Samkoon

PRO, MAX Series Bus-type PLC

G Code Programming Manual

Version 1.0

2022. 6

www.samkoon-automation.com

Copyright Notice

Document version history

Version Number	Revision Date
Version 1.0	June 3, 2022

CONTENTS

Chapter 1 Overview of G-code Programming

1.1 Overview

The MAX series bus PLC is based on the functions of the PRO series, and adds advanced G code functions that support macro B. This manual will focus on the G code programming instructions supported by the MAX series. The G code programming rules in this manual are based on the standardized RS274/NGC language and are based on code line programming. Each line of programming statement (or block) can include different programming functions.

1.2 Row Rules

- **A legal program statement line cannot exceed 256 characters.**
- **Spaces and tabs can be placed anywhere on each line, and blank lines will be ignored.**
- **Input is not case sensitive, and characters in comments are not important.**

1. Row Delete

Adding / at the beginning of the programming line can comment out the line without affecting the program.

2. Line number

Each line can be marked with a line number at the beginning, starting with the letter N, such as N1234, N56.78. The numbers are only to help programmers identify and have no practical meaning.

3. Character

G code key characters:

Character	Significance
A	A- axis
B	B -axis
C	C -axis
D	Tool compensation number

F	Feed rate
G	G -Code
H	Tool offset index
I	X- axis offset G87
J	Y axis offset G87
K	Z axis offset G87
L	Universal characters G10, M66 and others
M	M Code
N	Line Number
P	G4 Waiting Time
Q	Feed rate G73, G83 cycle
R	Arc radius and loop plane
S	Spindle speed
T	Tool selection
U	U -axis
V	V -axis
W	W -axis
X	X -axis
Y	Y -axis
Z	Z -axis

4.number

The following rules are used to define certain numbers, where the numbers are between 0 and 9.

- A number includes an optional plus or minus sign in front, and may also contain a decimal point. It includes two types: integers and decimals. Only decimal representation is supported, such as +2, -1968, 90.87, -45.39;
- The length of the number is unlimited, but it cannot exceed the maximum line character limit. Note that only 17 valid digits are retained, which is sufficient to meet the usage requirements;
- Unsigned numbers are positive by default.

1.3 Parameters

G codes in this manual support parameter programming, which is called variables in high-level programming languages. There are three ways to express and use parameters. The parameter value type only supports floating point types, no string, Boolean, or integer types. Logical expressions can use Boolean operators AND, OR, XOR; comparison operators EQ, NE, GT, GE, LT, LE, MOD, ROUND; FUP and FIX support integer operations.

There are three syntax forms for expressing parameters:

- **Number expression numbered - #4711**
- **Local naming style local - #<localvalue>**
- **Global variables - #<_globalvalue>**

Scope: The scope of parameters includes the global scope and the local scope in the subroutine. Subroutine parameters and local named variables have local scope. Numeric expression parameters and global named parameters starting from #31 have global scope.

Initialization: Uninitialized global numeric expression parameters and unused subroutine parameters return a value of zero when used in an expression. Uninitialized named parameters are an error when used in an expression.

Properties: Most parameters are read / write and can be assigned values directly. However, for many predefined parameters this does not make sense, so they are read-only - they may appear in expressions but not on the left side of an assignment statement.

Power-off Saving: Volatile parameters lose their values when the system is powered off. All parameters except the numbered parameters in the current persistent range are volatile, and volatile parameters are reset to zero.

Purpose: Digital parameters with digital codes ranging from #31 to #5000. In addition to predefined parameters, global and local parameters are also specified. They can be used to store and convert floating-point values during the entire program execution, such as intermediate results, flags, etc., and are readable and writable; subroutine parameters can be used to save the actual parameters passed to the subroutine; system parameters are read-only.

1.3.1 Digital parameters

Numeric parameters start with # and are represented by integers from 1 to 5602. You can store any number and assign a value using the equal sign =, for example, #3=5.

The value assigned to the same parameter in the same line does not directly affect the subsequent parameter values. For example, the value of #3 is 15. When #3=6 G1 X#3 is executed, the coordinate position of the X axis will be 15, and then the value of #3 will become 6.

symbol takes precedence over other operators. For example, #1+2 means the value of #1 plus 2, not the value of parameter #3 . #[1+2] means the value of parameter #3 . The # character can also be repeated. For example, ##2 means the numeric parameter corresponding to the value of parameter #2.

The specific planning definitions of the system-defined digital parameters are as follows:

#31-#5000: G code user parameters, these parameters are global in the G code file

#5061-#5096: Coordinates of G38 probe results X, Y, Z, A, B, C, U, V, W

#5070: G38 probe result: 1 if successful, 0 if failed

#5161-#5169: Zero point coordinates X, Y, Z, A, B, C, U, V, and W for "G28"

#5181-#5189: Zero coordinates of "G30" X, Y, Z, A, B, C, U, V, and W

#5211-#5219: "G92" offsets X, Y, Z, A, B, C, U, V, and W

#5210: If the current "G92" offset is valid, it is 1, otherwise it is 0

#5211-#5219: G92 offset XYZABCUVW

#5220: Coordinate system 1-9 values corresponding to G54—G59.3

#5221-#5230: Coordinate system 1, G54 is X, Y, Z, A, B, C, U, V, W and R (R represents the rotation angle of XY around the Z axis)

#5241-#5250: Coordinate System 2, G55 for X, Y, Z, A, B, C, U, V, W, and R

#5261-#5270: Coordinate System 3, G56 for X, Y, Z, A, B, C, U, V, W, and R

#5281-#5290: Coordinate System 4, G57 for X, Y, Z, A, B, C, U, V, W, and R

#5301-#5310: Coordinate System 5, G58 for X, Y, Z, A, B, C, U, V, W, and R

#5321-#5330: Coordinate System 6, G59 for X, Y, Z, A, B, C, U, V, W, and R

#5341-#5350: Coordinate System 7, G59.1 for X, Y, Z, A, B, C, U, V, W, and R

#5361-#5370: Coordinate System 8, G59.2 for X, Y, Z, A, B, C, U, V, W, and R

#5381-#5390: Coordinate System 9, G59.3 for X, Y, Z, A, B, C, U, V, W, and R

#5399: M66 check or wait for input execution result

#5400: Tool number

#5401-#5409: Tool offsets X, Y, Z, A, B, C, U, V, and W

#5410: Tool diameter

#5411: Tool rake angle

#5412: Tool clearance

#5413: Tool Axis

#5420-#5428: Relative position coordinates in the current coordinate system X, Y, Z, A, B, C, U, V, W

#5600: Tool change fault indicator

#5601: Tool change fault code

 # 5700-#6567: Registers for interacting with PLC, 32 -bit floating point data, readable and writable, corresponding internal registers are GHWord [0], GHWord [2], GHWord [4]...

1.3.2 Subroutine parameters

Parameters #1-#30 are local parameters for calling subroutines and are not retained when power is off. For details, see the O code programming section.

1.3.3 Named Parameters

Named parameters work similarly to numeric parameters, but are easier to understand and read. The G -code programming in this manual is not case-sensitive, so all parameter names are converted to lowercase characters and spaces and TAB characters are removed, so <param> and <PARAM> represent the same parameter. Named parameters must be accompanied by <> marks.

#<named parameter> is a local named parameter. By default, named parameters are local to the scope in which they are assigned. Local parameters are not accessible outside of the subroutine. Two subroutines can use the same parameter name without worrying about one subroutine overwriting the parameter value in the other.

#<_global named parameter> are global named parameters. They can be accessed from within a subroutine and their values can be set in other programs, but their scope is global.

Example:

To declare a named global variable:

#<_endmill_dia> = 0.049

Refer to the previously declared global variable:

#<_endmill_rad> = [#<_endmill_dia>/2.0]

Mixing Numeric and Named Variables

o100 call [0.0] [0.0] [#<_inside_cutout>-#<_endmill_dia>] [#<_Zcut>] [#<_feedrate>]

1.3.4 Predefined system parameters

Predefined system named parameters can be used to obtain the internal status of the interpreter and machine operation. They are read-only parameters and are defined as follows:

- #<_line> - the current interpreter execution line number
- #<_motion_mode> - current interpreter motion state
- #<_plane> - currently valid plane
- #<_ccomp> - Cutter compensation status
- #<_metric> - G21 returns 1 if valid, otherwise returns 0
- #<_imperial> - G20 returns 1 if it is valid, otherwise it returns 0
- #<_absolute> - G90 returns 1 if it is valid, otherwise it returns 0
- #<_incremental> - Returns 1 if G91 is valid, otherwise returns 0
- #<_inverse_time> - G93 returns 1 if it is valid, otherwise it returns 0
- #<_units_per_minute> - Returns 1 if G94 is valid, otherwise returns 0
- #<_units_per_rev> - Returns 1 if G95 is valid, otherwise returns 0
- #<_coord_system> - Returns the current workpiece coordinate system, 550.0 if G55 is active, 591.0 if G59.1 is active, and so on
- #<_tool_offset> - Returns 1 if G43 is valid, otherwise returns 0
- #<_retract_r_plane> - G98 returns 1 if it is valid, otherwise it returns 0

- #<_retract_old_z> - G99 returns 1 if it is valid, otherwise it returns 0
- #<_spindle_rpm_mode> - Spindle rpm mode G97 returns 1 if valid, otherwise returns 0
- #<_spindle_css_mode> - Spindle css mode G96 returns 1 if it is valid, otherwise it returns 0
- #<_ijk_absolute_mode> - Absolute arc distance mode G90.1 returns 1 if it is valid, otherwise it returns 0
- #<_lathe_diameter_mode> - Returns 1 if G7 is valid for lathe diameter mode, otherwise returns 0
- #<_lathe_radius_mode> - Returns 1 if G8 is in lathe radius mode, otherwise returns 0
- #<_spindle_on> - Returns 1 when the spindle is running M3 or M4, otherwise returns 0
- #<_spindle_cw> - Spindle returns 1 in clockwise direction M3, and returns 0 in reverse direction
- #<_mist> - Cooling mist M7 returns 1, otherwise returns 0
- #<_flood> - Coolant M8 returns 1, otherwise returns 0
- #<_speed_override> - Feedrate override (M48 or M50 P1) is valid, otherwise it returns 0
- #<_feed_override> - Feed override (M48 or M51 P1) is valid, otherwise it returns 0
- #<_adaptive_feed> - Adaptive feed (M52 or M52 P1) is valid, otherwise it returns 0
- #<_feed_hold> - Feed hold (M53 P1) is valid, otherwise it returns 0
- #<_feed> - Feed F value
- #<_rpm> - spindle S value
- #<_x> - current X relative coordinate including offset, equivalent to #5420
- #<_y> - Current Y relative coordinate including offset, equivalent to #5421
- #<_z> - current Z relative coordinate including offset, equivalent to #5422
- #<_a> - current relative coordinates of A including offset, equivalent to #5423
- #<_b> - current B relative coordinates including offset, equivalent to #5424

- #<_c> - current C includes the relative coordinates of the offset, equivalent to #5425
- #<_u> - current U relative coordinate including offset, equivalent to #5426
- #<_v> - current V relative coordinates including offset, equivalent to #5427
- #<_w> - current W relative coordinate including offset, equivalent to #5428
- #<_current_tool> - tool number, equivalent to #5400
- #<_current_pocket> - pocket number

1.4 Expression

An expression is a group of characters starting with a left bracket [and ending with a right bracket] . Between the brackets are numbers, parameter values, mathematical operations, and other expressions. Expressions are evaluated to produce a number. Expressions on a line are evaluated before the line is read and before any operations are performed on the line. An example of an expression is [1 + acos [0] - [#3 ** [4.0/2]]] .

1.5 Binary Operators

Binary operators include four basic mathematical operators: addition +, subtraction -, multiplication *, and division /; three logical operators: OR, XOR, and AND; modulus operator MOD, power operation (**); relational comparison operators: EQ equal to, NE not equal to, GT greater than, GE greater than or equal to, LT less than, and LE less than or equal to.

Binary operators are divided into groups based on their precedence. Logical operations and modulo are performed on any real number, not just integers. Zero is equivalent to logical false, and any non-zero number is equivalent to logical true.

Operator precedence	
Operators	**Priority**
**	Highest
*/MOD	
+ -	
EQ NE GT GE LT LE	
AND OR XOR	lowest

1.6 Floating-point comparisons

Comparing two floating-point numbers for equality involves precision issues. This manual supports that if the difference between the two numbers is within the range of 0.0001, they are considered equal.

1.7 Functions

The functions in the following table are the unary operation functions supported by this manual . The parameter input of trigonometric functions such as COS, SIN and TAN is in angle units, and the return calculation values of inverse trigonometric functions such as ACOS, ASIN and ATAN are also in angle units.

Function Name	Function
ATAN[arg]/[arg]	Four-quadrant inverse tangent function
ABS[arg]	Absolute value function
ACOS[arg]	Inverse cosine function
ASIN[arg]	Inverse sine function

COS[arg]	Cosine function
EXP[arg]	Exponential function
FIX[arg]	Left rounding function
FUP[arg]	Right round function
ROUND[arg]	Rounding function
LN[arg]	Natural logarithm function
SIN[arg]	Sine function
SQRT[arg]	Square root function
TAN[arg]	Tangent function
EXISTS[arg]	Does the parameter exist in the function?

The FIX function rounds to the left so that the calculated value is not greater than the original value, for example, FIX[2.8]=2 and FIX[-2.8]=-3 . The FUP function rounds to the right so that the calculated value is not less than the original value, for example, FUP[2.8]=3 and FUP[-2.8]=-2 . The EXISTS function checks whether a single named parameter exists and returns 1 if it does, and 0 if it does not .

1.8 Repetition

Each line may contain any number of G codes, but two G codes from the same modal group are not allowed to appear on the same line and must follow the G code modal rules. Repeated M codes also need to follow the modal rules.

If you repeat the parameter setting of the same parameter on a line, for example #3=15 #3=6, only the last setting takes effect. Setting the same parameter twice on the same line is not a recommended programming practice, but it is not illegal.

If multiple comments appear on a line, only the last one is used.

1.9 Modality

G code programming can be divided into two categories: modal and non-modal. Modal code instructions will exist until the same modal group instruction changes. For example, if the first line is a G1 linear interpolation command and no other type of motion is specified in the second line, the second line defaults to a G1 linear interpolation command; if a G2 arc command is specified in the second line, the mode is changed to run the arc command. Non-modal code instructions only work on the current line, such as a G4 wait command that only takes effect on the current line.

The modal instructions are divided into the following groups. The modal commands of the same group cannot appear in the same line. The modal commands that appear after the same group will change the current mode. There can be multiple groups of modes in programming, and only one instruction in each group can be effective.

G -code modal group classification:

Modal Group Meaning	Member Words
Non-modal codes (Group 0)	G4, G10 G28, G30, G53 G92, G92.1, G92.2, G92.3,
Motion (Group 1)	G0, G1, G2, G3, G33, G38.n, G73, G76, G80, G81
	G82, G83, G84, G85, G86, G87, G88, G89
Plane selection (Group 2)	G17, G18, G19, G17.1, G18.1, G19.1
Distance Mode (Group 3)	G90, G91
Arc IJK Distance Mode (Group 4)	G90.1, G91.1
Feed Rate Mode (Group 5)	G93, G94, G95
Units (Group 6)	G20, G21
Cutter Diameter Compensation (Group 7)	G40, G41, G42, G41.1, G42.1
Tool Length Offset (Group 8)	G43, G43.1, G49
Canned Cycles Return Mode (Group 10)	G98, G99

Coordinate System (Group 12)	G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3
Control Mode (Group 13)	G61, G61.1, G64
Spindle Speed Mode (Group 14)	G96, G97
Lathe Diameter Mode (Group 15)	G7, G8

M code modal group classification:

Modal Group Meaning	Member Words
Stopping (Group 4)	M0, M1, M2, M30, M60
Spindle (Group 7)	M3, M4, M5
Coolant (Group 8)	(M7 M8 can both be on), M9
Override Switches (Group 9)	M48, M49
User Defined (Group 10)	M100-M199

Notice:

- **When the machine is powered on again and enters the running state, the modes of each group will have initialized default values.**
- **Modal commands of the same group cannot be placed in the same line at the same time, otherwise an alarm will be generated.**
- **Modal groups Group0 and Group1 cannot be edited in the same row because they use the same axis number.**
- **O -code subroutine call code lines are not allowed to have any other programs or comments.**

1.10 Polar coordinate system

Polar coordinates can be used to describe the plane motion of XY coordinates. @n represents distance and ^n represents angle. In absolute position mode, distance and angle are calculated from the absolute zero position of the XY coordinate system, zero angle, and the counterclockwise direction is the positive angle direction. For example, G1 @1 ^90 is equivalent to G1 Y1 . In relative position mode, angle and position are expressed as incremental values.

Example:

```
F100 G1 @.5 ^90
G91 @.5 ^90
@.5 ^90
@.5 ^90
@.5 ^90
G90 G0 X0 Y0 M2
```

1.11 Program comments

G -code program comments can be added to the program line to understand the programmer's intention. You can use parentheses () to write comments in it, or you can use colons; to comment the rest of the line. If the colon; is in parentheses, it is not considered a comment symbol.

Bracket () comments can appear between characters, but not between characters and corresponding numbers. For example, S100(set speed)F200(feed) is correct, but S(speed)100F(feed) is not allowed.

Example:

```
G0 (Rapid to start) X1 Y1
G0 X1 Y1 (Rapid to start; but don't forget the coolant)
M 2; End of program
```

Note that comments cannot be used in O -code program lines.

1.12 NC program files

G code program file contains several lines of G code programs and ends with a program end mark. The program after the program end mark will not take effect. If the program end mark is not used, the percentage sign % can be used to implement the program end function. Use the percentage sign % in the first line and the program end position %, and the program after the second percentage sign % will not take effect.

Please note that the functions of using two percent signs % and the program end sign are different. When the percent sign % is used, the current mode and other states of the program will not change. If other programs are run subsequently, the previous mode may be maintained, resulting in inconsistent programming expectations.

G -code includes approximately 4 GB of internal user memory for storing NC programs.

1.13 G code execution order

The execution order of G codes in the same line is not the order of program positions, but is based on the following rules:

- O code instructions
- Notes
- Set feed mode (G93, G94)
- Set feed speed (F)
- Set spindle speed (S)
- Set Tool (T)
- Change tool (M6) Set tool number (M61)
- Spindle on/off (M3, M4, M5)
- Save state (M70, M73), restore state (M72), invalid state (M71)
- Cooling on/off (M7, M8, M9)

- Ratio on/off (M48, M49, M50, M51, M52, M53)
- Wait (G4)
- Setting the active plane (G17, G18, G19)
- Set length units (G20, G21)
- Tool radius compensation on/off (G40, G41, G42)
- Tool length compensation on/off (G43, G49)
- Coordinate system selection (G54, G55, G56, G57, G58, G59, G59.1, G59.2, G59.3)
- Set path mode (G61, G61.1, G64)
- Set position mode (G90, G91)
- Set the retract mode (G98, G99)
- Return to reference position (G28, G30) Change coordinate system settings (G10), set axis offset (G92, G92.1, G92.2, G94)
- Interpolation motion (G0 to G3, G33, G38.n, G73, G76, G80 to G89)
- Stop (M0, M1, M2, M30, M60)

1.14 G code programming style recommendation

Use floating point precision correctly

3 decimal places when using millimeters and at least 4 decimal places when using inches .

Use consistent spacing

Although G -code programming spaces are ineffective for programming, maintaining consistent space spacing helps with G -code reading and debugging.

Use Center Format Arc

Use center-format arcs (IJK- instead of R-) when possible. This will give you a more consistent description than R- format arcs, especially for angles around 180 or 360 degrees.

Use modal central settings at the start of the program

When the correct execution of a program depends on the modal settings, it is recommended to set them centrally at the beginning of the program.

- **Do not set and use parameters on the same line**

Do not use and set parameters on the same line, updating a variable to a new value, e.g. #1=[#1+#2] is OK.

- **Correct use of program terminators**

Program ends without % or without program end - G code files must end with M2 or M30, or with % .

Chapter 2 G -code Programming

2.1 Introduction

This manual follows the following conventions:

In the G code usage examples, - represents a number and <> represents an optional item.

If L- is used in a G -code example, it usually refers to the number L, or any other letter.

In the G -code examples, axes represents any available axis, i.e. XYZABCUVW .

An optional value will be written as *<L->* .

The actual number is as follows:

Determined number, 4

Expression, *[2+2]*

Parameter value, *#88*

Unary function value, acos [0]

In most cases, if axis numbers are given, they refer to specific destination points.

Unless explicitly described as being in an absolute coordinate system, axis numbers are described in the currently active coordinate system.

If the axis number is optional, any omitted axis will retain its last value. Anything not explicitly described as optional in the G -code prototype is mandatory. The value following the letter is usually given as an explicit number. Unless otherwise specified, the explicit number can be a real value. For example, G10 L2 can be written equally well as G[2*5] L[1+1] . If the value of parameter 100 is 2, G10 L#100 also means the same. If L- is written in a prototype - it will often be called L number, like any other letter.

2.2 G-code quick lookup table

G0	**Fast Movement**
G1	Straight line movement
G2 G3	Arc movement
G4	wait
G5	Cubic spline
G5.1	Quadratic Spline
G5.2	NURBS Curves
G10 L2	Set the workpiece coordinate coefficient value
G10 L20	Set the workpiece coordinate coefficient value
G17 - G19.1	Plane Selection
G20 G21	Unit Settings
G28 - G28.1	Fast forward to predefined position 1
G30 - G30.1	Fast forward to predefined position 2
G38.2 - G38.5	Detecting motion
G40	Tool radius compensation cancel
G41.1 G42.1	Dynamic tool radius compensation
G43.1	Dynamic tool length compensation
G43.2	Incremental tool length compensation
G49	Cancel tool length compensation
G53	Machine tool coordinate system movement
G54-G59.3	Workpiece coordinate system selection
G61 G61.1	Exact path mode

G64	Smooth Path Mode
G90 G91	Path Mode
G90.1 G91.1	Arc Path Mode
G92	Coordinate system offset
G92.1 G92.2	Reset G 92 offset
G92.3	Restore G 92 offset
G93 G94 G95	Feed speed mode

2.3 Detailed description of G code function instructions

2.3.1 G0 Rapid Movement

G0 axes

G0 Positions to the target position in a straight line at the maximum programmed rapid traverse speed. G0 is optional if the current motion mode is G0 . This will result in a coordinated arrival at the destination point at the rapid traverse speed. G0 is mainly used for rapid positioning.

Fast moving speed:

The maximum rapid composite moving speed MAX_VELOCITY can be set in the upper software SamProIDE . During the composite feed process, the maximum rapid composite moving speed can be greater than the MAX_VELOCITY setting of a single axis. At the maximum rapid composite moving speed, the component speed allocated to each axis cannot exceed the MAX_VELOCITY setting of a single axis, otherwise the composite rapid moving speed must be reduced to meet the settings of each axis.

G0 Example

```
G90 (set absolute distance mode)
G0 X1 Y-2.3 (Rapid linear move from current location to X1 Y-2.3)
M2 (end program)
```

If tool compensation is effective, the motion trajectory will not be consistent with the above.

If G53 is in the same line, the motion trajectory will also be inconsistent with the above.

a G0 rapid motion can be circular when changing direction and depends on the path control settings and the maximum acceleration of the axis .

The following will generate a syntax error:

1. The axis number is given, but the position coordinates are not given.

2.3.2 G1 linear movement

G1 axes

G1 performs linear motion at the programmed feedrate F. G1 is optional if the current motion mode is G1 . This results in a coordinated arrival at the destination point at the resulting feedrate .

G1 Example

G90 (Set absolute path mode)

G1 X1.2 Y-3 F10 (Move to coordinate X1.2 Y-3 at speed 10)

Z-2.3 (Move to coordinate Z-2.3 at the same speed 10)

Z1 F25 (Move to coordinate Z1 at speed 25)

M2 (Program end)

Notice:

If tool compensation is effective, the motion trajectory will not be consistent with the above.

If G53 is in the same line, the motion trajectory will also be inconsistent with the above.

The following will generate a syntax error:

1. The current feed speed F mode is not set;

2. An axis number was given, but no position coordinates were given .

2.3.3 G2, G3 arc movement

G2 or G3 axes offsets (center distance mode)

G2 or G3 axes R- (radius mode)

G2 or G3 offsets|R - <P-> (full circle)

A circle or helix is specified using G2 (clockwise) or G3 (counterclockwise) at the current feedrate . The direction (CW, CCW) is from the normal axis of the circular motion, following the right-hand rule.

The normal axis of the arc or the axis of the helix must be parallel to the X, Y or Z axis of the machine coordinate system. Among them, G17 (Z axis, XY plane), G18 (Y axis, XZ plane) or G19 (X axis, YZ plane) . Planes 17.1, 18.1 and 19.1 are not supported at present . If the arc is planar circular, it lies in a plane parallel to the selected plane.

Helix programming should include the axis that is perpendicular to the arc, for example, if in the G17 plane, include the Z axis so that it moves to the programmed point during the circular XY motion.

To program one or more complete arcs, use P to specify the number of complete arc revolutions plus the programmed arc. P must be an integer. If P is not specified, it defaults to P1, which means only a complete or partial arc is generated.

For each P increment greater than 1, an additional full circle is added to the programmed arc. This allows the creation of a multi-turn helix, which is primarily used for milling holes or tapping threads.

If a line of code forms an arc and includes a rotary axis motion, the rotary axis rotates at a constant rate so that the rotary motion starts and ends at the same time as the XYZ motion starts and ends.

If tool compensation is effective, the motion trajectory will be different from the above process. Please refer to the tool compensation section for details.

Whether the arc center is absolute coordinates or relative coordinates is set by G90.1 and G91.1 .

Arcs can be specified in two formats: center-to-center format and radius format.

The following will generate a syntax error:

1. The current feed speed F mode is not set

2. P is not an integer

Center distance format arc:

The arc in center distance format is more accurate than the arc in radius format and is the recommended format for programming. The offset of the arc end point and the offset of the arc center are used together to program an arc smaller than a full circle. The arc end point can be the same as the current position.

When programming arcs, it is recommended to use a precision greater than 4 decimal places (0.0000) for inches and greater than 3 decimal places (0.000) for millimeters, otherwise errors may occur due to rounding.

Incremental arc distance mode:

The arc center offset is a relative offset from the starting position of the arc, and the default is incremental arc distance mode.

For circles less than 360 degrees, one or more axis numbers or one or more offsets must be programmed.

Without axis number or without one or more offsets, a full circle is programmed. The P word defaults to 1 and is optional.

Absolute arc distance mode:

The arc center offset is the absolute distance from the current 0 position of the axis.

For arcs less than 360 degrees, one or more axis numbers and the corresponding offsets must be programmed.

Without axis number or without one or more offsets, a full circle is programmed. The P word defaults

to 1 and is optional.

XY- plane (G17)

G2 or G3 <X- Y- Z- I- J- P->

- ***Z –*** Helix Z- axis height
- ***I –*** X -axis offset
- ***J -*** Y axis offset
- ***P –*** Number of laps

XZ- Plane (G18)

G2 or G3 <X- Z- Y- I- K- P->

- ***Y -*** Helix Y- axis height
- ***I -*** X -axis offset
- ***K -*** Z -axis offset
- ***P -*** Number of laps

YZ- plane (G19)

- ***G2 or G3 <Y- Z- X- J- K- P->***
- ***X -*** Helix X- axis height
- ***J -*** Y- axis offset
- ***K -*** Z -axis offset
- ***P -*** Number of laps

The following will generate a syntax error:

1. The current feed speed F mode is not set

2. Offset not set

3. When the arc is projected onto the programming plane, the difference between the distance from the starting point to the center point and the distance from the end point to the center point exceeds (0.05 inch/0.5 mm) or ((0.0005 inch/0.005mm) and 0.1% of the radius

G2 or G3 axes R- <P->

- *R* – The radius of the arc calculated from the starting point

The method of programming a full circle or nearly a full circle using an arc in radius format is not recommended, because a small change in the end point position will cause a large change in the center position. The rounding error amplification effect during the calculation process will make the error larger.

In the radius format, the coordinates of the arc's endpoint in the selected plane are specified together with the arc's radius. axes are optional, and at least one of the two axes in the selected plane must be included. R is the radius, a positive number indicates that the arc is less than 180 degrees, while a negative number indicates that it is more than 180 degrees. If the arc is a helix, the value of the arc's endpoint on the coordinate axis parallel to the helix axis should also be specified.

The following situations will generate syntax errors:

• Omit two axes for the selected plane

• The end point of the arc is the same as the current point

G2 Example

G17 G2 X 10 Y 15 R 20 Z 5 *(radius format with arc)*

The example above is a clockwise (from the positive Z axis) arc or spiral, with its axis parallel to the Z axis, end point coordinates X=10 Y=15 Z=5, and a radius of 20. If the starting value of Z is 5, it is an arc parallel to the XY plane, otherwise it is a spiral.

2.3.4 G4 Wait

G4 P-

- P – Waiting time, in seconds (s)

P parameter is the time in seconds that all axes remain stationary. The P number is a floating point number, so decimals can be used. G4 does not affect the spindle, coolant, or any I/O .

G4 Example

G4 P0.5 (wait 0.5s and then continue to execute)

The following will generate a syntax error:

- P parameter is not specified or is set to a negative number

2.3.5 G5 cubic spline

G5 X- Y- <I- J-> P- Q-

- I – X increment from the start point to the first control point
- J - Y increment from the start point to the first control point
- P – The X increment from the end point to the second control point
- Q – Y increment from the end point to the second control point

G5 can only generate cubic B- spline curves in the XY plane. P and Q must be included in each G5 command line.

For the first G5 command in a series of G5 commands, both I and J parameters must be specified . For subsequent G5 commands, both I and J must be specified, or neither must be specified. If I and J are not specified, the starting direction of the curve will automatically connect to the ending direction of the previous curve.

The following will generate a syntax error:

- Neither P nor Q is specified
- Specify only one of I and J
- The first G5 command in a series of G5 commands does not specify I or J
- Specifying an axis other than X and Y

- The current effective plane mode is not G17

2.3.6 G5.1 Quadratic Spline

G5.1 X- Y- I- J-

- I – The X- axis increment from the start point to the control point
- J – Y increment from the start point to the control point

G5.1 creates a quadratic B- spline curve in the XY plane only . If you do not specify I or J, it means that the offset of the unspecified axis is zero, so one or both parameters must be specified.

Programming example: Program a parabola from X-2 Y4 to X2 Y4 .

G90 G17

G0 X-2 Y4

G5.1 X2 I2 J-8

The following will produce an error:

- I and J are specified or both specified parameters are 0
- Specify an axis other than the X and Y axes
- The current effective plane mode is not G17

2.3.7 G5.2 G5.3 NURBS curve

G5.2 <P-> <X- Y-> <L->

X- Y- <P->

...

G5.3

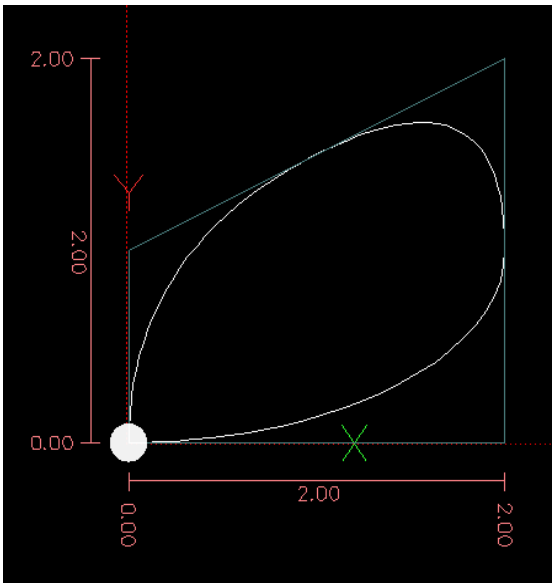
G5.2 is used to open the programming data block that defines the NURBS, and G5.3 is used to close the data block. In the line between G5.2 and G5.3, the NURBS curve control points are determined by their weight coefficients P and parameters L.

the first G5.2 command, use the current coordinates as the first NURBS control point. If you need to set the weight of the first control point, first program G5.2 P- without providing any XY .

The default weight when P is not specified is 1. If L is not specified, the default value is 3 .

G5.2 Example:

```
G0 X 0 Y 0 (rapid move)
F10 (set feed rate)
G5 .2 P 1 L 3
    X 0 Y 1 P 1
    X 2 Y 2 P 1
    X 2 Y 0 P 1
    X 0 Y 0 P 2
G5 .3
; The rapid moves show the same path without the NURBS Block
G0 X 0 Y 1
  X 2 Y 2
  X 2 Y 0
  X 0 Y 0
M2
```



2.3.8 G10 L2 sets the workpiece coordinate coefficient value

G10 L2 P- <axes R->

- P – Workpiece coordinate system number (0-9)
- R – Rotation angle around the Z axis

G10 L2 is used to set the workpiece coordinate coefficient value. Its value will replace the corresponding workpiece coordinate system group value. The workpiece coordinate system value without axis number will not be changed.

P0-P9 represents the selected workpiece coordinate system.

P Value	Coordinate System	G code
0	Active	n/a
1	1	G54
2	2	G55
3	3	G56
4	4	G57
5	5	G58
6	6	G59
7	7	G59.1
8	8	G59.2
9	9	G59.3

The value R is not required, and represents the angle of rotation of the XY axis around the Z axis, which conforms to the right-hand rule.

The axis numbers of all axes are not mandatory.

G91 incremental mode has no effect on G10 L2 .

G10 L2 Pn will not change the current workpiece coordinate system by the current P value, and G54-G59.3 needs to be used to set the current workpiece coordinate system.

The following will produce an error:

- The value range represented by P is not between 0 and 9

G10 L2 Example:

G10 L 2 P 1 X 3.5 Y 17.2

2.3.9 G10 L20 sets the workpiece coordinate coefficient value

G10 L20 P-axes

- *P* – Workpiece coordinate system number (0-9)

The function of G10 L20 is similar to that of G10 L2, but the value set by G10 L20 is an incremental value relative to the workpiece coordinate system, which is different from G10 L20 which is relative to the machine origin.

G10 L20 Example:

G10 L20 P1 X1.5 (set the X axis current location in coordinate system 1 to 1.5)

The following will produce an error:

- P value is not in the range of 0 to 9

2.3.10 G17 - G19.1 Plane Selection

Set the current plane according to the following G codes:

- G17 - XY (Default)
- G18 - ZX

- G19 - YZ
- G17.1 - UV
- G18.1 - WU
- G19.1 - VW

Among them, UV, WU and VW planes do not support arcs. It is recommended to include the above plane selection instructions at the beginning of the G code file, which will affect the G2/G3 arc instructions and G81/G89 instructions.

2.3.11 G20, G21 Unit Settings

- G20 – Length in inches .
- G21 – Length in millimeters .

It is recommended to set a unified length unit at the beginning of the G -code file.

2.3.12 G28, G28.1 Fast forward to predefined position 1

G28 uses the predefined position 1 stored in parameters #5161-#5169 as the coordinate value XYZABCUVW, and G0 quickly moves to this position. The parameter coordinate value represents the absolute coordinate, and the unit is configured in the motion control parameters. If G28.1 does not store data, all axes will return to the machine zero point.

G28 – Fast forward from the current position to the absolute coordinate position predefined by parameters #5161-#5166 .

G28 axes – First fast forward to the position coordinates defined by axes, and then fast forward to the absolute coordinate position defined by axes in parameters #5161-#5166 .

G28.1 – Store the current absolute coordinate position in parameters #5161-#5166 .

G28 Example:

G28 Z2.5 (First fast forward to Z2.5, then fast forward to the absolute Z coordinate defined by parameter #5163)

The following will produce an error:

- Tool compensation is effective

2.3.13 G30, G30.1 Fast forward to predefined position 2

G30, G30.1 and G28, G28.1 have the same function, the only difference is that G28, G28.1 use parameters #5161-#5169, while G30, G30.1 use parameters #5181-#5189 .

2.3.14 G38.n detection movement

G38.n axes

- *G38.2* – Approaching object detection, stop moving after contact signal, generate error signal after failure
- *G38.3* - Approaching object detection, stop movement after contact signal
- *G38.4* – Stay away from object detection, stop moving after losing signal, generate error signal after failure
- *G38.5* - Stay away from object detection and stop moving after losing signal

G38.n can be used to trigger linear probing motion. The probe signal input source must be set before use. If the probe signal is not detected before G 38.2 and G 38.4 reach the end coordinate, an alarm error will be generated. If the probe signal is successfully detected, parameters #5061 to #5069 will save the X, Y, Z, A, B, C, U, V, W coordinate values after probing. If the signal is not successfully detected, these parameter values will be assigned to the programmed coordinates. If the probe is successful, parameter # 5070 will be assigned to 1, otherwise it will be 0.

The following will produce an error:

- The current coordinates are consistent with the programmed coordinates

- No axis coordinate movement
- Tool compensation is effective
- Feed rate F is 0
- The detection signal is effective when the movement starts

2.3.15 G40 tool radius compensation cancellation

- G40 – Tool compensation canceled .

Tool compensation cancellation must be a linear move, and the moving length must be greater than the tool diameter.

When tool compensation is canceled, the cancel tool compensation command can still be executed, but it has no practical significance.

G40 Example:

```
; current location is X1 after finishing cutter compensated move
G40 (turn compensation off)
G0 X1.6 (linear move longer than current cutter diameter)
M2 (end program)
```

The following will produce an error:

- G2/G3 circular interpolation instructions in one line after G40 .
- The distance of linear movement is less than the tool diameter .

2.3. 16 G41.1, G42.1 Dynamic tool radius compensation

G41.1 D- <L-> (left tool compensation)

G42.1 D- <L-> (Right tool compensation)

- *D* – Tool diameter
- *L* – Tool direction

The functions of G41.1 & G42.1 are similar to those of G41 & G42, except that G41.1 & G42.1 can program tool diameter parameters, and the default value of L parameter is 0 if it is not specified.

The following errors will occur:

- YZ plane effective
- L parameter range is not within the range of 0 to 9
- L parameter is used when the XZ plane is not valid
- Repeatedly enable tool radius compensation function

2.3.17 G43.1 Dynamic tool length compensation

G43.1 axes

- G43.1 changes the subsequent motion trajectory by directly changing the tool offset. G43.1 itself does not generate any motion.

G43.1 Example:

```
G90 (set absolute mode)
T1 M6 G43 (load tool 1 and tool length offsets, Z is at machine 0 and DRO shows Z1.500)
G43.1 Z0.250 (offset current tool offset by 0.250, DRO now shows Z1.250)
M2 (end program)
```

The following will produce an error:

- The G43.1 line also includes other motion instructions

2.3.18 G43.2 Incremental Tool Length Compensation

G43.2 H-

- H – Tool No.

G43.2 enables incremental tool length compensation .

G43.2 Example:

G90 (set absolute mode)

T1 M6 (load tool 1)

G43 (or G43 H1 - replace all tool offsets with T1's offset)

G43.2 H10 (also add in T10's tool offset)

M2 (end program)

The following will produce an error:

- H parameter is not specified
- The tool table does not contain the H parameter tool number

2.3.19 G49 cancels tool length compensation

- G49 – Cancel tool length compensation

2.3.20 G53 Machine tool coordinate system movement

G53 axes

G53 performs linear movement in the currently set machine tool coordinate system. G53 is not modal and must be programmed when using the current line .

For example, G53 G0 X0 Y0 Z0 moves the axes to the home position, even if the current machine coordinate system is valid.

G53 Example:

G53 G0 X0 Y0 Z0 (rapid linear movement to the machine origin)

G53 X2 (rapid linear movement to absolute coordinate X2)

The following situations will generate errors:

- using G53, G0 or G1 is not active
- G53 with tool compensation on

2.3.21 G54-G59.3 Workpiece Coordinate System Selection

- G54 – Select workpiece coordinate system 1
- G55 - Select workpiece coordinate system 2
- G56 - Select workpiece coordinate system 3
- G57 - Select workpiece coordinate system 4
- G58 - Select workpiece coordinate system 5
- G59 - Select workpiece coordinate system 6
- G59.1 - Select workpiece coordinate system 7
- G59.2 - Select workpiece coordinate system 8
- G59.3 - Select workpiece coordinate system 9

Select	CS	X	Y	Z	A	B	C	U	V	W	R
G54	1	#5221	#5222	5223	5224	5225	5226	5227	5228	5229	5230
G55	2	#5241	#5242	5243	5244	5245	5246	5247	5248	5249	5250
G56	3	#5261	#5262	5263	5264	5265	5266	5267	5268	5269	5270
G57	4	#5281	#5282	5283	5284	5285	5286	5287	5288	5289	5290
G58	5	#5301	#5302	5303	5304	5305	5306	5307	5308	5309	5310
G59	6	#5321	#5322	5323	5324	5325	5326	5327	5328	5329	5330

G59.1	7	#5341	#5342	5343	5344	5345	5346	5347	5348	5349	5350
G59.2	8	#5361	#5362	5363	5364	5365	5366	5367	5368	5369	5370
G59.3	9	#5381	#5382	5383	5384	5385	5386	5387	5388	5389	5390

The following situations will generate errors:

- The above instructions are used when tool compensation is valid

2.3.22 G61, G61.1 Exact Path Mode

G61- Exact path mode, the movement moves exactly according to the programmed points. The speed will slow down or stop when reaching each programmed point. If two consecutive movements pass through the programmed points exactly, there will be no stop in the middle.

G61.1 - Exact stop mode, the motion will stop at the end point of each programmed segment.

2.3.23 G64 Smooth Path Mode

G64 P-

- P – Maximum motion fitting error

G64 P - fits motion errors with P parameters. If there is no P parameter, the motion will maintain a constant optimal moving speed, and the errors of the moving trajectory and programming points will be ignored.

G64 P - Example:

G64 P0.015 (set path following to be within 0.015 of the actual path)

to specify path control at the beginning of each G -code file.

2.3.24 G90, G91 Path Mode

G90 - Absolute path mode. In absolute path mode, axis numbers X, Y, Z, A, B, C, U, V, W represent the absolute coordinates in the current coordinate system. Special cases have been explained in G80

G89 .

G91- Relative path mode. In relative path mode, the axis number represents the incremental position coordinate relative to the coordinate.

G90 Example:

G90 (set absolute distance mode)

G0 X2.5 (rapid move to coordinate X2.5 including any offsets in effect)

G91 Example:

G91 (set incremental distance mode)

G0 X2.5 (rapid move 2.5 from current position along the X axis)

2.3.25 G90.1, G91.1 Arc Path Mode

G90.1 – Arc IJK offset in absolute path mode . In this mode, when programming G2/3 in the XY plane, the IJ offset represents the absolute coordinate and cannot be omitted during programming. The same applies to the other planes.

G91.1 – Arc IJK offset in relative path mode . The corresponding parameters can be omitted as needed.

2.3.26 G92 Coordinate System Offset

G92 axes

When running to the target point, use G92 to set the current point coordinates as the coordinate system offset. All axis numbers are optional, but at least one axis number must be used. If the programmed axis number is omitted, the offset of the axis is 0 .

When G92 is executed, the origins of all coordinate systems are moved. They are moved so that the value of the current control point in the currently active coordinate system becomes the specified value. The origins of all coordinate systems (G53-G59.3) are offset by the same distance.

G92 uses the values stored in parameters #5211-#5219 as the XYZABCUVW offset values for each axis. The parameter values are absolute machine coordinates in the local machine units specified in the

ini file. When G92 is active, all axes defined in the ini file will be offset. If no axis is entered after G92, the offset for that axis will be zero.

For example, suppose the current point is at X=4 and there is no G92 offset currently active. Then G92 X7 is programmed. This moves all origins -3 in X, which makes the current point X=7 . This -3 is saved in parameter 5211 .

Being in incremental distance mode (G91 instead of G90) has no effect on the effect of G92 . When G92 is called, the G92 offset may already be in effect. If this is the case, the offset is replaced with the new offset so that the current point becomes the specified value.

2.3.27 G92.1, G92.2 Reset G92 offset

G92.1 – Turns off G92 offset and resets parameters # 5211 – # 5219 to 0

G92.2 – turns off G92 offset, parameters # 5211 – # 5219 are still valid

G92.1 only clears the G92 offset. To change the G53-G59.3 offset, use G10 L2 or G10 L20.

2.3.28 G92.3 Restore G92 Offset

- **G92.3** – Sets the G92 offset from current parameter #5211 to #5219

to set axis offsets in one G -code program and use the same offsets in another G -code program. G92 in the first program, which will set parameters #5211 to #5219 . The parameter values will be saved when the first program is exited and restored when the second program is started. Use G92.3 at the start of the second program. This will restore the offsets saved in the first program. Do not use G92.1 in the rest of the first program, or the function will fail.

2.3.29 G93, G94, G95 feed speed mode

G93 Time-limited mode. In time-limited feed mode, the F value indicates that the feed should be completed within 1/F minute. For example, an F value of 2 means that the feed should be completed

within 1/2 minute. When the time-limited mode is effective, each line in the feed of G1, G2/G3 must include the F value. This mode does not affect the movement of G0 .

G94 is the feed per minute mode. In uints / minute feed mode, the feed rate F should move in mm/ min, inch /mm or degree/min, depending on the programming unit used.

G95 is the feed rate per revolution mode. In this mode, the feed rate F should be the unit of the spindle's rotation .

The following will produce an error:

- When the limited time mode is active, there is no F letter for G1, G2 or G3 movements
- switching to G94 or G95, no new feedrate is specified

Chapter 3 M -code Programming

3.1 M code quick lookup table

M2 M30	**End of program**
M3 M4 M5	Spindle control
M7 M8 M9	Cooling control
M62-M65	Digital Output
M66	Input Waiting
M67	Analog synchronous output
M68	Analog output immediately

3.2 Detailed description of M code function instructions

3.2.1 M2, M30 program ends

M2 – End of program

M30 – Pallet change, program ends. Press the Cycle Start button to restart the program from the file.

M2/M30 will affect the program mode:

- Automatic mode will switch to MDI mode
- Axis offsets are set to default (G54)
- The current plane is set to XY plane (G17)
- Set to absolute position mode (G90)
- Feed rate set to units/mm (G94)
- Feedrate override and spindle override are effective (M48)
- Tool compensation off (G40)
- Spindle stop (M5)
- The motion mode returns to feed (G1)
- Cooling off (M9)

The program code after M2/M30 will not be executed. Press the cycle start button to restart the program

from the file. Using the percentage sign % to end the program is inconsistent with the above process and there will be no mode change. Please pay attention to the difference.

3.2.2 M3, M4, M5 spindle control

- **M3** – Spindle starts at S speed clockwise
- **M4** - Spindle counterclockwise S speed start
- **M5** – Stop the spindle

When using M3/M4, it is OK to set the value of S to 0, or when the spindle override is set to 0, the spindle will not rotate. When the subsequent spindle speed is set to a non- zero value, the spindle starts to rotate. If the spindle is already rotating, you can also program using M3/M4 . If the spindle has stopped, you can also program using M5 .

3.2.3 M7, M8, M9 cooling control

- **M7** – Turn on cooling mist
- **M8** – Turn on the coolant
- **M9** – Close M7, M8

The above commands can be used regardless of the current cooldown status.

3.2.4 M62 - M65 digital outputs

- **M62 P-** -Motion control synchronous digital output ON, P represents the digital output channel number.
- **M63 P-** - Motion control synchronous digital output OFF, P represents the digital output channel number.
- **M64 P-** - Immediate digital output ON, P represents the digital output channel number.
- **M65 P-** - Immediately turn off the digital output, where P represents the digital output channel number.

P ranges from 0 to XX, corresponding to XX

The output of M62/M63 command will be at the beginning of the next motion command. If there is no

motion command next, M62/M63 command will not work. Therefore, the best programming method is to program the motion G command immediately after M62/M63 command, such as G0, G1, etc. M 62 /M63 is an IO operation between the previous and next motion commands, which will not affect the fitting smooth action of the two trajectories.

M64/M65 will switch the IO action after the previous motion ends and stops.

3.2.5 M66 input waiting

M66 P- | E- <L-> M66 P- | E- | **R-** <L-> **J- K-**

P- - Set the digital input channel number 0-63

E- - Set analog input channel number 0-63

L- - Set standby mode

Mode Mode 0: Immediate mode – Return value immediately, current input value is read directly from #5399

Mode 1: Rising edge mode – wait for the input signal rising edge event

Mode 2: FALL falling edge mode – wait for the falling edge event of the input signal

Mode 3: HIGH high level mode – waiting for input signal high level mode

Mode 4: LOW low level mode – waiting for input signal low level mode

Q- - specifies the timeout wait event, in seconds (s) . If a timeout occurs, the wait event will be interrupted and the parameter value #5399 will remain unchanged at -1 . If the L value is 0 (ie Mode0), the Q value will be invalid. If the L value is not 0, an error will occur if the Q value is 0 .

Analog input is only allowed to use Mode 0 .

M66 example:

M66 P0 L3 Q5 (wait up to 5 seconds for digital input 0 to turn on)

M66 can be used to wait for external input signals to determine whether to execute the next program.

M66 cannot use the P and E keywords at the same time (i.e. select analog and digital input at the same time). In addition, the timing time cannot be guaranteed to be particularly accurate, so please pay attention to the approximate range of the setting.

3.2.6 M67 analog synchronous output

- **M67 E-Q-**

M67 - Set analog command for synchronous output

E- -Output channel number, range

Q- - Set the analog value. Set it to 0 to turn off the analog value.

The M67 analog synchronous output will occur before the next motion command. If there is no motion command next, the M67 command will not be executed. Therefore, the best programming method is to program the motion G command immediately after the M67 command, such as G0, G1, etc. The M67 command function logic is similar to M62-M63 . M67 is an analog operation between the previous and next motion commands, which will not affect the fitting smoothing action of the two trajectories .

3.2.7 M68 analog output immediately

- **M68 E- Q-**

M68 – Set analog command for immediate output

E- -Output channel number, range

Q- - Set the analog value. Set it to 0 to turn off the analog value.

M68 outputs analog value after the last motion instruction reaches the stop. The working mode of M68 is similar to M64-65 .

Chapter 4 O - Code Programming

O code can provide sequential logic control in NC program. Each O code block has an associated digital label, which follows the O code and is used to uniquely identify the O code block.

Example:

```
O100 sub
( Note that the if-endif blocks use different numbers )
    O110 if [#2 GT 5 ]
        …( some code here)
    O110 endif
    …( some code here)
O100 endsub
```

The following actions are illegal operations:

The same O code number is used for more than one block

Include other code sections in O code lines

Using comments within O code lines

4.1 Subroutine Call

Subroutines range from Oxxx sub to Oxxx endsub . Oxxx sub to Oxxx endsub will not be executed in the NC program until the subroutine is called by Oxxx call .

Subroutine example:

```
O100 sub
    G53 G0 x0 y0 Z0 (rapid move to machine zero point)
O100 endsub
…
O100 call (call subroutine)
M2
```

Inside a subroutine, Oxxx return can be executed. This returns immediately to the calling code, ending the subroutine execution and returning.

Oxxx return example:

```
O100 sub
    O110 if [#2 GT 5] ( determines if parameter #2 is greater than 5)
        O100 return ( If true, return to the top of the subroutine )
    O110 endif

    ( Execute the code where parameter #2 is less than or equal to 5 )
O100 endsub
```

Oxxx calls can take up to 30 optional parameters, which are passed to the subroutine. #1, #2, …, #N . When returning from the subroutine, the values of parameters #1 to #30 (regardless of the number of parameters) are restored to the values before the call. Parameters #1 to #30 are local subroutine calls. Because 1 2 3 will be parsed as the number 123, the parameters must be enclosed in square brackets . The following calls a subroutine with 3 parameters:

Calling example:

```
O200 call [1] [2] [3]
```

Parts of subroutine definitions cannot be nested, and they can only be called after the full definition. They can be called from other functions, and can call themselves recursively if it makes sense, with a maximum subroutine nesting depth of 10. Subroutines have no return value, but can change the values of parameters above #30, and these changes will be visible to the calling code. Subroutines can also change the values of global named parameters.

4.2 Loop

Loop calls include two calling modes: while/ endwhile and do/while . In each calling mode, the loop body will continue to exist until the while loop condition is not satisfied. The do/while mode runs the loop

body first and then determines the execution condition, while the while/ endwhile mode determines the execution condition first and then runs the loop body.

while/ endwhile Examples:

```
( Draw a zigzag shape )
G0 x1 y0 ( move to the starting position )
#1 = 0 ( assigns the value of parameter #1 to 0 )
F25 ( Set feed speed )

O101 while[ #1 LT 10 ]
    G1 x0
    G1 y[ #1/10] x1
    #1 = [#1 +1] ( #1 increases by 1 )
O101 endwhile

M2 ( End program )
```

do/while example:

```
#1 = 0 ( assign parameter #1 to 0)
O100 do
    O110 if [#1 EQ 2 ]
        #1 = 3 ( assign the value of 3 to parameter #1)
        (MSG, #1 is assigned value 3)
        O100 continue ( jump to the beginning of the loop )
    O110 endif
    …
    #1 = [#1 +1] (#1 adds 1 to itself)
O100 while [#1 LT 3]
(MSG, loop complete! )
M2
```

In the while loop, you can use O-break to exit the loop immediately, and use O-continue to jump to the next while condition. If the condition is still met, the loop body will be restarted. If the condition is not met, the loop will be exited. This is the same as the use of break and continue in the loop body in C language .

4.3 Conditional Execution

if conditional statement contains a series of conditions, starting from if and ending with endif . Each condition must start from the same O code in front. At the same time, the elseif and else conditions must be within the range from if to end with endif.

If the if condition is true, the following statements will be executed.

If the judgment condition of if is false, the following elseif judgment conditions will be judged in order until one is judged to be true. If the judgment condition of elseif is true, the following statements of elseif will be executed. If there is no if or elseif judgment condition is true, the following statements of else will be executed.

The above process is similar to the logical judgment of C language.

if endif example:

```
O101 if [#31 EQ 3] ( If parameter #31 is equal to 3, set S 2000)
    S2000
O101 endif
```

if elself else endif Example:

```
O102 if [#2 GT 5 ] (if parameter #2 is greater than 5, set F100)
    F100
O102 elseif [#2 LT 2] ( If parameter #2 is less than 2, set F200)
    F200
O102 else ( If parameter #2 does not meet the above conditions, set F150)
    F150
O102 endif
```

4.4 Repeated Execution

repeat will execute the statements inside repeat/ endrepeat for the specified number of times .

Example:

```
G91 ( Incremental Mode )
O103 repeat [5]
    …
    G0 X1 Y1 ( Move diagonally to the next position )
O103 endrepeat
G90 ( Absolute mode )
```

4.5 Implicit O code

O code may not give the subsequent numbers directly, but may give them implicitly through parameters or calculation expressions.

Example:

```
O[ #101 +2] call
```

4.6 Calling Files

in the NC program is similar to calling a subroutine. The called file must contain sub/ endsub in the file. The file path is set in the PROGRAM_PREFIX or SUBROUTINE_PATH in the ini initialization file . The file name can only contain lowercase letters, numbers, hyphens and underscores. The called file can only contain the definition part of the subroutine.

Example of naming a file:

```
O< myfile > call
```

Example of a numeric name file:

```
o123 call
```

The called subroutine file must contain oxxx sub and endsub .

Example of a subroutine file:

```
( file name myfile.nc)
O< myfile > sub
    ( code here)
O< myfile > endsub
M2
```

Subroutine file call names are all lowercase letters, so o< MyFile > will become o< myfile >

4.7 Subroutines with return values

A subroutine can optionally return a value in an endsub or return expression.

Return example:

```
o123 return [#2 *5]
…
o123 endsub [3*4]
```

The subroutine return value is stored in the <_value> predefined parameter to predict that the value has been returned. The parameter is a global variable and will be cleared to zero the next time the subroutine is called.