/**********
****
* Hardware version: CoreV4D2, BASE-BOARDV61, IO-BOARDV6.2;
* Update content: Cancel G04 command, cancel G24 command,
Increase G00 command, immediate value of absolute value, variable; incremental immediate value, variable
Add 2 formats (angle and radians) Trigonometric operation instructions SIN, COS, TAN, ASIN, ASOS, ATAN
****
**********/

1. Serial port instruction
1.1 serial port mode

Serial port parameters: baud rate 115200; data bit 8; stop bit 1.
Instruction format:
The control commands are hexadecimal 0X10 (idle mode), 0X11 (file mode), 0X12 (zero return mode), 0X13 (run mode), 0X14 (debug mode), 0X15 (reset mode); 0X05 is the interrogation mode;
The running instructions all start with 'GXX' and end with a carriage return;
The teaching instruction ends with a carriage return and a new line;
In the system version of 180521 and above, when _flagSoftRst=1 is set in the parameter file, the serial port 1 can send 0X18 to output the output port o14 in any case. According to this function, the e14 relay control NRST restart (emergency stop).

0X05: == query mode ==
The control card is in any working mode (except file mode), and the serial port sends 0X05 to ask the current working mode of the control card. The return value is 0X10, 0X12, 0X13, 0X14, 0X15.

0X10: == enter idle mode ==
When the control card is in idle mode (0X10): it is in idle mode.
In file mode, 0X10 is valid after the file operation instruction is run.
In the zero return mode, the robot stops moving (send 0X30) and sends 0X10 to take effect.
In the running mode, the robot stops moving (send 0X30) and sends 0X10 to take effect.
In reset mode, the reset completes the brake and exits to idle mode. It is in idle mode when it is turned on.

0X11: == enter file mode ==
Download the running file and select “Send File” after sending 0X11. Among them, the first act in the file
"wr=ST" is the run file;
"WR=INI" is a parameter file;
"WR=oq" is a zero return file;
"WR=T_0" is the default run file, equivalent to "wr=ST";

"WR=T_1" is the run 1 file;
"WR=T_2" is the run 2 file;
...
"WR=T_120" is the run 120 file;
The second behavior file name; the third behavior file length, the length is the file size except the first three lines.
After reading the running file and sending 0X11, send "WR=R,##" ("##" is the length of the read content) to receive the running file.
After reading the parameter file, send "WR=P,##" ("##" is the length of the read content) after sending 0X11 to receive the parameter file.
After reading the parameter file, after sending 0X11, send "WR=Q,##" ("##" is the length of the read content) to receive the zero return file.
The read parameter file can receive the run 0 file after sending 0X11 and sending "WR=H_0,##" ("##" is the read content length).
After reading the parameter file, send "WR=H_1,##" ("##" is the length of the read content) after sending 0X11 to receive the run 1 file.
...
After reading the parameter file, after sending 0X11, send "WR=H_120,##" ("##" is the read content length) to receive the run 120 file.

0X12: == Motor zero return mode ==
After entering the zero return mode, the system automatically runs the file content and the instructions sent by the serial port in real time. Pause is 0X30, continue to 0X12, exit is 0X10
Instructions can be obtained from the run file and can also be obtained from the serial port.
Automatically exit idle mode after running the contents of the file

0X13: == enter running mode ==
After entering the running mode, the system automatically runs the file content and the instructions sent by the serial port in real time. The pause is 0X30, the continuation is 0X13, and the exit is 0X10.
Instructions can be obtained from the run file and can also be obtained from the serial port.
After running the contents of the file, it automatically runs from the beginning.

0X14: == enter debug mode ==
Send 0X05 as the inquiry mode.
Send 0X10 as the exit mode.
Send "0" to stop and send the angle of each axis of the machine.
Send "SVON=1" for servo enable.
Send "SVON=0" to disable the servo.
Send "SVBM=1" to the servo without holding the brake.
Send "SVBM=0" as the servo brake.
Send "CLR=ST_PUL" to clear the position of the control card.
Send "CLR=SV_ALM" to clear the drive alarm.
Send "J#+" to the #axis angle plus motion (#=1, 2, 3, 4, 5, 6).

Send "J#-" to the ##axis angle minus motion.
Send "J#0" to stop at ##axis angle.
Send "J#<" to repeat the motion for the #axis angle (@0° ~ -90°).
Send "J#>" to repeat the motion for the #axis angle (@ 0° ~ 90°).
Send "P VE=####" to change the running speed to #### pulses/sec.
Send "P AC=####" to change the acceleration to #### pulses/square sec.
Send "P DE=####" to change the deceleration to #### pulses/square sec.
Send "G00 J#=####,...." to get the robot joints to the appropriate angle.
Send "G07 **=####" to specify parameters for changing the robot.
TX+: Cartesian coordinates X+;
TX-: Cartesian coordinates X-;
TY+: Cartesian coordinates Y+;
TY-: Cartesian coordinates Y-;
TZ+: Cartesian coordinates Z+;
TZ-: Cartesian coordinates Z-;
TA+: Cartesian coordinates A+;
TA-: Cartesian coordinates A-;
TB+: Cartesian coordinates B+;
TB-: Cartesian coordinates B-;
TC+: Cartesian coordinates C+;
TC-: Cartesian coordinates C-;
P CYCP1=800000: The J1 axis is subdivided into 800,000 pulses per week
....
P J1=24000 Assignment to J1
....
P=VE Query VE value There are VE, AC, DE, CYCP1,...,J1,...,H0,D1,H2...

0X15: == enter reset mode ==
Return the machine to zero.

2. Button control instructions
When starting up, press the "Start" button to run the robot to run mode.
In the operating mode, at runtime, press the "Stop" button to pause the machine.
In the operating mode, when stopped, press the "Start" button and the machine continues to run.
In the operation mode, when stopping, press the "Stop" button and then press the "Start" button, the machine resets and exits the idle mode.

Instruction description:
//========== ==========
G0n=GGN such as G06 O=P1.1 is equivalent to GG6 O=P1.1
Point-to-point command G00, G20
G00 J1=0 J2=0 J3=-90 J4=0 J5=-90 J6=0 //with acceleration and deceleration absolute joint

position command
G00 J1=VXX J2=0 J3=VXX J4=VXX J5=-90 J6=0 //With acceleration and deceleration Absolute joint position command
G00 ... J2'4 J3'VXX ... //with acceleration and deceleration relative joint position command
G00 ... J2'4 J3'VXX J4=VXX J5=-90... //J2 is
G20 X=300 Y=100 Z=500 A=0 B=180 C=0 D=0 //with acceleration and deceleration
G20 X=300 Y=VXX Z=VXX A=VXX B=180 C=VXX D=0 //with acceleration and deceleration
G30 EQ01 J1=UXXXX, J2=UXXXX, J3=UXXXX Read memory contents G30 EQ01 J1=U30100 J2=U30120 J3=U30140 J4=U30160
G30 EQ02 U3060=XX.XX Write memory contents G30 EQ02 U30500=10
G30 EQ03 ASK=UXXXX Query memory contents
G30 EQ04 Vxx=UXXXXXX
G30 EQ05 UXXXXXX=Vxx
G40 X=300 Y=100 Z=500 A=0 B=180 C=0 D=0 //No acceleration/deceleration

Linear command G01 G21
G01 J1=0 J2=0 J3=-90 J4=0 J5=-90 J6=0
G21 X=300 Y=100 Z=500 A=0 B=180 C=0 D=0
G21 X=VXX Y=VXX Z=500 A=0 B=VXX C=0 D=0
G41 X=300 Y=100 Z=500 A=0 B=180 C=0 D=0 //with acceleration and deceleration
G41 X=300 Y=VXX Z=VXX A=0 B=180 C=0 D=0 //with acceleration and deceleration

Arc command G02 G03 G04 G22 G23 G06 DEGREE=ARC (or degree)
When DEGREE=ARC is going to be a three-point arc

Current point: G21 X=200 Y=0 Z=200 A=-180 B=150 C=0 D=0
Second point: G22 X=300 Y=100 Z=200 A=-180 B=150 C=90 D=0
Third point: G23 X=400 Y=0 Z=200 A=-180 B=150 C=300 D=0
Arc running: G06 DEGREE=300

//========== ==========
G06 directive
G06 T=500 delay 500 milliseconds
G06 t=VXX Delay VXX content milliseconds (for example: G08 MOV V400=#2000 G06 t=V400 is a delay of 2000 milliseconds)
G06 I=P1.1 Wait for P1 to be high
G06 I=P2.0 Wait for P2 to be low
G06 O=P1.1 makes output port P1 high (different port from input port)
G06 O=P2.0 makes the output port P2 low
G06 SCAN=I Read input port value, stored in V144-V150 unit
G06 SCAN=O Read output port value, stored in V160-V166 unit
G06 SCAN=RTC Read system clock value and store it in V176-V179 unit
G06 DEGREE=ARC robot takes a three-point arc
G06 DEGREE=35.2 The robot walks 35.2 degrees

G06 REPOS=J# J# Axis looks for HOME# sensor from angle increase direction (return to low level)
G06 REPOS=-J# J# axis finds HOME# sensor from angle reduction direction (return to low level)
G06 REPOS=J1 J1 axis finds HOME0 sensor from angle increase direction (return to low level)
G06 REPOS=-J1 J1 axis finds HOME0 sensor from angle reduction direction (return to low level)
G06 REPOS=JH# J# Axis finds HOME# sensor from angle increase direction (return to high level)
G06 REPOS=-JH# J# axis finds HOME# sensor from angle reduction direction (return to high level)
G06 REPOS=JH1 J1 axis finds HOME0 sensor from angle increase direction (return to high level)
G06 REPOS=-JH1 J1 axis finds HOME0 sensor from angle decrease direction (return to high level)
//========== ==========
G07 command
G07 VE=250000 speed is 250,000 pulses per second
G07 AC=250000 Acceleration is 250,000 pulses per second squared
G07 DE=250000 Deceleration is 250,000 pulses per second squared
G07 VPP=250000 Maximum speed 250,000 pulses per second
G07 VP=20 Speed VE is 20% of the highest speed, VE=VPP*VP*0.01
G07 vp=VXX Speed VE is the content of the highest speed VXX, VE=VPP*VXX unit value *0.01
G07 _h0=xx height change
G07 RCM=1 print run command, 0 does not print run command
G07 GCM=1 is output at right angle coordinates when teaching, 0 is joint coordinate output
G07 MARKPOS_HERE X1=0 Y1=0 X2=100 Y2=100 Template MARK point
G07 MARKPOS X1=10 Y1=10 X2=110 Y2=110 Match MARK point
G07 UNIT=1.0 Interpolation accuracy (in mm)
G07 P J#=XXXX calibration J# axis
// ========== ==========
G08 instruction
G08 XXXX: Tags No comments are allowed after the tags. The length of the tags must be less than 15 bytes.
G08 ACALL XXXX Call XXXX label to G08 END
G08 END call ended
G08 AJMP XXXX Jump to XXXX operation
G08 FLTAB = # file jumps to T_ # file to run;

G08 IF VXX> = VXX ACALL XXXX Integer comparison Immediate number is # comparison> =, <=, ==,! =,>, <; (G08 IF VXX> = VXX AJMP XXXX)
G08 IF_ELSE VXX> = VXX? ACALL XXXX: ACALL XXXX Before the true value runs the colon, after the false value runs the colon ACALL is the call and AJMP is the jump (G08 IF_ELSE VXX> = VXX? AMMP XXXX: AJMP XXXX)
G08 IFF floating point comparison
G08 IF_ELSEF floating-point comparison

G08 MOV VXX = # XX Integer transmission (immediate G08 MOV V10 = # 12) (unit G08 MOV V10 = V12)
G08 MOVF VXX = # XX integer transmission (immediate G08 MOVF V10 = # 12.5) (unit G08 MOVF

V10 = V12)
G08 PRINT VXX prints the value of VXX as an integer
G08 PRINTF VXX prints the value of VXX, floating point

G08 INT VXX converts floating point to integer **********
G08 FLOAT VXX converts integer to floating point

G08 ADD VXX = VXX + VXX Integer The immediate value is #
G08 SUBB VXX = VXX-VXX Integer The immediate value is #
G08 MUL VXX = VXX * VXX Integer The immediate value is #
G08 DIV VXX = VXX / VXX Integer The immediate value is #
G08 SQRT VXX = VXX Integer The immediate value is #

G08 ADDF VXX = VXX + VXX Floating point The immediate value is #
G08 SUBBF VXX = VXX-VXX floating point immediate number is #
G08 MULF VXX = VXX * VXX Floating point The immediate number is #
G08 DIVF VXX = VXX / VXX Floating point The immediate value is #
G08 SQRTF VXX = VXX Floating point The immediate value is #

G08 STO program automatically pauses ********** 170817 **********
G08 EXIT exits and enters idle mode ********** 170817 **********
// === ================
G09 instruction
G09 COPYRIGHT Inquiry System YN
G09 COM2 = XXXX COM2 port sends XXXX characters
For example: G09 COM2 = G00 J1 = 10 J2 = 10 J3 = -90 J4 = 0 J5 = 0 J6 = 0
This controller sends instructions to another controller COM1,
G09 COM2 10H Serial port 2 sends 0X10
G09 COM2 12H Serial port 2 sends 0X12
G09 COM2 13H Serial port 2 issues 0X13
G09 COM2 14H Serial port 2 sends 0X14
G09 COM2 15H Serial port 2 sends 0X15
G09 COM2 18H Serial port 2 sends 0X18

G09 ENC .... It is a special instruction to connect with the absolute value servo motor, refer to the zero return file example;

// === ================

VXX: Unit
#XX: Immediate
User arithmetic unit V0-V127, V400-V511
The system has used the computing unit to read the input port value and store it in the

V144-V150 unit.

Read the output port value and store it in V160-V166 unit

Read the system clock value and store it in V176-V179

Sending '0' as stop in running mode or when running G code, it is this instruction when continuing; '.' Is also stop, it is the next instruction when continuing;

STA real-time key value is stored in V180 unit, key value is stored in V183 unit, V183 is cleared by software

STO real-time key value is stored in V181 unit, key value is stored in V184 unit, V184 is cleared by software

RST real-time key value is stored in V182 unit, key value is stored in V185 unit, V185 is cleared by software

V186: Input port i03 or i04 is cancelled by command. V186 = 0 is not enabled (default), V186 = 1 is enabled.

V187: Limit and servo alarm master switch Not enabled when V187 = 1 (default), enabled when V187 = 0

V188: Soft limit stop master switch Disabled when V188 = 1 (default), enabled when V188 = 0

V189: Input signal capture master switch Not enabled when V189 = 1 (default), enabled when V189 = 0

V143: Running pulse width (value range 0 ~ 400, automatically modified)

V142: Pulse width threshold (value range 4 ~ 400, artificial modification)

V141: G41, G06 DGREE smooth value (stop speed, artificial modification)

V140: Speed timing value (automatic change)

Port corresponding register value

(G06 SCAN = I during scanning, V144 ~ V159, V192 ~ V207: light on value is 0, light off value is 1)

i00: p0 v144

i01: p1 v145

i02: p2 v146

i03: p3 v147

i04: p4 v148

i05: p5 v149

i06: p6 v150

i07: p7 v151

i08: P8 v152

i09: P9 v153

i10: P10 v154

i11: P11 v155

i12: P12 v156

i13: P13 v157

i14: P14 v158

i15: P15 v159

HM0: P16 v192
HM1: P17 v193
HM2: P18 v194
HM3: P19 v195
HM4: P20 v196
HM5: P21 v197

LP0: P22 v198
LP1: P23 v199
LP2: P24 v200
LP3: P25 v201
LP4: P26 v202
LP5: P27 v203

ALM0: P28 v204
ALM1: P29 v205
ALM2: P30 v206
ALM3: P31 v207
O00: P0 v160
O01: P1 v161
O02: P2 v162
O03: P3 v163
O04: P4 v164
O05: P5 v165
O06: P6 v166
O07: P7 v167
O08: P8 v168
O09: P9 v169
O10: P10 v170
O11: P11 v171
O12: P12 v172
O13: P13 v173
O14: P14 v174
O15: P15 v175

(V256 ~ V271: value 1 is capture enable, V189 = 0 takes effect)
ENi00: p0 v256
ENi01: p1 v257
ENi02: p2 v258
ENi03: p3 v259
ENi04: p4 v260
ENi05: p5 v261
ENi06: p6 v262

ENi07: p7 v263
ENi08: P8 v264
ENi09: P9 v265
ENi10: P10 v266
ENi11: P11 v267
ENi12: P12 v268
ENi13: P13 v269
ENi14: P14 v270
ENi15: P15 v271

(V288 ~ V303: capture time value, unit is millisecond, need to be cleared by software)
FLAGi00: p0 v288
FLAGi01: p1 v289
FLAGi02: p2 v290
FLAGi03: p3 v291
FLAGi04: p4 v292
FLAGi05: p5 v293
FLAGi06: p6 v294
FLAGi07: p7 v295
FLAGi08: P8 v296
FLAGi09: P9 v297
FLAGi10: P10 v298
FLAGi11: P11 v299
FLAGi12: P12 v300
FLAGi13: P13 v301
FLAGi14: P14 v302
FLAGi15: P15 v303

// == ================
Visual example
G08 CCD2:
G08 MOV V20 = # 0
G08 CCD2A:
G06 T = 401 // RCCD
G08 IF V20 == # 0 AJMP CCD2A
G08 END
When the host computer receives T = 401,
The upper computer sends instructions such as:
G20 X = 300 Y = 200 Z = 270 A = 0 B = 150 C = 0 D = 0 MOV V20 = # 1
G41 X = 300 Y = 200 Z = 270 A = 0 B = 150 C = 0 D = 0 MOV V20 = # 1
// ....
Examples of input signals
1, initialization, enable enable ENi07: p7 v263
G08 CSH:

G08 MOV V189 = # 0
G08 MOV V263 = # 1
G08 END
2.Read the captured FLAGi07: p7 v295 value during the run
G08 INSCAN:
G06 T = 500
G08 IF V295> # 400 ACALL ZHOUO14
G08 MOV V295 = # 0
G08 AJMP INSCAN
G08 END
3, capture the v295 value greater than 400 milliseconds, then call ZHOUO14
G08 ZHOUO14:
G06 O = P14.1
G06 T = 400
G06 O = P14.0
G06 T = 400
G06 O = P14.1
G06 T = 400
G06 O = P14.0
G08 END

// == ================
C language for, while implementation on PFOP

[1] For example:
int V0;
int V1 = 100;
for (V0 = 2, V0 <V1, V0 = V0 + 3)
{....}
...

[1] Implementation on PFOP:
G08 MOV V0 = # 2
G08 MOV V1 = # 100
G08 AJMP FOR1
G08 LABEL1:
G08 ADD V0 = V0 + # 3
....
G08 FOR1:
G08 IF_ELSE V0 <V1? AJMP LABEL1: AJMP LABEL2
G08 LABEL2:
...

[2] For example:

```
int V0;
int V1 = 100;
for (V0 = 2, V0 <V1, V0 = V0 + 3)
{
   ....
   if (V0> 50) {break;}
   ....
}
...
```

[2] Implementation on PFOP:

```
G08 MOV V0 = # 2
G08 MOV V1 = # 100
G08 AJMP FOR1
G08 LABEL1:
G08 ADD V0 = V0 + # 3
....
G08 IF V0> 50 AJMP LABEL2
....
G08 FOR1:
G08 IF_ELSE V0 <V1? AJMP LABEL1: AJMP LABEL2
G08 LABEL2:
...
```

[3] For example:

```
int V0 = 2;
int V1 = 100;
while (V0 <V1)
{
   V0 = V0 + 1
   ....
}
...
```

[3] Implementation on PFOP:

```
G08 MOV V0 = # 2
G08 MOV V1 = # 100
G08 AJMP FOR1
G08 LABEL1:
G08 ADD V0 = V0 + # 1
....
G08 FOR1:
G08 IF_ELSE V0 <V1? AJMP LABEL1: AJMP LABEL2
G08 LABEL2:
...
```

[4] For example:

```
int V0 = 2;
int V1 = 100;
while (V0 <V1)
{
   V0 = V0 + 1
   ....
   if (V0> 50) {break;}
   ....
}
...
```

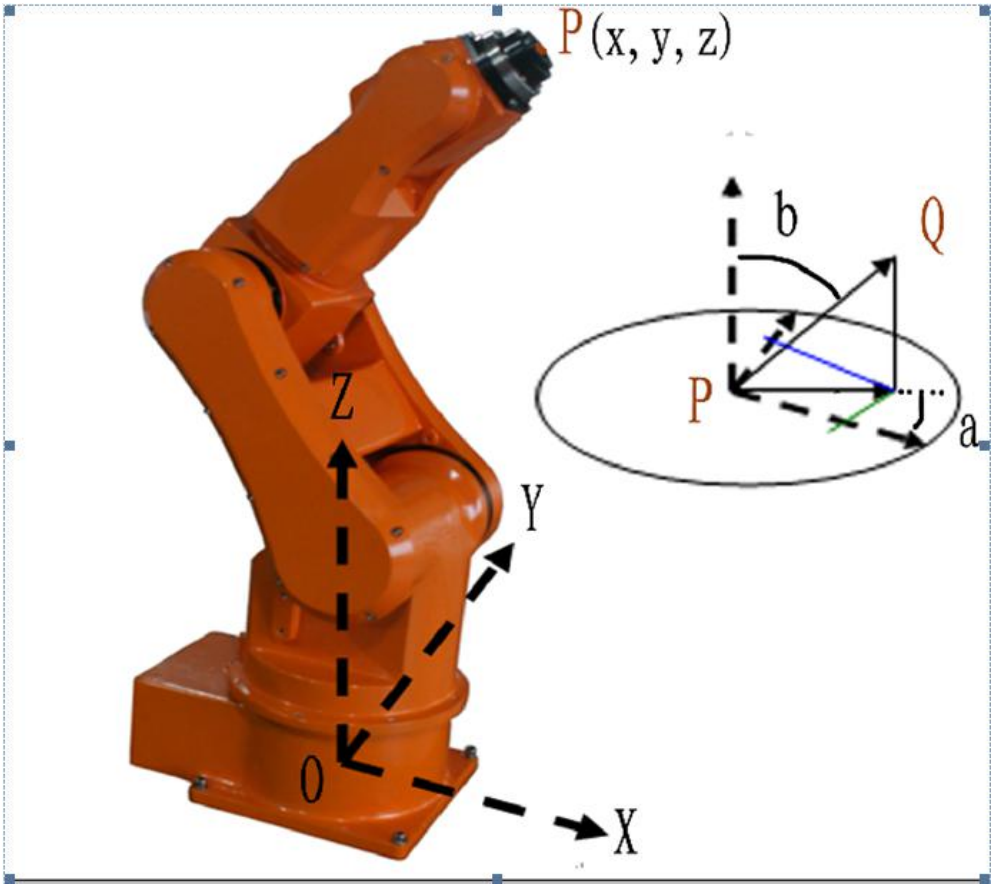
[4] Implementation on PFOP:

```
G08 MOV V0 = # 2
G08 MOV V1 = # 100
G08 AJMP FOR1
G08 LABEL1:
G08 ADD V0 = V0 + # 1
....
G08 IF V0> 50 AJMP LABEL2
....
G08 FOR1:
G08 IF_ELSE V0 <V1? AJMP LABEL1: AJMP LABEL2
G08 LABEL2:
...
```

```
// ================================================== ================
Parameter file
FILE = INI // File path
Parameter.ini // File name
510 // File length (counted from the fourth line to the end)
//// DH_PARAMETER: // DH parameters (unit: mm; h-bit length, d is distance)
_h0 = 203; // can be a dummy value
_d1 = 80;
_h2 = 260;
_d3 = 80;
_h4 = 288;
_h5 = 64.5;
_d6 = 40; // can be a dummy value
//// MOTOR_DIR: // Motor direction
_dir1 = -1;
```

```
_dir2 = 1;
_dir3 = 1;
_dir4 = -1;
_dir5 = 1;
_dir6 = -1;
//// JOINT_PUL: // Organization breakdown
_j1pul = 320000; // The mechanism needs 320000 pulses per circle
_j2pul = 326400;
_j3pul = 326400;
_j4pul = 320000;
_j5pul = 320000;
_j6pul = 320000;
//// VELOCITY:
_ac = 100000.0; // Acceleration
_de = 100000.0; // Deceleration
_vpp = 100000.0; // Full speed
_vp = 10; // speed degree
//// MODE:
_pve = 8.887 // Not used
_getCodeMode = 0; // 0 means teach for joint output, 1 teach for right angle output
_runCodeMode = 1; // 1 is the printing code, 0 is not printing the code
_prmTPUL = 120; // Pulse maximum width adjustment threshold (4 ~ 400) (V142)
_prmVLIM = 2000; // G41 smoothing value (V141)
// REPOS // ======== power-on default angle =======
_rePosJ1 = 0;
_rePosJ2 = 0;
_rePosJ3 = -90;
_rePosJ4 = 0;
_rePosJ5 = -90;
_rePosJ6 = 0;
// IRQ // ========== Interrupt switch ===========
_iIRQ = 1 // Inductor limit enable switch: _iIRQ = 1 enable off, _iIRQ = 0 enable on, default 1;
_sIRQ = 1 // Soft limit enable switch: _sIRQ = 1 enable off, _sIRQ = 0 enable on, default 1;
_flagSoftRst = 1 // Serial port restart controller enable: _flagSoftRst = 1 enable on, _flagSoftRst =
0 enable off, default 0;
_brkMotorEn = 0 // brake enable: _brkMotorEn = 1 enable on, _brkMotorEn = 0 enable off,
default 1;
// LIMIT // ========== software limit ===========
_sLp0 = 175 // Software positive limit angle of the 1st axis
_sLn0 = -175 // Software negative limit angle of the 1st axis
_sLp1 = 90 // The second axis limit angle
_sLn1 = -90 // The second axis limit angle
_sLp2 = 30 // 3rd axis limit angle
_sLn2 = -180 // 3rd axis limit angle
```

_sLp3 = 175 // 4th axis limit angle
_sLn3 = -175 // 4th axis limit angle
_sLp4 = 120 // 5th axis limit angle
_sLn4 = -120 // 5th axis limit angle
_sLp5 = 359 // The 6th axis limit angle
_sLn5 = -359 // The 6th axis limit angle



========= Six-axis robot Cartesian coordinate system ========
P (x, y, z, a, b, c, d)
x: X-axis distance (P-point X-axis component);
y: Y-axis distance (P-point Y-axis component);
z: Z-axis distance (P-point Z-axis component);
a: attitude plane angle;
The angle is the angle between the vector mapped by the attitude vector PQ in the XOY plane and the OX axis;
Angle range: (-180, 180);
b: attitude line face angle;
The angle b is the angle between the attitude vector PQ and the OZ axis;
Angle range: (0, 180);
c: attitude rotation angle;
The angle c is the angle between the vector mapped by the tool vector in the XOY plane and the OX axis;

Angle range: (-360, 360);
d: angle state of each joint in the model;
D is an integer, and no value defaults to 0;
Provisions:
0: The end is mapped to the positive direction, the same direction as D1 of the DH parameter, the third axis angle (-) sign, and the fifth axis angle (-) sign;
1: The end map is in the positive direction, the same direction as D1 of the DH parameter, the third axis angle (-) sign, and the fifth axis angle (+) sign;
2.The end is mapped in the positive direction, the same direction as D1 of the DH parameter, the third axis angle (+) sign, and the fifth axis angle (-) sign;
3: The end map is in the positive direction, the same direction as D1 of the DH parameter, the third axis angle (+) sign, and the fifth axis angle (+) sign;
4: The end is mapped in the negative direction, inverse to D1 of the DH parameter, the third axis angle (-) sign, and the fifth axis angle (-) sign;
5: The end is mapped in the negative direction, inverse to D1 of the DH parameter, the third axis angle (-) sign, and the fifth axis angle (+) sign;
6: The end is mapped in the negative direction, inverse to D1 of the DH parameter, the third axis angle (+) sign, and the fifth axis angle (-) sign;
7: The end is mapped to the negative direction, inverse to D1 of the DH parameter, the third axis angle (+) sign, and the fifth axis angle (+) sign;

Progaram Case

FILE=ST
AM.ST
1719 //Sequence number of the program
code:
G07 VP=20 //Speed set to 20%
G00 J1=0 J2=0 J3=-90 J4=0 J5=-90 J6=0 //Run to door position
G20 X=300 Y=131 Z=55 A=0 B=180 C=0 D=0 //Run to a location
G06 O=P0.1 //Turn on output 00
G20 X=300 Y=131 Z=20 A=0 B=180 C=0 D=0 //Run to a location
G06 O=P0.0 //Turn off output 00
G06 T=500 //Delay 500 ms
G00 J1=0 J2=0 J3=-90 J4=0 J5=-90 J6=0 //Back door position
G08 EXIT //Exit the loop,Program ends