CRP-S80
Operation instruction

This system supports several types of robot, and make a part instruction about the six-joint robot. Please refer to this instruction and debug manual for other type of robots.

Please make sure the operator of this product accepted the instruction related.

CRP Co., Ltd 2016-11-4
Safety Attentions

Please read carefully and understand totally of these instructions and other related documents BEFORE using this system. Please start using the system ONLY AFTER mastering all equipment knowledge, safety knowledge, and notices.

Safety attentions in this instruction are graded into four categories- DANGER, CAUTION, MUST, and FORBID indicating that death or severe injury will result if wrong operation is taken.

indicating that medium or slight injury or equipment breakdown will result if wrong operation is taken.

instructions that must be obey.

operations that mustn’t be taken.

It should be noticed that things labeled with CAUTION may also result into severe consequences in different situation. So it’s very important to pay attention to every CAUTION instructions, and stay strictly with them.

Some instructions even may not be labeled with DANGER or CAUTION, but users should also obey them.
Before operating the robot, press the “emergency stop” button on the teach box and make sure main power supply of servo is off and motor is in “off-power and brake” status. After cutting off the servo power, servo power button on the teach box is red.

Under an emergency, personal injury or equipment damage may result if robot cannot be stopped in time.

When emergency is removed and servo power need connecting, please remove and deal properly with the incidence which causes the emergency, and then connect the servo power.

Personal injury may result due to the unwanted robot movement from wrong operation.

When teaching within the robot movement region, please obey the following principle: keep observe the robot from the front

- follow the operation manual strictly
- consider the backup plan if robot move suddenly toward yourself
- set up the shelter just in case

Personal injury may result due to the unwanted robot movement from wrong operation.

When following processes are operated, make sure nobody is in the robot movement region and operator is in the safe position: powering the robot control electric tank controlling the robot using teaching programmer

test run of robot

reoccurrence automatically

Personal injury may result when entering the robot movement region or contacting the robot.
Press the emergency stop button immediately when exception occurs.

CAUTION

★ Following items need confirming before operating robot:
Operator is well-trained with robot-operation;
Operator has adequate knowledge of the movement feature of robot;
Operator is fully aware of the danger of robot;
Operator hasn’t drink alcohol;
Operator hasn’t take drugs that affect the nerve system and make reaction slow.

★ Following items need checking before robot teaching process. If any abnormal is found, fix it or take other necessary measurement immediately.
Movement of robot is normal or not;
Zero point adjustment is correct or not;
Auxiliary equipment connected to robot is normal or not.

★ Please put the teach equipment to original position after usage, and place it firmly.
Please don’t put the teach programmer on robots, claps or ground. Otherwise, personal injury or equipment damage may result when teach programmer has collision with robot or claps.
Please be cautious with the incidental falling of teach programmer, which may leads to wrong movement of robot, personal injury and equipment damage.
MUST

Safe operation protocols

★ All the operators of robot systems should participate in system training, study safety protection measurements and understand robot functionality.

★ Make sure no abnormal or dangerous situation of robot and auxiliary equipment occurs, before starting operating robot.

★ Unplug the power or press emergency stop button before entering the operation region, even robots aren’t running.

★ When programming within the movement region of robot, corresponding guard should be set to make sure that robot can be stopped under emergency circumstances. Please don’t wear gloves when teach and inch the robot. Please take the slow mode when inching the robot, in case emergency occurs and robot can be stopped efficiently.

★ Operators must be familiar with the emergency stop buttons on the robot controller and periphery controller, to the degree that these buttons can be pressed exactly under emergency.

★ Never think that static status of robot means the finish of program. Static robot is likely to receive the input signal to get moving.
# catalog

Safety Attentions .................................................................................................................. 1

Chapter 1 Introduction to System ......................................................................................... 9

1 Introduction to system hardware ....................................................................................... 10
    1.1 Basic information of system .................................................................................. 10
    1.2 System specification parameters ....................................................................... 10

2 System exterior .................................................................................................................. 11
    2.1 Teach .................................................................................................................... 11

3 Display of Teach ................................................................................................................. 17
    3.1 Main interface domains ....................................................................................... 17
    3.2 General display area .......................................................................................... 17
    3.3 Monitor area ......................................................................................................... 19
    3.4 Information area ................................................................................................... 20
    3.5 Main menu area .................................................................................................... 21
    3.6 Status control area .............................................................................................. 25
    3.7 Coordination button area ................................................................................... 27
    3.8 Status display area ............................................................................................. 28
    3.9 Sub menu area ...................................................................................................... 29

Chapter 2 TEACH MODE .................................................................................................. 30

1 Available movement in Teach mode ................................................................................. 31

2 Simple manual movement ............................................................................................... 33
    2.1 Correct gesture to use teach box ........................................................................ 31
    2.2 Simple manual movement .................................................................................. 33

3 Coordination system ........................................................................................................ 34
    3.1 Types of coordination system ............................................................................ 34
    3.2 Invoke and switch coordination system ............................................................... 36
    3.3 Set the coordination system .............................................................................. 39

4 **Zero** return operaion .................................................................................................. 51
    4.1 Absolute programmer servo institute: ................................................................. 51
    4.2 Incremental programmer servo institute: ............................................................ 51

5 Program instructions ...................................................................................................... 56
    5.1 Invoke of program instruction ............................................................................ 56
    5.2 List of program instructions .............................................................................. 57
    5.3 Intruduction of program instructions ................................................................ 59

6 Teach program .................................................................................................................. 101
    6.1 Welding teach program ...................................................................................... 102
6.2 Lifting program example.................................................................108
7 Program edit functions........................................................................113
  7.1 Edit in Program list........................................................................113
  7.2 Edit in program edit interface........................................................117
8 Program test-run..................................................................................126
  8.1 Preparation: ................................................................................126
  8.2 Steps for program test-run: ............................................................126

Chapter 3 PLAY & REMOTE mode.........................................................128
  1 PLAY mode.......................................................................................129
    1.1 Preparation..................................................................................129
    1.2 Program call...............................................................................129
    1.3 Program execution.......................................................................129
  2 remote mode...................................................................................136
    2.1 Remote operation mode...............................................................136
    2.2 Reservation operation...............................................................140

Chapter 6 File operation.........................................................................146
  1 Save files into U disk.........................................................................147
    1.1 Save parameters into U disk.......................................................147
    1.2 Save users’ program to U disk....................................................148
    1.3 Save user PLC to U disk.............................................................148
    1.4 Save system PLC to U disk.........................................................149
    1.5 One key system backup..............................................................149
  2 Import from U disk............................................................................150
    2.1 Import parameters to system.......................................................150
    2.2 Import user program into system..............................................151
    2.3 Import users PLC into system....................................................153
    2.4 Import system PLC to the system..............................................153
    2.5 One key restore system.............................................................154
  3 Software upgrade.............................................................................155
    4 Save notes to................................................................................161
      4.1 Save IO input notes to U disk....................................................162
      4.2 Save IO output notes to U disk................................................163
      4.3 Save analog notes to U disk.....................................................163
      4.4 Save M relay notes to U disk....................................................164
      4.5 Save timer notes to U disk.......................................................164
      4.6 Save counter notes to U disk...................................................164
      4.7 Save GP parameters to U disk................................................164
4.8 Save GI parameters to U disk................................................................. 165
4.9 Save GD parameters to U disk................................................................. 165
5 Import notes from U disk................................................................................. 165
  5.1 Import IO input notes to system................................................................. 165
  5.2 Import IO output notes to system............................................................... 166
  5.3 Import analog notes to system................................................................. 166
  5.4 Import M relay notes to system................................................................. 166
  5.5 Import timer notes to system................................................................. 166
  5.6 Import counter notes to system............................................................... 166
  5.7 Import GP para notes to system.............................................................. 166
  5.8 Import GI para notes to system.............................................................. 166
  5.9 Import GD para notes to system.............................................................. 166
6 Uninstallation of U disk.................................................................................. 166
Chapter 1 Introduction to System
1 Introduction to system hardware

1.1 Basic information of system

CRP-S80 industrial robot is designed with the control system based on the international-popular open platform for software and hardware, with the specified multi-axes movement control card, data collector card, specified robot connection terminal, and security connector, which are developed independently. Program design is modelized, specified to different tool structure, applied industry, and functions. In structure, this system can control the vertical multi-joint tandem robot, vertical multi-joint parallelogram robot, vertical multi-joint L-shape wrist robot, vertical multi-joint spherical wrist robot, polar coordination robot, Delta robot, and so on. In functions, this system can control the robot to work in the areas, such as lifting, welding, painting, palletizing, cutting, polishing, and so on. Standard computer main wire can extend to on-site main wire, robot vision system, and so on.

Technology features:

- Optimized structure algorithm, which can adapt to various structure formation of structure control.
- Modularized functions settings, which can adapt to various applied scenarios.
- 8-axies control to realize the control of outside axis (walking axis, positioner).
- Adaptive to multi-pulse shape, Incremental, absolute servo.
- Open structure, synchronized internationally, easy to extend.
- Software PLC functions, easy for logical control.
- Process function, simplified program operation.
- Improved connecting method of computer main wires for reliability and practicability.
- Security module for safe production of robot.

1.2 System specification parameters

<table>
<thead>
<tr>
<th>Robot</th>
<th>CRP-S40</th>
<th>CRP-S80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of control axis</td>
<td>4 axis</td>
<td>6+2 axis</td>
</tr>
<tr>
<td>User storage space</td>
<td>200M</td>
<td></td>
</tr>
<tr>
<td>Teach</td>
<td>8-inch TET-LCD, full keyboard, mode-selection switch, security switch + emergency stop (touch screen is optional)</td>
<td></td>
</tr>
<tr>
<td>Ports</td>
<td>Digital I/O connector, 24 input/24 output (can be extended)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 output of analogous signal of 0-10V, 12 digitals.</td>
<td></td>
</tr>
<tr>
<td><strong>Security Module</strong></td>
<td>Connected emergency stop, abnormal signal of robot leads to quick stop of robot (optional)</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Controllable servo</strong></td>
<td>Incremental, absolute servo: domestic, Taida, Anchuan, Sanling, Sanyang, Songyang, and so on.</td>
<td></td>
</tr>
<tr>
<td><strong>Operation mode</strong></td>
<td>Teach, replay, remote</td>
<td></td>
</tr>
<tr>
<td><strong>Program mode</strong></td>
<td>Teach replay, process program</td>
<td></td>
</tr>
<tr>
<td><strong>Movement functions</strong></td>
<td>Point to point, linear, circle</td>
<td></td>
</tr>
<tr>
<td><strong>Instruction system</strong></td>
<td>Movement, logic, process, calculate</td>
<td></td>
</tr>
<tr>
<td><strong>Control mode</strong></td>
<td>Position</td>
<td></td>
</tr>
<tr>
<td><strong>Coordination system</strong></td>
<td>Joint Coordination, Right-angle coordination, user coordination, and tool coordination.</td>
<td></td>
</tr>
<tr>
<td><strong>Software PLC functions</strong></td>
<td>Edited using ladder diagram, 5000 steps</td>
<td></td>
</tr>
<tr>
<td><strong>Abnormal detection functions</strong></td>
<td>Emergency stop abnormal, servo abnormal, anti-collision, security maintenance, arc start abnormal, user coordination abnormal, tool coordination abnormal, etc.</td>
<td></td>
</tr>
<tr>
<td><strong>Structure algorithm</strong></td>
<td>vertical multi-joint tandem robot</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vertical multi-joint parallelogram robot</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vertical multi-joint L-shape wrist robot</td>
<td></td>
</tr>
<tr>
<td></td>
<td>polar coordination robot</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Horizontal joint robot (SCARA)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deltarobot</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Special structure robot</td>
<td></td>
</tr>
<tr>
<td><strong>Applications</strong></td>
<td>Lifting, welding, painting, palletizing, cutting, polishing, forging and stamping, pouring, etc.</td>
<td></td>
</tr>
<tr>
<td><strong>Zero point functions</strong></td>
<td>Absolute: Battery memory; Incremental: reset after opening</td>
<td></td>
</tr>
</tbody>
</table>

2 System exterior

2.1 Teach

Operation button is on the teach machine for teach and program.
The Front view of teach machine

- Switch of mode selection
- Emergency stop
- Electrical hand-wheel
- Power
- Alarm
- Display/Touch
- Buttons
- Teach box cable

The back view of teach machine

- Security switch
- Electrical hand-wheel
- Handrail
- Touch pen
- Teach machine cable
- USB (Spare)
2.1.1 Emergency stop button

rotate the button according the white arrow to remove the emergency stop.

Notice: The use of emergency stop should cooperate with circuit design, and they should be safe and reliable. Otherwise, robot may not be stopped in emergency, which can be a safe problem.

2.1.2 Switch of mode selection

Used to choose the operation mode of robot. In this system, there are three modes: TEACH, PLAY, REMOTE.

2.1.3 Security switch

In teach mode, when security switch is in the middle gear, the robot will be powered. If hold tightly or release the security switch, power will be cut off and electric motor will be in the brake state.

Description: There are three gears of security switch, outside gear and inside gear are off-power, middle gear means power connected.
Notice: When security switch is in middle gear, robot will have power to move. So nobody is allowed within the range of robot movement in case of accidents.

2.1.4 Electrical hand-wheel

Used as electrical scroll wheel, which is effective in Manu categories, parameters interface, variables interface, etc.

In teach mode, electrical hand-wheel can also be used to control slight movement of robot (note: only versions after V2.0 have this function).

2.1.5 Coordination buttons

Coordination buttons are used to manually control joint of robot in teach mode.

Coordination buttons are used to modify the operation speed and operation mode in replay mode.

Coordination buttons are used to switch the correspondent functions in non-axis movement interface.

2.1.6 State control buttons in hardware

State cancel button, cancel current state, return to previous interface.
Windows switch button, switch the cursor position in different windows among general display, monitor, and information prompt.

Stop button, stop the robot in replay mode.

Notice:
There will be a time period between pressing the button and robot total stop, for the reasons such as up/down speed of system, servo driver parameters, and mechanical structure tenacity.

After pressing the stop button to stop the robot, the robot stop the program movement, but it has power to move at any time. So at this time, no operator is allowed within the movement area of robot, otherwise accident may occur.

Operation button, used to test the current program segment selected by current cursor; used to start the program in replay mode.

Notice: After the press of start button, robot will move. Please make sure that robot is in normal condition and no people or obstacles are within the robot movement area before the press of this button. Otherwise, accident may occur.

2.1.7 State control buttons in software

Display and operation may differ in different software functions. Usually, control buttons are to control the power state and movement mode, and to switch the coordination systems.
2.1.8 Main menu bar

| File operation | Run | Parameter setting | Monitoring | Run prepare | Contact | Technical | PLC |

Main menu bar is used to invoke the correspondent menus. Menus include file operation, program edit, parameters setting, monitor, operation preparation, program instructions, user process, PLC.

2.1.9 Sub menu bar

Display and operation may differ in different software functions.

2.1.10 Confirm button

: To confirm certain operations, such as parameters modification, data instruction input, and so on.

2.1.14 Speed adjustment button

Notice: Speed adjustment button can only be found on v1.1 version keyboard.

To adjust the movement speed of robot, and result will show in the status bar. Speed50%

This button can only adjust the speed in teach mode. In replay and remote mode, user should use auto-operation speed multiplier and correspondent coordination button+/- to increase or decrease the speed. result will show in the status bar. Speed85%.
3 Display of Teach

3.1 Main interface domains

Screen of teach is a 8-inches colorful screen, which can display digital, character, and symbols.

Display interface consists three main display area (general display area, monitor are, and information area), in addition, main menu, operation 1, operation 2, status bar, sub menu are located around.

The three main areas can switch by . When certain area is selected, the background will change or cursor bar appears. When switching display areas, main menu and submenu will change accordingly.

Monitor area can be closed. When monitor area is shown, general display area will automatically shrink into half-screen display. After closing the monitor area, general display area will automatically enlarged into whole screen display.

Status control area, main menu area, coordination area, sub menu area can be operated by the correspondent button.

Specific display is shown as below, monitor area is selected.

3.2 General display area

General display area is used for program list, program edit, parameters setting, coordination setting, and process setting, and so on.
General display area will automatically shrink into half display; and extend to full display after monitor window is closed.

Interface example is as following:

Program edit and input monitor (half display)

Program list(full display)

Program edit (full display)
3.3 Monitor area

Monitor area is used to show: coordination data, time data, TV data, I/O data, PLC internal relay, timer, counter condition, main wire info, hardware info, software info, appointment status, program variables status, and so on. Monitor area can be closed. Monitor area is shown on the right side of general display area. When general output ports and PLC auxiliary relay
M960M799 are displayed in monitor area, status control area will show the mode-switch logo, which can be used to switch the Y** or M** status of current cursor position.

Details can refer to “Chapter 5 Monitor”.

Example onitor area interface is as follows:

![Monitor area interface](image)

3.4 Information area

Information area mainly shows the latest operation, movement operated by system, log info such as alert. Information area will record the latest information. If unknown problem occurs, users can use the up/down cursor button to check the info and find out the reason.

Information area is shown below:

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Run</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>996</td>
<td>11-04 11:08:18:886</td>
<td>1</td>
<td>抛球后运行键</td>
</tr>
<tr>
<td>996</td>
<td>11-04 11:09:04:220</td>
<td>1</td>
<td>In case moving after power off, please confirm...</td>
</tr>
<tr>
<td>997</td>
<td>11-04 11:00:04:286</td>
<td>1</td>
<td>Please back to zero, can not present and aut...</td>
</tr>
<tr>
<td>998</td>
<td>11-04 11:21:52:944</td>
<td>1</td>
<td>Open file at the moment, please wait</td>
</tr>
<tr>
<td>999</td>
<td>11-04 11:31:58:010</td>
<td>1</td>
<td>Open file success</td>
</tr>
</tbody>
</table>
3.5 Main menu area

Main menu area is divided into eight main menus: file, program, parameters, monitor, operation preparation, program instruction, user process, PLC. Menu structures are as follows:

3.5.1 File

File menu is used to update/backup program files, robot parameters, and PLC file. File menu also consists system software update. The drop-down menu is shown as below figure.

For detailed file operation, please refer to [-----].

<table>
<thead>
<tr>
<th>File operation</th>
<th>Edit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Save file to U disk</td>
<td></td>
</tr>
<tr>
<td>2 Import from U disk</td>
<td></td>
</tr>
<tr>
<td>3 Software upgrade</td>
<td></td>
</tr>
<tr>
<td>4 Save notes to U disk</td>
<td></td>
</tr>
<tr>
<td>5 Import note from U disk</td>
<td></td>
</tr>
<tr>
<td>7 Eject U disk</td>
<td></td>
</tr>
</tbody>
</table>

3.5.2 Program

Program menu is used to edit program line in program interface. For detailed description, please refer to [Program edit chapter].
3.5.3 Parameter setting:

Parameter setting is used to set the robot parameters, in order to control the robot. The drop-down menu is shown as below figure.

Detailed parameters description can be found in [parameter chapters] and “Commissioning manual of S80/40 control system”.
3.5.4 Operation preparation:

The operation preparation is used to set the zero point of robot coordination system and to set the variables. The drop-down menu is shown as below figure.

3.5.5 Monitor

Monitor menu is used to check the information related with robot and periphery ports. The drop-down menu is shown as below figure.
### 3.5.6 Program instructions

Structures info can be found in [Chap2 2.5 program instruction].

<table>
<thead>
<tr>
<th>Command</th>
<th>Technical</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Movement</td>
<td></td>
</tr>
<tr>
<td>2 Logic</td>
<td></td>
</tr>
<tr>
<td>3 Arithmetic</td>
<td></td>
</tr>
<tr>
<td>4 Palletizing</td>
<td></td>
</tr>
<tr>
<td>5 Welding</td>
<td></td>
</tr>
<tr>
<td>6 Assistant command</td>
<td></td>
</tr>
<tr>
<td>7 Visual</td>
<td></td>
</tr>
<tr>
<td>8 Follow</td>
<td></td>
</tr>
<tr>
<td>9 Correspondence</td>
<td></td>
</tr>
<tr>
<td>10 Special instruct</td>
<td></td>
</tr>
</tbody>
</table>
3.5.7 User process:

User process menu is used to set the specific process applied to robot. The drop-down menu is shown as below figure.

Detailed description of user process can be found in “Welding process”, “Palletizing process”, “Track process”, “vision process”.

Description: Among the palletizing, painting, welding process, only one process can be chosen, i.e. two processes cannot be applied at the same time. Painting process is the internal function of system, which is not available temporarily.

3.5.8 PLC:

PLC menu is used to observe the operation status of PLC in trapezoid diagram.

Detailed PLC descriptions can be found in “CRP-S40/S80 manual”.

Notice: PLC edit function in PLC menu is not prefect now, please don’t use this function.

3.6 Status control area

This area is used to switch and display: manual control, coordination system, speed multiplier, output ports, auxiliary relay, and so on.

Follows are list of frequently used logos. Logos may differ in different process. Details can be found in “Process manual”.
Manual control status:

Axis forbidden. Meaning: In teach mode, move robot through coordination button is forbidden.

Axis permit. Meaning: it’s allowed to control robot movement through coordination button on the right side of screen.

Robot coordination status:

Robot joint coordination status: showing the current joint coordination system.

Robot rectangular coordination system: showing the rectangular coordination system is used currently.

Robot tool coordination status: showing the tool coordination system is used currently.

Robot user coordination status: showing the user coordination system is used currently.

Auxiliary M16* relay switch:

Pressing the correspondent status control button can switch the auxiliary relay M16*, e.g. open M160. Pressing the correspondent status control button can switch the auxiliary relay M16*, e.g. close M160.

Servo electric machine status button:
Servo permit: red means servo electric machine is not allowed to work. Green means servo permit.

As for absolute servo drive, red means system will communicate among the different divers and collect the absolute position data.

"R" reset button. When system info area shows the prompt alert or notice, pressing this button can reset the prompt alert or notice.

In auxiliary monitor interface, it’s effective to select relay using cursor. In auxiliary monitor interface, it’s ineffective to select relay using cursor. In general output ports monitor interface, it’s ineffective to select relay using cursor. In general output ports monitor interface, it’s effective to select relay using cursor.

3.7 Coordination button area

Coordination button area is used to operate the correspondent movement by coordination button in teach mode.

Frequently used buttons are listed as below. Buttons may differ in different process. Details can be found in process manuals.

In joint coordination system, move the joint by pressing the correspondent +/- button.

In rectangular, tool, user coordination system, moves the robot along the coordination axis by pressing the correspondent +/- button

Manually adjust speed multiplier. Speed can be adjusted by pressing the correspondent +/- button or pressing the or button. Results are shown in the status bar Speed100%
Automatically speed multiplier. Speed can be adjusted by pressing the correspondent +/- button or pressing the button. Results are shown in the status bar.

Play/remote mode: Continuous cyclic operation. Program can be operated in a continuous cyclic way. After program is suspended, users can switch among continuous mode, single program mode, and single line mode by pressing the correspondent +/- button.

Play/remote mode: Single program operation. Program is operated from cursor position to the end of current program. After program is suspended, users can switch among continuous mode, single program mode, and single line mode by pressing the correspondent +/- button.

Play/remote mode: Single line operation. Single line is operated, then suspend, and operated next line after pressing starting button. After program is suspended, users can switch among continuous mode, single program mode, and single line mode by pressing the correspondent +/- button.

3.8 Status display area

Status display area is used to show the current working status: such as number panel condition, robot movement condition, work mode, operation speed, tool coordination number, work piece coordination number, current time.

Meaning: current working status of robot, including manual stop, manual operation, automatic stop, automatic operation, remote stop, remote operation.

Working mode, correspondent with working condition.

In teach, working conditions are manual stop, manual operation. In replay mode, working conditions are automatic stop, automatic operation. In remote mode, working conditions are remote stop, remote operation.
Meaning: current speed multiplier. In teach, it shows manual operation speed multiplier; in replay and remote mode, it shows automatic operation speed multiplier.

Meaning: current tool coordination number. In tool coordination setting interface, after selecting certain coordination number, number will show in this area and the setting coordination is invoked. Please make sure the tool coordination number is correct.

Meaning: current user coordination number. In user coordination setting interface, after selecting certain coordination number, number will show in this area and the setting coordination is invoked. Please make sure the user coordination number is correct.

Current time

Synergy condition: when showing SYN1 SYN2, it means the open condition of synergy axis.

3.9 Sub menu area

This area has different content according to the active interface. Pressing the buttons to operate in correspondent way.
Chapter 2 TEACH MODE
1 Available movement in Teach mode

★ Related parameters setting, equipment commissioning, and equipment maintenance. Reference: 【】 and 《》.
★ Coordination system setting, zero return operation. Reference: 【---】 and 《Commissioning manual》.
★ Code, modify the teach program. Reference: 【----】.
★ Monitor of working data. Reference: 【---】.
★ Peripheral USB disk operation. Reference 【---】.

2 Simple manual movement

Please obey the following rules within the robot movement region:

<table>
<thead>
<tr>
<th>Observe the robot from the front.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep strictly with operation steps.</td>
</tr>
<tr>
<td>Consider the backup solution if robot move towards the people suddenly.</td>
</tr>
<tr>
<td>Make sure to set the shelter in case of emergency.</td>
</tr>
<tr>
<td>Notice: People injury may result due to the wrong movement of robot.</td>
</tr>
</tbody>
</table>

Following conditions should be satisfied before robot movement:

■ Equipments have been commissioned properly, and they are in good condition.
■ Self-check has finished, and no error or alarm is found.

2.1 Correct gesture to use teach box

1. Grab the standing hole with the security switch on the back of teach box by left hand through the rope. Index finger is on the position of security switch. (Gesture is shown in below figure).
2. Raise the teach box, turn it with the front side up, and hold the box at the proper position on the belly. Operate the teach box button or switch using right hand. (Gesture is shown in below figure).
3. Stand beyond the movement region of robot on the front or side direction. Pay attention to observe the robot movement. Pay attention to the obstacles or peoples that enter the movement region. If any emergency occurs, press the emergency stop button to stop the robot.

2.2 Simple manual movement

1. Open power supply of the robot control box, power the robot.

2. Make sure no alarm info is shown in info bar.
3. Switch the mode into teach mode.

Press related button, power the servo. After the power, the logo turns into.

Notice:

When servo is powered, system will receive the feedback of programmer to calculate the current coordination. As for the absolute programmer servo institute, system can directly receive the feedback and calculate the coordination and system allows the replay and remote operation. As for incremental programmer servo institute, system cannot calculate using the feedback of programmer after servo is powered. So it’s necessary to operate the zero return in <operation preparation>-<zero point setting of lathe> manually. For detailed info, please refer to “Zero return” and “commission manual”.

4. Press & to manipulate the speed multiplier.

5. Choose the proper coordination system. Joint coordination system is suggested.

6. Press related button, cancel the forbidden movement robot, and logo should turn into, which means robot is allowed to move.

7. Press the +/-button to move every joint slowly.

3 Coordination system
3.1 Types of coordination system

When axis operation is required for robot, the following coordination systems are useful.

**Joint coordination system**

Robot can move along the axis independently in joint system. Joint coordination system is set after the robot commission and unchanged afterwards.
Since this system can support various robot types, please refer to commission manual to check the move direction of different joint coordination system.

**Rectangular (ground) coordination system**

Robot Rectangular system is also called ground coordination system. Axis direction and zero point position may differ of different rectangular system.

After the setting of related robot parameters, zero point and axis directions are determined in rectangular system. And rectangular system cannot be modified unless parameters are changed.

Robot can move along the preset x, y, z axis, no matter where the robot is. As for 6-axis robot, robot can also rotate along A, B, C, A axis rotates around the X axis, B axis rotates around the Y axis, C axis rotates around the Z axis. Rule is right-handed twisted rule.

**Tool coordination system**

In tool coordination system, z axis is defined along the effective direction of tool held in the flange plate of robot wrist, and zero point is defined at the sharp end point of tool.

0# tool coordination system is the basic tool coordination system, unchanged, unmodified and same as rectangular system. 1-49# tool coordination systems can be set by users according to the real situation. Detailed setting steps can be found in [---].
**User coordination system**

Robot can move along the specific user coordination system axis.

In the coordination systems except joint system, it’s allowed to change the tool gesture without changing the tool sharp point position (control point), which is called static control point movement.

0# user coordination system is the basic user coordination system, unchanged, unmodified and same as rectangular system. 1-49# user coordination systems can be set by users according to the real situation. Detailed setting steps can be found in [---].

3.2 Invoke and switch coordination system

Joint coordination system and rectangular coordination system can be switched by pressing the status control button, then using +/- button to move the robot in the current system, or programming to invoke the correspondent coordination system.

User coordination system and tool coordination system should satisfy two conditions: invoke proper coordination number, switch to desired coordination system. After the conditions are satisfied, desired system can be used by move the robot in manual or programmable way, in teach mode.
3.2.1 Invoke the coordination system:

Steps to invoke the tool coordination system: <Run prepare>-<Tools coordinate>-Tool number-<close>

In the above interface, using cursor button to move cursor into tool coordination number area, then using up/down button to select the desired number. Then press the submenu close. At this time, tool coordination invoke is finished.

Steps to invoke the user coordination system: <Run prepare>-<User coordinate>-<confirm>-User number-<close>. 
In the above interface, using cursor button to move cursor into user coordination number area, then using up/down button to select the desired number. Then press the submenu close. At this time, user coordination invoke is finished.

3.2.2 Switch the coordination system:

In teach mode, pressing correspondent coordination button in condition control area can switch the coordination system in a cyclic way: joint, rectangular, tool, user coordination system. Display button is the current used coordination system.

Following is the display logo and correspondent coordination system in the system switch

Robot joint coordination, meaning the current coordinationsystem is joint coordination system.

Robot Rectangular (ground) coordination, meaning the current coordination system is Rectangular (ground) coordination system.
Robot tool coordination, meaning the current coordination system is tool coordination system.

Robot user coordination, meaning the current coordination system is user coordination system.

In the Edit process in teach mode, selected coordination system will be transferred into the teach program. So please select and set the coordination system properly.

3.3 Set the coordination system

Setting of joint coordination system and rectangular system is finished after the robot commissioning, which cannot be changed afterwards.

Tool coordination system and user coordination system can be set by users. Related content can be found in “welding process”.

3.3.1 User coordination system setting

It’s convenient to program in teach mode after establish the user coordination system. As shown in below figure, where there is a tooling surface, there should be a user system.

Setting steps for user coordination system:

<Run prepare>-<User coordinate>. Set the user coordination system in the prompt interface (as shown in below). Every tool needs a user coordination system (also called tool coordination system).
Setting interface of user coordination system

As shown in above, select the user coordination system number, click the <Calibration>, and enter the user coordination setting, as shown in below.
In the above figure, firstly set the zero point of user (tool) coordination system (ORG value), i.e. move the robot sharp end (welding wire in the welding machine) to one terminal of an angle on the tool. Then click <record current point> button to record the zero point. At this time, the logo below the ORG value on the screen should become .

Select recoding point , switch to “XX direction” to determine the x edge, as shown in below.
In the above interface, set the x direction of user (tool) coordination system, i.e. move the terminal end point at the edge of tool. Then press <record current point> button to record the XX direction in user (tool) coordination system. At this time, logo below the XX direction should
In the above interface, set the Y direction of user (tool) coordination system, i.e. move the terminal end point at the other edge of tool. Then press <record current point> button to record the YY direction in user (tool) coordination system. At this time, logo below the YY
After determining the zero point (ORG), XX direction, YY direction (all logos become green), press <calculate> button. Then system will automatically calculate the current user (tool) coordination system, determining the coordination system and axies on the tool.

**Discription**

User coordination takes the right-handed spin rule (shown in below figure), i.e. orientation of Z axis is defined along the thumb direction when right hand is spin from x axis to y axis. Usually when tool coordination is setup, orientation of Z axis points away from the tool, so it should be considered which edge on the tool should be x axis and which edge should be y axis.
After the calculation of user coordination system, users can switch to the user coordination system and verify if the tool direction is set as desired.

After verification, press <cancel> to quit.

3.3.2 Tool coordination system setting

In order to control the robot for the correct linear/circle interpolation etc. movement, size information should be inputted correctly and position of control point should be defined. User coordination is determined by setting up 6 sets of terminal end of robots, then automatically calculating the position of tool control point, and inputting into the tool files.

The coordination of control point in flange plate coordination system can be inputted using the tool verification, as shown in below.
Tool size is based on the coordination of robot’s terminal (left figure); Relationship of tool coordination and terminal flange plate coordination (right figure)

When tool verification is operated, 6 different gesture should be taught based on the control point. Then tool size can be automatically calculated according to these 6 data. Selection of point is shown as below:

As shown above, gesture of P1-P4 should be changed as largely as possible. At P5 point, welding wire (the straight part of welding gun) must in a straight line with verification machine. P6 point is for determination of x axis of tool coordination system, which means line connected P5 and P6 is the x axis of tool coordination system.

Steps for tool coordination system setup are as following: <Run preparation>-><Tools coordinate>, then set the tool coordination system in setting interface.
Tool coordination system setup interface.
In the above figure, checks if “Tool number” is correct. Then select “record point”. Then select point, and move the welding machine sharp end (welding wire) to the correspondent position, press <Record point>, then logo below P1 will become green. Repeatedly, select point and move the welding machine sharp end (welding wire) to the correspondent position, press <Record point>, then logo below P2 will become green.

In a cyclic way, set the P3-P6, make sure the logo below the P1-P6 turns green, as shown in below figure.

**Description**

As shown above, gesture of P1-P4 should be changed as largely as possible. At P5 point, welding wire (the straight part of welding gun) must in a straight line with verification machine. P6 point is for determination of x axis of tool coordination system, which means line connected P5 and P6 is the x axis of tool coordination system.
Click `<calculate>` in above interface, system will automatically calculate the current tool coordination, determine the coordination system and orientation on the tool. Click `<cancel>`, size of tool sharp point in obtained according to the flange plate at the terminal of robot.
Tool size is determined according to the above figure.

Tool coordination system is shown in above figure, based on the Cartesian
After the calculation of coordination system, user can verify if the setup coordination has the desired orientation of tool axis by switching to user coordination. After the verification, press <close> to quit.

4 Zero return operation

Zero return operation: System reads the feedback of servo electric machine programmer after certain operations (e.g. institute operation to zero position switches, servo electric machine seek for zero position at low speed). Then system will calculate the current system coordination based on the preset relationships.

4.1 Absolute programmer servo institute:

After fixing the absolute servo electric machine onto the mechanical body, since the absolute programmer has absolute unique position, system can calculate the coordination value based feedback position. So coordination calculation can be finished at the time servo is powered and reading the programmer. In this way, extra zero return process is unnecessary.

4.2 Incremental programmer servo institute:

Since the incremental programmer cannot guarantee the absolute and unique position, system cannot calculate the coordination value based feedback position. So it’s necessary to operate the zero return to every incremental axies.

Following is the description of zero return process in incremental programmer servo institute.

Firstly, <Run prepare> - <Robot home>, as shown in below.
System has 5 ways to operate zero return process for incremental programmer servo institute. Specific operation may differ in different methods. Every method is described below:

Method 0: Detect switch

Meaning: At the current position find the zero switch at high speed (positive speed) according to the preset direction. After touching the switch, reverse the direction at low speed (negative speed), detouch the switch and find the z signal. Stop with one circle of electric machine, otherwise failure.

Please make sure that zero switch movement is normal, zero switch installation position is normal, and switch is positioned ahead of the current point according to the zero return direction. Move the cursor to the corresponding axis position, press <security switch>-><move to zero>-><confirm>. Correspondent axis will move quickly to touch the zero return switch, stop, reverse the direction, detouch with the zero switch, detect the programmer z signal in one circle starting from separation with zero switch. Then system will show zero return successful, indicating that light of zero return becomes green, and joint coordination system will be refreshed to the correspondent coordination value.
It's suggested to repeat the zero return operation to verify the accuracy of zero return operation. Feedback position fluctuations of electric machine on the driver should also be observed; usually fluctuation is the 2-3 units of instruction pulse (it's needed to shift the effective value of electric machine feedback according to the electric gear wheel).

If no Z signal is found within the one circle, system will control the electric machine to reverse the direction to the starting position of the previous circle, and seek for the z signal at the low speed. If the above process has been operated for three times and no z signal is found, the system will show zero return failure. If zero return failure is frequently occurs, users can lower the reverse speed.

**Method 1:** Seek for Z pulse without zero return switches

**Meaning:** Seek for the z signal from the current position according the preset direction at the low speed (negative speed). If signal can be found within one circle, zero return is finished. Otherwise, zero return fails.

It’s necessary to set a zero return range to guarantee the zero return consistency, i.e. the value of electric machine at certain position.

**Steps to determine the zero return range:**

Operate the zero return at positive direction (Move cursor to correspondent axis position, <security switch> -> <back to zero> -> <confirm>, and then z signal is found within one circle of electric machine. System shows zero return succeeds, indicating light of “zero condition” turns green and joint coordination is refreshed to the calculated coordination value).

After the zero return, draw a line at the static/move part of joint.

Set the speed at the lowest in teach mode, inch forwar the axis joint (slight movement).
Operate the zero return again. After success, draw a line at the static joint along the rotation line of joint.

In this way, there is one line on the move joint and two lines on the static joint (These two lines determine the one circle range of electric machine).

Move into this range before zero return, which can guarantee the consistency of zero return every time.

Description

1. The lines drew on the static joint should be observed easily. This is related with speeddown ratio and joint outside diameter. If speeddown ratio is large and joint diameter is small, the two lines will be close and easy to wrong operation.

2. To guarantee the two lines on the static joint is the one circle of electric machine, it’s necessary to operate the zero return in two lines. After the zero return, the lines on the static and move joint will be on one line.

3. If system cannot find the z signal within one circle, system will control the electric machine to reverse the speed, move back to the starting point, and seek for z signal at low speed. If this process has been operated three times without z signal, system will show zero return failed. At this time, it’s necessary to operate zero return instructions. If zero return failed frequently, set the “reverse speed” lower.

4. To test the consistency of zero return, it’s suggested to repeat the zero return operation to verify the success percentage and accuracy.

Method 2: Detection of detouch with switch at positive direction

Meaning: Starting according to the preset speed and direction to seek the zero switch, until detouch with the switch.

Method 3: Detection of detouch with switch at negative direction
Meaning: Starting according to the preset direction to seek the zero switch. If switch is touched, move at the preset negative speed until detouch with the switch, stop.

Before the operation, please make sure zero return reference normal, switch movement normal, and switch installation properly.

Move the cursor to the correspondent axis position, press <security press>-<zero return>-<confirm>, correspondent axis will touch the decelerate switch quickly and then stop after touching the switch. Then axis will rotate along the positive direction, and system will show zero return succeed once the switch is off-touched. At this time, zero position indicating light will turn green, and joint coordination will be refreshed by correspondent coordination value.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This return method isn’t accurate based on zero return speed and switch conditions.</td>
</tr>
</tbody>
</table>

Method 4: Direct record operation:

<Run preparation>-<Robot home>-<confirm>. In the prompt interface, press <parameters modification> to modify the “correspondent coordination” bar (Parameters modification switch should be open with the users’ factory access.) into angular value in zero position of different types of robots (The specific angular value can be referred to “Commissioning manual”).

In teach mode, move every joint to the zero position of the certain type of robot (refer to “commissioning manual”), press <record>, then the indicating light of zero position will turn green, and joint coordination will be refreshed by correspondent coordination value.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This method cannot guarantee the reproducibility, for the inconsistency is not guaranteed in every observation.</td>
</tr>
</tbody>
</table>
Method 5: Zero return when pressing the switch:

Before the operation, please make sure the switch movement of zero return reference point is normal and switch installation position is normal.

Move the cursor to the correspondent axis position, press <security press> to move the correspondent axis sensor switch direction. After the zero position indicating light will turn green, press <zero return>-<confirm> to detouch the axis. System will seek for programmer Z pluse within the one circle of electric machine. If system finds the z pulse, she will say the zero return succeed and, zero point position logo turn green, and joint coordination will be refreshed by correspondent coordination value.

If system cannot find the programmer z pulses, system will control the electric machine move backwards, and return to starting position and seek for the z signal slowly. If above three search isn’t work, system will advise the zero return failed. At this time, system will show zero return, and users should operate the zero return process.

Description

1. Please make sure the indicating light of "position zero" should become green, otherwise repeat of experiment is common.

2. If system cannot find the programmer z pulses, system will control the electric machine move backwards, and return to starting position and seek for the z signal slowly. If above three search isn’t work, system will show the zero return failed. At this time, system will show zero return, and users should operate the zero return process. Usually, if system continuously reminds you of zero return failed, users can set the lower negative speed.

5 Program instructions

5.1 Invoke of program instruction

Turn the mode security switch to teach mode.

Switch to the programlist interface, create or open a program.
Click the < Command > button in the status control bar, or click the < Movement > button in the functional button area to invoke the frequently used instructions. When using functional area buttons, repeat clicking the button will result into the different instructions under the menu. Click < Last inst > button to invoke the instructions used last time.

After the input of correct instruction parameters, click < Inst correct > button to input the instruction; click the < Inst exit > button to cancel the instructions input. These two operation should be carried out while pressing the security switch.

### 5.2 List of program instructions

<table>
<thead>
<tr>
<th>Type</th>
<th>Instructions in English</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Movement</td>
<td>MOVJ</td>
</tr>
<tr>
<td></td>
<td>MOVL</td>
</tr>
<tr>
<td></td>
<td>MOVC</td>
</tr>
<tr>
<td>2 Logic</td>
<td>DOUT</td>
</tr>
<tr>
<td></td>
<td>AOUT</td>
</tr>
<tr>
<td></td>
<td>WAIT</td>
</tr>
<tr>
<td></td>
<td>TIME</td>
</tr>
<tr>
<td></td>
<td>PAUSE</td>
</tr>
<tr>
<td></td>
<td>JUMP</td>
</tr>
<tr>
<td></td>
<td>CALL</td>
</tr>
<tr>
<td></td>
<td>;</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>RET</td>
</tr>
<tr>
<td></td>
<td>TDOUT</td>
</tr>
<tr>
<td>3 Calculation</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>MUL</td>
</tr>
<tr>
<td></td>
<td>DIV</td>
</tr>
<tr>
<td></td>
<td>INC</td>
</tr>
<tr>
<td></td>
<td>DEC</td>
</tr>
<tr>
<td>4 Lifting</td>
<td>1 Pallet</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>5 Welding</td>
<td>1 ARC START</td>
</tr>
<tr>
<td></td>
<td>2 ARC END</td>
</tr>
<tr>
<td></td>
<td>3 WEAVE</td>
</tr>
<tr>
<td></td>
<td>4 WEAVE END</td>
</tr>
<tr>
<td>6 Painting</td>
<td>1 GUN ON</td>
</tr>
<tr>
<td></td>
<td>2 GUN OFF</td>
</tr>
<tr>
<td></td>
<td>3 GAS ON</td>
</tr>
<tr>
<td></td>
<td>4 GAS OFF</td>
</tr>
<tr>
<td>7 Pouring</td>
<td>1 SPEED</td>
</tr>
<tr>
<td></td>
<td>2 IF</td>
</tr>
<tr>
<td></td>
<td>1 IF</td>
</tr>
<tr>
<td></td>
<td>2 ELSE IF</td>
</tr>
<tr>
<td></td>
<td>3 ELSE</td>
</tr>
<tr>
<td></td>
<td>4 END IF</td>
</tr>
<tr>
<td></td>
<td>3 FOR</td>
</tr>
<tr>
<td></td>
<td>4 WHILE</td>
</tr>
<tr>
<td></td>
<td>1 WHILE</td>
</tr>
<tr>
<td></td>
<td>2 END WHILE</td>
</tr>
<tr>
<td></td>
<td>5 SWITCH</td>
</tr>
<tr>
<td></td>
<td>1 SWITCH</td>
</tr>
<tr>
<td></td>
<td>2 CASE</td>
</tr>
<tr>
<td></td>
<td>4 DEFAULT</td>
</tr>
<tr>
<td></td>
<td>5 BREAK</td>
</tr>
<tr>
<td></td>
<td>6 END SWITCH</td>
</tr>
<tr>
<td>8 Auxiliary Instructions</td>
<td>6 PULSE TARANSFORM</td>
</tr>
<tr>
<td></td>
<td>7 PULSE TARANSFORM END</td>
</tr>
<tr>
<td></td>
<td>8 IMPUTED</td>
</tr>
<tr>
<td></td>
<td>9 ZERO</td>
</tr>
<tr>
<td></td>
<td>10 MOTION FINISH</td>
</tr>
<tr>
<td></td>
<td>11 CHANGE USE</td>
</tr>
<tr>
<td></td>
<td>12 FORBID AXIS</td>
</tr>
<tr>
<td></td>
<td>13 FORBID AXIS END</td>
</tr>
<tr>
<td></td>
<td>14 COUNT COORD</td>
</tr>
<tr>
<td></td>
<td>15 CHANGE USE</td>
</tr>
<tr>
<td>10 Vision</td>
<td>1 RUN VISION</td>
</tr>
<tr>
<td></td>
<td>2 GET VISON DATA</td>
</tr>
<tr>
<td></td>
<td>3 CLEAR VISON DATA</td>
</tr>
<tr>
<td></td>
<td>4 TRIGGER VISON</td>
</tr>
<tr>
<td></td>
<td>5 CHANGE VISON</td>
</tr>
<tr>
<td>11 Track</td>
<td>1 TARCK START</td>
</tr>
<tr>
<td></td>
<td>2 TARCK END</td>
</tr>
<tr>
<td></td>
<td>3 GET TRACK DATA</td>
</tr>
<tr>
<td></td>
<td>4 CLEAR TARCK DATA</td>
</tr>
<tr>
<td></td>
<td>5 RUN IO CUTIN</td>
</tr>
<tr>
<td>12 Communication</td>
<td>1 READ IO BCD</td>
</tr>
<tr>
<td></td>
<td>2 READ DATA TO</td>
</tr>
<tr>
<td>13 Special instructions</td>
<td>1 RESET</td>
</tr>
<tr>
<td></td>
<td>2 CHANG EPOINT</td>
</tr>
</tbody>
</table>

5.3 Introduction of program instructions

Description of instructions:

1. <> contains the selected or input content. [Blank] contains nothing, meaning don’t use.

3. Program instruction line should select or input the following types of content: instructions, judgement symbol, data, status, number.

Instructions: instructions need invoking, e.g. MOVJ, JUMP, GP, LI, etc.

Judgement symbol: Used to judge the condition relationship, e.g. ==, >, <, >=, <=.

Assignment: Set the certain port into the certain condition, e.g. Y0=ON, M24=OFF.

Data: Data can be divided into digital, integer, minus, and characters according to the different instructions.

Status: (ON=1, OFF=0). The status can be used as the number 0, 1.

Number: Number should be integer larger than 0 (0 included) according to the different instructions. The number range may also differ.

4. The additions may differ attached to the different instructions. Please pay attention to the additions.

5. Positions with downwards arrow symbol should be inputted by <up/down arrows> selecting rather than directly input. Otherwise errors may occur.
6. Please select the instructions according to the reality where many instructions are available in additional instructions position.

7. 【Shortcut: 8 5 1】Invoke the instructions using shortcut. After showing the program instruction list, instructions can be invoked by using number button, arrow button, and enter button. Examples are as following:

【Shortcut: 8 5 1】Press 8,5,1 to invoke SWITCH

【Shortcut: 2 1 E】Press 2,1, enter button to invoke RET

【Shortcut: 8 ↑ 1 E】Press 8, upward arrow, upward arrow, enter button to invoke FORBIDAXISEND.

Structure of instruction lines:

<table>
<thead>
<tr>
<th>Additional option formats and definition</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{VJ} = % ) : Joint movement speed</td>
<td>Units: 1%, Max: 100%. Actual speed = highest speed of the axies in parameters × VJ × automatic operation multiplier</td>
</tr>
<tr>
<td>( \text{VL} = \text{straight movement speed} ) : straight movement speed</td>
<td>Units: MM/S, Max: Highest straight movement speed in parameters. Actual speed = VL × automatic operation multiplier</td>
</tr>
<tr>
<td>Smoothness (&lt;\text{Smoothness}&gt;) :</td>
<td>Range 0-9. Basically, it’s the radian of transition, which determines if the transition is performed in rectangular or circle way. Suppose two straight lines need connecting, parameters should be set to determine the radian of the transition line. The below figure is the scheme of value selection.</td>
</tr>
</tbody>
</table>
UNITL <condition>:  
Condition judgement  

Parameters can be used to judge: $X<$variable #>, $M<$variable #>, $T<$variable #>, $C<$variable #>; The condition of user judgment is effective (ON=1), ineffective (OFF=0). E.g. UNTIL $X#$(0)==ON to judge X0 port is effective; UNTIL $M#$(3)==OFF to judge the M3 auxiliary relay is ineffective.

<table>
<thead>
<tr>
<th>X&lt;Variable &gt;: General input X port</th>
</tr>
</thead>
<tbody>
<tr>
<td>General input X port: this port is related to hardware physical port.</td>
</tr>
<tr>
<td>Port status can be monitored in the “Monitor” - “I/O port” - “general input port monitor”. variable # can be referred to “Hardware operation manual”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M&lt;variable #&gt;: Internal auxiliary M relay. Status variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal auxiliary M relay. Status variable (ON=1, OFF=0), range: 0-800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gl&lt;variable #&gt;: global I variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>range: 0-99, values should be integer, with positive or nagetive symbol. So when Program first invokes the same variable number, the value of the variable is same. When Program secondly invokes the variable number, the value of the variable will be covered.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Li&lt;variable #&gt;: local I variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>range: 0-1000, values should be integer, with positive or nagetive symbol. When different programs invoke the same variable number, the correspondent values are not same, and will not interface and independent.</td>
</tr>
</tbody>
</table>
This variable can only be shown in the variable list when the program is open.

| GD<variable #> : Global D variable | range: 0-99, values should be float, with three decimal places (.000) and positive or negative symbol.  
So when Program first invokes the same variable number, the value of the variable is same. When Program secondly invokes the variable number, the value of the variable will be covered.  
This variable can be monitored in “operation preparation” - “variables” - “global D variables” list. |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------|
| LD<variable #> : local D variable | range: 0-1000, float variable, values should be float, with three decimal places (.000) and positive or negative symbol.  
When different programs invoke the same variable number, the correspondent values are not same, and will not interface and independent.  
This variable can only be shown in the variable list when the program is open. |
| GP<variable #> <data number> : Global P variable | GlobaleP variable, combination of multi-data, used to record the joint gesture, coordination, etc., of lathe. Variable range: 0-999, number range: 0-11.  
Data number refers to the certain data in the GP variables combination. Data number is defined as:  
0: operate the whole data combination;  
1: operate the x axis data;  
2: operate the y axis data;  
3: operate the z axis data;  
4: operate the J1 axis data;  
5: operate the J2 axis data;  
6: operate the J3 axis data;  
7: operate the J4 axis data;  
8: operate the J5 axis data;  
9: operate the J6 axis data;  
10: operate the J7 axis data;  
11: operate the J8 axis data;  
So when Program first invokes the same variable number, the value of the variable is same. When Program secondly invokes the variable number, the value of the variable will be covered. So when using the previously used variable numer, please pay attention in case of wrong movement or danger.  
This variable can be monitored in “operation preparation” - “variables” - “global P variables” list. |
| LP<variable #> <data number> : Local P variable | Local P variable, combination of multi-data, used to record the joint gesture, coordination, etc., of lathe. Variable range: 0-1000, number range: 0-11.  
Data number refers to the certain data in the LP variables combination. |
Data number is defined as:
0: operate the whole data combination;
1: operate the x axis data;
2: operate the y axis data;
3: operate the z axis data;
4: operate the J1 axis data;
5: operate the J2 axis data;
6: operate the J3 axis data;
7: operate the J4 axis data;
8: operate the J5 axis data;
9: operate the J6 axis data;
10: operate the J7 axis data;
11: operate the J8 axis data;
When different programs invoke the same variable number, the correspondent values are not same, and will not interface and independent. For example, when one program invokes variable L90, and another program invokes variable L90, these two L90 variables are independent and will not interface. This variable can only be shown in the variable list when the program is open.

Y<variable #: general output port
General output port, which is related with hardware physical port.
Status variable (ON=1, OFF=0), range 0-79.
This variable can be monitored in “Monitor” - “IO port” - “general output monitor”. Variable number is referred to “Hardware manual”.

T<variable #: Timer
T<variable #> Internal timer number, status variable (ON=1, OFF=0), range 0-59. The status of this variable is ON or OFF. Status of timer can be monitored in “Monitor” - “PLC” - “monitor of timer”.

TC<variable #: value in the timer
T<variable #> internal timer data, number variable. This data can be monitored in “Monitor” - “PLC” - “monitor of timer”.

C<variable #: counter
C<variable #> internal counter number, status variable (ON=1, OFF=0), range 0-19.
This counter status can be monitored in “Monitor” - “PLC” - “monitor of counter”.

CC<variable #: value in the counter
C<variable number > internal counter data, number variable, range 0-19.
This data can be monitored in “Monitor” - “PLC” - “monitor of counter”.

GS<variable #: global character string variable
GS character string variable, range 0-99. This variable is mainly used to read the barcode and to edit data from external I/O port. This variable should be used with READ DATA TOor READ IO BCD instructions. E.g.:
READDATATO GS# (0) : read barcode to variable GS0
PAUSE IF GS#(0)=123456 : pause when GS0=123456
### User-defined Variable

User-defined numer variable, which is the specific data user can input. This variable can using +/- symbol, and can be a decimal.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==:</code></td>
<td>judge equal</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>judge greater than</td>
</tr>
<tr>
<td><code>&lt;:</code></td>
<td>judge less than</td>
</tr>
<tr>
<td><code>&gt;=:</code></td>
<td>judge equal or greater than</td>
</tr>
<tr>
<td><code>&lt;=:</code></td>
<td>judge equal or less than</td>
</tr>
<tr>
<td><code>=</code></td>
<td>Setting symbol</td>
</tr>
</tbody>
</table>

#### MOVE J (MOVJ)

Move to the teach position in joint interpolation way. Movement speed of each joint is axis speed × VJ × automatic multiplier.

**Additions**

- `[blank]`
- `GP<variable>`
- `LP<variable>`
- `VJ=<%>`
- `PL=<smoothness>`

**Position Data**

- Position data, blank is shown in this bar on screen.
- `GP variable`, `variable#`: 0-999
- LP variable, `variable#`: 0-999
- VJ speed percentage, percentage: 1%-100%
- PL smoothness: 0-9

**UNTIL Condition**

- Not available
- Condition. When condition is satisfied, the current line will be stopped, and the next will be executed. Detailed can be found in.

---

5.3.1 Movement instrument

Movement instructions include three instructions: MOVJ, MOVL, MOV C.
Addition description 1:

Example of UNTIL:

Execution condition. Then condition is satisfied, current program line will be stopped, and next line will be executed. Otherwise, execute current line before executing the next line. X, M, T, C can be used as condition.

This function could be used in digging, scooping aluminium, and plate loading.

<table>
<thead>
<tr>
<th>position</th>
<th>line</th>
<th>program</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1</td>
<td>MOVJ VJ=30% PL=0</td>
</tr>
<tr>
<td>P2</td>
<td>2</td>
<td>MOVL VL=500MM/S PL=0</td>
</tr>
<tr>
<td>P3</td>
<td>3</td>
<td>MOVL VL=200MM/S PL=0 UNTIL X#(1)==ON</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>JUMP *22 IF X#(1)==ON</td>
</tr>
<tr>
<td>P4</td>
<td>5</td>
<td>MOVL VL=200MM/S PL=0</td>
</tr>
</tbody>
</table>

Example MOVJ VJ=30% PL=3
MOVJ VJ=30 GP#0 PL=3 UNTIL M#(0)==ON
<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>*22</th>
</tr>
</thead>
<tbody>
<tr>
<td>P5</td>
<td>7</td>
<td>MOVJ VJ=30% PL=0</td>
</tr>
</tbody>
</table>

**Notice:**

After X1 is effective, system will jump out of the current 3# line, and execute the 7# line.

The effective time of X1 should be long, otherwise system will execute 5 lines.

<table>
<thead>
<tr>
<th>2Straight move (MOVL)</th>
<th>functions</th>
<th>Move to the teach position in a straight interpolation way. Each joint will move in straight interpolation way with the speed of VL×multiplier.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[blank]</td>
<td>Position data, blank in this bar on the screen.</td>
</tr>
<tr>
<td></td>
<td>GP&lt;variable&gt;</td>
<td>GP variable, variable #: 0-999</td>
</tr>
<tr>
<td></td>
<td>LP&lt;variable&gt;</td>
<td>LP variable, variable#: 0-999</td>
</tr>
<tr>
<td></td>
<td>VL=&lt;straight move speed&gt;</td>
<td>Straight move speed, unitMM/S, max parameter of straight move highest speed.</td>
</tr>
<tr>
<td></td>
<td>PL=&lt;smoothness&gt;</td>
<td>PLsmoothness: 0-9</td>
</tr>
<tr>
<td></td>
<td>[blank]</td>
<td>Not available</td>
</tr>
<tr>
<td></td>
<td>UNTIL&lt;condition&gt;</td>
<td>UNTIL: When condition is satisfied, stop executing the current line and move to next line. Details can be found in addition description 1 of MOVJ.</td>
</tr>
<tr>
<td></td>
<td>DOUT&lt;addition&gt;</td>
<td>DOUT: output signal in movement. Details can be found in addition description 2 of MOVL.</td>
</tr>
<tr>
<td></td>
<td>AOUT&lt;addition&gt;</td>
<td>AOUT: output analogous index in movement. Details can be found in addition description 3 of MOVL.</td>
</tr>
<tr>
<td></td>
<td>[blank]</td>
<td>Not available</td>
</tr>
<tr>
<td></td>
<td>COORD</td>
<td>Added axis 1 and 2 is cooperated simultaneously</td>
</tr>
<tr>
<td></td>
<td>COORD1</td>
<td>Added axis 1 cooperated independently</td>
</tr>
<tr>
<td></td>
<td>COORD2</td>
<td>Added axis 2 cooperated independently</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Detailed refer to “chap9 1 cooperation manual”</td>
</tr>
<tr>
<td></td>
<td>example</td>
<td>MOVL VL=500MM/S PL=0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOVL VL=500MM/S GP#3 PL=5 UNTIL X#(0)==ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOVL VL=500MM/S PL=0 DOUT Y#(0)==ON START 10.0 END 10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MOVL VL=500MM/S PL=0 AOUT A1=2.0 A2=2.0</td>
</tr>
</tbody>
</table>
Addition description 2: DOUT example:

This addition is used to set a distance where the program should switch certain signal (M, Y) effective or ineffective, and to set a distance where restore that signal.

Example: MOV L V L=500MM/S P L=0 DOUT Y#= ON START 10.0 END 10.0 Description:
After executing this program, 10MM later, Y0 port change into open (ON). Robot keep moving to the point 10MM away from the end, and Y0 change to close (OFF). Then program come to the end.

This function can be used to open/close the gun in painting operation.

Addition description 3: AOUT example:

This addition is used to output the preset analogous index in the movement or at the end point.

1) V I analogous output:

VI analogous index output can only be used among the welding instructions. If V1 is not among the welding instructions, when this line is executed, system will alert and prompt: no welding info, AOUT is not allowed to change the welding current and voltage during the movement.

V is correspondent to welding voltage, while I is correspondent to welding current. When this instruction is executed from the beginning to the end, analogous voltage will change linearly from welding voltage to setting voltage.

Example:

Welding current 200A~analogous index 4.95V, welding voltage 20V~ analogous index 4.95V; Welding current 400A~analogous index 9.95V, welding voltage 40V~analogous index 9.95V.

ARCSTART# (0)

MOV L V L=500MM/S P L=0 AOUT V=40.0 I=400.0A
ARCEND# (0)

Description:

When 1st line is executed, A1 and A2 will output the arc start voltage. When 1st line is finished, A1 and A2 analogous index output is 4.95V (Welding current and voltage). When 2nd line is finished, A1 and A2 analogous index output change linearly from 4.95V to 9.95V. When 3rd line is executed, voltage become arc ending voltage.

This function can be used in the situation where welding current and voltage need changing.

A1 A2 analogous output:

When this instruction is between the welding instructions, it’s ineffective, and system will output according with the welding process settings.

When this instruction isn’t between the welding instructions, system will output the A1 and A2 analogous indet at the end of this instruction line. This is called point output for short.

A3 A4 analogous output:

No matter if this instruction is between the welding instructions or not, when system execute this instruction, robot will operate from the beginning to end, A3 and A4 analogous index will change linearly from strating point to the end point. This is called linear output for short.

Example:

MOVJ VJ=30% PL=0 ; Point A A3=0 A4=0
MOVL VL=500MM/S PL=0 AOUT A3=10.0 A4=10.0 ; Point B

Description: Robot moves from point A to point B, and A3 and A4 analogous index change linearly from 0V to 10V.

<table>
<thead>
<tr>
<th>3.Circle move (MOVC)</th>
<th>functions</th>
<th>Position data, blank in the bar on the screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>additions</td>
<td>[blank]</td>
<td>GP&lt;variable&gt;, variable# : 0-999</td>
</tr>
<tr>
<td></td>
<td>GP&lt;variable&gt;, variable# : 0-999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LP&lt;variable&gt;</td>
<td>LPvariable, variable#: 0-999</td>
</tr>
<tr>
<td>Description</td>
<td>Details</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>VL= &lt;straight move speed&gt;</td>
<td>Straight move speed, units MM/S, max is parameters of highest straight move speed.</td>
<td></td>
</tr>
<tr>
<td>PL= &lt;smoothness&gt;</td>
<td>PL smoothness: 0-9</td>
<td></td>
</tr>
<tr>
<td>&lt;variable&gt;POINT</td>
<td>Range: 1-3, One circular track must be realized by three circular instructions. Three circular instructions define the arc starting point, middle point, and end point. 1POINT is the starting point of arc, 2POINT is the middle point of arc, 3POINT is the end point of arc.</td>
<td></td>
</tr>
<tr>
<td>[blank] UNTIL&lt;condition&gt;</td>
<td>Not available</td>
<td></td>
</tr>
<tr>
<td>DOUT&lt;addition&gt;</td>
<td>UNTIL: When condition is satisfied, system will stop executing the current line and move to the next line. Details can be found in the MOVJ addition description 1. DOUT: output signal during movement. Details can be found in MOVL additions description 2.</td>
<td></td>
</tr>
<tr>
<td>AOUT&lt;addition&gt;</td>
<td>AOUT: output analogous index during movement. Details can be found in MOVL additions description 3.</td>
<td></td>
</tr>
<tr>
<td>[blank] COORD</td>
<td>Not available</td>
<td></td>
</tr>
<tr>
<td>COORD1</td>
<td>Added axis 1 and 2 is cooperated simultaneously</td>
<td></td>
</tr>
<tr>
<td>COORD2</td>
<td>Added axis 1 cooperated independently</td>
<td></td>
</tr>
<tr>
<td>example</td>
<td>Added axis 2 cooperated independently</td>
<td></td>
</tr>
<tr>
<td>[blank] COORD</td>
<td>Detailed refer to “chap9 1 cooperation manual”</td>
<td></td>
</tr>
</tbody>
</table>
Integrated example of circular program without gesture:

**Description**

In order to move smoothly, please set the PL of every point (including starting point and every point on the arc) as 9.

Whole circle consists of ABC arc and CDA arc. Terminal of robot is located at the starting point. Program as follows:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV L VL=500MM/S PL=9</td>
<td>Move from starting point to point A, taking A as the first point on the arc.</td>
</tr>
<tr>
<td>MOV C VL=500MM/S PL=9 POINT=2</td>
<td>Move from A to B, taking B as the second point on the arc.</td>
</tr>
<tr>
<td>MOV C VL=500MM/S PL=9 POINT=3</td>
<td>Move from B to C, taking C as the third point on the arc and as the first point on the arc CDA.</td>
</tr>
<tr>
<td>MOV C VL=500MM/S PL=9 POINT=2</td>
<td>Move from C to D, taking D as the second point on the arc.</td>
</tr>
<tr>
<td>MOV C VL=500MM/S PL=9 POINT=3</td>
<td>Move from D to A, taking A as the third point on the arc.</td>
</tr>
</tbody>
</table>

Whole circle with gesture consists of three arcs.

6 points is needed (2 3 4 5 6 7) as follows:
5.3.2 Logic Instructions

| Functions | Control the status of variables. Only two types of index exists, i.e. “ON (effective)” and “OFF (ineffective)”.
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Digital quantity output (DOUT)</td>
<td></td>
</tr>
<tr>
<td>Additions</td>
<td>Y&lt;variable&gt; = M&lt;variable&gt; =</td>
</tr>
<tr>
<td></td>
<td>Control the output port Y&lt;variable&gt; in ON or OFF status. Range of variables numbers should refer to “hardware manual” or “PLC manual”. Control the auxiliary relay M&lt;variable&gt; in ON or OFF status. Range of variables numbers should refer to “PLC manual”.</td>
</tr>
</tbody>
</table>
| example | DOUT Y#(0)=ON Control output port Y0 into ON status  
DOUT M#(0)=OFF Control the auxiliary relay M0 into OFF status |
<table>
<thead>
<tr>
<th>2 Analogous output (AOUT)</th>
<th>functions</th>
<th>Output the analogous voltage of analogous port.</th>
</tr>
</thead>
<tbody>
<tr>
<td>additions</td>
<td>AO#&lt;variable&gt; =</td>
<td>Set the analogous output port, range from A1-A2 in S40 and A1-A4 in S80.</td>
</tr>
<tr>
<td></td>
<td>&lt;variable&gt;</td>
<td>Variable is the output analogous voltage, range 0.000-10.000V</td>
</tr>
<tr>
<td>example</td>
<td>AOUT AO#(1)=0.000</td>
<td>output analogous 0V voltage in A1 port.</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(2)=10.000</td>
<td>output analogous 10V voltage in A2 port.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3 conditional wait (WAIT)</th>
<th>functions</th>
<th>When condition is satisfied, program will execute the following code. When condition is not satisfied, program will wait until condition is satisfied. However, there is a waiting time setting, when the condition is not satisfied but waiting time reaches the set time, program will move on.</th>
</tr>
</thead>
<tbody>
<tr>
<td>additions</td>
<td>X&lt;variable&gt; == ON</td>
<td>Judge if the input port X&lt;variable&gt; is ON or OFF. Range of variable number should refer to “hardware manual”.</td>
</tr>
<tr>
<td></td>
<td>M&lt;variable#&gt; == OFF</td>
<td>Judge of the auxiliary relay M&lt;variable#&gt; is ON or OFF. Range of variable number should refer to “PLC manual”.</td>
</tr>
<tr>
<td></td>
<td>DT= &lt;variable&gt;</td>
<td>Instruction waiting time, units 1MS. DT = 0 means waiting the condition forever. DT ≠0 means whether condition is satisfied or time is reached, program will move on.</td>
</tr>
<tr>
<td></td>
<td>CT= &lt;variable&gt;</td>
<td>Signal last time, units 1MS. CT = 0 means once condition is satisfied, program will move on. CT ≠0 means condition need to last for a set time, then condition is regard as satisfied and program will move on.</td>
</tr>
<tr>
<td>example</td>
<td>WAIT X#(0)==ON T=0</td>
<td>Waiting X0 into On status forever.</td>
</tr>
<tr>
<td></td>
<td>WAIT M#(1)==OFF T=500</td>
<td>Waiting M1 relay into ON status in 500MS, otherwise, program will move on.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4 Delay (TIME)</th>
<th>functions</th>
<th>Waiting for setting time (relay time), then program can move on.</th>
</tr>
</thead>
<tbody>
<tr>
<td>additions</td>
<td>T= &lt;variable&gt;</td>
<td>Set the waiting (relay) time, units 1MS, range: 1-9999999MS</td>
</tr>
<tr>
<td>example</td>
<td>TIME T=50</td>
<td>waiting for 50MS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5 pause (PAUSE)</th>
<th>functions</th>
<th>Pause without condition: pause until pressing “operation” button. Pause with condition: when following condition is satisfied, pause, or operate continuously.</th>
</tr>
</thead>
</table>
### additions

<table>
<thead>
<tr>
<th>X&lt;variable#&gt;</th>
<th>ON</th>
<th>Pause without condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>M&lt;variable#&gt;</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>Y&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GI&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LI&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GD&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GP&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LP&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GS&lt;variable#&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;character string&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### example

<table>
<thead>
<tr>
<th>PAUSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAUSE IF X#(0)==ON</td>
</tr>
<tr>
<td>PAUSE IF GD#(1)==LD#(1)</td>
</tr>
</tbody>
</table>

### 6 condition jump (JUMP)

#### functions

Jump with condition and without condition.

**description**: 1. When using this instruction, star symbol (*) should be used to lable which line program need to jump to.

Jump without condition: once program is executed here, jump to the labeled position.

Jump with condition: when this line is executed, if condition is satisfied, program will jump to the labeled line. If condition is not satisfied, program will executed in order.

<table>
<thead>
<tr>
<th>*&lt;variable&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump to symbol: jump to the position labeled with *&lt;variable&gt;; variable can be any character or number.</td>
</tr>
</tbody>
</table>

### addition

<table>
<thead>
<tr>
<th>[blank]</th>
<th>jump without condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF X&lt;variable#&gt;</td>
<td>== ON</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>CALL</td>
<td>Call the sub program, including call with condition and call without condition.</td>
</tr>
<tr>
<td></td>
<td>descriptions: 1. The difference of main program and sub program is that sub program is ended with RET.</td>
</tr>
<tr>
<td></td>
<td>% is the subprogram users want to call. If no condition attached, program will call the sub program at this line. If condition is attached with call instruction, program will judge if the condition is satisfied. If satisfied, then call the subprogram.</td>
</tr>
<tr>
<td></td>
<td>When using call without condition, there are fixed internal</td>
</tr>
</tbody>
</table>

### Example

- 1 * 345T : label *345T position
- 6 JUMP *SES IF M#(1) == ON : if M1 = ON, jump to *SES position
- 12 JUMP *345T : Jump to *345T position without condition
- 13 *SES : label *SES position
- ..... : Following lines

### Function List

- M< variable#> > OFF condition: If condition is satisfied, jump to the position labeled with *<variable>. If condition is not satisfied, program will be executed in order.
- Y< variable#> <
- T< variable#> >
- C< variable#> =
- GI< variable#> <
- LI< variable#> =
- GD< variable#> <
- LD< variable#> <
- GP< variable#> <data#>
- LP< variable#> <data#>
- TC< variable#>
- CC< variable#>
- GS< variable#> <character string>
Subprograms inside the robot to control the sliding table and spray gun (e.g. auto rotation 90, quick strat, etc.).

<table>
<thead>
<tr>
<th>%&lt;program name&gt;</th>
<th>name of sub program users want to call</th>
</tr>
</thead>
<tbody>
<tr>
<td>[blank]</td>
<td>Call without condition. Call&lt;program name &gt;sub program when this line is executed.</td>
</tr>
</tbody>
</table>

**Additions**

**IF**

<table>
<thead>
<tr>
<th>X&lt;variable#&gt;</th>
<th>ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>M&lt; variable#&gt;</td>
<td>OFF</td>
</tr>
<tr>
<td>Y&lt; variable#&gt;</td>
<td>=</td>
</tr>
<tr>
<td>T&lt; variable#&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>C&lt; variable#&gt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>GI&lt;variable#&gt;</td>
<td>=</td>
</tr>
<tr>
<td>LI&lt;variable#&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>GD&lt;variable#&gt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>LD&lt;variable#&gt;</td>
<td>=</td>
</tr>
<tr>
<td>GP&lt;variable#&gt; &lt;data#&gt;</td>
<td></td>
</tr>
<tr>
<td>LP&lt;variable#&gt; &lt;data#&gt;</td>
<td></td>
</tr>
<tr>
<td>TC&lt; variable#&gt;</td>
<td></td>
</tr>
<tr>
<td>CC&lt; variable#&gt;</td>
<td></td>
</tr>
<tr>
<td>GS&lt; variable#&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

- CALL 125 call the subprogram with name of 125
- CALL %SBS IF LD#(1) > LD#(2) If LD1>LD2, call SBS sub program.

**8 Notes (; ;)**

- Note for explain. When program comes to this line, notes will not be executed. Notes are used to help users to understand the program.

**Addition**

| ; <notes> | Users can write the notes to check and understand the program. |

**Example**

| ; tqzl | notes are tqzl, which will not be executed. |

**9 Jump Symbol**

- Label the jump position. It should cooperate with jump instruction.

**Addition**

| *<variable> | Variable can be any character or number. |
### 5.3.3 Calculation instructions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
</table>
| **Add** | Add the former variable with later variable, and the result will be set to the former variable. E.g.: A=A+B. This instruction can only be used with number variables. | ADD TC#(4) GP#1(1) : TC4=TC4+GP1 x axis coordination  
ADD CC#(1) 20.5 : CC1=CC1+20.5 |

**Example:** ADD TC#(4) GP#1(1) : TC4=TC4+GP1 x axis coordination  
ADD CC#(1) 20.5 : CC1=CC1+20.5
<table>
<thead>
<tr>
<th>Function</th>
<th>Addition</th>
<th>Subtraction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SUB</strong></td>
<td>GL&lt; variable#&gt;</td>
<td>GL&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>LI&lt; variable#&gt;</td>
<td>LI&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>GD&lt; variable#&gt;</td>
<td>GD&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>LD&lt; variable#&gt;</td>
<td>LD&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>GP&lt; variable#&gt;</td>
<td>GP&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;data#&gt;</td>
<td>&lt;data#&gt;</td>
</tr>
<tr>
<td></td>
<td>LP&lt; variable#&gt;</td>
<td>LP&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>TC&lt; variable#&gt;</td>
<td>TC&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>CC&lt; variable#&gt;</td>
<td>CC&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>D&lt; variable&gt;</td>
<td>D&lt; variable&gt;</td>
</tr>
<tr>
<td><strong>Multiplication (MUL)</strong></td>
<td>Multiply the former variable with the latter one and the result will be set to the former one. E.g.: A=A×B. This instruction can only be used with number variables.</td>
<td></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>SUB TC#(4) GP#1(1) : TC4=TC4-GP1 x axis coordination</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SUB CC#(1) 20.5 : CC1=CC1-20.5</td>
<td></td>
</tr>
<tr>
<td><strong>Division (DIV)</strong></td>
<td>Devide the former variable with the latter one and the result will be set to the former one. E.g.: A=A÷B. This instruction can only be used with number variables.</td>
<td></td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>MUL TC#(4) GP#1(1) : TC4=TC4×GP1 x axis coordination</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MUL CC#(1) 20.5 : CC1=CC1×20.5</td>
<td></td>
</tr>
<tr>
<td>【shortcut：3 4】</td>
<td>additions</td>
<td>functions</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>GI&lt; variable#&gt;</td>
<td>LI&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>GD&lt; variable#&gt;</td>
<td>LD&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>LP&lt; variable#&gt;</td>
<td>TC&lt; variable#&gt;</td>
</tr>
<tr>
<td>Devide the former variable with the latter one and the result will be set to the former one.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>example</td>
<td>DIV TC#(4) GP#1(1) : TC4 = TC4 ÷ GP1 x axis</td>
<td>DIV CC#(1) 20.5 : CC1 = CC1 ÷ 20.5</td>
</tr>
<tr>
<td>functions</td>
<td>Add 1 to the variable, and the result will be set to the variable. E.g.: A = A + 1. This instruction can only be used with number variables, and usually it’s used to count.</td>
<td></td>
</tr>
<tr>
<td>additions</td>
<td>GI&lt; variable#&gt;</td>
<td>LI&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>GD&lt; variable#&gt;</td>
<td>LD&lt; variable#&gt;</td>
</tr>
<tr>
<td></td>
<td>LP&lt; variable#&gt;</td>
<td>TC&lt; variable#&gt;</td>
</tr>
<tr>
<td>Add 1 to the variable, and the result will be set to the variable.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>example</td>
<td>INC TC#(4) : TC4 = TC4 + 1</td>
<td>INC CC#(1) : CC1 = CC1 + 1</td>
</tr>
<tr>
<td>functions</td>
<td>Subtract 1 from the variable and the result will be set to the variable. E.g.: A = A - 1. This instruction can only be used with number variables, and usually it’s used to count.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>incresement</td>
<td>+1 (INC)</td>
</tr>
</tbody>
</table>
### 5.3.4 Pallet instruction

#### 1 Pallet (PALLET)【shortcut: 4E】

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operate pallet process. When this instruction is operated, program will call the preset palleting process. Variablenumber is the process number, range 0-9. Detailed palletizing infos can be found in “palletizing process manual”.</td>
<td>PALLET#(1) : Call palleting process 1#</td>
<td></td>
</tr>
</tbody>
</table>
### 5.3.5 Welding instructions

1. **Start arc (ARC START)**【shortcut: 5】
   - **Function**: When this instruction is executed, system will call the preset welding parameters and start the arc. Variable number is the welding document number, range 0-7. The ARC START and ARC END are used in pair. Speed of program line will not be controlled by the auto multiplier. Only MOVL and MOVC can be used between ARC START and ARC END. Detailed welding infos can be found in “Welding process manual”.
   - **Addition**: `<variable#>` Variable number is the welding parameter document number that user wants to call.
   - **Example**: ARCSTART#(1) 8 MM/S V=20V I=200A ...... ARCEND#(1) Call welding parameter document 1#, start arc, welding speed 8 MM/S, welding voltage is 20V, welding current is 200A. starting welding Arc end, and welding finish.

2. **End arc (ARC END)**【快： 5 2】
   - **Function**: When this instruction is executed, system will call the preset welding parameters and end the arc. Variable number is the welding document number, range 0-7. The ARC START and ARC END are used in pair. Detailed welding infos can be found in “Welding process manual”.
   - **Addition**: `<Variable#>` Variable number is the welding parameter document number that user wants to call.
### Integrated welding example:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCSTART#(1) 10MM/s V=20V I=200A</td>
<td>Call the welding parameter document, speed 10MM/S, welding voltage 20V, welding current 200A</td>
</tr>
<tr>
<td>WEAVESINE#(1)</td>
<td>Call the welding weave parameters</td>
</tr>
<tr>
<td>MOV L VL=100MM/S PL=0</td>
<td>Move along the welding track, speed is 100MM/S according to the ARCSTART</td>
</tr>
<tr>
<td>WEAVEEND</td>
<td>End weave</td>
</tr>
<tr>
<td>ARCEND#(1)</td>
<td>End the arc according to 1# welding process, end.</td>
</tr>
</tbody>
</table>

5.3.6 Painting Instruction (Backup Fuction)

5.3.7 Pouring Instruction (Backup Fuction)

5.3.8 Auxiliary Function

**1 Speed change (SPEED)**

This instruction is used to change the speed after this instruction by multiply the additional percentage with the program line speed.

- **VJ<percentage>** is effective to the VJ speed after this line.
- **VL<percentage>** is effective to the VL speed after this line.
- **SPEED VJ=100 or SPEED VL=100** can cancel the speed change function.

This instruction is mainly used to change the speed of certain part in the program. In replay mode, the operation multiplier is effective to the whole program.
### Addition

<table>
<thead>
<tr>
<th>VJ = &lt;percentage&gt;</th>
<th>VL = &lt;percentage&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range: 1-100, effective to the VJ speed afterwards;</td>
<td>Range: 1-100, effective to the VL speed afterwards;</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>SPEED VL = 30</th>
<th>MOVL VL = 500MM/S</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL = 0</td>
<td>SPEDE VL = 100</td>
</tr>
<tr>
<td>MOVL VL = 500MM/S</td>
<td>PL = 0</td>
</tr>
</tbody>
</table>

VL speed afterwards will multiply 30%. VL speed change to: 500 * 30% = 150MM/S.

VL speed afterwards will multiply 100%. VL speed change to: 500 * 100% = 500MM/S. (Equal to no speed change or cancel of speed change.)

### Function

**IF condition judgement instruction.** The whole instruction should consist of four parts: IF, ELSEIF (can be omitted or be used repeatedly), ELSE, ENDIF.

Firstly, judge if the condition is satisfied, if satisfied, the instruction after the IF will be executed; If not satisfied, judge the condition after ELSEIF (ELSEIF can be used according the situation or not used). If condition after ELSEIF is satisfied, the instructions after ELSEIF will be executed. If not satisfied, the instruction after ELSE will be executed. At the end of the IF condition judgement, use ENDIF to label the end.

<table>
<thead>
<tr>
<th>X&lt;Variable#&gt;</th>
<th>M&lt;Variable#&gt;</th>
<th>Y&lt;Variable#&gt;</th>
<th>T&lt;Variable#&gt;</th>
<th>C&lt;Variable#&gt;</th>
</tr>
</thead>
</table>
| GI<Variable#>| LI<Variable#>| GD<Variable#>| LD<Variable#>| GP<Variable#>| GP<Variable#>| LP<Variable#>| TC<Variable#>| CC<Variable#>| GS<Variable#>
|              |              |              |              |              |              |              |              |              |
| ON=OFF        |              |              |              |              |              |              |              |              |

### Addition

<table>
<thead>
<tr>
<th>GI&lt;Variable#&gt;</th>
<th>LI&lt;Variable#&gt;</th>
<th>GD&lt;Variable#&gt;</th>
<th>LD&lt;Variable#&gt;</th>
<th>GP&lt;Variable#&gt;</th>
<th>GP&lt;Variable#&gt;</th>
<th>LP&lt;Variable#&gt;</th>
<th>TC&lt;Variable#&gt;</th>
<th>CC&lt;Variable#&gt;</th>
<th>GS&lt;Variable#&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>&gt;</td>
<td>&lt;</td>
<td>=</td>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**IF condition judgement instruction.** If condition is satisfied, the instructions after the IF will be executed. If IF condition is not satisfied, judge the ELSEIF condition. If ELSEIF condition is satisfied, the instructions after the ELSEIF will be executed. If IF condition and ELSEIF
<table>
<thead>
<tr>
<th>Condition(s) are not satisfied, the instructions after the ELSE will be executed. ENDIF to end the IF judgement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction structure label. In a complete structure, number should be accordant, otherwise, program will alert or jump out. The numbers used in one complete instruction structure can be identical with the numbers used in another complete instruction structure. Range: 0-8</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Instruction structure label. In a complete structure, number should be accordant, otherwise, program will alert or jump out. The numbers used in one complete instruction structure can be identical with the numbers used in another complete instruction structure. Range: 0-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF X#(0)==ON 0 IF X0=ON, execute INC TC#(0) TC0=TC0+1 ELSEIF X#(1) = OFF 0 IF X0=OFF, execute DEC TC#(0) TC0=TC0-1 ELSE 0 If above two conditions are not satisfied, ADD TC#(0) TC#(0) execute TC0=TC0+TC0 END IF 0 End IF instruction</td>
</tr>
</tbody>
</table>

### Function

IF condition judgement instruction. The whole instruction should consist of four parts: IF, ELSEIF (can be omitted or be used repeatedly), ELSE, ENDIF. Firstly, judge if the condition is satisfied, if satisfied, the instruction after the IF will be executed; If not satisfied, judge the condition after ELSEIF (ELSEIF can be used according the situation or not used). If condition after ELSEIF is satisfied, the instructions after ELSEIF will be executed. If not satisfied, the instruction after ELSE will be executed. At the end of the IF condition judgement, use ENDIF to label the end.

### Addition

<table>
<thead>
<tr>
<th>X&lt;Variable#&gt;</th>
<th>M&lt;Variable#&gt;</th>
<th>Y&lt;Variable#&gt;</th>
<th>T&lt;Variable#&gt;</th>
<th>C&lt;Variable#&gt;</th>
<th>OP</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>=</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

83
If IF condition is not satisfied, judge the ELSEIF condition. If ELSEIF condition is satisfied, the instructions after the ELSEIF will be executed. If IF condition and ELSEIF condition(s) are not satisfied, the instructions after the ELSE will be executed. ENDIF to end the IF judgement.

Example

<table>
<thead>
<tr>
<th>Instruction structure label. In a complete structure, number should be accordant, otherwise, program will alert or jump out. The numbers used in one complete instruction structure can be identical with the numbers used in another complete instruction structure. Range: 0-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;number&gt;</td>
</tr>
<tr>
<td>GI&lt; Variable#&gt;</td>
</tr>
<tr>
<td>LI&lt; Variable#&gt;</td>
</tr>
<tr>
<td>GD&lt; Variable#&gt;</td>
</tr>
<tr>
<td>LD&lt; Variable#&gt;</td>
</tr>
<tr>
<td>GP&lt; Variable#&gt; &lt;data#&gt;</td>
</tr>
<tr>
<td>LP&lt; Variable#&gt; &lt;data#&gt;</td>
</tr>
<tr>
<td>TC&lt; Variable#&gt;</td>
</tr>
<tr>
<td>CC&lt; Variable#&gt;</td>
</tr>
<tr>
<td>GS&lt; Variable#&gt;</td>
</tr>
<tr>
<td>&lt;Character string&gt;</td>
</tr>
<tr>
<td>GI&lt; Variable#&gt;</td>
</tr>
<tr>
<td>LI&lt; Variable#&gt;</td>
</tr>
<tr>
<td>GD&lt; Variable#&gt;</td>
</tr>
<tr>
<td>LD&lt; Variable#&gt;</td>
</tr>
<tr>
<td>GP&lt; Variable#&gt;</td>
</tr>
<tr>
<td>LP&lt; Variable#&gt; &lt;data#&gt;</td>
</tr>
<tr>
<td>TC&lt; Variable#&gt;</td>
</tr>
<tr>
<td>CC&lt; Variable#&gt;</td>
</tr>
<tr>
<td>D&lt;Variable#&gt;</td>
</tr>
<tr>
<td>IF X#(0)==ON 0 IFX0=ON, execute</td>
</tr>
<tr>
<td>INC TC#(0) TC0=TC0+1</td>
</tr>
<tr>
<td>ELSEIF X#(1)==ON 0 IF X1=ON, execute</td>
</tr>
<tr>
<td>ADD TC#(0) 2.000 TC0=TC0+2</td>
</tr>
<tr>
<td>ELSEIF X#(2)==ON 0 IF X2=ON, execute</td>
</tr>
<tr>
<td>ADD TC#(0) 3.000 TC0=TC0+3</td>
</tr>
<tr>
<td>ELSE 0 Otherwise, execute</td>
</tr>
<tr>
<td>ADD TC#(0) TC#(0) TC0=TC0+TC0</td>
</tr>
<tr>
<td>ENDIF 0 0#Instruction structure end</td>
</tr>
</tbody>
</table>
### Function

IF condition judgement instruction. The whole instruction should consist of four parts: IF, ELSEIF (can be omitted or be used repeatedly), ELSE, ENDIF.

Firstly, judge if the condition is satisfied, if satisfied, the instruction after the IF will be executed; If not satisfied, judge the condition after ELSEIF (ELSEIF can be used according the situation or not used). If condition after ELSEIF is satisfied, the instructions after ELSEIF will be executed. If not satisfied, the instruction after ELSE will be executed. At the end of the IF condition judgement, use ENDIF to label the end.

<table>
<thead>
<tr>
<th>2 condition judge (IF→ELSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
</tr>
</tbody>
</table>

Instruction structure label. In a complete structure, number should be accordant, otherwise, program will alert or jump out.
The numbers used in one complete instruction structure can be identical with the numbers used in another complete instruction structure. Range: 0-8

<table>
<thead>
<tr>
<th>Addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;number&gt;</td>
</tr>
</tbody>
</table>

### Example

IF X#(1)==ON 0  If X1=ON,  
INC TC#(0)  execute TC0=TC0+1.  
ELSE 0  Otherwise,  
ADD TC#(0) 3.000  execute TC0=TC0+3  
END IF 00#IF instruction structure end

IF X#(1)==OFF 0  If X1=OFF,  
ADD TC#(0) 4.00  execute TC0=TC0+4  
ELSE 0  otherwise,  
ADD TC#(0) 5.00  execute TC0=TC0+5  
END IF 00#IF instruction structure end

### 2 Condition (IF→ENDIF)  

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
</table>

IF condition judgement instruction. The whole instruction should consist of four parts: IF, ELSEIF (can be omitted or be used repeatedly), ELSE, ENDIF.

Firstly, judge if the condition is satisfied, if satisfied, the instruction after the IF will be executed; If not satisfied, judge the condition after ELSEIF (ELSEIF can be used according the situation or not used). If condition after ELSEIF is satisfied, the instructions after ELSEIF will be executed. If not satisfied, the instruction after ELSE will be executed. At the end of the IF condition judgement, use ENDIF to label the end.

<table>
<thead>
<tr>
<th>2 Condition (IF→ENDIF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
</tr>
</tbody>
</table>

Addition <number> Instruction structure label. In a complete structure, number should be accordant, otherwise, program will alert or jump out.
The numbers used in one complete instruction structure can be identical with the numbers used in another complete instruction structure. Range: 0-8

IF X#(1)==ON 0  If X1=ON,  
INC TC#(0)  execute TC0=TC0+1.  
ELSE 0  Otherwise,  
ADD TC#(0) 3.000  execute TC0=TC0+3  
END IF 00#IF instruction structure end

IF X#(1)==OFF 0  If X1=OFF,  
ADD TC#(0) 4.00  execute TC0=TC0+4  
ELSE 0  otherwise,  
ADD TC#(0) 5.00  execute TC0=TC0+5  
END IF 00#IF instruction structure end
Instruction structure label. In a complete structure, number should be accordant, otherwise, program will alert or jump out.

The numbers used in one complete instruction structure can be identical with the numbers used in another complete instruction structure. Range: 0-8

IF <condition><number> IF condition
...... Program if IF condition is satisfied.
ELSEIF <condition><number> ESLEIF condition
...... Program if ELSEIF condition is satisfied.
ELSE<number>
Program if the previous conditions are not satisfied.
......
END IF <number> ENDIF to end IF structure

WHILE loop instruction. The complete structure should consist of WHILE and ENDWHILE.

When WHILE condition is satisfied, i.e. WHILE condition is ON, the program between WHILE and ENDWHILE will be executed until WHILE condition is no longer satisfied. Then jump out of this loop. WHILE condition must be set inside the loop part, otherwise, endless loop may occur.

WHILE loop condition instruction.

When WHILE condition is satisfied, i.e. WHILE condition is
### Function

WHILE loop instruction. The complete structure should consist of WHILE and ENDWHILE.

### Addition

<table>
<thead>
<tr>
<th>&lt;number&gt;</th>
</tr>
</thead>
</table>

### Example

```
SET TC#(1) 0.000
WHILE TC#(1) < 20 0
IF X#(1)==ON 0
INC TC#(1)
ELSE 0
ADD TC#(1) 2.000
ENDIF 0
ENDWHILE 0
SET TC#(1) 15.000
```

```
SET TC1=0
IF TC1<20, Execute IFstructure
IFjudge: if X1=ON, then
TC1=TC1+1
Otherwise if X1≠ON, then
TC1=TC1+2
IFstructureend
If TC1<20, execute TC1=15
```
<table>
<thead>
<tr>
<th>Example</th>
<th>WHILE &lt;condition&gt; &lt;number&gt; ...... ENDWHILE &lt;number&gt;</th>
<th>Judge condition, Satisfied: Execute ......part; Unsatisfied: Execute the program behind ENDWHILE.</th>
</tr>
</thead>
</table>

**Function**

Switch condition selection instruction. The complete selection structure consists of SWITCH,CASE (can be used repeatedly), BREAK, DEFAULT, ENDSWITCH. Calculate the variable value behind the SWITCH, judge the value is equal to which value behind CASE. After finding the matched CASE, program will be executed from that line until the ENDSWITCH. If no identical value can be found, program behind DEFAULT will be executed until the ENDSWITCH. Notice: The different positions of BREAK may result into different result.

<table>
<thead>
<tr>
<th>X&lt;Variable#&gt;</th>
<th>M&lt;Variable#&gt;</th>
<th>Y&lt;Variable#&gt;</th>
<th>T&lt;Variable#&gt;</th>
<th>C&lt;Variable#&gt;</th>
<th>Status variables, CASE = 0 or 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>GI&lt;Variable#&gt;</td>
<td>LI&lt;Variable#&gt;</td>
<td>GD&lt;Variable#&gt;</td>
<td>LD&lt;Variable#&gt;</td>
<td>GP&lt;Variable#&gt;&lt;data#&gt;</td>
<td>Number variables, CASE equal to specific number</td>
</tr>
<tr>
<td>LP&lt;Variable#&gt;&lt;data#&gt;</td>
<td>TC&lt;Variable#&gt;</td>
<td>CC&lt;Variable#&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 switch (SWITCH) →SWITCH) 【shortcut: 851】

<table>
<thead>
<tr>
<th>GS&lt;Variable#&gt;</th>
<th>Character string, CASE equal to specific character.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>&lt;number&gt;</th>
<th>Instruction structure label. In a complete structure, number should be accordant, otherwise, program will alert or jump out. The numbers used in one complete instruction structure can be identical with the numbers used in another complete instruction structure. Range: 0-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>example</td>
<td>SWITCH TC#(2) 0</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>CASE 10 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=1.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>CASE 20 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=2.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>CASE 30 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=3.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>CASE 40 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=4.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>CASE 50 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=5.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>CASE 60 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=6.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>CASE 70 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=7.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>CASE 80 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=8.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>CASE 90 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=9.000</td>
</tr>
<tr>
<td></td>
<td>BREAK 0</td>
</tr>
<tr>
<td></td>
<td>DEFAULT 0</td>
</tr>
<tr>
<td></td>
<td>AOUT AO#(1)=10.000</td>
</tr>
</tbody>
</table>

5 condition selection (SWITCH CASE H → CAS E)  
【shortcut: 8 5 2】

| Function | SWITCH condition selection instruction. The complete selection structure consists of SWITCH, CASE (can be used repeatedly), BREAK, DEFAULT, END SWITCH.
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Calculate the variable value behind the SWITCH, judge the value is equal to which value behind CASE. After finding the matched CASE, program will be executed from that line until the END SWITCH. If no identical value can be found, program behind DEFAULT will be executed until the END SWITCH.</td>
</tr>
<tr>
<td>Addition</td>
<td>Possible values behind SWITCH.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>&lt;data&gt;</td>
<td>Instruction structure label. In a complete structure, number should be accordant, otherwise, program will alert or jump out. The numbers used in one complete instruction sturture can be identical with the numbers used in another complete instruction structure. Range: 0-8</td>
</tr>
<tr>
<td>&lt;number&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```
SWITCH TC#(1) 0
CASE 1 0
CASE 2 0
CASE 3 0
CASE 4 0
CASE 5 0
DOUT Y#(18)==ON
BREAK 0
DEFAULT 0
DOUT Y#(18)==OFF
ENDSWITCH 0
```

**Function**

SWITCH condition selection instruction. The complete selection structure consists of SWITCH, CASE (can be used repeatedly), BREAK, DEFAULT, ENDSWITCH.

Calculate the variable value behind the SWITCH, judge the value is equal to which value behind CASE. After finding the matched CASE, program will be executed from that line until the ENDSWITCH. If no identical value can be found, program behind DEFAULT will be executed until the ENDSWITCH.

**Notice:** The different positions of BREAK may result into different behaviors.
<table>
<thead>
<tr>
<th>8 5 5</th>
<th>Addition</th>
<th>Number range: 0-8,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;number&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```
SWITCH <variable><number>
CASE <value><number>
......
BREAK <number>
DEFAULT <number>
......
ENDSWITCH <number>
```

**Functions**

SWITCH condition selection instruction. The complete selection structure consists of SWITCH, CASE (can be used repeatedly), BREAK, DEFAULT, ENDSWITCH.

End symbol of SWITCH structure.

5 switch (SWITCH  
→ENDSWITCH)  
【shortcut: 8 5 6】

<table>
<thead>
<tr>
<th>8 5 6</th>
<th>Addition</th>
<th>Number range: 0-8,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;number&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

```
SWITCH <variable><number>
CASE <value><number>
......
BREAK <number>
DEFAULT <number>
......
ENDSWITCH <number>
```

**Functions**

This complete instruction structure consists of PULSETTRANSFORM and PULSETTRANSFORMEND.

This function will transfer the pulse from output axis to input axis, i.e. originally system sends pulse to output axis to move; after the execution of this instruction, pulse will be sent to input axis, and input axis will move according to the input pulse, and output axis will stop, and original input pulse will become ineffective.

This instruction is effective to joint coordination.

6 Pulse transform between axis (PULSETTRANSFORM)  
【shortcut: 8 6】

<table>
<thead>
<tr>
<th>8 6</th>
<th>Addition</th>
<th>Output axis is the axis which user wants to transfer pulse from; input axis is the axis which user wants to transfer pulse into. Axis number range: 1-8.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt;output axis#&gt; &lt;input axis#&gt;</td>
<td></td>
</tr>
</tbody>
</table>
| Example | PULSETRANSFORM#(1)(2)  

......  

PULSETRANSFORMEND |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>This complete instruction structure consists of PULSETRANSFORM and PULSETRANSFORMEND. PULSETRANSFORMEND will end the pulse transform. And each axis will operate according to the pulse received by itself.</td>
</tr>
<tr>
<td>Addition</td>
<td>[none]</td>
</tr>
</tbody>
</table>
| Example | PULSETRANSFORM#(1)(2)  

......  

PULSETRANSFORMEND |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Treat the specific axis joint coordination: divide the coordination by 360, and the remainder will be set as the current coordination. In this way, when zero return, joint will not rotate the whole angle. E.g.: J1 coordination is 860.245. After inputing, coordination is 140.245. In zero return operation, J1 should rotate 140.245 rather than 860.245. This function is mainly used to the joints with incremental programmer.</td>
</tr>
</tbody>
</table>
| Addition | <axis#>  

Axis need inputing.range: 1-8 |
| Example | IMPUTED#(1)  

IMPUTED#(2)  

Impute J1  

Impute J2 |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>This zero return instruction is specifically used for incremental programmer. When zero return instruction is executed, system will execute the mechanical zero return movement according to the set axis number and the zero return parameters of this axis.</td>
</tr>
</tbody>
</table>
| Addition | <axis#>  

Axis need zero return.range: 1-8 |
| Example | ZERO #(1)  

ZERO #(2)  

Zero return of axis 1  

Zero return of axis 2 |
### 10 Finish of motion (MOTION FINISH)

**Function**

Motion finish is effective to the MOVJ, MOVL instructions before the motion finish line.

Working process: The move instruction will move to the set position precisely. In motion finish process, system will compare the electrical machine feedback data with the setting data of program until two data is identical (i.e. when system pluse is about to finish, electric machine will also stop at the position, and position error is within the allowed limit.), MOTIONFINISH will end. Otherwise, when time is longer than the internal set time, system will prompt that motion finish time out.

<table>
<thead>
<tr>
<th>Addition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>[none]</td>
<td>MOVJ VJ=10% PL=5 MOTIONFINISH</td>
</tr>
<tr>
<td></td>
<td>Effective to the previous MOVL instruction to make sure the robot can move to the accurate position.</td>
</tr>
</tbody>
</table>

### 11 Change tool coordination (CHANGE USE)

**Function**

Change tool coordination instruction. When this instruction is executor, program will change from the current coordination number to the coordination set by CHANGUSE.

If change is from 0# coordination to other coordination or none-0# coordination to 0# coordination, there’s no need to use this instruction.

If change is from one none-0# coordination to another none-0# coordination, this instruction is needed.

This operation may cause danger, please check the coordination number carefully and make sure there’s no danger if robot works in the coordination.

<table>
<thead>
<tr>
<th>Addition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;coordination#&gt;</td>
<td>MOVJ VJ=10% PL=0 tool coordination 0#</td>
</tr>
<tr>
<td></td>
<td>MOVJ VJ=20% PL=0 USE=3 change to 3#</td>
</tr>
<tr>
<td></td>
<td>CHANGEUSE#(4) change to 4#</td>
</tr>
<tr>
<td></td>
<td>MOVJ VJ=20% PL=0 USE=4 change to 4#</td>
</tr>
<tr>
<td></td>
<td>MOVJ VJ=20% PL=0 change to 0#</td>
</tr>
</tbody>
</table>
### Function

**FORBIDAXIS and FORBIDAXISEND** should be used in pair.

- **FORBIDAXIS** forbids the set axis to move, and cancels this forbid.
- **FORBIDAXISEND** removes the forbid status of the axis.

**Function**

- **FORBIDAXIS(x)**: Forbids setting of axis x.
  - Example: `FORBIDAXIS(1)`: Forbids axis J1 to move.
  - Example: `FORBIDAXISEND(1)`: Removes the forbid status of axis J1.

**Addition**

- `<axis#>`: The axis that needs forbidding move or removing the forbid. Range: 1-8~J1-J8 axis.

### Addition

**COUNTCOORD**

- **Function**: When this instruction is executed, the system will read the current axis programmer data and then calculate the current coordination.
- **Parameters**:
  - 0: Coordination calculation time out or beyond the error range, system will alert.
  - 1: Coordination calculation operating, no alarm.

**Example**

- **COUNTCOORD(0)**: Reads all axis programmer data, calculates coordination, and alerts if any error.
- **COUNTCOORD(1)**: Calculates coordination without an alarm.

### Change Tool Coordination

**Function**

- **CHANGEUSE**
  - **Function**: Changes tool coordination instruction. When this instruction is executed, the program will change from the current coordination number to the coordination set by **CHANGEUSE**.
  - If change is from 0# coordination to another coordination or none-0# coordination to 0# coordination, there’s no need to use this instruction.
  - If change is from one none-0# coordination to another none-0# coordination, this instruction is needed.
  - This operation may cause danger, please check the coordination number carefully and make sure there’s no danger if the robot works in the coordination.

**Addition**

- `<coordination#>`: Number of coordination need changing, range: 0-49.
### 5.3.9 Vision Instruction

<table>
<thead>
<tr>
<th>Example</th>
<th>MOVJ VJ=10% PL=0 tool coordination 0#</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MOVJ VJ=20% PL=0 USE=3 change to 3#</td>
</tr>
<tr>
<td></td>
<td>CHANGEUSE#(4) change to 4#</td>
</tr>
<tr>
<td></td>
<td>MOVJ VJ=20% PL=0 USE=4 change to 4#</td>
</tr>
<tr>
<td></td>
<td>MOVJ VJ=20% PL=0 change to 0#</td>
</tr>
</tbody>
</table>

**1. Vision Run Function**

**Function:** Vision run instruction: system will call the process number related parameters, and establish the connection with the system.

- If process document has start based on time or touch, after this instruction, vision system will start continuous start according the preset. Data will be stored in the system.
- If connection process is during the program execution, connection process will continue until program finish and quit. If there are many vision-run instructions in one program, only the first one is effective, i.e. when program is executed continuously, the first execution of vision run instruction is effective and the rest instructions will not be effective.

**Example**

```
RUNVISION#(0) Establish the connection of system and vision with the parameters in 0# process document.
```

**2. Get Vision Data Function**

**Function:** Import vision cache data into the GP52 and GP52 (the values of GP52 and GP52 is same).

**Example**

```
GETVISIONDATA#(0) Import data into GP52 and GP53.
```

**3. Clear Vision Data Function**

**Function:** Clear the vision data in current cache area.

**Example**

```
CLEARVISIONDATA#(0) Clear the tool data acquired by 0# process camera in the cache.
```
### Vision Trigger (TRIGGERVISION)

**Function:**
When trigger instruction is chosen in certain process document, this instruction can trigger the vision system, calculate the tool data, write the acquired data into vision cache, and write the number of data into GI50.

In order to get rid of delay, this instruction will directly trigger the vision system via output port Y.

**Shortcut:** 1 E 4

| Example | TRIGGERVISION#(0) | trigger the 0# process vision system |

### Camera Program Change (CHANGEVISION)

**Function:**
When this instruction is executed, system will change the tool calculation parameters document by the data connect wire to the vision system.

**Shortcut:** 1 1 E 5

| Example | TRIGGERVISION#(0) | trigger the 0# process vision system |

| Example | CHANGVISION#(222) | Vision system will call the tool B calculation parameters document (222). |

### Example

**Integrated Vision Example:**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUNVISION#(0)</td>
<td>run 0# vision process</td>
</tr>
<tr>
<td>*22</td>
<td>jump symbol *22</td>
</tr>
<tr>
<td>TRIGGERVISION#(0)</td>
<td>Trigger for first time, acquire camera calculation data of tool (相机计算工件数据).</td>
</tr>
<tr>
<td>TIME T= 200</td>
<td>delay 200ms</td>
</tr>
<tr>
<td>JUMP *22 IF GI#(50)&lt;=0.000</td>
<td>Judge if GI50 has data. If no, jump to *22, trigger again.</td>
</tr>
<tr>
<td>GETVISONDATA#(0)</td>
<td>Acquir data from vision cache, write them into GP52 and GP53.</td>
</tr>
<tr>
<td>ADD GP#53(3) 10.000</td>
<td>Move GP53 for 10MM in Z direction, in case of collision.</td>
</tr>
<tr>
<td>MOVL VL=500MM/S GP#53 PL=0</td>
<td>Move to GP53 safe point.</td>
</tr>
<tr>
<td>MOVL VL=100MM/S GP#52 PL=0</td>
<td>Move to GP52 grab point (抓取点).</td>
</tr>
<tr>
<td>DOUT Y#(1)=ON</td>
<td>Grabe</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>1. <strong>Strat</strong>&lt;br&gt;track(TARCK START)<strong>&lt;br&gt;【shortcut: 1</strong>&lt;br&gt;11**&lt;br&gt;E1】</td>
<td>When this instruction is executed, system will call the track process parameters, write the current tool position of cache into the GP50 and GP51 (the values of GP52 and GP52 is same). At this time, the data in GP50 and GP51 is real-time changing. Then system will call the GP50 and GP51 data to start tracking.</td>
</tr>
<tr>
<td>example</td>
<td>TRACKSTART#(0) Prepare the GP50 and GP51 data</td>
</tr>
<tr>
<td>2. <strong>End</strong>&lt;br&gt;track(TARCK END)<strong>&lt;br&gt;【shortcut: 1</strong>&lt;br&gt;11**&lt;br&gt;E2】</td>
<td>Track end, robot will stop tracking tool. Data of GP50 and GP51 will not be real-time any longer.</td>
</tr>
<tr>
<td>example</td>
<td>TRACKEND# (0) End of tool tracking</td>
</tr>
<tr>
<td>3. <strong>Acquir</strong>&lt;br&gt;track data(GET TRACK DATA)<strong>&lt;br&gt;【shortcut: 1</strong>&lt;br&gt;111 E3】&lt;br&gt;</td>
<td>Call the track document set parameters, judge the data in tracking cache. If no data in tracking cache, wait. If tool hasn’t entered the AB region, wait. If tool has gone beyond the B, abandon the data and wait.</td>
</tr>
<tr>
<td>example</td>
<td>GERRTACKDATA# (0) X# (0.0) Y# (0.0) Z# (0.0) Judge if the tracking cache has effective data. No compensation for XYA axis.</td>
</tr>
<tr>
<td>4. <strong>Clear</strong>&lt;br&gt;track data&lt;br&gt;(CLEAR STACK DATA)<strong>&lt;br&gt;【shortcut: 1</strong>&lt;br&gt;11 E4】&lt;br&gt;</td>
<td>Clear the tracking cache data related with the set tool number. This instruction is used in the cache cleanup situation.</td>
</tr>
<tr>
<td>example</td>
<td>CLEARTRACKDATA#(0) Clear the 0# tracking document-related data in cache.</td>
</tr>
</tbody>
</table>
### 5.3.11 Communication instructions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Read BCD data of IO port</strong></td>
<td>Read the status of input port X, and write the data into set variable.</td>
</tr>
<tr>
<td><strong>Addition</strong></td>
<td><strong>Example</strong></td>
</tr>
<tr>
<td>G&lt;variable&gt;</td>
<td>READIOBCD #(20) #(0) #(4)</td>
</tr>
<tr>
<td>X&lt;port&gt;</td>
<td>Read the status data of X0-X3 (4 ports) into G120 variable.</td>
</tr>
<tr>
<td>B&lt;number&gt;</td>
<td></td>
</tr>
</tbody>
</table>

### Integrated tracking example:

- **MOVVL VL=500.00MM/S PL=0** Move to the air above the middle point, waiting for grab.
- **RUNIOCUITIN #**(1) Start backstage IO detection. If there is object passing, the current position data will be written into cache.
- **GETTRACKDATA#**(1) X#(0.0) Y#(0.0) Z#(0.0) Judge the data in tracking cache. If no data in tracking cache, wait. If tool hasn’t entered the AB region, wait. If tool has gone beyond the B, abandon the data and wait.
- **TREACKSTART #**(1) Track start; write the current tool data of cache into GP50 and GP51.
- **ADD GP#51(3) 10.000** Move GP51 for 10mm in Z direction, set this point as safe point. The meaning of (3) can refer to the data single-axis operation of GP variable.
- **MOVVL VL=500MM/S GP#51 PL=0** Move ot GP51 safe point.
- **MOVVL VL=100MM/S GP#50 PL=0** Move to GP50 tracking point.
- **DOUT Y#(0) =ON** Output Y0 is effective; grab objective.
- **MOVVL VL=500MM/S GP#51 PL=0** Lift the objective, move to GP51 safe point.
- **TRACKEND #**(1) Track end.
- **…** Following operation.
### 2 Read data to (READ DATA TO)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicate the data to the latter variable through serious port. Communication port # and communication function should be in operation parameters. This function is mainly to read barcode, code, etc.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Addition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GI&lt;variable#&gt;</td>
<td>Communicate data to GI variable.</td>
</tr>
<tr>
<td>GD&lt;variable#&gt;</td>
<td>Communicate data to GD variable.</td>
</tr>
<tr>
<td>GS&lt;variable#&gt;</td>
<td>Communicate data to GS variable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>READDATATO GI# (20)</td>
<td>Read the port data into variable GI20.</td>
</tr>
</tbody>
</table>

### 5.3.12 Special Instructions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>When this instruction is executed, system will return backwards to the first line of the program line to line. Even if current execution is within the subprogram, system will quit the subprogram and return to the first line of main program after this instruction is executed.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Addition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[none]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ......</td>
<td></td>
</tr>
<tr>
<td>CALL 123</td>
<td>Main program XX calls 123 subprogram</td>
</tr>
<tr>
<td>......</td>
<td>Lines after subprogram is end</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>--------</td>
<td>subprogram123</td>
</tr>
<tr>
<td>1 ......</td>
<td>program reset</td>
</tr>
<tr>
<td>RESET</td>
<td>subprogram return</td>
</tr>
<tr>
<td>RET</td>
<td></td>
</tr>
</tbody>
</table>

In the above program, when the RESET line in the 4th line of subprogram is executed, system will directly return to main program XX untile the dirt line. If there is no RESET instruction in subprogram 123, when RET isexecuted, system will return to main program and execute the 9th line in the main program.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move the automatical line pointer when operation. E.g. line pointer stops at 18th line in automatic operation. After the use of CHANGEPOINT# (-2), the program stop line will jump to (18-2=16)th line.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Addition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;line number&gt;</td>
<td>The lines of jump users want. Positive value means jump backwards, and negative value means jump forwards.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHANGEPOINT#(-2)</td>
<td>Stop line move 2 lines afterwards. E.g. line pointer stops at 18th line in automatic operation. After the use of CHANGEPOINT# (-2), the program stop line will jump to (18-2=16)th line.</td>
</tr>
</tbody>
</table>
6 Teach program

Teach program is the process of creating a user program in teach mode, including selecting correct coordination system, manually move the terminal of robot into the desired position; call the current coordination data, operation track, and process instructions by certain operation (e.g. press button to select the instructions, coordination data is invisible).

Movement instruction line includes move instruction, additional speed, smoothness, and joint data of robot (joint data is invisible in the data program edit interface).

If variables are used, the shutdown data of robot is contained in the variables.

The following sections are the examples to describe the teach program steps. For more program example, please refer to “process manual”.

This system supports the Chinese character display of program.

Correspondent parameters:

<table>
<thead>
<tr>
<th>parameter type</th>
<th>parameter</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operation parameters</td>
<td>program display</td>
<td>0</td>
<td>Letters: program is shown with letters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Characters: Program is shown with Chinese characters.</td>
</tr>
</tbody>
</table>

Specific steps:

Firstly, `<parameter settings>`-<7 system parameters>-<1 operation authority selection>-<confirm>. Input the integrator password in the prompt interface, click `<confirm>`, the authority will be changed to integrator authority.

Then, `<parameter settings>`-<3 operation parameters>-<confirm>. Choose `[program display]` using cursor in the prompt program list. Click `<modify>` in the sub menu. Input number 1 in the prompt input window, click `<confirm>`, as shown in below figure.
At this time, teach program in the system will be displayed in Chinese character, as shown in below:

<table>
<thead>
<tr>
<th>Number</th>
<th>Operating Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Fourth axis for delta code back to zero detection switch 0-...</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Fifth axis for delta code back to zero detection switch 0-d...</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Sixth axis for delta code back to zero detection switch 0-d...</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>Seventh axis for delta code back to zero detection switch 0...</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Eighth axis for delta code back to zero detection switch 0-...</td>
<td>1</td>
</tr>
</tbody>
</table>

Program display 0—letters 1—characters

0—display instruction 1—
display text

6.1 Welding teach program

Scheme as follows:

![Diagram showing welding teach program]

Related instructions:

MOVJ: joint move
MOVL: Line move
VJ: speed multiplier of joint move
VL: Speed multiplier of straight move
PL: smoothness
USE: tool coordination
ARCSTART: arc strat
ARCEND: arc end

Program list

<table>
<thead>
<tr>
<th>Program Instructions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

101
<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>In user coordination system USE=1, MOVJ joint move, VJ=50% (multiplier), PL=9 (smoothness), move to program point 1, reach the prepare point.</td>
</tr>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>In user coordination system USE=1, MOVJ joint move, VJ=50% (multiplier), PL=9 (smoothness), move to program point 2, approach to objective.</td>
</tr>
<tr>
<td>MOVJ VJ=25% PL=0 USE=1</td>
<td>In user coordination system USE=1, MOVJ joint move, VJ=25% (multiplier), PL=0 (smoothness), move to program point 3, touch the objective.</td>
</tr>
<tr>
<td>ARCSTART#(0)</td>
<td>arc start</td>
</tr>
<tr>
<td>MOVL VL=100.0MM/S PL=0 USE=1</td>
<td>In user coordination system USE=1, MOVL joint move, VL=100MM/S, PL=0 (smoothness), move to program point 4, operate in welding track.</td>
</tr>
<tr>
<td>ARCEND#(0)</td>
<td>arc end</td>
</tr>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>In user coordination system USE=1, MOVJ joint move, VJ=50% (multiplier), PL=9 (smoothness), move to program point 5, depart from the objective.</td>
</tr>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>In user coordination system USE=1, MOVJ joint move, VJ=50% (multiplier), PL=9 (smoothness), move to program point 1, return to prepare point.</td>
</tr>
</tbody>
</table>

Teach program steps

1. Switch to teach mode.
2. Choose proper coordination system.
3. Enter the program list interface.
4. Click **button.

5. Input the new program name (e.g. 555555) in the prompt interface. After the input, click **confirm**. At this time, the new program will shown in the program list interface, as follows:

If input the new program name, system will create a new program and highlight it by blue bar. If input the already-exist program name, system will highlight the program.
6. Click submenu button, open program edit window:

7. Holding security switch, move the terminal of robot into program point 1 by coordination button on the teach box. Click <Movement><MOVJ>.
Prompt instruction edit window.

**MOVJ**  GP  0  VJ 10.0 PL 0

Input related parameters, press . Instruction will show in program edit window.

1. MOVJ VJ=50.0% PL=0

Finish of instruction edit of program point 1.

8. Holding security switch, move the terminal of robot into program point 2 by coordination button in the teach box. Click <Movement> - <MOVJ>. MOVJ instruction edit window will prompt as follows:

**MOVJ**  VJ 50 PL 0

Input related parameters, press . Instruction will show in program edit window.

1. MOVJ VJ=50.0% PL=0
2. MOVJ VJ=50.0% PL=0
Finish of instruction edit of program point 2.

9. Repeat step 7 or 8, change VJ speed to 25%, PL=0. Input the instruction line of program point 3.

10. Click <program instruction>-<5 welding>-<1 ARC START>-<confrim>, following window will prompt.

Input related parameters, press , Instruction will show in program edit window.

11. Repeat the above steps, and input the every program points and instructions as follows:

The welding program is created by the above steps.

12 Click sub menu , then , close program edit window.
6.2 Lifting program example

scheme as follows:

```
MOVJ: joint move  MOVL: joint move  VJ: speed multiplier of joint move  VL: Speed multiplier of straight move
PL: smoothness  USE: tool coordination  DOUT: number index output
WAIT: conditional wait
```

6.2.1 program list

<table>
<thead>
<tr>
<th>program instructions</th>
<th>descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>In user cooordination system USE=1, MOVJ joint move, VJ=50% (multiplier), PL=9(smoothness), move to program point 1, reach the prepare point.</td>
</tr>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>Move to program point 2, approach to objective (before grab)</td>
</tr>
<tr>
<td>MOVL VL=100MM/S PL=0 USE=1</td>
<td>Move to program point 3, touch the objective (grab position)</td>
</tr>
<tr>
<td>DOUT Y#(0)=ON</td>
<td>grab</td>
</tr>
<tr>
<td>WAIT X#(0)=ON T=0</td>
<td>detect if grab is finished</td>
</tr>
<tr>
<td>MOVL VL=200MM/S PL=9 USE=1</td>
<td>Move to program point 4, leave objective (after grab)</td>
</tr>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>Move to program point 5, back to initial position</td>
</tr>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>Move to program point 6, approach put position</td>
</tr>
<tr>
<td>MOVJ VJ=50% PL=9 USE=1</td>
<td>Move to program point 7, reach auxiliary put point</td>
</tr>
<tr>
<td>MOVL VL=100MM/S PL=0 USE=1</td>
<td>Move to program point 8, reach put position</td>
</tr>
</tbody>
</table>
DOUT Y#(0)=OFF
WAIT X#(0)=OFF T=0
MOV L VL=200MM/S PL=9 USE=1
MOV J VJ=50% PL=9 USE=1

6.2.2 Steps for teach program

1. Switch to teach mode.

2. Choose proper coordination system.

3. Enter the program list interface.

<table>
<thead>
<tr>
<th>File</th>
<th>operation</th>
<th>name</th>
<th>Duration</th>
<th>size(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>1111</td>
<td></td>
<td>2016-09-27 10:48</td>
<td>2500</td>
</tr>
<tr>
<td>123</td>
<td>1234</td>
<td></td>
<td>2016-10-26 16:50</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>99999</td>
<td></td>
<td>2016-10-26 16:50</td>
<td>616</td>
</tr>
<tr>
<td>abc</td>
<td>ARCLINE</td>
<td></td>
<td>2016-10-31 10:39</td>
<td>0</td>
</tr>
</tbody>
</table>

4. Click the button.

5. Input the new program name (e.g. transport) in the prompt interface. After the input, click <<confirm>>. At this time, the new program will show in the program list interface, as follows:

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>If input the new program name, system will create a new program and highlight it by blue bar. If input the already-exist program name, system will highlight the program.</td>
</tr>
</tbody>
</table>
6. Click sub menu **Open** button, open program edit window.

7. Holding security switch, move the terminal of robot into program point 1 by coordination button on the teach box. Click `<Command>-<Movement>-<MOVJ>`-
Prompt instruction edit window.

```
MOVJ -  -  YJ=50.0  PLG 9
```

### Description

*YJ=50.0*  Is the input window, data can be written into it.

Input related parameters, press *Inst correct*. Instruction will show in program edit window.

```
1  MOVJ VJ=50.0  PL=0
```

Finish of instruction edit of program point 1.

8. Holding security switch, move the terminal of robot into program point 2 by coordination button in the teach box. Click `<Command><1 MOVJ>`. MOVJ instruction edit window will prompt as follows:

```
MOVJ -  -  YJ=50.0  PLG 9
```

Input related parameters, press *Inst correct*. Instruction will show in program edit window.
Finish of instruction edit of program point 2.

9. Repeat step 7 or 8, change VJ speed is 100MM/S, PL=0. Input the instruction line of program point 3.

```
1  MOVJ VJ=50. 0% PL=0
2  MOVJ VJ=50. 0% PL=0
3  MOVJ VJ=50. 0% PL=0
```

Input related parameters, press Instruction will show in program edit window.

```
1  MOVJ VJ=50.0% PL=0
2  MOVJ VJ=50.0% PL=0
3  MOVJ VJ=50.0% PL=0
4  MOVJ VJ=50.0% PL=0
5  MOVJ VJ=50.0% PL=0
```

10. Click <Command>-<2 Logic>-<1 DOUT>, following window will prompt.

Input related parameters, press Instruction will show in program edit window.

```
1  MOVJ VJ=50. 0% PL=0
2  MOVJ VJ=50. 0% PL=0
3  MOVJ VJ=50. 0% PL=0
4  MOVJ VJ=50. 0% PL=0
5  MOVJ VJ=50. 0% PL=0
6  DOUT Y=(0)=ON
```

11. Click <Command>-<2 logic>-<3 WAIT>, prompt follow window:

Input related parameters, press Instruction will show in program edit window.

```
1  MOVJ VJ=50. 0% PL=0
2  MOVJ VJ=50. 0% PL=0
3  MOVJ VJ=50. 0% PL=0
4  MOVJ VJ=50. 0% PL=0
5  MOVJ VJ=50. 0% PL=0
6  MOVJ VJ=50. 0% PL=0
7  DOUT Y=(0)=ON
8  WAIT X=(0)=ON DT=O CT=O
```

12. Repeat the above steps, and input the every program points and instructions as follows:
13. Click sub menu then click close program edit window.

The lifting program is created by the above steps.

7 Program edit functions

Program edit functions mainly include: program list edit, mainly used to create, rename, backup, delete, and etc. of program; Program interface edit, mainly used to copy, cut, delete, find, replace and etc. of program line.

7.1 Edit in Program list

Create, rename, backup, delete, open U disk, oped or copy to U disk.
7.1.1: Create program

Click the button, Create program window will prompt. Type the program name in the blank, click. Close input window. At this time, cursor will jump to the new program just created, and highlight it. If input the already-exist program name, system will highlight the program.

Programming name can be any letters, number, Chinese character, and their combination.

If users want to cancel the input, press to quit.

7.1.2: Rename the already-exist program

Users should move the cursor at the program need renaming, then click. Windows as follows will prompt: type the modified name in the blank, and click. The input window will close. At this time, cursor will jump at the renamed program in the program list.

Programming name can be any letters, number, Chinese character, and their combination.

If the typed name is the same as one already-exist program, the info area will show:

1 test 更名为 test 存在相同的名称，更名失败！

If users want to cancel the input, press to quit.

7.1.3: Copy the current program and paste into another program with another name.

Users should move the cursor at the program need backup, then click. Windows as follows will prompt: Type the new name in the blank, and click. The input window will close. At this time, cursor will jump at the renamed program in the program list.

Programming name can be any letters, number, Chinese character, and their combination.

If the typed name is the same as one already-exist program, the info area will show:

1 test 更名为 test 存在相同的名称，更名失败！
If users want to cancel the input, press Exit to quit.

7.1.4 Delete : Delete the current program

Users should move the cursor at the program need deleting, then click Delete, windows as follows will prompt:

> Whether delete test Program?

Yes, confirm delete. The program will disappear.

If user wants to cancel press No button to quit.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete is irreversible, so be cautious with delete operation.</td>
</tr>
</tbody>
</table>

7.1.5 Open U disk : Open the content list of U disk

This list only shows the user program that system can recognize; other types of document will not show. If U disk is not inserted or not recognized, system will show: please check if U disk is inserted or disinstalled. Users should insert U disk again.

7.1.5.1 Copy to system : Copy the program to system.

This button is to copy the selected program (by cursor) to the system program list. Click Copy to system, system will prompt:

> Whether to copy program from U disk To system?

Click Yes, Copy the selected program to the system program list. If users want to cancel, press No button to quit.
7.1.5.2 Uninstall the current U disk:

Uninstall the current U disk, and quit the U disk content disk, return to system program list. Info bar shows: U disk uninstall succeed.

7.1.5.3 EXIT:

Quit from U disk content, and return to system program list.

7.1.6: Open the selected program and enter the program edit interface.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Teach program and program edit should be operated in program edit interface.</td>
</tr>
<tr>
<td>2. Operation in play mode should also require program is open.</td>
</tr>
<tr>
<td>3. Operation in remote mode doesn’t need open the program manually. After the power-on of driver, system will automatically open the program, whose name has been set in the remote process.</td>
</tr>
</tbody>
</table>

Users should move cursor at the program users want to open, and then click the button; system will enter the program edit interface of selected program. The already-exist program line will show as follows:

1. MOVJ VJ=10.0% PL=0
2. MOVJ VJ=10.0% PL=0
3. MOVVL V1=100.0MM/S PL=0

7.1.7 Copy the program to U disk:

Insert the U disk system can recognize, then select the program (by cursor) that users want to copy. Click the button, the following window will prompt:

Click Yes to copy the selected program and send it to U disk. If users want to cancel, press No button to quit. After copy, click to check if there is the program in the U disk conten list. Check if the copy has completed.
7.2 Edit in program edit interface

- copy–copy current line, copy–copy section, paste, cut–cut current line, cut–cut section, delete, find, replace, turn into, modify as time sequence, reset.

<table>
<thead>
<tr>
<th>Files operation</th>
<th>Edit</th>
<th>Para setting</th>
<th>Monitoring</th>
<th>Run program</th>
<th>Command</th>
<th>Technical</th>
<th>PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7.2.1 1.copy→1 copy current line:

Copy cursor located lineto backstage

**description**

When copy the program line, teach point data will also be copied. So please pay attention to the execution position of the copied program line.

Move the cursor to the line need copying, select <copy current line>; move cursor to the position of paste, use <paste> to past the line before the cursor line. E.g., copy the line 4 ARCSTART# (0) to the line before line 8. The specific operation is following:

1. Move cursor at the line 4, as above figure.

2. Choose <Edit>–<1 copy>–<1 copy current line>–<confirm>, info bar shows multiple program line copy succeed.
3. Move the cursor at the line 8, as below figure.

4. Choose <Edit>-<2 paste>-<confirm>, info bar shows paste succeed. The pasted line will appear at the position between line 7 and line 8, while cursor is at this line, as below figure:

7.2.2 _copy->2 copy section :

Copy the contains between strating line and end line to the backstage

<table>
<thead>
<tr>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>When copy the program section, teach point data will also be copyed. So please pay attention to the execution position of the copyed program line.</td>
</tr>
</tbody>
</table>

Copy the contents between starting line and end line to the backstage, and then move the cursor the position of paste, use <paste> to past the section before the cursur line. E.g., copy the line 2 and line 3 to the line before line 8. The specific operation is following:

1. Choose <Edit>-<1 copy>-<2 copy section>-<confirm>. The following window will prompt:

input 2 in the starting line, and input 3 at the end line, click , and info bar shows multiple program lines copy succeed. If users want to cancel the copy operation, click button to quit.
2. Move cursor to line 8, as below figure:

```
1. M01 VJ=60.0% P1=9 TOOL=1
2. M01 VJ=60.0% P1=9 TOOL=1
3. M01 VJ=25.0% P1=9 TOOL=1
4. ARCSTART# (0)
5. M01 VJ=60.0% P1=9 TOOL=1
6. ARCEND# (0)
7. M01 VJ=25.0% P1=9 TOOL=1
8. M01 VJ=60.0% P1=9 TOOL=1
```

3. Choose <Edit> - <2 paste> - <confirm>, info bar shows past succeed. And the copied section (line 2 and line 3) will appear between the line 7 and line 8, while cursor is at the line 8, as below figure shows:

```
1. M01 VJ=60.0% P1=9 TOOL=1
2. M01 VJ=60.0% P1=9 TOOL=1
3. M01 VJ=25.0% P1=9 TOOL=1
4. ARCSTART# (0)
5. M01 VJ=60.0% P1=9 TOOL=1
6. ARCEND# (0)
7. M01 VJ=25.0% P1=9 TOOL=1
8. M01 VJ=60.0% P1=9 TOOL=1
9. M01 VJ=25.0% P1=9 TOOL=1
```

7.2.3 2paste : used to past after copy and cut

Paste the copied or cut program into the line where cursor is located.

7.2.4 3cut->1 cut the current line :

Copy the line where cursor is and delete this line.

Move the cursor to the line need cutting, then click “cut current line”. The selected line will be copyed to backstage and current line will be deleted. Move cursor to the desired position, click <paste>, and the cut line will appear at the cursor position. E.g. cut the line 4 ARCSTART# (0) to the position before line 8. The specific steps are as follows:

```
1. M01 VJ=60.0% P1=9 TOOL=1
2. M01 VJ=60.0% P1=9 TOOL=1
3. M01 VJ=25.0% P1=9 TOOL=1
4. ARCSTART# (0)
5. M01 VJ=60.0% P1=9 TOOL=1
6. ARCEND# (0)
7. M01 VJ=25.0% P1=9 TOOL=1
8. M01 VJ=60.0% P1=9 TOOL=1
9. M01 VJ=25.0% P1=9 TOOL=1
```

1. Move xursor to the line 4, as above figure.

2. Click <Edit> - <3 cut> - <1 cut current line> - <confirm>. System will show: current line has benn cut successfully. At this time, line 4 ARCSTART# (0) is copyed to the backstage. And line 4 is deleted in program edit interface, as below figure:
If users want to cancel the cut operation, press **Cancel** to quit before clicking **Confirm**. If users want to restore the cut line, move cursor to the proper position and paste the cut line back.

3. Move the cursor to the line 8 (line 7 after cut).

Click **Edit**->**2 paste**->**<confirm>**. At this time, the cut line will appear at the position above line 8, as below figure:

7.2.5 cut->2 cut section:

Copy the content between starting line and end line to the backstage, and delete them.

Cut the content between starting line and end line to the backstage, and then delete them.

Move the cursor the position of paste, use <paste> to paste the section before the cursor line.

E.g., copy the line 4, 5, and 6 to the line before line 8. The specific operation is following:

Choose **Edit**->**3 shear**->**2 cut block**. The following window will prompt:

Cut first 4 Cut end 6

caution: when operating block, line number displaying is
Input 4 in the starting line. Input 6 at the end line, click , and info bar shows multiple program lines cut succeed. At this time, the content between line 4 and line 6 has been deleted:

```
1  MOVJ VJ=50.0% PL=0
2  MOVJ VJ=50.0% PL=0
3  MOVJ VJ=50.0% PL=0
7  MOVJ VJ=50.0% PL=0
8  MOVJ VJ=50.0% PL=0
```

If users want to cancel the cut operation, click button to quit. If users want to restore the cut lines, move cursor to the proper position and paste the cut lines back.

2. Move cursor to line 8.

3. Choose <Edit>-<2 paste>, info bar shows paste succeed. And the cut section (line 4, 5, 6) will appear between the line 7 and line 8, while cursor is at the line 8, as below figure shows:

```
1  MOVJ VJ=50.0% PL=0 TOOL=1
2  MOVJ VJ=50.0% PL=0 TOOL=1
3  MOVJ VJ=50.0% PL=0 TOOL=1
4  AROSTART(0)
5  MOVJ VJ=50.0% PL=0 TOOL=1
6  AROENDX(0)
7  MOVJ VJ=50.0% PL=0 TOOL=1
8  MOVJ VJ=50.0% PL=0 TOOL=1
9  
```

7.2.6 Delete:

Delete the line where cursor is located

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete of program line is irreversible, please be cautious with delete operation!</td>
</tr>
</tbody>
</table>

Move cursor at the line users want to delete. Click <Edit>-<4 delete>-<confirm>. Info bar will show:

If users want to cancel the delete operation, press cancel button to quit before pressing confirm.

Notice: After pressing confirm button, the deleted line cannot be restored.
7.2.7 Find:

To find specific content, cursor will locate at the found line.

Firstly, move the cursor at the first line, click <edit>-<5 Search>, and following window will prompt:

Then select the addition users want to search, and input the addition values, as shown in above. Then click 5 Search, system will search. The nearest line with the same addition value will be found, and cursor will locate at the line. System prompt: find. If no addition value has been found, system will prompt: no result.

If there are many addition value is the same with the search condition, every time user clicks 5 Search button, system will search from the cursor position and find the nearest addition value. The rest can be found in the same manner.

If users don’t want to continue, click to quit.

Addtitions users can search include: PL、VJ、VL、X、Y、M、GP、LP、GD、LD、GI、LI。

7.2.8 Replace:

Replace the original content with the new content.

Firstly, move the cursor at the first line, click <Edit>-<6 replace> , and following window will prompt:

Then select the addition users want to replace, and input the original addition values 0 and replaced value 9, as shown in above. Then click Replace, system will search. The nearest line with the same addition value will be found, original value will be replaced by new one and cursor will locate at the line. System prompt: replace succeed. If no addition value has been found, system will prompt: no result.

If there are many addition value is the same with the search condition, every time user clicks Replace button, system will search from the cursor position and find the nearest addition value. The rest can be found in the same manner, until every addititions with the original value have been replaced.
If users don’t want to continue, click to quit.

Addtitions users can search include: PL, VJ, VL, X, Y, M, GP, LP, GD, LD, GI, LI.

7.2.9 Turn:

Locate the cursor at specific line.

E.g.: move cursor to line 6 using Turn function.

Click <Edit>-<7 Go to>, following window will prompt.

Type the number 6 into the blank. And click > button. Then cursor will be located at the line 6, as below figure.

If users don’t want to move the cursor at the turn window, users can press > button to quit.

7.2.10 Modify as timesequence:

Backup functions, under development, not available now

7.2.11 Program reset:

Reset to original condition

Click <Edit>-<9 turn>-<confirm>, system will prompt: Reset.

7.2.12 Modify instructions:

Modify the current program line.
Move cursor to the line users want to modify, click [Change inst], windows prompt the current program edit interface. After the modification, users can click [Inst correct] to confirm modification, or click [Inst exit] to cancel the modification.

The specific steps are as follows:

1. MOVJ VJ=50.0% PL=9 TOOL=1
2. MOVJ VJ=50.0% PL=9 TOOL=1
3. MOVJ VJ=50.0% PL=9 TOOL=1

Move cursor at the line 2, click [Change inst], following window will prompt:

In this status, click submenu <move> or <logic>, or click main menu <Command > to select other instructions to modify the current line. It’s also allowed to modify the addition without program modification.

Description

As for modification of the move instructions with move gesture, it’s necessary to hold the security switch to record the position and gesture.

As for modification of the move instructions without move gesture, it’s unnecessary to hold the security switch.

As for modification of the non-move instructions, it’s unnecessary to hold the security switch.

Modify the VJ speed to 30%, PL to 5, as following:

Click [Inst correct] confirm modification. Then program is:

Click [Inst exit] button to cancel modification.

7.2.13 move: Call move instruction

In program edit interface, click this button, system will prompt the move instructions windows as follows (can also be called by < Command >-<Movement>-<MOVJ>):
Click this button for many times, the instructions will change in the order of the list <Command>-<Movement>, e.g. MOVJ-MOVL-MOVC-MOVJ......

7.2.14 logic: Call logic instructions

In program edit interface, click this button, system will prompt the logic instructions windows as follows (can also be called by <Movement>-<1 logic>-<1 DOUT>-<confirm>):

Click this button for many times, the instructions will change in the order of the list <Command>-<logic>-<confirm>, e.g. DOUT-AOUT-WAIT-TIME-PAUSE-JUMP......

7.2.15 Open process: Open process button

When program line uses CALL instruction, and program calls the process document, users can open the process interface by pressing this button.

7.2.16 last instruction: Call last instruction

If other instructions have been used before, the last instruction window will be open after pressing this button.

e.g. In program edit interface, MOVJ line is created as follows:

Click <previous instruction>, the MOVJ window used last time will appear:

This button is mainly for convenient call of the same instructions.

7.2.17 save: Save the program

After the program editing, click the save button to save the program. Then status bar will show document saved.

7.2.18 close: Close the program edit interface.

Clicking this button can close the program edit window. However, if program hasn’t been saved, the edited content may lost.
8 Program test-run

Program test-run refer to: after the editing of program, users can allow robot to operate line by line of the program by certain operation. This is useful to help user to compare the actual move and move track and to judge is move or track is correct.

8.1 Preparation:

8.1.1 Related parameters

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation parameters</td>
<td>Cursor move in order in test-run</td>
<td>0</td>
<td>no move: Cursor stop at the line after test-run</td>
</tr>
<tr>
<td>(integrator authority)</td>
<td></td>
<td>1</td>
<td>move downwards: Cursor stop to the next line after test-run</td>
</tr>
</tbody>
</table>

Specific steps for parameters modification:

Firstly, <parameter settings>-<7 system parameters>-<1 operation authority selection>-<confirm>. Input the integrator password in the prompt interface, click <confirm>, the authority will be changed to integrator authority.

Then, <parameter settings>-<3 operation parameters>-<confirm>. Choose [cursor move in test-run] using cursor in the prompt program list. Click <modify> in the sub menu. Input number 0 or 1 in the prompt input window, click <confirm> to finish the modification of parameter of cursor move in test-run.

8.1.2 Other preparation

1. Switch to robot movement allowance status

2. Modify the speed into a proper speed by pressing or button or speed multiplier. It’s suggested that speed multiplier is less than 50%. The modified speed multiplier can be found in statu display area

8.2 Steps for program test-run:

1. Return to program list interface as follows:
2. Move cursor to the line users want to run, as the 555555 in above figure. Then click submenu <open> button to open this program. Users will enter the program edit interface, as follows:

```
1. MOVJ VJ=50.0% PL=O
2. MOVJ VJ=50.0% PL=O
3. MOVJ VJ=50.0% PL=O
7. MOVJ VJ=50.0% PL=O
8. MOVJ VJ=50.0% PL=O
```

3. Move cursor at the beginning of the line that users want to test-run, for example, at the beginning of second line.

4. Holding security switch, then press button in character keyboard. System will control the robot to execute the current line, such as robot movement, IO output, calculation, logic......

<table>
<thead>
<tr>
<th>Notice</th>
</tr>
</thead>
<tbody>
<tr>
<td>When cursor is located within the IF, WHILE, SWITCH structure, system will prompt error alert. So before the test-run, move the cursor beyond the above instruction structure. When MOVC is test-run of define the points, robot move track will be straight.</td>
</tr>
</tbody>
</table>
Chapter 3 PLAY & REMOTE mode
1 PLAY mode

1.1 Preparation

- Please make sure the program is correct using test-run before execute it in play mode.
- Please make sure there is no person or obstacle in the movement area of robot.

1.2 Program call

Return to the program list interface (resource manager).

Move cursor by <1><△> at the program users want to execute. For example, execute 555555 program, the example program in the chapter 2 6.1 welding teach program.

Click <Open> button, open the program edit interface, as follows:

```
1  MOVJ TI=55.0% PL=9 TOOL=1
2  MOVJ TI=55.0% PL=9 TOOL=1
3  MOVJ TI=55.0% PL=9 TOOL=1
4  ABORTMOD(0)
5  MOVJ PL=100.0% TI=0 PL=0 TOOL=1
6  ABORTMOD(0)
7  MOVJ TI=55.0% PL=0 TOOL=1
8  MOVJ TI=55.0% PL=9 TOOL=1
```

Move the cursor to the beginning of the program (first line).

1.3 Program execution

1.3.1 START

1. Switch mode to play mode ( "PLAY" ). Then “play mode” will show in status bar.

2. Choose the proper operation method: single line execution, single program execution, cyclic execution.

Notice

It’s suggested to use single-line mode for first-time execution, which is beneficial for deal with the problem. Please pay attention that program will be executed slowly in single line mode. After the single line operation is successful, choose single program execution. After the single program operation is successful, choose cyclic execution and robot starts to work.

3. Choose proper operation speed. Click multiplier logo, and the status bar will show: Speed05%
It’s suggested the speed is slow at first. Then accelerate the speed after the first successful operation.

4. After the previous preparation, press button, and program will start operation according to the position (点位), move, and logic in teach mode. The operation interface is shown in below figure:

![Image of operation interface]

1.3.2 Pause (stop)

**Notice**

The pause and stop is the same status in this system, i.e. after pressing the stop button, system will stay at the stop (pause) status. In this status, speed can be modified, and operation mode can also be switched.

Press button, program can continue. Switching to the teach mode will result in quit of program.

- During the execution of the program, if users want to suspend or stop the execution, press button, then system decelerate the execution of program and the movement of robot till stop. If stop the program by this way, the related internal status, output prods, counter, variables will maintain. After start again, press button, and program will be executed as normal. This way is high recommended.
To ensure the security, it’s suggested to click button for several times or press it for a longer time and to observe the info area to show: program stops, please continue after reset (程序停止，请复位后运行！).

- When program is executed in the single line mode, after the execution of the last line, system will decelerate and stop the program ansd robot. But robot is in still status rather than stop status. Users should click button to stop the program.

- Switching to teach mode or play mode will stop the program forcibly. In this way, system will cut off the pulse, power off the supply, start brake, which may cause the lash (冲击) on the robot. So it’s not suggested to switch mode during program execution.

1.3.3 Modify speed, switch operation mode.

- Modify speed:

In pause (stop) status, click button and coordination button<+> and <-> to modify the speed multiplier. Status bar will show.
Description: When program is executed in single line mode and the current line has been executed and robot is static, users should click button to stop the program, then modify the speed multiplier by multiplier logo and <+> and <-> button.

- **Switch the operation mode:**

  1. In cyclic operation mode, click the correspondent button and can switch to the single line operation mode.

  2. In pause (stop) status, click operation logo and coordination button <+> and <-> to switch and operation mode.

  3. In play or remote mode, if program is executed in cyclic mode, if M216 auxiliay relay is effective, program will switch directly to single program operation mode. After the single program is executed and stop, system will reset the M216 relay into ineffective, and program operation mode is reset to cyclic mode. This function is mainly used to stop the cyclic program after one single execution, which is beneficial for stop the robot at a certain status.

Notes: When this function is needed, PLC need modifying as follows:

- **Switch the operation mode during execution:**

  In play mode, if users want to switch operation mode when program is being executed, it’s needed to press the stop button to stop the execution. Then switch to the desired mode (teach mode or remote mode).

1.3.4 **Continue after stop**

- **Operation mode is still play mode**

  In this mode, press the stop button, program will stop and robot will decelerate the execution and move. Program-related internal status, output ports, counters, variables are all kept.
Press or button, program will start again and be executed continuously.

- Operation mode changed to teach mode.

1. Related parameters (Integrator authority): <Parameters setting>-<3 operation parameter>

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>parameter</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation parameters</td>
<td>18 Initial cursor position in cyclic execution mode.</td>
<td>0</td>
<td>Current line: When operation mode is changed from teach mode to play and remote mode in cyclic mode, cursor will locate at the current line as in the teach mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>Beginning of program: When operation mode is changed from teach mode to play and remote mode in cyclic mode, cursor will locate at the beginning of the program.</td>
</tr>
</tbody>
</table>

2. Strat process with different parameters

When <operation parameter>- “18 Initial cursor position in cyclic execution mode” is zero:
Cyclic execution mode: switch to play mode, and click the button, program will be executed from the current line where cursor is located in teach mode, as shown in the following example:

<table>
<thead>
<tr>
<th>File operation</th>
<th>Edit</th>
<th>Para setting</th>
<th>Monitoring</th>
<th>Run prepare</th>
<th>Command</th>
<th>Technical</th>
<th>PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MOVJ VJ=10.0% PL=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 MOVJ VJ=10.0% PL=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 MOVY VL=100.0 MW/S PL=5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 ARGSTAR(#1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 ARGEND(#1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 MOVY VL=100.0 MW/S PL=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 WAVEEND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 ARGEND()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 MOVJ VJ=10.0% PL=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When \(<operation parameter>-"18\) Initial cursor position in cyclic execution mode" is 1:

Cyclic execution mode: switch to play mode, cursor will jump to first line of the program. After clicking the button, program will be executed from the start of the program as shown in follows:
1.3.5 Emergency stop

warning

1. In automatic operation, if abnormal is found, users should press the emergency stop button as soon as possible.
2. After the emergency stop, the status of robot may be abnormal. So pay careful attention when system alert after reset the machine.

When robot is operated in play mode, stop the program while program is executed. Restart of robot should be done as the following steps:

Firstly, check the robot: if tool clamps is abnormal? Can the program be executed again? (The number in the following figure starts from 2)

1. Swirl to loosen the emergency stop button
2. Press R button, reset the current alert info
3. Click servo power button, power the servo
4. Decelerate the speed, switch to single line operation mode
5. Press start button for several times, judge if program can be executed normally.
6. After confirm no abnormal, accelerate the speed and change to cyclic operation mode.
7. Click start button, robot start to work.
2 remote mode

2.1 Remote operation mode.

The remote operation mode refers to: control the move and stop of the robot from the position away from the robot teach box. This operation mode is mainly used in following situations: remote control of several robots, working position is far away from operator, etc.

After switching to remote mode, the following buttons is ineffective:
run, reverse run, program stop buttons on the teach box.

Details can be found in "CRP-S40 S89 remote function description".

2.1.1 preparation

1 related parameters (integrator authority): <parameters setting>-<3 operation parameters>

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>parameter</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation parameter</td>
<td>External IO confirm time</td>
<td>1000</td>
<td>This parameter is the lasting time of remote/appoint signal. Remote/appoint signal is effective only in the complete form: uphill edge, uphill lasting time, downhill edge. Appoint stop signal and timely response is not controlled by this parameter.</td>
</tr>
</tbody>
</table>

1. Connection of lines can refer to "CRP-S40 S80 remote function description".
2. Edit of operation program

It’s necessary to edit the working program in teach mode, and to test the correctness of program in play mode.

Notice: The program should end with RET if users want to execute it in remote mode, as shown in follows:

```
1 G1000
2 W001 V=1500, CMM/S PL=0
3 W002 V=1500, CMM/S PL=0
4 W003 V=1200, CMM/S PL=0
5 W004 V=1000, CMM/S PL=0
6 W005 V=1200, CMM/S PL=0
7 W006 V=1500, CMM/S PL=0
8 RET
```

3. Working conditions preparation of robot

Check the robot clamps is ready, the used product is reasonable. In the test-run process, robot should work with all clamps and products to test if program, product and clamps can work properly.

Notes: When robot is going to interact with other robots or external equipments, make sure to properly deal with each signal in logic, connection, timeliness, etc. Otherwise, interaction may be abnormal, which may result into equipment damage and personal injury.

2.1.2 Program call

1. Choose remote mode
Click <technical>-<other>-<remote>-<confirm> to open the following window. Click submenu <remote/appoint> switch button, choose [remote choose], and will turn into.

2. Edit the remote working program

Click sub-menu <next page> to enter remote program edit interface, as shown in below figure:
Type the program name users want to operate in remote mode. Click <quit> to quit the remote setting. If the typed program name is wrong or the program doesn’t exist, info area will show: program name wrong, please type the name again after check.

2.1.3 Remote operation

2.1.3.1 Remote start

Switch to remote mode using mode switch;

Power the servo by pressing servo power button, and open the remote operation program simultaneously.

Press the start button, hold the button and release the button. At this time, the system will start to execute the working program.

Notice: When system is collecting the remote start signal, it’s necessary to collect three conditions including uphill edge, voltage duration, and downhill edge. (这三个术语不太会翻，上升沿 下降沿 电平保持时间，或者翻成 rising edge, voltage duration, trailing edge)

2.1.3.2 Remote speed adjust

Only after stop the program can users adjust the speed. So when users want to adjust the speed, press the remote stop button to stop the robot, and then adjust the operation speed on the teach box.

2.1.3.3 Remote pause/stop

Press the remote stop button, then program will stop.

Notice: There may be delay after pressing the remote stop button until the stop of robot movement. So if emergency occurs, press the emergency button first, rather stop button.

2.1.3.4 Remote restart after stop

After remote stop, users can adjust the robot clamps, or switch to teach mode to teach the robot.

When users want to restart, move the cursor to the program line users want to start, then press the remote operation button, hold it and then release. In this way, program will be executed from the line where cursor is located.

2.1.3.5 Remote reset and restart

When robot has alert, robot will stop the movement. At this time, users can press the remote reset button to reset the alert status. Or press the “R” button on the teach box to reset the alert status.

Restart after reset:
High-level alert will clear the power status on the servo. When restart, users need to power the servo by remote servo power button, then press the remote operation button to start the program.

2.2 Reservation operation

Reservation operation is to preform the functions of programs on every tool in the appointed order, by the start button on every tool.

When switch to reservation operation, the following buttons will be ineffective:
<operation>, <reverse operation>, <program stop> buttons on the teach box.
<EX-RUN>, <EX-HLAT> buttons in remote ports.

Details can refer to “CRP-S40, S80 appointment operaion instructions”.

2.2.1 Preparation

1. Related parameters (Integrator authority): <Parameters setting>-<3 operaion parameters>

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>parameter</th>
<th>value</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation parameter</td>
<td>External IO confirm time</td>
<td>1000</td>
<td>This parameter is the lasting time of remote/appoint signal. Remote/appoint signal is effective only in the complete form: uphill edge, uphill lasting time, downhill edge. Appoint stop signal and timely response is not</td>
</tr>
</tbody>
</table>
1. The connection of wire can refer to “CRP-S40, S80 appointment operaion instructions”

2. Edit of operation program

It’s necessary to edit the working program in teach mode, and to test the correctness of program in play mode.

Notice: The program should end with RET if users want to execute it in remote mode, as shown in follows:

```
1   [F] =10000
2   WOVL VL=1500, OMR/S PL=0
3   WOVL VL=1500, OMR/S PL=0
4   WOVL VL=1200, OMR/S PL=0
5   WOVL VL=1000, OMR/S PL=0
6   WOVL VL=1200, OMR/S PL=0
7   WOVL VL=1500, OMR/S PL=0
8   RET
9
```

3. Working conditions preparation of robot

Check the robot clamps is ready, the used product is reasonable. In the test-run process, robot should work with all clamps and products to test if program, product and clamps can work properly.

Notes: When robot is going to interact with other robots or external equipments, make sure to properly deal with each signal in logic, connection, timeliness (时效性), etc. Otherwise, interaction may be abnormal, which may result into equipment damage and personal injury.

2.2.2 Program call

1. Choose remote mode
Click <technical>- <other>- <remote>-<confirm> to open the following window. Click submenu <remote/appoint> switch button, choose [remote choose], and will turn ... will turn into.

2. Edit the reservation operation working program

Click sub-menu <next page> to enter reservation program edit interface, as shown in below figure:
In the above interface, type the correspondent program name in every station name box, for example, station 1 is correspondent to program test1, station 2 is correspondent to program test 2, and station 3 is correspondent to program test 3.

Then move cursor to the station name bar that users want to open. Clicking <choose/cancel> button will switch the indicating signal light. When the light shows , it means the station reservation is open. When the light shows , it means the station reservation is closed.

After the setting, click <quit> to quit the setting. When quitting, if typed program name is not exists or has error, system will prompt: file name error, please type the name again after checking. Then system will clear the wrong program name, and close the reservation funtions of this station.

2.2.3 Reservation operation

2.2.3.1 Reservation start

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The effectiveness of M33X and M34X of correspondent station is required for the start of reservation operation.</td>
</tr>
<tr>
<td>2. During the reservation operation, when program is stopped, system will switch to teach mode. If users close the opened reservation program, then all the reservation status will be cancelled. If users don’t close the opened reservation program, all the reservation status will be cancelled except the current program.</td>
</tr>
</tbody>
</table>

Switch to remote mode using mode switch;

Power the servo by pressing servo power button.

After the preparation of station 1, press the station 1 reservation start button, and hold it for a certain time period. Then release the button. At this time, the Station 1 program is starting to operate.

After the preparation of station 2, press the station 2 reservation start button, and hold it for a certain time period. Then release the button. At this time, if robot is operating the other station program, the Station 2 will be in queue status.

After the preparation of station 3, press the station 3 reservation start button, and hold it for a certain time period. Then release the button. At this time, if robot is operating the other station program, the Station 2 will be in queue status.
Attention

When system collects the reservation start signal, it’s necessary to collect three conditions, including rising edge, voltage duration time, and trailing edge.

2.2.3.2 Cancel of queue reservation

When users want to cancel the reservation of one station which is in queue status, it’s needed to press the reservation start button. Hold it and release. At this time, the reservation status will be cancelled of this station.

If users want to set reservation again, press the reservation start button. Hold it and release. At this time, the station will be in queue status.

2.2.3.2 Speed adjust in reservation operation

Only after stop the program can users adjust the speed. So when users want to adjust the speed, press the remote stop button to stop the robot, and then adjust the operation speed on the teach box.

2.1.3.3 Pause/stop in Remote

Press the remote stop button, then program will stop.

Notice: There may be delay after pressing the remote stop button until the stop of robot movement. So if emergency occurs, press the emergency button first, rather stop button.

2.1.3.4 Restart after stop in reservation mode

After reservation stop, users can adjust the robot clamps, or switch to teach mode to teach the robot.
Notice: When closing the opened reservation program, all the reservation in queue status will be cleaned.

After switching to teach mode, when users want to restart, move the cursor to the program line users want to start, and switch to remote mode. Then press the reservation operation button, hold it and then release. In this way, program will be executed from the line where cursor is located. After the finish of this station, system will move on to the following station in queue.

<table>
<thead>
<tr>
<th>Attention</th>
</tr>
</thead>
<tbody>
<tr>
<td>If program is closed in teach mode, the reservation status will be cleaned. In this situation, restart is similar to first-time reservation start.</td>
</tr>
</tbody>
</table>

2.1.3.5 Reset and restart in reservation mode

When robot has alert, robot will stop the movement. At this time, users can press the remote reset button to reset the alert status. Or press the “R” button on the teach box to reset the alert status.

Restart after reset:

**High-level alert (高等级报警)** will clear the power status on the servo. When restart, users need to power the servo by remote servo power button, then press the remote operation button to start the program.
Chapter 4 File operation
File operation is mainly used for the interaction between system and U disk. For example, software update, parameters backup, data import/export, and etc. After the commissioning of new equipment, it’s suggested to backup the parameters in case of unexpected need. Meanwhile, if users is confronted with problems, users can also backup the system data, and send the complete data (5 files, 1 folder) to our company to analyze and deal with the problem.

In this chapter, "" refer to U disk symbol.

Pre-requisites:
1. One recognizable, formatted U disk with the storage more than 20M.
2. When it’s needed to read the system, the file route and name is correct.

1 Save files into U disk
1.1 Save parameters into U disk

Insert the U disk into the USB connector on the main case. Click <file operation>-<1 save files into U disk>-<1 save para to U disk>-<confirm>.

The system parameters will be saved in the root dictionary on U disk (\:para.txt), ans info area will prompt: Parameters file has been saved in U disk. If you want to pull the U disk out, please uninstall the U disk.

Warning
System parameters are the key parameters of the equipment. Please don’t modify the backup data, otherwise data error may occur, which may cause incident.

If U disk is not inserted or not recognized, info area will prompt: **Faiuer of parameters file save, please check if U disk is inserted.**

1.2 Save users’ program to U disk

Insert the U disk into the USB connector on the main case. Click <file operation>-<1 save files into U disk>-<2 save program to U disk>-<confirm>. The following window will prompt:

[Copy progress bar]

If there has been ROBOT folder, the following window will prompt:

[Whether overwrite the current folder] Using cursor to select [Yes to All] and confirm.

The users’ program will be saved in the root dictionary on U disk (\*:\ ROBOT), and info area will prompt: **ROBOT folder has been saved in U disk. If you want to pull the U disk out, please uninstall the U disk.**

**warning**

There are many data related with system in ROBOT folder. Please don’t change or delete these data without knowing their meanings.

If U disk is not inserted or not recognized, info area will prompt: **Faiuer of users’ program save, please check if U disk is inserted.**

1.3 Save user PLC to U disk

Insert the U disk into the USB connector on the main case. Click <file operation>-<1 save files into U disk>-<3 save users PLC to U disk>-<confirm>.

147
The users’ PLC will be saved in the root dictionary on U disk (\:plc.plc plc.lad), and info area will prompt: plc.lad folder has been saved in U disk. If you want to pull the U disk out, please uninstall the U disk.

If U disk is not inserted or not recognized, info area will prompt: Failure of plc.lad folder save, please check if U disk is inserted.

### 1.4 Save system PLC to U disk

Insert the U disk into the USB connector on the main case. Click <file operation>-<1 save files into U disk>-<4 save system PLC to U disk>-<confirm>.

The system PLC will be saved in the root dictionary on U disk (\:system.plc system.lad), and info area will prompt: system.lad folder has been saved in U disk. If you want to pull the U disk out, please uninstall the U disk.

If U disk is not inserted or not recognized, info area will prompt: Failure of plc.lad folder save, please check if U disk is inserted.

### 1.5 One key system backup

One key backup can backup all the users’ data in the system to U disk. When users want to restore the data, users can use the <one key read system> function to read the system.

Or when users are confronted with obstacles or problems, users can backup the complete data (including 5 files and 1 folder, see more in the following notice) and send these data to our company. We can restore the users’ system situation using the complete data, and help users to deal with the obstacles or problems.

Insert the U disk into the USB connector on the main case. Click <file operation>-<1 save files into U disk>-<5 backup system>-<confirm>. The following window will prompt:

![Copying...]

[Copy progress bar]

If there has been ROBOT folder, the following window will prompt:
[Whether overwrite the current folder] Using cursor to select [Yes to All] and confirm.

After the complete of copy progress bar, the related files will be saved in the root dictionary on U disk (*:\), and info area will prompt: Parameters file has been saved in U disk. If you want to pull the U disk out, please uninstall the U disk.

ROBOT folder has been saved in U disk. If you want to pull the U disk out, please uninstall the U disk.

Plc.lad folder has been saved in U disk. If you want to pull the U disk out, please uninstall the U disk.

System.lad folder has been saved in U disk. If you want to pull the U disk out, please uninstall the U disk.

If U disk is not inserted or not recognized, info area will prompt: Failure of system.lad folder save, please check if U disk is inserted. Failure of system.plc folder save, please check if U disk is inserted.

**Attention**

1. It’s highly suggested to backup the system after the commissioning by equipment sailor.

2. After one key backup, there are 5 files and 1 folder, as shown in below:

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROBOT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>para</td>
<td>3 KB</td>
<td></td>
</tr>
<tr>
<td>plc.lad</td>
<td>7 KB</td>
<td></td>
</tr>
<tr>
<td>plc.plc</td>
<td>13 KB</td>
<td></td>
</tr>
<tr>
<td>system.lad</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Import from U disk

2.1 Import parameters to system

Insert the U disk into the USB connector on the main case. Click <file operation>-<2 import from U disk>-<1 import para>-<confirm>.

The files in the root dictionary on U disk (*:\para.txt) will be imported into the system, and overwrite the previous parameters. And info area will prompt: Import parameters succeed.
**Warning**

1. System parameters are equipment key parameters. Please be cautious, for the wrong imported parameters will result into abnormal operation of equipment and cause damage further!

2. The zero position parameters are also in parameters files. Please make the imported data are accordent with the equipment. Otherwise, the zero position of robot will be changed. So please be cautious!

If U disk is not inserted or not recognized, info area will prompt: *Please check if U disk is inserted or has been uninstalled.*

If the parameters route is incorrect or files don’t exist, info area will prompt: *Failure of parameters import, please check the files and U disk.*

### 2.2 Import user program into system

Insert the U disk into the USB connector on the main case. Click `<file operation>`-<2 import from U disk>-<2 import user program>-<confirm>. The following window will prompt:
[Whether overwrite the current folder] Using cursor to select [Yes to All] and confirm. The following progress bar will prompt:

![Copy progress bar]

When the progress bar is finished and disappear, the folder in the root dictionary (\*:\robot) will be imported in to the system, and overwrite the files with the same name. And info area will prompt: import users program succeed. And the following window will prompt simultaneously:

![User program import completed]

Click <confirm> button to restart the system.

Users cannot pull the U disk out until the presence of the above interface. After the finish of system restart, the user program import is completed.

If U disk is not inserted or not recognized, info area will prompt: Please check if U disk is inserted or has been uninstalled.
If the parameters route is incorrect (\robot) or files don’t exist, info area will prompt:

**Failure of parameters import, please check the files and U disk.**

2.3 Import users PLC into system

Insert the U disk into the USB connector on the main case. Click <file operation>-<2 import from U disk>-<3 import user PLC>-<confirm>. The info area will prompt: **pcl.lad has been copied to the system. System will restart.** And the following window will prompt simultaneously:

![Warning message](image1)

Click <confirm> button, and system will restart. The following steps are the same as the steps in [2.2 import user program into system].

If U disk is not inserted or not recognized, info area will prompt: **Please check if U disk is inserted or has been uninstalled.**

If the parameters route is incorrect (\plc.plc plc.lad) or files don’t exist, info area will prompt: **Incomplete user PLC files, plc.plc plc.lad are needed.**

2.4 Import system PLC to the system

Insert the U disk into the USB connector on the main case. Click <file operation>-<2 import from U disk>-<4 import system PLC>-<confirm>. The info area will prompt: **system.lad has been copied to the system. System will restart.** And the following window will prompt simultaneously:

![Warning message](image2)

Click <confirm> button, and system will restart. The following steps are the same as the steps in [2.2 import user program into system].

If U disk is not inserted or not recognized, info area will prompt: **Please check if U disk is inserted or has been uninstalled.**

If the parameters route is incorrect (\system.plc system.lad) or files don’t exist, info area will prompt: **Incomplete system PLC files, system.plc system.lad are needed.**
2.5 One key restore system

One key restore system can restore the backup data from the U disk by <one key backup system> function to the system. One key restore system is mainly used to restore the system data.

Insert the U disk into the USB connector on the main case. Click <file operation>-<2 import from U disk>-<5 restore system>-<confirm>. The following window will prompt:

![Screenshot of the Confirm Folder Replace window]

[Whether overwrite the current folder] Using cursor to select [Yes to All] and confirm. The following progress bar will prompt:

![Screenshot of the Copying window]

After the [copy progress bar] is finished and disappear, the related files and folders will be imported into the system and overwrite the files with the same name. Info area will prompt: Import of parameters files succeed. Import of user program folder succeed. Plc.lad folder has been copied into the system, and system will restart. Plc.plc folder has been copied into the system, and system will restart. System.lad folder has been copied into the system, and system will restart. System.plc folder has been copied into the system, and system will restart.

The following window will prompt simultaneously:

![Screenshot of the XPESvr window]

Click <confirm> to restart the system.
Users cannot pull the U disk out until the presence of the above interface. After the finish of system restart, one key restore is completed.

If U disk is not inserted or not recognized, info area will prompt: **Please check if U disk is inserted or has been uninstallled.**

If the files in U disk is not complete, info area will prompt: **Several items import succeed, and several items failed.**

<table>
<thead>
<tr>
<th>Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The files and folders in U disk have strict requirement of route. If these files and folders have wrong route, they cannot be imported.</td>
</tr>
<tr>
<td>2. Some unnecessary files can be deleted. When one key restoring the system, files and folders in the correspondent route will be imported. Others will not be imported.</td>
</tr>
<tr>
<td>3. Please be cautious about the treat of U disk files. If treat is improper, failure of data import may result, and damage may occur.</td>
</tr>
<tr>
<td>4. Users mustn’t modify the Parameter files (para.txt) and import them.</td>
</tr>
</tbody>
</table>

3. **Software upgrade**

   **Preparation**

   Prepare a empty U disk to copy the update software (as shown in below figure) and the files need updating into the root dictionary of U disk.
Notice: UpdateCRP.exe file is need to locate in the root dictionary of U disk, which will copy all the upgrated files into root dictionary of U disk.

Insert the U disk into the USB connector in system main case.

Press "file operation" in main menu, move cursor to "3 software update", as shown in below figure:

 Press confirm button on the teach box, and the following window will prompt:

press "1": forced stop
After 9s, the following window will prompt automatically

Press "2": software upgrade
Press "2": next step

After the progress bar shows 100%,
press "2" to next step
CRP软件升级

软件升级

软件升级成功。
点击<完成>返回主页面，选择进入下一项操作。
或主页面选择<系统重启>，结束[CRP软件升级]操作。

成功！
升级包路径：E:\

完成 2 >

Press "2": next step

CRP软件升级

重要提示
1. 升级操作的版本有可能造成不兼容，破坏数据，请谨慎选择1。
2. 升级过程，直到系统重启完成，切勿切断电源。
3. 为了安全起见，建议先备份当前系统再升级软件。
4. 升级路径无CRP.exe文件，<软件升级>按钮无效。
5. 升级路径无开机界面文件，<升级开机画面>按钮无效。

<table>
<thead>
<tr>
<th>升级版本</th>
<th>未知</th>
<th>目前系统版本</th>
<th>未知</th>
</tr>
</thead>
<tbody>
<tr>
<td>日期：未知</td>
<td>版本：未知</td>
<td>日期：未知</td>
<td>版本：未知</td>
</tr>
</tbody>
</table>

1. 软件备份 2. 软件升级 3. 升级 开机画面 5. 系统重启

Press "5": system restart

CRP软件升级

本操作将自动启动系统，系统关闭到重新启动正常期间，严禁断电和取下U盘。

1 取消 2 重启

158
5. Please don’t cut off the power or pull out the disk until the normal system starting interface is shown:

<table>
<thead>
<tr>
<th>File operation</th>
<th>Date</th>
<th>Size</th>
<th>PLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>2016-9-7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARCLINE</td>
<td>2016-10-22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>call</td>
<td>2016-9-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xx</td>
<td>2016-10-21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>视觉抓取</td>
<td>2016-9-5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Run</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-26</td>
<td></td>
<td>Can not open system plc instruction list doc...</td>
</tr>
<tr>
<td>10-26</td>
<td></td>
<td>Can not open plc ladder diagram file</td>
</tr>
<tr>
<td>10-26</td>
<td></td>
<td>System initialization complete</td>
</tr>
</tbody>
</table>

6. After the system restart, check if the version upgraded into 16****(Modified**--**) in “monitor” -> “software info” interface:
Notice

*Power mustn’t be cut off during the restart process.*

4 Save notes to

Notes files store the notes of system internal relay, timer, IO port, variables, which is useful to help users to understand the functions when checking the related info. Users can outport the correspondent files, modify the files, and import them.

Format of notes files: number + “,” + notes + “;” + “enter” . Please pay attention to the comma, semicolon, and enter symbol.

The following will be an example of modification of IO input notes file InNotes.txt:

Open the InNotes.txt using the notepad software:
We can modify the notes of X06 into 防护门开 (attention: don’t change the format) as follows:

Then save, and import this file into the system according to the steps in【5.1 Import IO input notes】.

Click system <monitor>-<4 IO port>-<1 general input port monitor>-<confirm>, and open the input monitor interface as follows:

Then we can see the modified notes of X06.

Warning
Notes files have strict format requirement. So please modify the notes files according the format. Otherwise system cannot recognize them.

4.1 Save IO input notes to U disk

Insert the U disk into the USB connector on the main case. Click<file operation>-<4 save notes to U disk>-<1 save IO input notes>-<confirm>.
System will send the notes files to U disk (*:InNotes.txt). Info area will prompt: file copied to U disk, please uninstall the U disk before pulling the U disk out.

If U disk is not inserted or not recognized, info area will prompt: Please check if U disk is inserted or has been uninstalled.

4.2 Save IO output notes to U disk

Insert the U disk into the USB connector on the main case. Click <file operation>-<4 save notes to U disk>-<2 save IO output notes>-<confirm>.

System will send the notes files to U disk (*:OutNotes.txt). Info area will prompt: file copied to U disk, please uninstall the U disk before pulling the U disk out.

If U disk is not inserted or not recognized, info area will prompt: Please check if U disk is inserted or has been uninstalled.

4.3 Save analog notes to U disk

Insert the U disk into the USB connector on the main case. Click <file operation>-<4 save notes to U disk>-<3 save analog notes>-<confirm>. 
System will send the notes files to U disk (\AOutNotes.txt). Info area will prompt: file copied to U disk, please uninstall the U disk before pulling the U disk out.

If U disk is not inserted or not recognized, info area will prompt: Please check if U disk is inserted or has been uninstalled.

4.4 Save M relay notes to U disk

Insert the U disk into the USB connector on the main case. Click <file operation> - <4 save notes to U disk> - <4 save M relay notes> - <confirm>.

System will send the notes files to U disk (\MNotes.txt). Info area will prompt: file copied to U disk, please uninstall the U disk before pulling the U disk out.

If U disk is not inserted or not recognized, info area will prompt: Please check if U disk is inserted or has been uninstalled.

4.5 Save timer notes to U disk

Insert the U disk into the USB connector on the main case. Click <file operation> - <4 save notes to U disk> - <5 save timer notes> - <confirm>.

System will send the notes files to U disk (\TimeNotes.txt). Info area will prompt: file copied to U disk, please uninstall the U disk before pulling the U disk out.

If U disk is not inserted or not recognized, info area will prompt: Please check if U disk is inserted or has been uninstalled.

4.6 Save counter notes to U disk

Insert the U disk into the USB connector on the main case. Click <file operation> - <4 save notes to U disk> - <6 save counter notes> - <confirm>.

System will send the notes files to U disk (\CountNotes.txt). Info area will prompt: file copied to U disk, please uninstall the U disk before pulling the U disk out.

If U disk is not inserted or not recognized, info area will prompt: Please check if U disk is inserted or has been uninstalled.

4.7 Save GP parameters to U disk

Insert the U disk into the USB connector on the main case. Click <file operation> - <4 save notes to U disk> - <7 save GP para notes> - <confirm>.

System will send the notes files to U disk (\GPNotes.txt). Info area will prompt: file copied to U disk, please uninstall the U disk before pulling the U disk out.
If U disk is not inserted or not recognized, info area will prompt: **Please check if U disk is inserted or has been uninstalled.**

4.8 Save GI parameters to U disk

Insert the U disk into the USB connector on the main case. Click<file operation>-<4 save notes to U disk>-<8 save GI para notes>-<confirm>.

System will send the notes files to U disk (*:\GINotes.txt) . Info area will prompt: **file copied to U disk, please uninstall the U disk before pulling the U disk out.**

If U disk is not inserted or not recognized, info area will prompt: **Please check if U disk is inserted or has been uninstalled.**

4.9 Save GD parameters to U disk

Insert the U disk into the USB connector on the main case. Click<file operation>-<4 save notes to U disk>-<9 save GD para notes>-<confirm>.

System will send the notes files to U disk (*:\GDNotes.txt) . Info area will prompt: **file copied to U disk, please uninstall the U disk before pulling the U disk out.**

If U disk is not inserted or not recognized, info area will prompt: **Please check if U disk is inserted or has been uninstalled.**

5 Import notes from U disk

Import notes function will import notes files on U disk to system. The save notes function can refer to **【4 save notes to U disk】**.

5.1 Import IO input notes to system

Insert the U disk into the USB connector on the main case. Click<file operation>-<5 import notes from U disk>-<1 import IO input notes>-<confirm>. The following overwrite window will prompt:
[whether to overwrite the files] click <yes> to overwrite or <no> to cancel.

System will import the notes files from U disk (\CountNotes.txt) to the system. Info area will prompt: file imported from U disk.

If U disk is not inserted or not recognized or file doesn’t exist, info area will prompt: Import from U disk to system failed, please check if U disk is inserted or files exist.

5.2 Import IO output notes to system
5.3 Import analog notes to system
5.4 Import M relay notes to system
5.5 Import timer notes to system
5.6 Import counter notes to system
5.7 Import GP para notes to system
5.8 Import GI para notes to system
5.9 Import GD para notes to system

Steps are similar to 【5.1 import IO input notes to system】, except the file name and route:

<table>
<thead>
<tr>
<th>Notes name</th>
<th>File name and route</th>
<th>Notes name</th>
<th>File name and route</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO output notes</td>
<td>*:\OutNotes.txt</td>
<td>Analog notes</td>
<td>*:\AOutNotes.txt</td>
</tr>
<tr>
<td>M relay notes</td>
<td>*:\MNotes.txt</td>
<td>Timer notes</td>
<td>*:\TimeNotes.txt</td>
</tr>
<tr>
<td>Counter notes</td>
<td>*:\CountNotes.txt</td>
<td>GP para notes</td>
<td>*:\GPNotes.txt</td>
</tr>
<tr>
<td>GI para notes</td>
<td>*:\GINotes.txt</td>
<td>GD para notes</td>
<td>*:\GDNotes.txt</td>
</tr>
</tbody>
</table>

6 Uninstallation of U disk

Warning
1. For the data consistency, file completeness, and system stability, it’s highly recommended to operate as the following steps before pulling the U disk out.

2. Pulling out the U disk forcibly is not allowed when files is being imported or uninstallization hasn’t completed.

When users want to pull the U disk out, click <file operation>-<eject U disk>-<confirm>. And system will uninstallize the U disk and info area will show: uninstallization of U disk succeed.