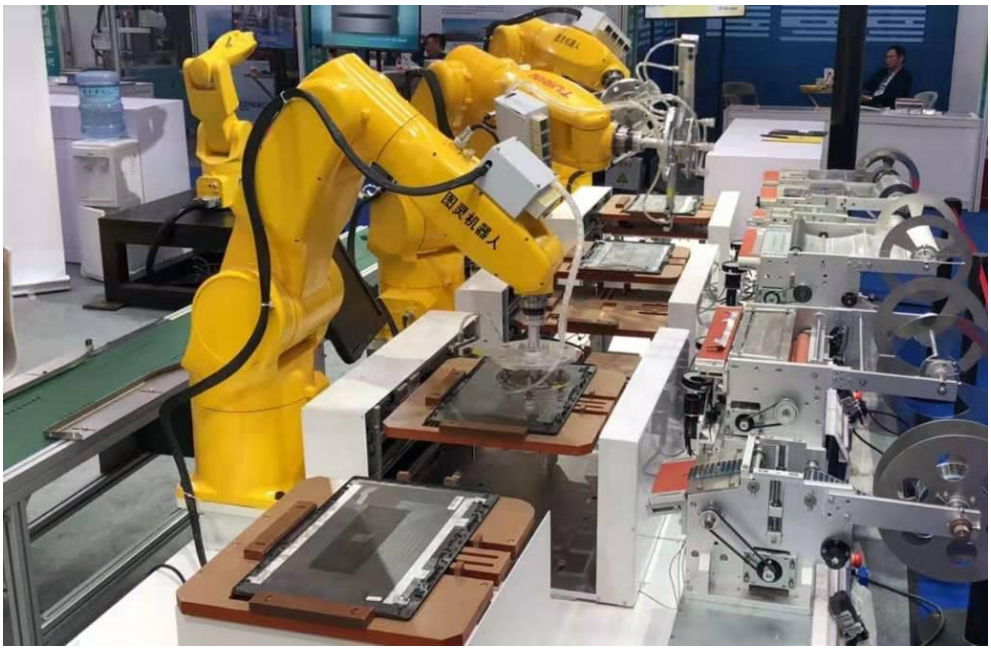
Disconnect client

Shut down the server

ENDWHILE

Once the return value of the communication related function is less than 0, it indicates the communication failure. Will jump to shut down the server and automatically reconnect.

7.3 Notebook back cover paste accessories assembly line



Action description：
Paste accessories are divided into four processing procedures, which are completed by four manipulators. Each manipulator takes materials from the stripping machine, completes the corresponding process on its own slide, and then transfers the materials to the next manipulator.
Program resolution:
1. main program (control manipulator movement only)

```
001   ---Initialization---
002   Get current pos to P10
003   Point P10 Cart Z set to 80
004   MoveL Point  P10 S=50%
005   I/O output 10=OFF ( no comment)
006   I/O output 07=OFF ( no comment)
007   I/O output 04=OFF ( no comment)
008   I/O output 09=OFF ( no comment)
009   Visual open, param 01
010   Visual trigger, param 01
011   Follow init, param 01
012   Sleep 200 ms
013   V6 = 0
014   ---Waiting for withdrawal---
015   Mark 0
016   I/O output 04=OFF ( no comment)
017   V0 = 0
018   MoveJ  S=100% U=01
019   I/O output 07=OFF ( no comment)
020   Visual recv point, param 01, save to P0
```

```
001  Mark 0
002  MoveJ Point  P15 S=100% B=100
003  ---First, judge whether there is material in the suction nozzle, and then throw the material  Take the large rubber strip ...
004  Wait for DI9 == 1
005  I/O output 10=ON ( no comment)
006  MoveJ Point  P17 S=100%
007  MoveJ Point  P16 S=50%
008  Sleep 200 ms
009  MoveJ Point  P17 S=100% B=100
010  Mark 5
011   Wait(DI6 == 1 || DI7 == 1|| DI8 == 1)
012  If DI6 == 1
013  MoveJ Point  P18 S=100% B=100
014  MoveJ Point  P19 S=50%
015  I/O output 09=ON ( no comment)
016  Sleep 100 ms
017  MoveJ Point  P18 S=100% B=100
018  MoveJ Point  P15 S=100% B=100
019  Goto mark3
020  Else if DI7 == 1
021  MoveJ Point  P18 S=100% B=100
022  MoveJ Point  P20 S=50%
```

```
001  I/O output 09=ON ( no comment)
002  Sleep 100 ms
003  MoveJ Point  P18 S=100% B=100
004  MoveJ Point  P15 S=100% B=100
005  Goto mark3
006  Else if DI8 == 1
007  MoveJ Point  P18 S=100% B=100
008  MoveJ Point  P21 S=50%
009  I/O output 09=ON ( no comment)
010  Sleep 100 ms
011  Goto mark3
012  Endif
013  MoveJ Point  P18 S=100% B=100
014  MoveJ Point  P15 S=100% B=100
015  ---At this time, judge whether the negative pressure gauge has taken the material or not, and which one has not been t..
016  Mark 3
017  If DI0 == 0
018  Goto mark0
019  Else if DI0 == 0
020  Goto mark5
021  Endif
```

```
001  Mark 1
002  ---B at the moving point, B with material, paste B---
003   If(DI28 == 1&& DI29 == 1)
004  S7 = "paste B"
005  I/O output 06=ON (起弧熄弧开关)
006  I/O output 07=ON ( no comment)
007  Wait for DI37 == 0
008  I/O output 02=OFF ( no comment)
009  I/O output 03=ON ( no comment)
010  Wait for DI28 == 1
011  Wait for DI29 == 1
012  MoveJ  S=100% B=100
013  MoveJ Point  P22 S=50%
014  I/O output 11=OFF ( no comment)
015  Sleep 100 ms
016  MoveJ  S=100% B=100
017  MoveJ Point  P24 S=100% B=100
018  MoveJ Point  P25 S=50%
```

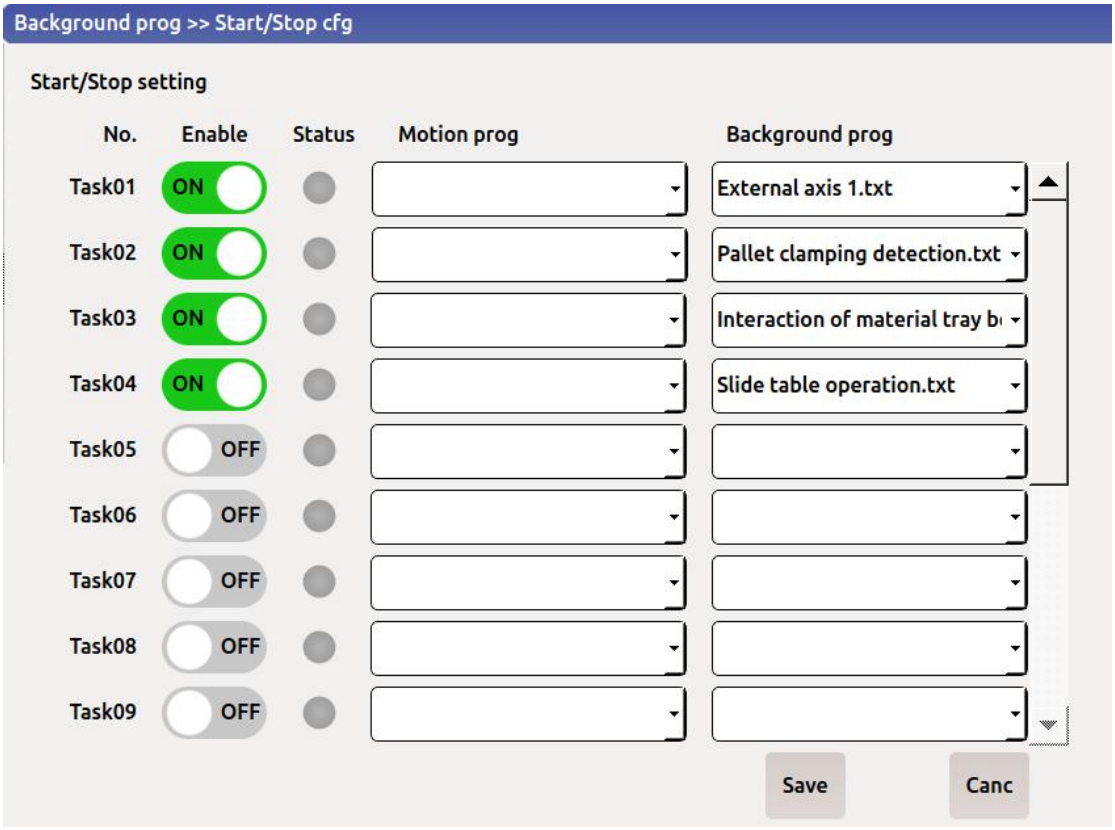
```
001  I/O output 09=OFF ( no comment)
002  Sleep 100 ms
003  MoveJ Point  P24 S=100% B=100
004  MoveJ Point  P28 S=100% B=100
005  MoveJ Point  P29 S=100% B=100
006  I/O output 08=ON ( no comment)
007  I/O output 06=OFF (起弧熄弧开关)
008  I/O output 07=OFF ( no comment)
009  Sleep 100 ms
010  MoveJ Point  P28 S=100% B=100
011  S7 = ""
012  MoveJ Point  P32 S=100% B=100
013  MoveJ Point  P33 S=100%
014  I/O output 08=OFF ( no comment)
015  Sleep 100 ms
016  MoveJ Point  P32 S=100% B=100
017  MoveJ Point  P15 S=100% B=100
018  Goto mark0
```

```
001   ---A at the moving point, a with material, B at the origin, paste a---
002    Elseif(DI25 == 1 && DI26 == 1&& DI27 == 1)
003   S8 = "paste a"
004   I/O output 04=ON ( no comment)
005   I/O output 05=ON ( no comment)
006   Wait for DI37 == 0
007   I/O output 13=OFF ( no comment)
008   I/O output 16=ON ( no comment)
009   Wait for DI25 == 1
010   Wait for DI27 == 1
011   Wait for DI26 == 1
012   MoveJ  S=100% B=100
013   MoveJ Point  P26 S=100% B=100
014   MoveJ Point  P27 S=50%
015   I/O output 09=OFF ( no comment)
016   Sleep 100 ms
017   MoveJ Point  P26 S=100% B=100
018   MoveJ Point  P30 S=100% B=100
019   MoveJ Point  P31 S=100%
020   I/O output 08=ON ( no comment)
021   I/O output 04=OFF ( no comment)
```

```
001   I/O output 05=OFF ( no comment)
002   Sleep 100 ms
003   MoveJ Point  P30 S=100% B=100
004   S8 = ""
005   MoveJ Point  P32 S=100% B=100
006   MoveJ Point  P33 S=100%
007   I/O output 08=OFF ( no comment)
008   Sleep 100 ms
009   MoveJ Point  P32 S=100% B=100
010   MoveJ Point  P15 S=100% B=100
011   Goto mark0
012   Endif
013   Sleep 50 ms
014   Goto mark1
```

Analysis:
In the first section of the program, reset is performed to raise the manipulator and initialize the signal. Then wait for the stripper to discharge. After picking, it is the judgment of several states. For the pallet that meets the conditions, the labeling action is directly executed. Control of the pallet and external axis is performed in the background.

2.Background configuration



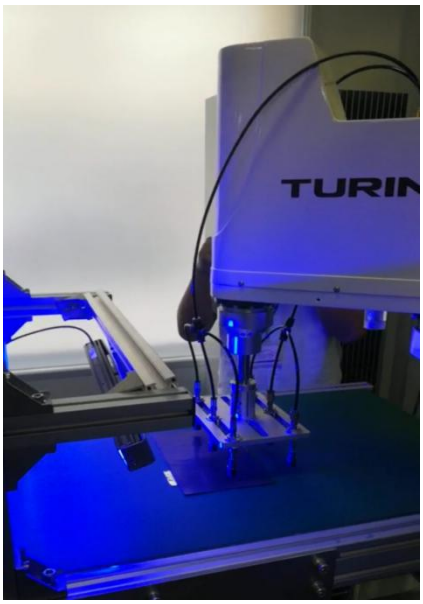
滑台动作.txt

```
WHILE(1)
    SLEEP(1)

#托盘无料，清掉准备好命令
If( DI26 == 0 )
        S5 = ""
ENDIF

If( DI29 == 0 )
        S6 = ""
ENDIF

#B没有料，汽缸松开
IF ( DI29 == 0 )
        SLEEP(1000)
        DO7 = 0
        DO6 = 0
ENDIF

#A没有料，汽缸松开
IF( DI26 == 0 )
        SLEEP(1000)
        DO4 = 0
        DO5 = 0
ENDIF

#A在原点，A有料，A准备好，不在取A动作中，A进
IF( DI24 == 1 && DI26 == 1 && S5 == "A盘准备好" && S8 == "" )
        DO13 = 0
        DO16 = 1
ENDIF
```

```
#A在动点，A无料，不在取A动作中，A回
If( DI25 == 1 && DI26 == 0 && S8 == "" )
        DO13 = 1
        DO16 = 0
ENDIF

#取A动作中，B回
IF( S8 == "贴A中" )
        DO2 = 1
        DO3 = 0
ENDIF

#B在原点，B有料，B准备好，A不在动点，不在取A动作中，不在取B动作中，B进
IF( DI27 == 1 && DI29 == 1 && DI25 == 0 && S6 == "B盘准备好" && S7 == "" && S8 == "" )
        DO2 = 0
        DO3 = 1
ENDIF

#B在动点，B无料，不在取B动作中，B回
IF( DI28 == 1 && DI29 == 0 && S7 == "" )
        DO2 = 1
        DO3 = 0
ENDIF

#A在动点或者无料，清除A准备好
IF( DI25 == 1 || DI26 == 0 )
        S5 = ""
ENDIF
#B在动点或者无料，清除B准备好
IF( DI28 == 1 || DI29 == 0 )
        S6 = ""
ENDIF

#A在原点并且有料，夹紧
IF( DI24 == 1 && DI26 == 1 )
        SLEEP(1000)
        DO4 = 1
        DO5 = 1
ENDIF

#B在原点并且有料，夹紧
IF ( DI27 == 1 && DI29 == 1 )
        SLEEP(1000)
        DO7 = 1
        DO6 = 1
ENDIF

ENDWHILE
```

纯文本 ▾　制表符宽度：8 ▾　行 33，列 21　插入

7.4 Pipeline tracking and grabbing battery



Action description：

The material state is detected by photoelectric switch. When the material enters the photographing area, the photoelectric switch detects the incoming material and sends IO signal to the robot. The robot receives IO signal and triggers the camera to take photos through the network command. After the camera takes photos, it sends the coordinates to the manipulator. At this time, when the material enters the tracking range, the manipulator performs correction and grabbing.

Program parsing：

```
001  ---Initialization---
002  Get current pos to P10
003  Point P10 Cart Z set to 80
004  MoveL Point  P10 S=50%
005  I/O output 10=OFF ( no comment)
006  I/O output 07=OFF ( no comment)
007  I/O output 04=OFF ( no comment)
008  I/O output 09=OFF ( no comment)
009  Visual open, param 01
010  Visual trigger, param 01
011  Follow init, param 01
012  Sleep 200 ms
013  V6 = 0
014  ---Waiting for withdrawal---
015  Mark 0
016  I/O output 04=OFF ( no comment)
017  V0 = 0
018  MoveJ  S=100% U=01
019  I/O output 07=OFF ( no comment)
020  Visual recv point, param 01, save to P0
```

```
001  ---OK/NG judge---
002  If S4 != "1.0"
003  I/O output 10=ON ( no comment)
004  Sleep 1000 ms
005  I/O output 10=OFF ( no comment)
006  Goto mark0
007  Endif
008  ---Tracking grasping---
009  Point P0 Cart X add offset -370
010  Set follow point P0
011  While V0 != 1
012  If V0 == 2
013  Sleep 10 ms
014  Goto mark0
015  Endif
016  Sleep 10 ms
017  Check if workpiece enters follow area, save result to V0
018  End while
019  Update follow point pos, save result to V5
020  If V5 >= V6
021  Goto mark0
022  Endif
```

```
001  Start follow sync
002  Point P0 Cart Z set to 50
003  MoveL Point  P0 S=100% U=01
004  Point P0 Cart Z set to 35
005  MoveL Point  P0 S=100% U=01
006  I/O output 04=ON ( no comment)
007  Sleep 150 ms
008  Point P0 Cart Z set to 56
009  MoveL Point  P0 S=100% U=01
010  Stop follow sync
011  Sleep 3000 ms
012  If DI 13==OFF goto mark 0 ( no comment)
```

```
001  Mark 1
002  ---To A material box---
003  If DI11 == 1
004  MoveJ  S=100% U=01
005  ---To B material box---
006  Else if DI12 == 1
007  MoveJ  S=100% U=01
008  Else
009  Sleep 100 ms
010  Goto mark1
011  Endif
012  ---Discharge---
013  Get current pos to P1
014  Point P1 Joint 3 set to -60
015  MoveJ Point  P1 S=60%
016  I/O output 08=OFF ( no comment)
017  I/O output 09=ON ( no comment)
018  Sleep 150 ms
019  I/O output 09=OFF ( no comment)
020  Point P1 Joint 3 set to -14
021  MoveJ Point  P1 S=60%
```

7.5 Determine the current position of the robot

Action description:

The manipulator moves to the standby position, obtains the current position to P0, compares the X value of P0 with the boundary value V2, and compares the Y value of P0 with the boundary value V3. According to the difference of the current stopping position, a safe path is selected and the corresponding transition point is moved.

Program analysis:

图片

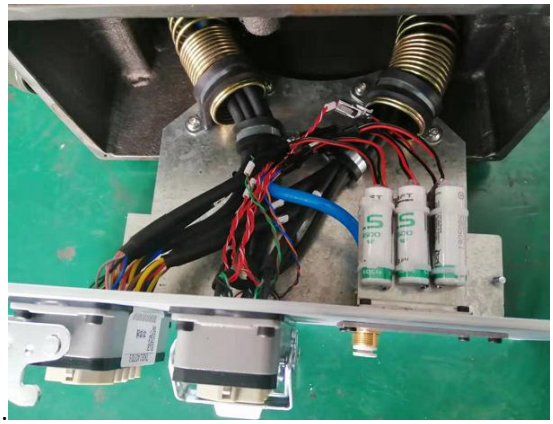
VIII. Frequently asked questions of users

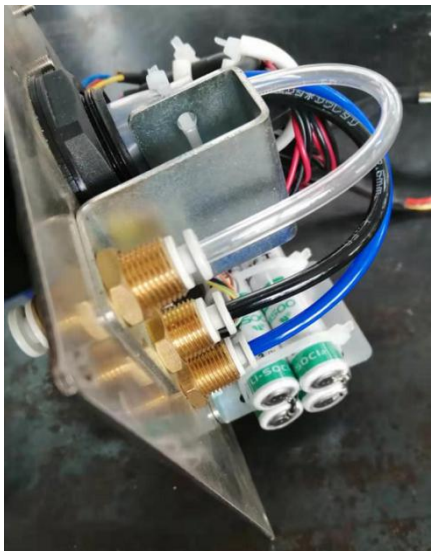
8.1 Location misalignment

First of all, repeat positioning at any time should be accurate. If the repeated positioning is out of alignment, first check the mechanical faults, such as: the mechanical structure is loose, the belt is aging, the reducer is damaged, etc.
The misalignment here refers to the displacement of the manipulator's own coordinate system caused by the power failure of the encoder, impact and other factors.

8.1.1Encoder battery replacement

When the low voltage alarm of encoder battery occurs in the system, the battery needs to be replaced. During annual maintenance, the battery voltage needs to be detected. The specific operation is: open the small battery cover plate of encoder on the back cover of robot base, and measure the battery voltage with a multimeter. The normal rated voltage is 3.6V. When the voltage is lower than 3.2V, the battery needs to be replaced.
The encoder batteries of Turing robot are all installed on the back cover of the base, and the batteries can be replaced by removing the back cover (some models have a small battery cover plate, which only needs to be removed to replace the batteries)





Note: when replacing the battery, the robot needs to return to the zero position and lock the robot joint with the motor holding brake. After replacement, the zero point needs to be recalibrated. If you accidentally lose the zero position, you need to recalibrate.

8.1.2 20 point calibration

When the encoder is powered off and impacted by external force, resulting in relative displacement between the motor shaft and the mechanical structure, the position records are all in an unrecoverable state. It is necessary to recalibrate the zero point, and the error of zero point calibration by naked eyes is large. At this time, it is necessary to use the 20 point calibration method.

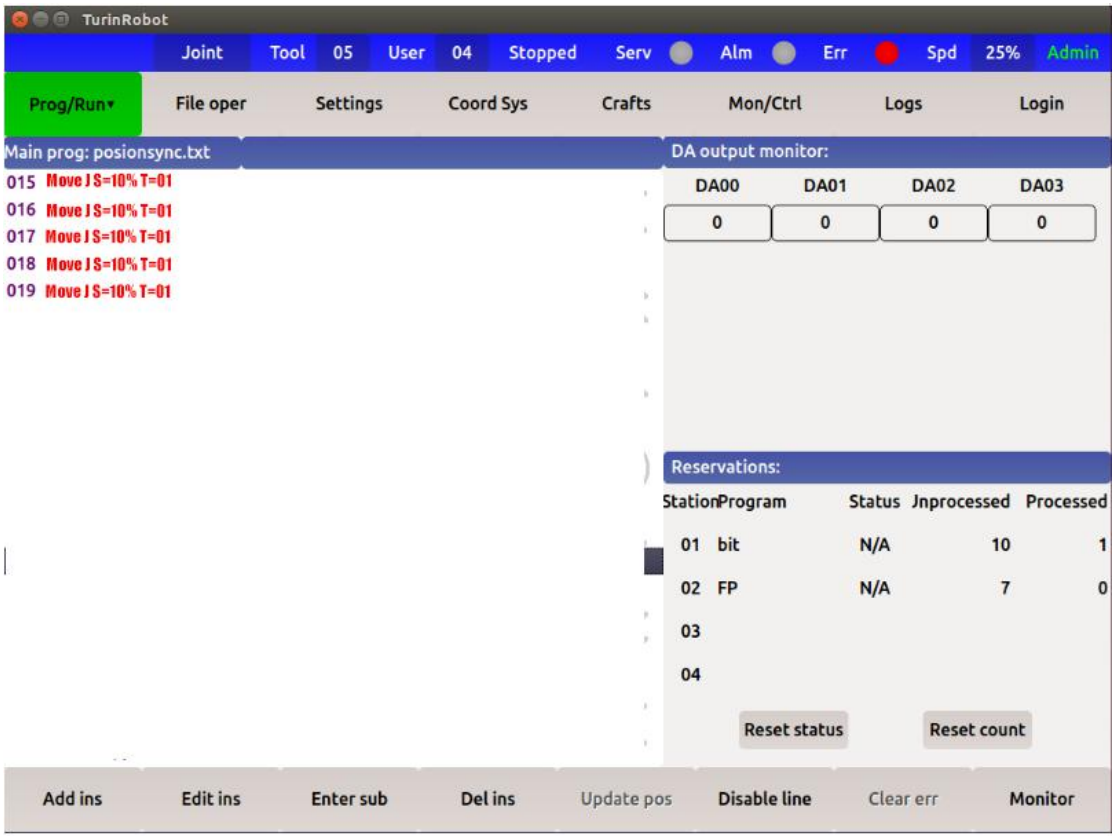
Before calibration, it is necessary to ensure that the parameters of the robot's linkage, deceleration ratio and coupling ratio are accurate, and the zero position should be within the allowable deviation.

The specific calibration steps are as follows.

First step： Make two tip calibration rods (the sharper the better), one is installed at the center point of the tool to be calibrated (the tip of the tool itself can be used, such as welding wire, but the sharpening is required), and the other is placed horizontally on the workbench (cannot be moved).

The second step： Create a new file on the teachpad with a custom file name.

The third step： Open the program. Run the robot in manual teaching mode, align the two tip calibration rods, and then add a joint motion command. Lift the manipulator, change the swing posture, repeat the above steps, and record 20 points in total. The following picture:

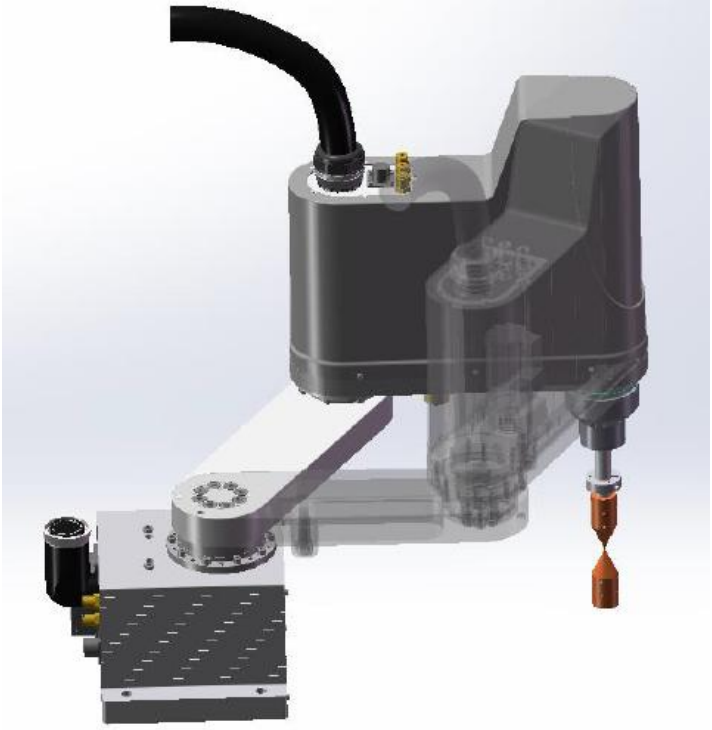


Explain：

➢ When making 20 points of a six axis robot, make the tip calibration bar on the robot form a certain angle with the tip calibration bar on the workbench, and align. Twenty points are scattered around, and the bigger the difference of robot's attitude, the better.

➢ When making 20 points of a four-axis robot, place the tip calibration bar on the workbench at any position within the robot's motion range, align the left and right hand posture of the manipulator to the same point of the lower thimble (the center of the thimble used by the manipulator shall be concentric with the rotation center of the screw rod), and then record the motion instructions of the two joints. Ten positions are scattered in the moving range of the robot, and the more scattered the better.

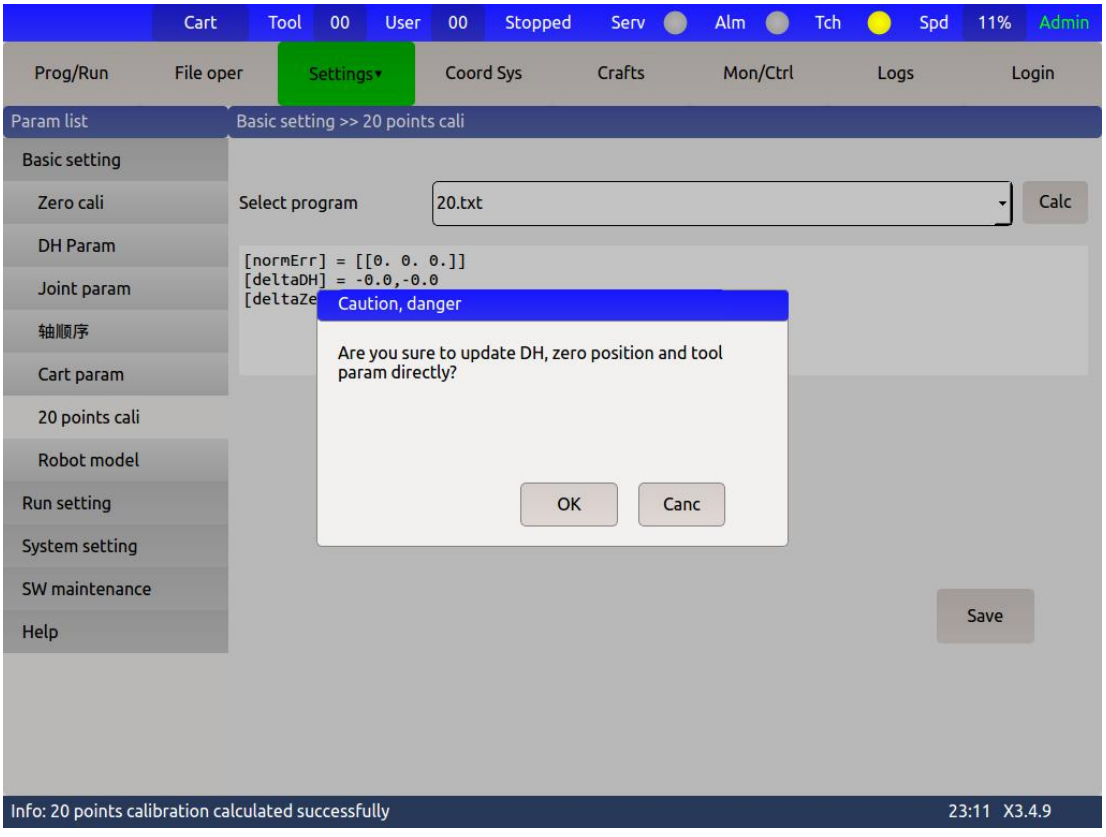
Six axis robot at 20 o'clock



Four axis robot at twenty

The fourth step： Switch to [parameter setting] - [basic setting] - [20 point calibration] on the teaching pendant, select the script file established in the second step, and click [calculation]. The progress will be displayed in the pop-up window during calculation. Click OK after calculation. The following information bar will show: 20 points are calculated successfully.

The fifth step： Click Save to open the window update to link, zero point and tool directly? 】If you click OK, the system parameters will be changed directly; if you click cancel or do not save, you need to update the system parameters manually.



The following description is made for the calculation results of 20 points:

【normErr】: Tool coordinate error

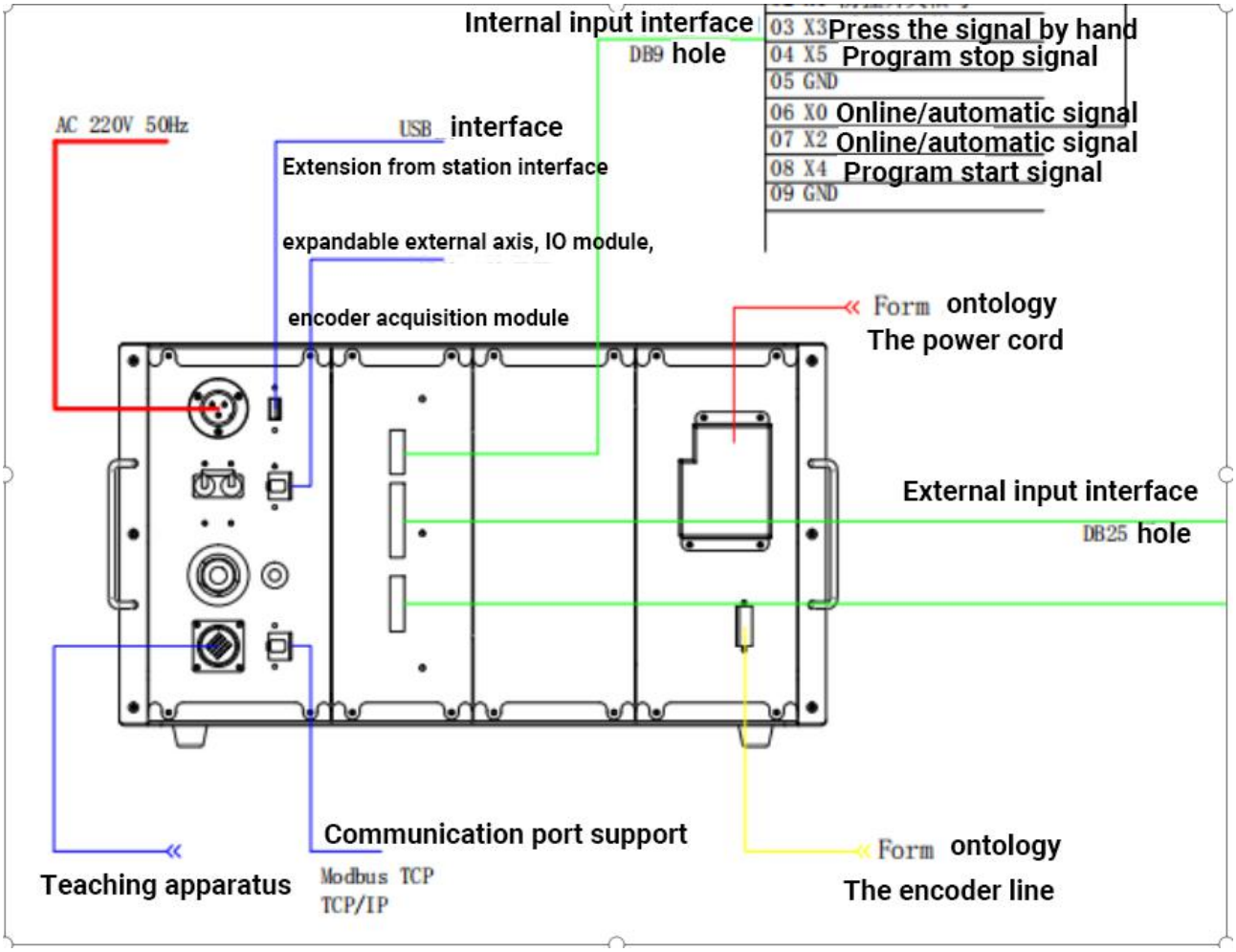
【deltaDH】: Calculated error of connecting rod parameters

【deltaZero】: Calculated zero error

Note: Generally speaking, the calibration value will not exceed 5mm. If it exceeds 5mm, it may be the joint parameter or zero point calibration setting problem. It is recommended to check it before proceeding

8.2 Insufficient physical IO points

8.2.1Add IO expansion board

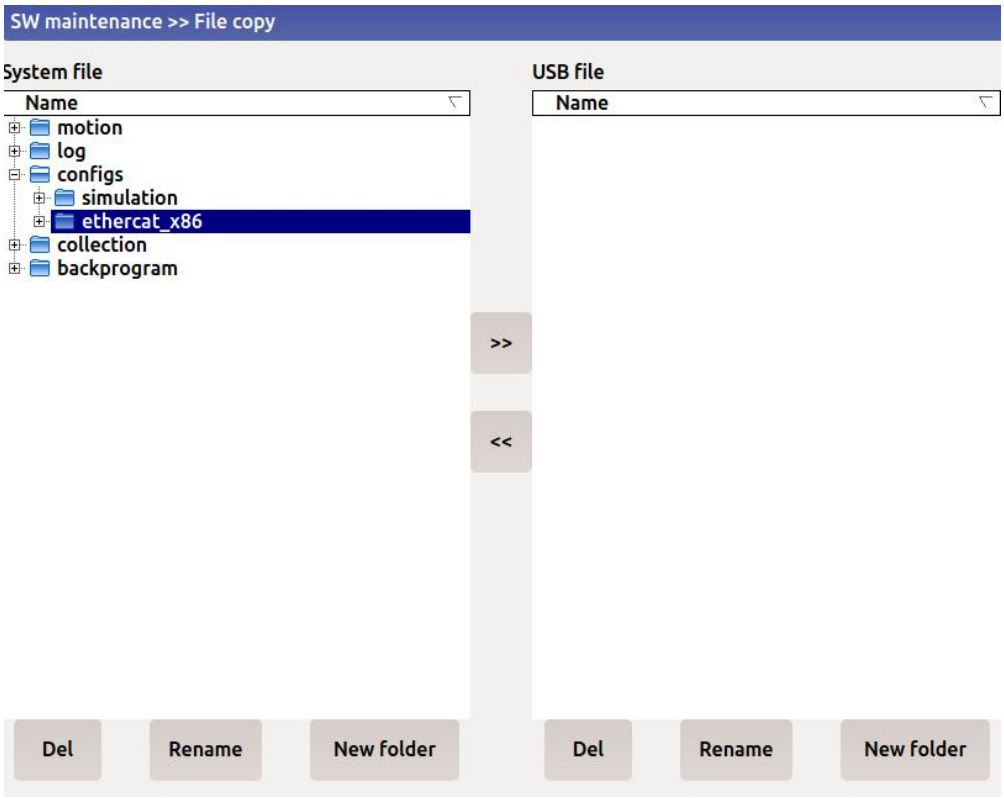


A series control cabinet is equipped with two network ports. TCP / IP, MODBUS and other protocol communication between Internet interface and external equipment. The Ethernet port is a network port that

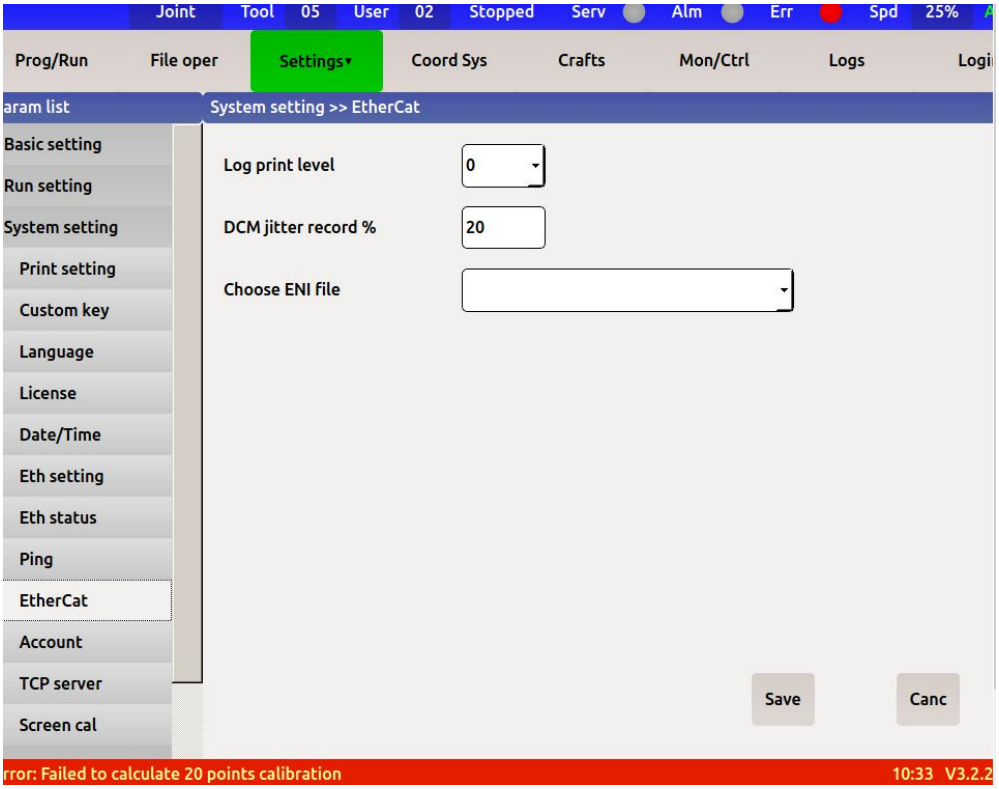
uses the Ethernet bus inside the robot.

After the extended IO board is powered on and connected with the Ethernet port of the control cabinet, the robot will alarm the master station for abnormal communication. At this time, the configuration of the slave station has changed, and the correct communication file has been replaced. That is to say, the expansion board can only be used after being added to the configuration.

First, use the U disk to copy the communication files to the x86 folder under configs.



Then, in the [system settings] - > EtherCAT] interface, replace the Eni file. Save and restart.



8.2.2 Using Modbus Protocol

See section 6.2.4 for details.

8.3 Emergency stop signal can not be reset

The teaching interface displays the emergency stop status, and the emergency stop status cannot be released, and the robot cannot move.

8.3.1Check communication status

As the master station, the controller reads the input signal of IO board through the EtherCAT communication. " The input point of port 0 "is the emergency stop state input, and the input point of port 0" should be the normally closed state so that the manipulator can work normally.
First, make sure that the EtherCAT communication is in a normal state. By contrast, press the manual enable switch, turn the teaching / playback switch key, check whether the controller can read the corresponding input signal, and check whether the IO board optocoupler has corresponding changes.
If the status changes correctly, the communication is normal. Otherwise, the cause of the emergency stop may be the abnormal software version of the controller, the mismatching of the EtherCAT communication configuration file, the overdue authorization of the controller system or the damage of the IO board.

8.3.2 check the emergency stop input signal

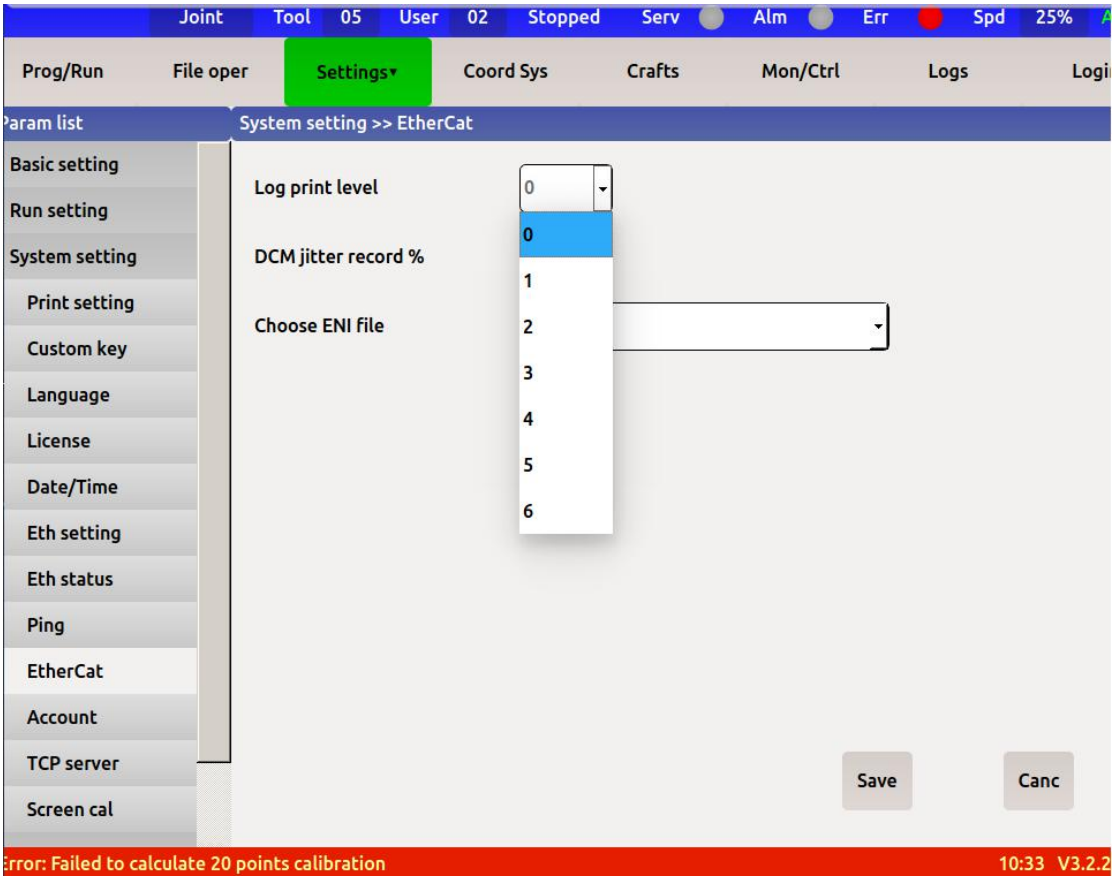
The manipulator can normally read the input signals of other input points. It indicates that the input source of emergency stop input signal is abnormal.
Check the emergency stop switch. The manipulator has three emergency stop switches, emergency stop button of teaching device, emergency stop button of control cabinet and external emergency stop input. The teaching pendant and control cabinet emergency stop buttons shall be raised.
Series a control cabinet has DB9 input connector and built-in passive input point. Check the connection status of connector and internal signal wire.
B series control cabinet, with external emergency stop input at the input terminal. Check the connection status of the corresponding contact signal wire of the terminal block.

8.4Main station related alarm

1.Alarm code 30002
30002 alarm usually occurs when the communication file does not match the actual connection state. For example, the communication file is not replaced when the external axis and IO board are expanded.
The processing method is to check the consistency of hardware connection and communication file order.
2.Alarm code 1000A
1000A alarm means lost frame in communication. Generally, it occurs in the case of strong external interference, unqualified network cable type, virtual connection of network port, etc.
Firstly, the external interference source is found, and the magnetic ring, lead blanket and other shielding electromagnetic interference equipment are added. Second, try to replace the network cable to check whether there is virtual connection。
3.Alarm code 0x65
0x65 alarm means that the slave station disappears. Generally, it occurs in the unstable power supply circuit,

which leads to the loss of power in the communication chip of a slave station. Or the network connection has an open circuit, and the slave station cannot be found.

First, check the status of network cable connection and network port light.



Set the output level of the master station log to "3", and copy the operation log for the manufacturer's assistance. (Note: the master station log printing level is set to "3", which will print a large number of master station related logs, resulting in too much hard disk garbage, affecting the system operation. Therefore, the general recurrence problem is to restore the log output level to 0 after collecting the related logs.)