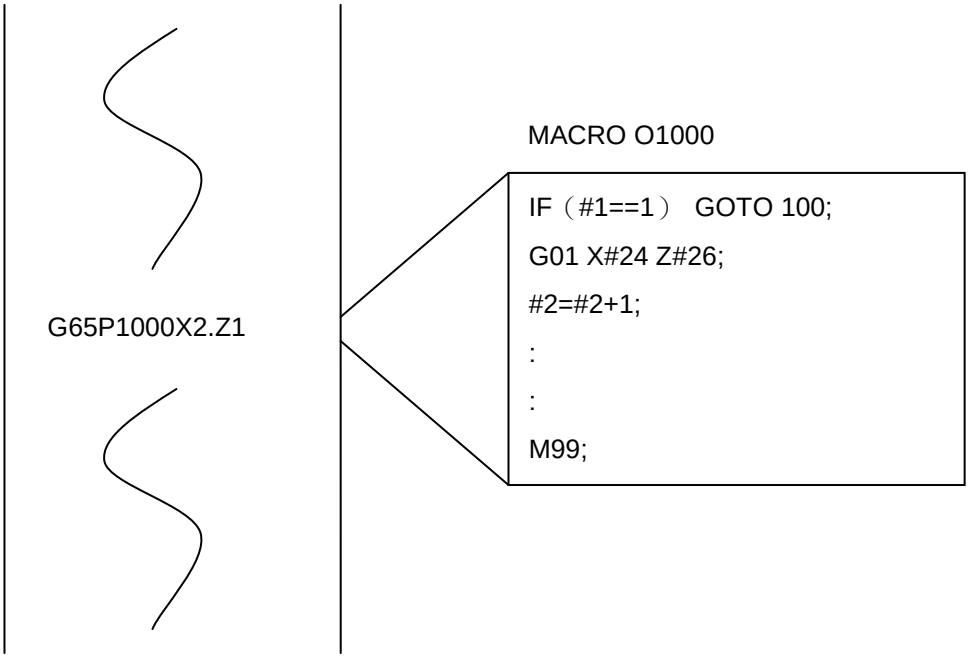
# 5 MACRO

## 5.1 Introduction of Macro

Traditional NC programs have limited functions, such as being unable to operate and have no if functions. Macro commands provide a higher level method of syntax utilization. Functions such as IF, GOTO, functions, variables, etc. are available and bring users more flexibility.

When adopted in systems, in order to let users to call and execute a series of often-used operations with a simple command, users can develop Macro programs to fulfill the need：



### 5.2 MACRO Call

| Command Format | Description | Example |
|---|---|---|
| M98 P_ L_<br>M98 “string” L_<br>M98 “string”P_ L_ | M98 call sub-program<br>L：Repeat times<br>P：4digits sub-program number (O+ P_No)<br>“string”：Appoint string arbitrarily<br>“string”P_：Combination file name(string+4 digits of P_NO) | ● M98 P1 L2;<br>Description：Call O0001 twice.<br>● M98 “HELLO” L2;<br>Description：Call HELLO twice.<br>● M98 “ABC”P1 L2;<br>Description：Call ABC0001 twice. |
| G65 P_ L_ <arguments…><br>G65 “string” L_ <arguments…><br>G65 “string”P_ L_ <arguments…> | G65MACRO single call<br><arguments…>：send to MACRO arguments<br>L：Repeat times<br>P：4digits sub-program number (O+ P_No)<br>“string”：Appoint string arbitrarily<br>“string”P_：Combination file name(string+4 digits of P_NO) | ● G65 P1 L2 A11 B12 C13;<br>Description：<br>Call O0001 twice. Send to arguments cooreponding to #1=11、#2=12、#3=13。<br>● G65 “HELLO” L2 A11 B12 C13;<br>Description：<br>Call HELLO twice. Send to arguments cooreponding to #1=11、#2=12、#3=13。<br>● G65 “ABC”P1 L2 A11 B12 C13;<br>Description：<br>Call ABC0001 twice. Send to arguments cooreponding to #1=11、#2=12、#3=13。 |
| G66 P_ L_ <arguments…><br>G66 “string” L_ <arguments…><br>G66 “string”P_ L_ <arguments…> | G66MACRO mode call<br><arguments…>：send to MACRO arguments<br>L：Repeat times<br>P：4digits sub-program number (O+ P_No)<br>“string”： Appoint string arbitrarily<br>“string”P_：Combination file name(string+4 digits of P_NO) | G66 P0001 L2 A11 B12 C13;<br>X21.Y21.<br>X22. Y22.<br>Description：<br>Call O0001 twice. Send to arguments cooreponding to #1=11、#2=12、#3=13;the following movement single block will call O0001 twice. |
| G_ <arguments…> | G code call MACRO<br><arguments…>：send to MACRO arguments | G22 A11 B12 C13;<br>Description：<br>If directory <maker_macro> has file maker_macro_g22, then system will call this file one time, and send to arguments to correspond to #1=11、#2=12、#3=13。 |
| M_ <arguments…> | M code call MACRO | M23 A11 B12 C13; |

| | | |
|---|---|---|
| | <arguments…>：send to MACRO arguments | Description：<br>If directory < maker_macro > has file maker_macro_m23，, then system will call this file one time, and send to arguments to correspond to #1=11、#2=12、#3=13。 |
| T_ <arguments…> | T code call MACRO<br><arguments…>：send to MACRO arguments | T24 A11 B12 C13;<br>Description：<br>If directory < maker_macro > has file maker_macro_t0, then system will call this file one time, and send to arguments to correspond to #1=11、#2=12、#3=13。 |

| Special Call Macro | Description |
|---|---|
| First time to do cyclestart to call MACRO | Version： Milling machine INT mode**mill_int_Ver03.01.53**<br>If directory<macro>has file sys_func_init1, then system will run this MACRO one time when doing first time cyclestart.<br>If directory <macro_maker> has file maker_func_init1, then system will run this MACRO one time when doing first time cyclestart.<br>If above two files exist at the same, then system will run sys_func_init1 and then maker_func_init1 in order. |
| Every time to do cyclestart to call MACRO | Version： Milling machine INT mode**mill_int_Ver03.01.53**<br>If directory <macro> has file sys_func_startup1, then system will run this MACRO one time when doing every time cyclestart.<br>If directory <macro_maker> has file maker_func_startup1, then system will run this MACRO one time when doing every time cyclestart.<br>If above two files exist at the same, then system will run sys_func_startup1 and then maker_func_init1. |

### 5.3 MACRO return method

M99：Finishing sub-program to go back to main program

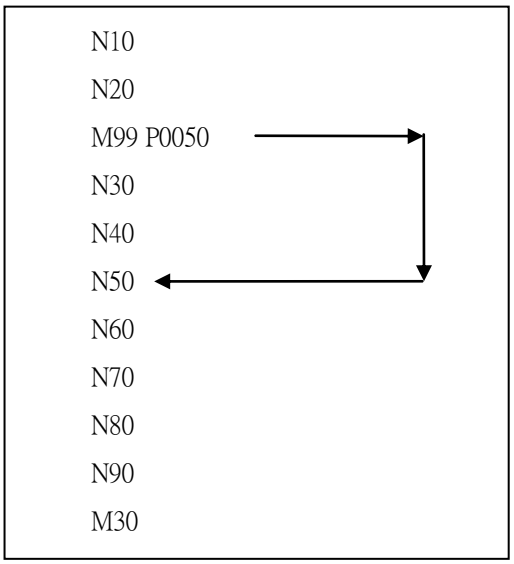
(1.) When executing M99 in the main program, system will go back to the top of program to run again. In the sub-program, you will need to use M99 to do the ending and make it go back to main program.

(2.) Command format：M99 P__ ;

P__：appointed return line number

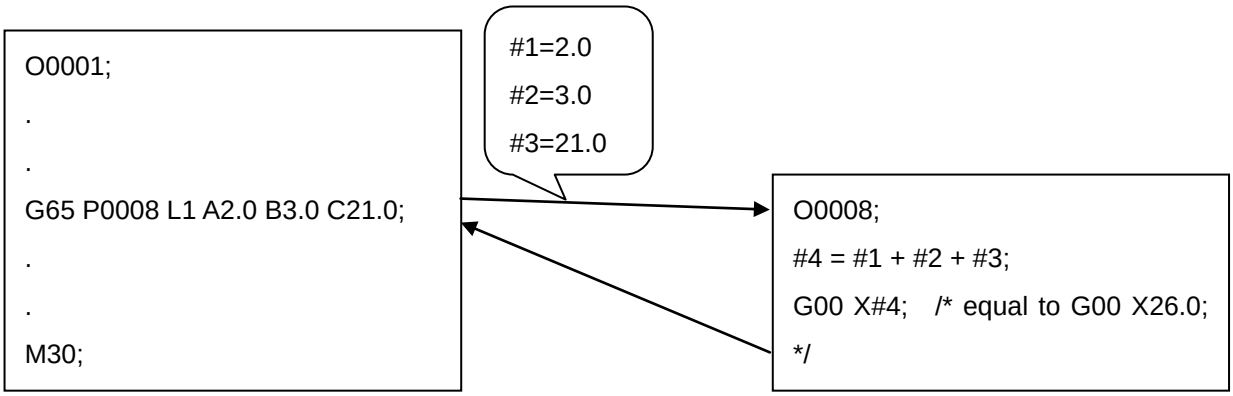
If main program has M99 P_, then system will find the line number that M99 appointed and run；

When using M99 P_at the ending of sub-program, M99 will run from the appointed line number, after sub-program finishes and returnes to main program, as below：


N10
N20
M99 P0050
N30
N40
N50
N60
N70
N80
N90
M30

### 5.4 Send MACRO arguments method

Call MACRO can be via NC address（english letters, expect G、N）to send in arguments, there is no order of them. These arguments has corresponding local variables from MACRO, as below：




Variables corresponding list as below：

| **A~Z** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| NC address | A | B | C | D | E | F | G | H | I | J |
| Local variables | #1 | #2 | #3 | #4 | #5 | #6 | | #8 | #9 | #10 |
| | | | | | | | | | | |
| NC address | K | L | M | N | O | P | Q | R | S | T |
| Local variables | #11 | #12 | #13 | | #15 | #16 | #17 | #18 | #19 | #20 |
| | | | | | | | | | | |
| NC address | U | V | W | X | Y | Z | | | | |
| Local variables | #21 | #22 | #23 | #24 | #25 | #26 | | | | |

### 5.5 MACRO saving directoty and file type description

There are 3 types of system program and MACRO saving directory. System directory is <macro>, machine maker's directory is <maker_macro>, and end-users' directory is <ncfile_1>, as below. Milling machine MACRO file list is at the appendix A.

| Directory | File Type | File Type Example | Description |
|---|---|---|---|
| macro | sys_macro_g## | sys_macro_g35 | G code call system MACRO |
| | sys_macro_m## | sys_macro_m35 | M code call system MACRO |
| | sys_macro_t0 | sys_macro_t0 | All T code call this system MACRO |
| | sys_modal_g## | sys_modal_g81 | System MACRO mode call is INT basic command |
| | sys_single_g## | sys_single_g53 | System MACRO single time call is INT basic command |
| | sys_func_mdi1 | | Inset MACRO manually |
| | sys_func_prog_restart1 | | Program restart to call this system MACRO |
| | sys_func_start_init1 | | When file exists,system will run this MACRO one time at the first time cycle start. |
| | sys_func_startup1 | | When file exists,system will run this MACRO one time in advance |
| maker_macro | maker_macro_g## | maker_macro_g35 | G code call machine maker MACRO |
| | maker_macro_m## | maker_macro_m35 | M code call machine maker MACRO |
| | maker_macro_t0 | maker_macro_t0 | All T code call this machine maker MACRO |
| | maker_func_start_init1 | | When file exists,system will run this MACRO one time at the first time cycle start. |
| | Maker_func_startup1 | | When file exists,system will run this MACRO one time in advance |
| ncfile_1 | o####(4 digits) | o1234 | This file format offer M98 call sub-program，or for G65、G66 to call MACRO |
| | ###...(32 digits) | test1234 | Normal file name, support 32 english letters |

### 5.6 System MACRO, machine maker MACRO and end-user MACRO usage rule

MACRO arguments usage rule

a. If there is GMT call MACRO in single block program, then all words in this line will trun to argument, except below：
   - G、N code
   - GMT call MACRO code

b. If there are 2 and above GMT call MACRO in single block program, then the first showing one will be call MACRO code, the coming one will be treated as arguments.

- When P#50072 MACRO mode call type(0:default,1:single level) setting is 1, system MACRO , machine maker MACRO and end-user MACRO has rules as below：

  a.If “MACRO mode call ”meets”MACRO single time call”, ”MACRO single time call” will be run first.

  b.“ End-user MACROmode call ” meets” system MACRO mode call ”, ” system MACRO mode call ” will be run first.

  c.The same type MACRO mode call in the same level, the following MACRO mode call setting will cover previous setting.

  d.MACRO mode call allows ”the same level, but different type next call” or ”the same type but different level next call”, but do not accept overlap call.

  e.GMT code call MACRO, machine maker MACRO will be run firstly, and then run system MACRO.
  EX：there is maker_macro_g33 and sys_macro_g33, system will run maker_macro_g33.

  f.MACRO mode call only valid in this level.

c. MACRO mode call features：

| | | |
|---|---|---|
| system MACRO mode call | Call timing | Every single block with axis command |
| | Cancel timing | 1. G80 command cancel this setting level<br>2. G code group [1] movement command cancel this setting level<br>3. M99 command cancel this setting level<br>4. M30 command cancel this setting level |
| end-user MACROmode call | Call timing | G code group[1] movement command call |
| | Cancel timing | 1. G67 command cancel this setting level<br>2. M99 command cancel this setting level<br>3. M30 command cancel this setting level |

d. Sub-progrma and MACRO level limitation：

| | |
|---|---|
| System | “System MACRO mode call” and”system MACRO single call”-maximum combination can be 8 levels. |
| | “System sub-program” , “system MACRO mode call ” and ”system MACRO single call” -maximum combination can be 12 levels. |
| User | “End-user MACROmode call” and ” user MACRO single call” -maximum combination can be 4 levels. |
| | “End-user MACROmode call” , “end-user MACROmode call ” and ”user MACRO single call” -maximum combination can be 6 levels. |

e. In the MACRO call program, if there is a GMT code call file with the same name, then it will be treated as a normal GMT code.

# 6 Operand Priority

This chapter is to explain whole series of INT mode.

Version： Milling machine INT mode **mill_int_Ver03.01.01**

## 6.1 Operand Priority

Description as below：

| Operand | Symbol | Priority |
|---|---|---|
| Parentheses | ( ) | 1 |
| Function | xxxx(); | 2 |
| Minus | - | 3 |
| NOT operation | ! | |
| Multiplication | * | 4 |
| Division | / | |
| Addition | + | 5 |
| Subtraction | - | |
| Comparison | >, <, >=, <=, ==, != | 6 |
| AND logic operation | && | 7 |
| OR logic operation | \|\| | 8 |
| XOR logic operation | ^ | 9 |

## 6.2 Mathematic Operation Command

(1). Substitution, =

#i = #j

(2). ADDITION , +

#i = #j + #k

(3). SUBTRATION , -

#i = #j - #k

(4). MULTIPLICATION , *

#i = #j * #k

(5). QUOTIENT , /

#i = #j / #k

(6). PARENRHESIS , （ ）

#i = #j * （ #k + #l ）

### 6.3 Logic Operation Command

(1). AND logic operation, &&

#i = #j && #k

0 for false and non-0 for true in logic operation

(2). OR logic operation, | |

#i = #j | | #k

(3). NOT logic operation, !

#i = ! #j

### 6.4 Compare Command

（1）Greater Than （GT）， >

#i = #j > #k，If #j is greater than #k, then the statement is true, #i=1.

（2）Less Than （LT），<

#i = #j < #k，If #j is less than #k then the statement is true, #i=1.

（3）Greater Than or Equal to （GE）， >=

#i = #j >= #k，If #j is greater than or equal to #k, then the statement is true, #i=1.

（4）Less Than or Equal to （LE），<=

#i = #j <= #k，If #j is less than #k, then the statement is true, #i=1.

（5）Equal，==

#i = #j == #k，If #j is equal to #k, then the statement is true, #i=1.

（6）Not Equal，!=

#i = #j != #k，If #j is not equal to #k, then the statement is true, #i=1.

# 7 Expression

This chapter is to explain whole series of INT mode.

Version： Milling machine INT mode **mill_int_Ver03.01.01**

## 7.1 IF~GOTO

A. Conditioned Jump

Method：IF （ <Conditional express> ） GOTO n

Description：IF <Conditional express> condition is true, then jump to the block numbered “n”, else continue the next block

Example：

```
IF(#1>3) GOTO 1001;
G01 X10.;
…
N1001 G01 X20.;
…
M30;
```

B. Unconditioned Jump

Method：GOTO n

Description：Jump to the block numbered “n” directly.

Example：

```
GOTO 1001;
G01 X10.;
…
N1001 G01 X20.;
…
M30;
```

### 7.2 IF…ELSE

Method：

```
IF(conditions described)
    Data Process;
ELSEIF(conditions described)
    Data Process;
ELSE
    Data Process;
END_IF
```

Description：IF…ELSE select described

Example：

```
#1=2
G0 X0.
IF(#1==1)
   G0 X10.
ELSEIF(#1==2)
   G0 X20.
ELSE
   G0 X30.
END_IF
G0 X40.
M30
```

## 7.3 SELECT

Method：

```
SELECT(Integer or calculating formula)
    CASE Integer：
        Data Process;
    CASE Integer, Integer, Integer, Integer：
        Data Process;
    CASE_ELSE
        Data Process;
END_SELECT
```

Description：SELECT (select described)

Example：

```
#1=2
G0 X0.
SELECT(#1)
    CASE 1:
        G0 X10;
    CASE 2:
        G0 X20;
    CASE 3,4,5,6:
        G0 X30.
    CASE_ELSE
        G0 X99.
END_SELECT
M30
```

### 7.4 FOR

A. Not use INC variables

Method：

```
FOR variables=loop’s initial value or calculating formula TO loop ending value or calculating formula
    Data Process;
END_FOR
```

Description：FOR loop.

Example：

```
G0 X0.
FOR #1=11 TO 23
    G0 X#1;
END_FOR
M0;                          /* X axis final movement position 23mm，#1 final is 24 */
M30
```

B. Use INC variables

Method：

```
FOR variables= loop’s initial value or calculating formula TO loop ending value or calculating formula
STEP loop acccumulated value or calculating formula
    Data Process;
END_FOR
```

Description：FOR loop.

Example：

```
G0 X0.
FOR #1=11 TO 23 STEP 2
    G0 X#1;
END_FOR
M0;                          /* X axis final movement position 23mm，#1 final is 25 */
M30
```

## 7.5 EXIT_FOR

Method：

```
FOR variables=loop’s initial value or calculating formula TO loop’s endingl value or calculating
formula STEP loop acccumulated value or calculating formula
        EXIT_FOR
END_FOR
```

Description：leave FOR loop

Example：

```
G0 X0.
FOR #1=11 TO 23
  G0 X#1;
  IF(#1>15)
      EXIT_FOR;          /* when #1=16, leave loop */
  END_IF
END_FOR
M0;                      /* X axis final movement position 16mm，#1final is 16 */
M30
```

## 7.6 WHILE

Method：

```
  WHILE(conditions described)
      Data Process;
  END_WHILE
```

Description：WHILE loop

Example：

```
G0 X0.
#1=2
WHILE(#1<10)
  G0 X#1
  #1=#1+1
END_WHILE
 M0;                     /* X AXIS FINAL MOVEMENT POSITION9mm，#1final is 10 */
M30
```

### 7.7 EXIT_WHILE

Method：

```
WHILE(conditions described)
    EXIT_WHILE
END_WHILE
```

Description：leave WHILE loop

Example：

```
G0 X0.
#1=2
WHILE(#1<10)
    G0 X#1
    IF(#1>8)
        EXIT_WHILE;        /* When #1=9, leave loop */
    END_IF
    #1=#1+1
END_WHILE
M0;                        /* X AXIS FINAL MOVEMENT POSITION 9mm，#1final is 9 */
M30
```

### 7.8 DO…UNTIL

Method：

```
DO
    Data Process;
UNTIL(conditions described)
```

Description：When the conditions are not established, repeat UNTIL loop

Example：

```
G0 X0.
#1=2
DO
    G0 X#1
    #1=#1+1
UNTIL(#1>=10)
M0;                        /* X AXIS FINAL MOVEMENT POSITION9mm，#1final is 10 */
M30
```

## 7.9 EXIT_DO

Method：

```
DO
    EXIT_DO
UNTIL(conditions described)
```

Description：leave DO…UNTIL loop

Example：

```
G0 X0.
#1=2
DO
    G0 X#1
    IF(#1>7)
        EXIT_DO;            /* when #1=8, leave loop */
    END_IF
    #1=#1+1
UNTIL(#1>=10)
 M0;                        /* X AXIS FINAL MOVEMENT POSITION8mm，#1final is 8 */
M30
```

### 7.10 CALL_SUB

Method：

CALL_SUB “function name”

SUB “function name”

Data Process;

END_SUB

Description：allow program to ccall the same file’s function

Example：

```
G0 X0.
#1=3;
CALL_SUB “ABC_123”
G0 X99.
M30

SUB “ABC_123”
    G0 Y10.
    IF(#1>2)
        EXIT_SUB
    END_IF
    G0 Y20.
END_SUB
```

### 7.11 EXIT_SUB

Method：

EXIT_SUB

Description：leave function command

Example：

Refer to CALL_SUB call function description

# 8 Function

This chapter is to explain whole series of INT mode.

Version： Milling machine INT mode **mill_int_Ver03.01.01**

## 8.1 Function List

| Mathematical function | Description | R/W |
|---|---|---|
| SIN(DEG) | SIN Number of function | R |
| COS(DEG) | COS Number of function | R |
| TAN(DEG) | TAN Number of function | R |
| ASIN(VALUE) | ASIN Number of function | R |
| ACOS(VALUE) | ACOS Number of function | R |
| ATAN(VALUE1,VALUE2) | ATAN Number of function | R |
| SQRT(VALUE) | Get SQRT value | R |
| ABS(VALUE) | Get ABS value | R |
| ROUND(VALUE) | Get ROUND value | R |
| FIX(VALUE) | Get FIX value | R |
| MOD(VALUE1,VALUE2) | Get MOD value | R |
| Normal function | | |
| WAIT(PATH,TYPE) | Stop interpretation | W |
| ALARM("STRING") | Command MACRO to send alarm | W |
| ALARM(PATH,ALARM_No) | Command system MACRO to send Alarm | W |
| TAN_FOLLOW(PATH,TYPE,VALUE) | Tan follow function setting | W |
| Read/write system info function | | |
| R_G_GROUP(PATH,GROUP) | Read G code group | R |
| R_SYS_INFO(PATH,TYPE) | Read system info | R |
| TIME(TYPE) | Read system time | R |
| R_ARG(AXIS) | Get MACRO to 1~6 axis argument value | R |
| R_AXID("NAME") | Get path axis order | R |
| W_HSHP(PATH,TYPE,VALUE) | HSHP setting | W |
| R_SKIP(PATH,TYPE) | Read G31 Skip coordinate info | R |
| W_SKIP(PATH,TYPE,VALUE) | Write G31 Skip coordinate info | W |
| R_RESTART(PATH,TYPE) | Read program restart info | R |
| R_BREAKPOINT(PATH,TYPE) | Read manual return info | R |
| Read/write R and variables function | | |

| | | |
|---|---|---|
| R_REG(R_No) | Read R value | R |
| R_REG_F(R_No) | Pre-read R value | R |
| R_REG_BIT(R_No,BIT) | Read R BIT value | R |
| R_REG_BIT_F(R_No,BIT) | Pre-read R BIT value | R |
| W_REG(R_No,VALUE) | Write R value | W |
| W _REG_SYNC(R_No,VALUE) | Write R value with SBK synchronizely | W |
| W_REG_BIT(R_No,BIT,ONOFF) | Write R BIT value | W |
| W_REG_BIT_SYNC(R_No,BIT,ONOFF) | Write R BIT by SBK SYN | W |
| R_LV_BIT(LV_No,BIT) | Read Local variables BIT | R |
| W_LV_BIT(LV_No,BIT,ONOFF) | Write Local variables BIT | W |
| R_GV_BIT(GV_No,BIT) | Read global variables BIT | R |
| W_GV_BIT(GV_No,BIT,ONOFF) | Write Global variables BIT | W |
| Read/write coordiante and coorinate function | | |
| R_ABS_COOR(PATH,AXIS) | Read MOT ABS coordinate | R |
| R_MACH_COOR(PATH,AXIS) | Read MOT MACHINE coordinate | R |
| W_ABS_SHIFT(PATH,AXIS,CASE,VALUE) | Command MOT coordinate Offset | W |
| R_INT_PROG_COOR(PATH,AXIS) | Read INT Program Coordinate | R |
| INT_UPDATE(PATH,AXIS) | Make INT coordinate & MOT coordinate SYN | R |
| R_G53G59_COOR(PATH,Coor,AXIS) | Read G53~G59 coordinate | R |
| W_G53G59_COOR(PATH,Coor,AXIS,VALUE) | Write G53~G59 coordinate | W |
| R_G54EXP_COOR(PATH,Coor,AXIS) | Read G54 extension coordinate | R |
| W_G54EXP_COOR(PATH,Coor,AXIS,VALUE) | Write G54 extension coordinate | W |
| Read/write tool info function | | |
| R_TOOL_DATA(PATH,Tool_No,TYPE) | Read INT tool info | R |
| W_TOOL_DATA(PATH,Tool_No,TYPE,VALUE) | Write INT tool info | W |
| I LATCH value function | | |
| W_I_LATCH(PATH,I_NO,HwifAxis_No,Ri_Fa) | Set up I Latch | W |
| R_I_LATCH(PATH,I_NO) | Get if I Latch happend | R |
| R_I_LATCH_COOR(PATH,I_NO,AXIS) | Get I Latch coordinate | R |
| Regularization | | |
| SPEED_STD(PATH,VALUE) | Speed value regularization | R |
| SPEED_STD_R(PATH,R_No) | R speed regularization | R |
| SPEED_STD_R_F(PATH,R_No) | Pre-read R Speed regularization | R |
| LEN_STD(PATH,AXIS,VALUE) | Length value regularization | R |
| LEN_STD_R(PATH,AXIS,R_No) | R Length value regularization | R |
| LEN_STD_R_F(PATH,AXIS,R_No) | Pre-read R length value regularization | R |

| In position function | | |
|---|---|---|
| C_INPOS(PATH,TYPE,VALUE) | Feedrate in position check | W |
| RP_INPOS(PATH,TYPE,VALUE) | Rapid in position check | W |
| BKCMP(PATH,TYPE,VALUE) | SBK compare | W |
| MACRO variables stack function | | |
| PUSH(VALUE) | Push to MACRO variable stack | W |
| POP( ) | Pop info from MACRO variables stack | R |
| STKTOP(STACK_NO) | Check info from MACRO variables stack | R |
| STKCLR() | Clear stack info | W |
| CCD function | | |
| W_CCD(PATH,TYPE,VALUE) | Set up CCD info | W |
| PASER_CCD(PATH,ITEM) | Get CCD reply data column value | R |
| MACRO dialogue function | | |
| MSG_OK("TITLE","TEXT","FILE") | Message dialogue | W |
| MSG_YES("TITLE","TEXT","FILE") | Inquiry dialogue | R |
| INPUT("TITLE","TEXT","FILE",MIN,MAX,DEF) | Input value dialogue | R |
| MENU("TITLE","TEXT","FILE") | Menu dialogue | R |
| MENU_ADD("TEXT","FILE") | Menu dialogue setting | W |
| PLOT_ADD("FILE","LEGEND") | Curve data file name | W |
| PLOT_SHOW("TITLE","TEXT","XLB","YLB") | Curve framce | W |
| MACRO write function | | |
| OPEN("FILE") | Open file | W |
| CLOSE() | Close file | W |
| PRINT("STRING”) | Write string | W |

## 8.2 Mathematical function

| | SIN(DEG) | SIN Number of function | R |
|---|---|---|---|
| Description | ● DEG => Angle degree。Value range****, unit：degree | | |
| Reply | Value, value range****, unit：N/A | | |
| Example | #1 = SIN(30); /* #1 Value=0.5 */ | | |

| | COS(DEG) | COS Number of function | R |
|---|---|---|---|
| Description | ● DEG => Angle degree。Value range****, unit：degree | | |
| Reply | Value, value range****, unit：N/A | | |
| Example | #1 = SIN(30); /* #1 Value=0.866 */ | | |

| | TAN(DEG) | TAN Number of function | R |
|---|---|---|---|
| Description | ● DEG =>Angle degree。Value range****, unit：degree | | |
| Reply | Value, value range****, unit：N/A | | |
| Example | #1 = TAN(30); /* #1 Value=0.57735 */ | | |

| | ASIN(VALUE) | ASIN Number of function | R |
|---|---|---|---|
| Description | ● VALUE=> Value。Value range-1~1, unit：N/A | | |
| Reply | Angle degree。Value range-90~90, unit：degree | | |
| Example | #1 = ASIN(0.5); /* #1 Value=30 */ | | |

| | ACOS(VALUE) | ACOS Number of function | R |
|---|---|---|---|
| Description | ● VALUE=> Value。Value range-1~1, unit：N/A | | |
| Reply | Angle degree。Value range-90~90, unit：degree。 | | |
| Example | #1 = ASIN(0.866); /* #1 Value=30 */ | | |

| | ATAN(VALUE1,VALUE2) | ATAN Number of function | R |
|---|---|---|---|
| Description | ● VALUE1=> On the other side of right triangle。Value range****, unit：N/A<br>● VALUE2=> On the next side of right triangle。Value range****, unit：N/A | | |
| Reply | Angle degree。Value range-180~180, unit：degree。 | | |
| Example | #1 = ATAN(0.57735,1); /* #1 Value =30 */ | | |

| | SQRT(VALUE) | Get SQRT value | R |
|---|---|---|---|
| Description | ● VALUE=> Value, value range****, unit：N/A | | |
| Reply | Value, value range****, unit：N/A | | |
| Example | #1 = SQRT(2); /* #1 Value=1.4142 */ | | |

| | ABS(VALUE) | Get ABS value | R |
|---|---|---|---|
| Description | ● VALUE=> Value, value range****, unit：N/A | | |
| Reply | Value, value range****, unit：N/A | | |
| Example | #1 = ABS(-2);  /* #1 Value=2 */ | | |

| | ROUND(VALUE) | Get Round value | R |
|---|---|---|---|
| Description | ● VALUE=>Value, value range****, unit：N/A | | |
| Reply | Value, value range****, unit：N/A | | |
| Example | #1 = ROUND(1.7);  /* #1 Value=2 */ | | |

| | FIX(VALUE) | Get FIX value | R |
|---|---|---|---|
| Description | ● VALUE=>Value, value range****, unit：N/A | | |
| Reply | Value, value range****, unit：N/A | | |
| Example | #1 = FIX(1.2);  /* #1 Value=1 */ | | |

| | MOD(VALUE1,VALUE2) | Get MOD value | R |
|---|---|---|---|
| Description | ● VALUE1=>Value, value range****, unit：N/A<br>● VALUE2=>Value, value range****, unit：N/A | | |
| Reply | Value, value range****, unit：N/A | | |
| Example | #1 = MOD(24.95,5.5);  /* #1 Value=2.95 */ | | |

### 8.3 Normal function

<table>
<tr><th></th><th>WAIT(PATH,TYPE)</th><th>Stop interpretation</th><th>W</th></tr>
<tr><td>Description</td><td colspan="3">
<ul>
<li>PATH =>path, No.。Value range0 ~6, unit：N/A<br>0：current belonging path, 1~6：first path~6th path.</li>
<li>TYPE =>Appointed type。Value range****, unit：N/A<br>0: type Ⅰ, Interpred single block has been executed.<br>1: type Ⅱ, Interpred single block has been executed.+arrived machine coordinate.</li>
</ul>
Below function has been doned stop interpretation(Type Ⅰ)
<table>
<tr><td>R_REG</td><td>R_REG_BIT</td><td>W_REG</td></tr>
<tr><td>W_REG_BIT</td><td>LEN_STD_R</td><td>SPEED_STD_R</td></tr>
<tr><td>R_ABS_COOR</td><td>W_ABS_SHIFT</td><td>TIME</td></tr>
<tr><td>W_I_LATCH</td><td>R_I_LATCH</td><td>R_I_LATCH_COOR</td></tr>
<tr><td>W_CCD</td><td>WAIT_CCD</td><td>MSG_OK</td></tr>
<tr><td>MSG_YES</td><td>INPUT</td><td>MENU</td></tr>
</table>
Below function has been doned stop interpretation (Type Ⅱ)
<table>
<tr><td>R_MACH_COOR</td><td>INT_UPDATE</td><td>R_RESTART</td></tr>
</table>
</td></tr>
<tr><td>Reply</td><td colspan="3">：N/A</td></tr>
<tr><td>Example</td><td colspan="3">…<br>WAIT(0,0);  /* wait previous single block to be executed and read the coming single block */<br>…</td></tr>
</table>

<table>
<tr><th></th><th>ALARM("STRING")</th><th>Command MACRO to send Alarm</th><th>W</th></tr>
<tr><td>Description</td><td colspan="3">
<ul>
<li>STRING =>At least 1 digit’s any string. If above arguments string has MACRO variables, system will replace it to be the value automatically. Refer to 5.5 string variables replace value chapter.</li>
</ul>
</td></tr>
<tr><td>Reply</td><td colspan="3">：N/A</td></tr>
<tr><td>Example</td><td colspan="3">Usage Example 1：<br>ALARM("did not open coolant");  /* send alarm 610000 and show string” did not open coolant” */<br>Usage Example 2：<br>#1= R_MACH_COOR(0,1);  /* If this path’s machine coordinate first axis =1.234mm, #1Value=1.234 */<br>ALARM("machine coordinateX#1correct");  /* send alarm 610000 and show string” machine coordinateX1.234 correct” */</td></tr>
</table>

| | ALARM(PATH,ALARM_No) | Command system MACRO to send Alarm | W |
|---|---|---|---|
| Description | **This function is only for system MACRO to use**<br>● PATH =>path number。Value range 0 ~6, unit：N/A<br>0：Current path，1~6：First path~6th path.<br>● ALARM_No =>Appoint alarm number(1~511)。Value range1 ~ 511, unit：N/A | | |
| Reply | ：N/A | | |
| Example | ALARM(0,1); /* send alarm 610001 */ | | |

| | TAN_FOLLOW(PATH,TYPE,VALUE) | Tan-follow Function Setting | W |
|---|---|---|---|
| Description | ● PATH =>Path No., value range0 ~6, unit：N/A<br>0：Current path, 1~6：First path~6th path.<br>● TYPE =>Setting item, value range****, unit：N/A<br>1: Tan-follow function setting axis(0:N/A,1~3:Cartesian coordinate previous 3 axis), unit：N/A<br>2: Tan-follow direction(0:+,1:-), unit：N/A<br>● VALUE =>Write info, value range0 ~ 2147483647, unit：N/A | | |
| Reply | Alarm：<br>-1 => Direction is over range | | |
| Example | TAN_FOLLOW(0,1,3); /* Tan-follow function setting=3rd axis */ | | |

### 8.4 Read/write system info function

| | R_G_GROUP(PATH,GROUP) | Read G code group | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● GROUP =>G group[0~22], value range0~22, unit：N/A | | |
| Reply | Part program read G code group, unit：N/A<br>Alarm：<br>-1 => Group No. is ove range | | |
| Example | #1= R_G_GROUP(0,1); /* If current group 1=G2, #1Value=2 */ | | |

| | R_SYS_INFO(PATH,TYPE) | Read system info | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● TYPE =>Type, value range****, unit：N/A. | | |

| | |
|---|---|
| | 0：Path No., unit：N/A.<br>1: Get system type, 0:Main INT 1:Preview INT 2:R-restart, unit：N/A<br>2: Get Mode T code, unit：N/A<br>4: Get system speed, unit=mm/min<br>5: Get extention coordinate number, unit：N/A<br>6: Path axis numbers, unit：N/A<br>7: MACRO to 2 M arguments, unit：N/A<br>8: MACRO to 3 M arguments, unit：N/A<br>9: MACRO to 2 S arguments, unit：N/A<br>10: MACRO to 3 S arguments, unit：N/A<br>11: If path 1 is ON(0:N,1:Y)<br>12: If path 2 is ON(0:N,1:Y)<br>13: If path 3 is ON(0:N,1:Y)<br>14: If path 4 is ON(0:N,1:Y)<br>15: If path 5 is ON(0:N,1:Y)<br>16: If path 6 is ON(0:N,1:Y)<br>20: If HMI is invalid(0:Valid,1:Invalid), corresponding command DIS_HMIPLOT_ON<br>21: If mute is ON(0: N/A,1:mute), corresponding command MUTE_ON<br>22: If offset is invalid(0:Valid,1:Invalid), corresponding command DIS_SHIFT_ON<br>23:If external SBK stop is invalid(0:Valid,1:Invalid), corresponding command SBK_OFF<br>24: If fixed G0 override(0:N/A,1:Fixed 100%), corresponding command FIX_RAPID_OR_ON。<br>25: If fixed G1 override(0: N/A,1:Fixed 100%), corresponding command FIX_CUT_OR_ON。<br>26: If fixed auto mode(0: N/A,1:Fixed ), corresponding command FIX_AUTOMODE_ON。<br>101 ~ 132: Get 1~32 axis corresponding software axis number, unit：N/A |
| Reply | Part program read system info, unit=as above |
| Example | #1= R_SYS_INFO(0,2);  /* If current situation T code=201, #1Value=201 */ |

| | TIME(TYPE) Read System Time R |
|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● TYPE =>Type, value range****, unit：N/A.<br>1: Get system interrupt times, unit：N/A<br>2: System date year：yyyy, unit：N/A<br>3 System date month：mm, unit：N/A<br>4: System date day：dd, unit：N/A<br>5: System time hour：hh, unit：N/A<br>6: System time 分：nn, unit：N/A |

| | |
|---|---|
| | 7: System time minute：ss, unit：N/A |
| Reply | Part program read system time, unit=as above |
| Example | #1= TIME(5);  /* If current time=15:30, #1Value=15 */ |

| | R_ARG(AXIS) | Get MACRO to 1~6 axis argument value | R |
|---|---|---|---|
| Description | ● AXIS=>Appoint axis number, value range1~32, unit：N/A | | |
| Reply | MACRO to this axis argument value, unit=mm<br>VACANT => This axis did not have arguments to pass in. | | |
| Example | G65 P1234 X11. Y12. Z13.  /* call O1234 MACRO */<br>;/* O1234 */<br>#1= R_ARG(1);  /* #1Value=11 */ | | |

| | R_AXID("NAME") | Get path axis order | R |
|---|---|---|---|
| Description | ● NAME=>Appoint axis, value range****, unit：N/A | | |
| Reply | Reply path axis order No., unit：N/A<br>VACANT => No this axis in the parameter setting<br>Alarm：<br>-1 => Did not appoint axis string<br>-2 => Axis string format error | | |
| Example | #1= R_AXID("X");  /* If path 1 axis=X, then #1Value=1 */<br>#2= R_AXID("Y2");  /* If path 2 axis =Y2, then #2Value=2 */<br>#3= R_AXID("Z3");  /* If path 3 axis =Z3, the #3Value=3 */ | | |

| | W_HSHP(PATH,TYPE,VALUE) | HSHP setting | W |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● TYPE =>Setting item, value range****, unit：N/A<br>1: Path feedrate linear ACC/DEC time, unitms。<br>7: Path feedrate 5mm arm allow speed, unitKLU/min。<br>9: Feedrate circular clamp minimum speed, unitKLU/min。<br>101 ~ 132: path feedrate 1~32 axis corner speed difference, unitKLU/min。<br>● VALUE =>Write info, value range0 ~ 2147483647, unit：N/A | | |
| Reply | Alarm：<br>-1 => Appoint value is illegal | | |
| Example | W_HSHP(0,1,50);  /* Set up path feedrate linear ACC/DEC time=50ms */ | | |

| | R_SKIP(PATH,TYPE) | Read G31 Skip coordinate info | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● TYPE =>Type, value range****, unit：N/A.<br>1: If SKIP signal has been triggered(0:N,1:Y), unit：N/A<br>101 ~ 132: SKIP ABS coordinate 1~32 axis, unit=mm<br>201 ~ 232: SKIPMACHINE coordinate1~32 axis, unit=mm | | |
| Reply | Part program Read G31 Skip coordinate info, unit=as above | | |
| Example | G91 G31 Z-100. F100;<br>#1= R_SKIP(0,103); /* Get 3 axis SKIP ABS coordinate */<br>G04 X20.; /* feedhold, you can check if #1=Z axis ABS coordinate */<br>M30; | | |

| | W_SKIP(PATH,TYPE,VALUE) | Write G31 Skip coordinate info | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● TYPE =>Type, value range****, unit：N/A.<br>1: If SKIP signal has been triggered (0:N,1:Y), unit：N/A<br>101 ~ 132: SKIPABScoordinate1~32 axis, unit=mm<br>201 ~ 232: SKIPMACHINE coordinate1~32 axis, unit=mm<br>● VALUE =>Write info, value rangedouble, unit=mm | | |
| Reply | ：N/A | | |
| Example | W_SKIP(0,101,0); /* White SKIP 1st axis ABS coordinate, write value=0 */ | | |

| | R_RESTART(PATH,TYPE) | Read program restart info | R |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅱ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● TYPE =>Type, value range****, unit：N/A.<br>1~15:Secircularhed M code<br>21: Secircularhed S code<br>31~32: Secircularhed T code<br>101 ~ 132: Target site ABS coordinate 1~32 axis, unit=mm<br>201 ~ 232: Target site MACHINE coordinate 1~32 axis, unit=mm<br>301 ~ 332: Offset from current position to target site 1~32 axis, unit=mm | | |
| Reply | Part program read program restart info, unit=as above<br>Alarm：<br>-1 => Read coordinate error | | |

| | |
|---|---|
| Example | #1= R_RESTART(0,101);  /* read R-restart firt axis target site ABS coordinate */ |

| | R_BREAKPOINT(PATH,TYPE) | Read manual return info | R |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅱ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● TYPE =>Type, value range****, unit：N/A.<br>101 ~ 132: Target site ABS coordinate 1~32 axis, unit=mm<br>201 ~ 232: Target site MACHINE coordinate 1~32 axis, unit=mm<br>301 ~ 332: Offset from current position to target site 1~32 axis, unit=mm | | |
| Reply | Part program Read manual return info, unit=as above<br>Alarm：<br>-1 => Read coordinate error | | |
| Example | #1= R_BREAKPOINT(0,101);  /* read manual return first axis target site ABS coordinate */ | | |

### 8.5 Read/write R and variables function

| | R_REG(R_No) | Read R value | R |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● R_No=> R location NO., value range0 ~ 179999, 1000000 ~ 5999999, unit：N/A | | |
| Reply | R VALUE, value range-2147483648 ~ 2147483647, unit：N/A<br>Alarm：<br>-1 =>R value NO. is over range. | | |
| Example | #1=R_REG(1);  /* If R[1]=12, #1Value=12 */ | | |

| | R_REG_F(R_No) | Pre-read R value | R |
|---|---|---|---|
| Description | This function will pre-read R bit, easy to have pre-INT problem, so need to use WAIT(….) to use, suitable for advanced programmer to use.<br>● R_No=> R location NO., value range0 ~ 179999, 1000000 ~ 5999999, unit：N/A | | |
| Reply | R VALUE, value range-2147483648 ~ 2147483647, unit：N/A<br>Alarm：<br>-1 =>R value NO. is over range. | | |
| Example | #1=R_REG_F(1);  /* If R[1]=12, #1Value=12 */ | | |

| | R_REG_BIT(R_No,BIT) | Read R BIT value | R |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● R_No=> R location NO., value range0 ~ 179999, 1000000 ~ 5999999, unit：N/A | | |

| | |
|---|---|
| | ● BIT=>Appoint R value Bit, value range0~31, unit：N/A |
| Reply | R Bit, value range0 ~ 1, unit：N/A<br>Alarm：<br>-1 =>R value NO. is over range.<br>-2 => BIT No. is over range |
| Example | #1= R_REG_BIT(3,2);　/* If R[3]=21, #1Value=1 */ |

| | R_REG_BIT_F(R_No,BIT) | Pre-read R BIT value | R |
|---|---|---|---|
| Description | This function will pre-read R bit, easy to have pre-INT problem, so need to use WAIT(….) to use, suitable for advanced programmer to use.<br>● R_No=> R location NO., value range0 ~ 179999, 1000000 ~ 5999999, unit：N/A<br>● BIT=>Appoint R Bit, value range0~31, unit：N/A | | |
| Reply | R Bit, value range0 ~ 1, unit：N/A<br>Alarm：<br>-1 =>R value NO. is over range.<br>-2 => R value NO. is over range. | | |
| Example | #1= R_REG_BIT_F(3,2);　/* If R[3]=21, #1Value=1 */ | | |

| | W_REG(R_No,VALUE) | Write R value | W |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● R_No=> R location NO., value range0 ~ 179999, 1000000 ~ 5999999, unit：N/A<br>● VALUE =>Write info, value range-2147483648 ~ 2147483647, unit：N/A | | |
| Reply | Alarm：<br>-1 =>R value NO. is over range. | | |
| Example | W_REG(2,13);　/* R[2]Value=13 */ | | |

| | W_ REG_SYNC(R_No,VALUE) | Write R value with SBK synchronizely | W |
|---|---|---|---|
| Description | Write R value by SBK order, no need to stop INT, has limit when writing R location, this SBK will need 1 interrupt time.<br>● R_No=> R location NO., value range0 ~ 179999, unit：N/A<br>● VALUE =>Write info, value range-1073741824 ~ 1073741823, unit：N/A | | |
| Reply | Alarm：<br>-1 =>R value NO. is over range.<br>-2 => R value writes over range | | |
| Example | W_ REG_SYNC(2,13);　/* R[2]Value=13 */ | | |

| | W_REG_BIT(R_No,BIT,ONOFF) | Write R BIT value | W |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)** | | |

| | |
|---|---|
| | ● R_No=> R location NO., value range0 ~ 179999, unit：N/A<br>● BIT=>Appoint R valueBit, value range0~31, unit：N/A<br>● ONOFF=>write value, 0 ~ 1, unit：N/A |
| Reply | Alarm：<br>-1 =>R value NO. is over range.<br>-2 => BIT No. is over value range<br>-3 => ON OFF is over value range |
| Example | W_REG_BIT(4,3,1);  /* If R[4]=0，R[4]Value=8 */ |

| | W_REG_BIT_SYNC(R_No,BIT,ONOFF) Write R BIT by SBK SYN W |
|---|---|
| Description | Write R value by SBK order, no need to stop INT, has limit when writing R location, this SBK will need 1 interrupt time.<br>● R_No=> R location NO., value range0 ~ 179999, unit：N/A<br>● BIT=>Appoint R valueBit, value range0~31, unit：N/A<br>● ONOFF=>write value, 0 ~ 1, unit：N/A |
| Reply | Alarm：<br>-1 =>R value NO. is over range.<br>-2 => BIT No. is over range<br>-3 => ONOFF is over range |
| Example | W_REG_BIT_SYNC(4,3,1); /* If R[4]=0，R[4]Value=8 */ |

| | R_LV_BIT(LV_No,BIT) Read Local Variables BIT R |
|---|---|
| Description | ● LV_No =>Read local variables location No., value range0 ~199, unit：N/A<br>● BIT=>Appoint local variables value Bit, value range0~31, unit：N/A |
| Reply | Local variables Bit, value range0 ~ 1, unit：N/A<br>Alarm：<br>-1 => Local variables No. is over range<br>-2 => BIT No. is over range |
| Example | #1= R_LV_BIT(3,2); /* if #3=21, #1Value=1 */ |

| | W_LV_BIT(LV_No,BIT,ONOFF) Write Local Variables BIT W |
|---|---|
| Description | ● LV_No =>Write local variables location No., value range 0 ~199, unit：N/A<br>● BIT=>Appoint local variables value Bit, value range0~31, unit：N/A<br>● ONOFF=>write value, 0 ~ 1, unit：N/A |
| Reply | Alarm：<br>-1 => Local variables No. is over range<br>-2 => BIT No. is over range<br>-3 => ONOFF is over range |
| Example | W_LV_BIT(4,3,1); /* if #4=0, #4Value=8 */ |

| | R_GV_BIT(GV_No,BIT) Read Global Variables BIT R |
|---|---|
| Description | ● GV_No =>Read global variables location No., value range0 ~1999, unit：N/A<br>● BIT=>Appoint global variables value Bit, value range0~31, unit：N/A |
| Reply | Global variables Bit, value range0 ~ 1, unit：N/A<br>Alarm：<br>-1 => Global variables No. over range |

| | |
|---|---|
| | -2 => BIT No. is over range |
| Example | #1= R_GV_BIT(3,2); /* If @3=21，#1Value=1 */ |

| | W_GV_BIT(GV_No,BIT,ONOFF) | Write Global Variables BIT | W |
|---|---|---|---|
| Description | ● GV_No =>Write global variables location No., value range0 ~1999, unit：N/A<br>● BIT=>Appoint global variables value Bit, value range0~31, unit：N/A<br>● ONOFF=>write value, 0 ~ 1, unit：N/A | | |
| Reply | Alarm：<br>-1 => Global variablesover range<br>-2 => BIT No. is over range<br>-3 => ONOFF is over range | | |
| Example | W_GV_BIT(4,3,1); /* if @4=0。@4Value=8 */ | | |

### 8.6 Read/write coordinate and coordinate function

| | R_ABS_COOR(PATH,AXIS) | Read MOT ABS Coordinate | R |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A | | |
| Reply | MOT appoint axis ABS coordinate, unit=mm<br>Alarm：<br>-1 =>Read coordinate error | | |
| Example | #1= R_ABS_COOR(0,1); /* If this path ABS coordinate 1st axis=1.234mm, #1Value=1.234 */ | | |

| | R_MACH_COOR(PATH,AXIS) | Read MOT MACHINE Coordinate | R |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅱ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A | | |
| Reply | MOT appoint axis MACHINE coordinate, unit=mm<br>Alarm：<br>-1 =>Read coordinate error | | |
| Example | #1= R_MACH_COOR(0,1); /* If this path MACHINE coordinate 1st axis=1.234mm, #1Value=1.234 */ | | |

| | W_ABS_SHIFT(PATH,AXIS,CASE,VALUE) | Command MOT Coordinate Offset | W |
|---|---|---|---|

| | |
|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A<br>● CASE =>Set up type, value range1~4, unit：N/A<br>1:Re-set program coordinate 2: Offset program coordinate 3:Re-set REL coordinate 4:Offset REL coordinate<br>● VALUE =>Write info, value rangedouble, unit=mm |
| Reply | ：N/A |
| Example | W_ABS_SHIFT(0,1,1,1.234); /* This path ABS coordinate 1st axis=1.234mm */ |

| | R_INT_PROG_COOR(PATH,AXIS) | Read INT Program Coordinate | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A | | |
| Reply | Part program read INT coordinate, unit=mm | | |
| Example | #1= R_INT_PROG_COOR(0,1); /* read INT coordinate 1st axis value */ | | |

| | INT_UPDATE(PATH,AXIS) | Make INT Coordinate & MOT Coordinate SYN | W |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅱ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A | | |
| Reply | Alarm：<br>-1 =>Read coordinate error | | |
| Example | INT_UPDATE(0,1); /* SYN this path 1st axis INT coordinate and MOT coordinate */ | | |

| | R_G53G59_COOR(PATH,Coor,AXIS) | Read G53~G59 Coordinate | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● Coor =>Appoint coordinate 53~59, value range 53~59, unit：N/A<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A | | |
| Reply | Part program read G53~G59 coordinate, unit=mm<br>Alarm：<br>-1 => Appoint coordinate No. over range | | |
| Example | #1= R_G53G59_COOR(0,55,1); /* If this path G55 coordinate 1st axis=1.234mm, #1Value=1.234 */ | | |

| | W_G53G59_COOR(PATH,Coor,AXIS,VALUE) Write G53~G59 Coordinate W |
|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● Coor =>Appoint coordinate 53~59, value range53~59, unit：N/A<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A<br>● VALUE =>Write info, value rangedouble, unit=mm |
| Reply | Alarm：<br>-1 => Appoint coordinate No. is over range |
| Example | W_G53G59_COOR(0,55,1,1.234); /* This path G55 coordinate 1st axis=1.234mm */ |

| | R_G54EXP_COOR(PATH,Coor,AXIS) Read G54 Extension Coordinate R |
|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● Coor =>Appoint extension coordinate 1~100, value range1~100, unit：N/A<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A |
| Reply | Part program read G54 extension coordinate, unit=mm<br>Alarm：<br>-1 => Appoint coordinate No. is over range |
| Example | #1= R_G54EXP_COOR(0,2,1); /* If this path P2 coordinate 1st axis=1.234mm,<br>#1Value=1.234 */ |

| | W_G54EXP_COOR(PATH,Coor,AXIS,VALUE) Write G54 Extension Coordinate W |
|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● Coor =>Appoint extension coordinate 系 1~100, value range1~100, unit：N/A<br>● AXIS=>Appoint axis number, value range1~32, unit：N/A<br>● VALUE =>Write info, value rangedouble, unit=mm |
| Reply | Alarm：<br>-1 => Appoint coordinate No. is over range |
| Example | W_G54EXP_COOR(0,2,1,1.234); /* this path P2 coordinate 1st axis=1.234mm */ |

### 8.7 Read/write tool info function

| | R_TOOL_DATA(PATH,Tool_No,TYPE) Read INT Tool Info R |
|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path. |

| | |
|---|---|
| | ● Tool_No =>Appoint get tool info's too number, value range1 ~ 400, unit：N/A<br>● TYPE =>Appoint get tool info's type, value range****, unit：N/A<br>1: Tool type, unit：N/A<br>2: Tool wear diameter, unit=mm<br>3: Tool geometry diameter,, unit=mm<br>4: Tool index, unit：N/A<br>5: Tool database index, unit：N/A<br>101 ~ 132: Tool wear length 1~32 axis, unit=mm<br>201 ~ 232: Tool geometry length 1~32 axis, unit=mm |
| Reply | Part program read tool info, unit=above<br>Alarm：<br>-1 => Appoint tool number is over range |
| Example | #1= R_TOOL_DATA(0,2,201);  /* If Tool 2's first axis geometry length =1.234mm, #1Value=1.234 */ |

| | W_TOOL_DATA(PATH,Tool_No,TYPE,VALUE) Write INT Tool Info | W |
|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● Tool_No =>Appoint write tool info's tool number, value range1 ~ 400, unit：N/A<br>● TYPE =>Appoint write tool info's type, value range****, unit=below<br>2: Tool wear diameter<br>3: Tool geometry diameter<br>101 ~ 132: Tool wear length 1~32 axis<br>201 ~ 232: Tool geometry length 1~32 axis<br>● VALUE =>Write info, value range double, unit=mm | |
| Reply | Alarm：<br>-1 => Appoint tool number is over range | |
| Example | W_TOOL_DATA(0,2,201,1.234); /* this path Tool 2 first axis geometry tool length=1.234mm */ | |

### 8.8 I LATCH value function

| | W_I_LATCH(PATH,I_NO,HwifAxis_No,Ri_Fa) Setup I Latch | W |
|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● I_NO => I No., value range****, unit：N/A<br>1~2 Card1 EPCIO LI1~2<br>11~12 Card2 EPCIO LI1~2 ~ 51~52 Card6 EPCIO LI1~2<br>101~106 Card1 CPLD LI1~6<br>111~116 Card2 CPLD LI1~6 ~ 151~156 Card6 CPLD LI1~6<br>● HwifAxis_No=>Hardware axis No., value range****, unit：N/A<br>1~6:Card1<br>11~16:Card2 ~ 51~56:Card6<br>● Ri_Fa=>1:Rising Edge,2:Falling edge, value range1~2, unit：N/A | |
| Reply | ：N/A | |
| Example | W_I_LATCH(0,101,1,1); /* Enable CPLD first I point, first axis at hardware, up trigger */ | |

| | R_I_LATCH(PATH,I_NO) Get if I Latch happend | R |
|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path. | |

| | |
|---|---|
| | ● I_NO => I No., value range****, unit：N/A<br>1~2 Card1 EPCIO LI1~2<br>11~12 Card2 EPCIO LI1~2 ~ 51~52 Card6 EPCIO LI1~2<br>101~106 Card1 CPLD LI1~6<br>111~116 Card2 CPLD LI1~6 ~ 151~156 Card6 CPLD LI1~6 |
| Reply | path Get if I Latch happend(0:N,1:Y), unit：N/A |
| Example | #1 = R_I_LATCH(0,101); /* Get CPLD first I signal, if triggered, #1Value=1 */ |

| | R_I_LATCH_COOR(PATH,I_NO,AXIS) Get I Latch Coordinate R |
|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● I_NO => I No., value range****, unit：N/A<br>1~2 Card1 EPCIO LI1~2<br>11~12 Card2 EPCIO LI1~2 ~ 51~52 Card6 EPCIO LI1~2<br>101~106 Card1 CPLD LI1~6<br>111~116 Card2 CPLD LI1~6 ~ 151~156 Card6 CPLD LI1~6<br>● AXIS=>Appoint axis number, value range1~6, unit：N/A |
| Reply | path Get I Latch coordinate, unit=mm |
| Example | #1 = R_I_LATCH_COOR(0,101,1); /* Get CPLDfirst I point LATCH MACHINE coordinate value */ |

### 8.9 Value Regularization

| | SPEED_STD(PATH,VALUE) | Speed Value Regularization | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● VALUE =>Write info, value rangedouble, unit(mm/min)。 | | |
| Reply | Converts the input value to the maximum metric or inch unit mode. Metric mode (mm/min), inch mode (inch/min). | | |
| Example | #1 = SPEED_STD(0,25.4);  /* if this SBK is under inch mode, #1Value=1 inch/min */ | | |

| | SPEED_STD_R(PATH,R_No) | R Speed Regularization | R |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● R_No=> R location NO., value range0 ~ 129999, 1000000 ~ 3999999, unit：N/A | | |
| Reply | Converts the input value to the maximum metric or inch unit mode. Metric mode (mm/min), inch mode (inch/min).<br>Alarm：<br>-1 =>R value NO. is over range. | | |
| Example | #1 = SPEED_STD_R(0,1);  /* If R[1]=25400, #1Value=1 inch/min */ | | |

| | SPEED_STD_R_F(PATH,R_No) | Pre-read R Speed Regularization | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● R_No=> R location NO., value range0 ~ 129999, 1000000 ~ 3999999, unit：N/A | | |
| Reply | Converts the input value to the maximum metric or inch unit mode. Metric mode (mm/min), inch mode (inch/min).<br>Alarm：<br>-1 =>R value NO. is over range. | | |
| Example | #1 = SPEED_STD_R_F(0,1);  /* If R[1]=25400, #1Value=1 inch/min */ | | |

| | LEN_STD(PATH,AXIS,VALUE) | Length Value Regularization | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● AXIS=>Appoint axis number, value range0~32, unit：N/A<br>0：Convert directly. 1~32：follow path axis situation to change<br>● VALUE =>Write info, value rangedouble, unit(mm)。 | | |

| | |
|---|---|
| Reply | Converts the input value to the maximum metric or inch unit mode. Metric mode (mm/min), inch mode (inch/min). |
| Example | #1 = LEN_STD(0,2,25.4); /* If 2 axis is linear axis and its SBK in under inch mode, #1Value=1 inch */ |

| | LEN_STD_R(PATH,AXIS,R_No) | R Length Value Regularization | R |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● AXIS=>Appoint axis number, value range0~6, unit：N/A<br>0：Convert directly. 1~32：follow path axis situation to change<br>● R_No=> R location NO., value range0 ~ 129999, 1000000 ~ 3999999, unit：N/A | | |
| Reply | Converts the input value to the maximum metric or inch unit mode. Metric mode (mm/min), inch mode (inch/min).<br>Alarm：<br>-1 =>R value NO. is over range. | | |
| Example | #1 = LEN_STD_R(0,2,1); /* If 2 axis is linear axis and its SBK in under inch mode，R[1]=25400, #1Value=1 inch */ | | |

| | LEN_STD_R_F(PATH,AXIS,R_No) | Pre-read R valuelength Value Regularization | R |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● AXIS=>Appoint axis number, value range0~6, unit：N/A<br>0：Convert directly. 1~32：follow path axis situation to change<br>● R_No=> R location NO., value range0 ~ 129999, 1000000 ~ 3999999, unit：N/A | | |
| Reply | Converts the input value to the maximum metric or inch unit mode. Metric mode (mm/min), inch mode (inch/min).<br>Alarm：<br>-1 =>R value NO. is over range. | | |
| Example | #1 = LEN_STD_R(0,2,1); /* If 2 axis is linear axis and its SBK in under inch mode, R[1]=25400, #1Value=1 inch */ | | |

### 8.10 In position function

| | C_INPOS(PATH,TYPE,VALUE) | Feedrate In Position Check | W |
|---|---|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path. | | |

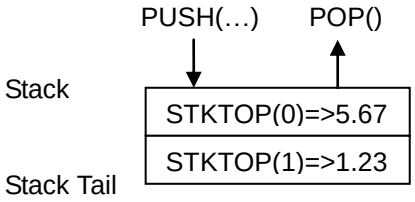
| | |
|---|---|
| | ● TYPE =>Type, value range****, unit：N/A.<br>1: Feedrate in postion check type(0:OFF,1:Axis,2:Tan), unit：N/A<br>2: Feedrate in postion check type reply parameter default setting (1: Notify tag), unit：N/A<br>3: Set up tan in postion check range, unit=mm<br>4: Set up tan in postion check range reply parameter default setting (1:Notify tag), unit：N/A<br>101~132: Set up every axis in postion check switch(0:OFF,1:ON), unit：N/A<br>201~232: Set up every axis in postion check switch reply parameter default setting (1:notify tag), unit：N/A<br>301~332: Set up every axis in position range, unit=mm<br>401~432: Set up every axis in position range reply parameter default setting (1:notify tag), unit：N/A<br>501~532: If every axis in tan in postion check switch(0:OFF,1:Combine&check), unit：N/A<br>601~632: If every axis in tan in postion check switch reply parameter default setting (1:notify tag), unit：N/A<br>● VALUE =>Write info, value range0 ~ 2147483647, unit：N/A |
| Reply | ：N/A |
| Example | C_INPOS(0,301,0.123);　/* Appoint first axis feedrate in postion check range 0.123mm */ |

| | RP_INPOS(PATH,TYPE,VALUE)　Rapid In Position Check　W |
|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~32：1 path ~6 path<br>● TYPE =>Type, value range****, unit：N/A.<br>1: Rapid in postion check type(0:OFF,1:every axis), unit：N/A<br>2: Rapid in postion check type reply parameter default setting (1:notify tag), unit：N/A<br>101~132: Set up every axis in postion check switch(0:OFF,1:ON), unit：N/A<br>201~232: Every axis in postion check switch reply parameter default setting (1:notify tag), unit：N/A<br>301~332: Set up every axis in position range, unit=mm<br>401~432: Set up every axis in position range reply parameter default setting (1:notify tag), unit：N/A<br>● VALUE =>Write info, value range0 ~ 2147483647, unit：N/A |
| Reply | ：N/A |
| Example | RP_INPOS(0,301,0.123);　/* Appoint first axis rapid in postion check range 0.123mm */ |

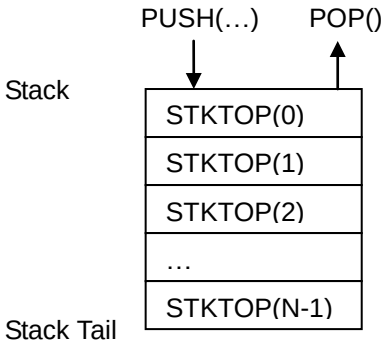
| | BKCMP(PATH,TYPE,VALUE)　SBK Compare　W |
|---|---|
| Description | ● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path. |

| | |
|---|---|
| | ● TYPE =>Type, value range****, unit：N/A.<br>0: Compare1, unit=mm<br>1: Compare2, unit=mm<br>● VALUE =>Write info, value range0 ~ 2147483647, unit：N/A |
| Reply | ：N/A |
| Example | BKCMP(0,1,0.123);  /* Appoint SBK compare2=0.123mm */ |

### 8.11 MACRO variables stack function

| | PUSH(VALUE) | Push to MACRO Variable Stack | W |
|---|---|---|---|
| Description | Maximum stack amount=50<br>● VALUE =>Write info, value rangedouble, unit：N/A | | |
| Reply | ：N/A | | |
| Example | #1=1.234<br>#2=5.678<br>STKCLR();  /* clear stacking info */<br>PUSH(#1);  /* push #1 variables to MACRO variables to stack, amount=1 */<br>PUSH(#2);  /* push #2 variables to MACRO variables to stack, amount=2*/<br>#11=STKTOP(0);  /* check 1 MACRO variables stack info, #11=5.678, amount=2*/<br>#12=STKTOP(1);  /* check 2 MACRO variables stack info, #12=1.234, amount=2*/<br>#13=STKTOP(2);  /* check 3 MACRO variables stack info,alarm #13=VACANT, amount=2*/<br>#21=POP();  /* withdraw MACRO variables stack last data, #21=5.678, amount=1*/<br>#22=POP();  /* withdraw MACRO variables stack last data, #22=1.234, amount =0*/<br>#23=POP();  /* MACRO variables stack has no data, #23=VACANT, amount =0*/<br>  <br> | | |

| | POP( ) | Pop info from MACRO variables stack | R |
|---|---|---|---|
| Description | N/A | | |
| Reply | Withdraw MACRO variables stack last data, this data will be deleted from the stack, unit：N/A<br>VACANT => MACRO variables stack has no data. | | |
| Example | Refer to PUSH function example | | |

| | STKTOP(STACK_NO) | Check info from MACRO variables stack | R |
|---|---|---|---|
| Description | ● STACK_NO =>Check info from stack's top, value range0~49, unit：N/A<br>  <br> | | |
| Reply | Check info from MACRO variables stack, data will not be deleted from stack, unit：N/A<br>Alarm：<br>-1 => Check serial number is over range | | |
| Example | Refer to PUSH function example | | |

| | STKCLR() | Clear data in stack | W |
|---|---|---|---|
| Description | N/A | | |
| Reply | N/A | | |
| Example | Refer to PUSH function example | | |

### 8.12 CCD function

| | W_CCD(PATH,TYPE,VALUE) | Setup CCD | W |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)；function will be ignored when preview & R-restart**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● TYPE =>Appoint type, refer to below description, value range****, unit：N/A<br>● VALUE =>Set up data, refer to below d Description, value range****, unit：N/A<br>0: CCD type(0:Keyence5xxx,1:Keyence7xxx), value range0~1, unit：N/A<br>1: Give CCD command, ex: switch to KEYENCE CCD 1 => "PW,IN,1"<br>2: Switch CCD program number, value range 0~999, unit：N/A<br>99: Set up overtime, value range1~2147483647, unit：ms<br>100: TCP/IP format => "IP,PORT",ex:"192.168.0.10,8500" | | |

<table>
<tr><td></td><td>101: RS232 format => "COM_PORT,BAUD,BITS& PARITY&STOP"<br>,ex:"COM1,9600,812"
<table>
<tr><td>RS232 Format</td><td>Description</td><td>Setting</td></tr>
<tr><td>COM_PORT</td><td>COM port</td><td>COM1,COM2</td></tr>
<tr><td>BAUD</td><td>Transmission speed(bps)</td><td>9600,19200,38400</td></tr>
<tr><td>BITS</td><td>Transmission bit(bit)</td><td>7,8</td></tr>
<tr><td>PARITY</td><td>Transmission even/odd check</td><td>0:N/A,1:Even,2:Odd</td></tr>
<tr><td>STOP</td><td>Transmission stop bit</td><td>1,2</td></tr>
</table></td></tr>
<tr><td>Reply</td><td>Alarm：<br>-3 => did not input string<br>-4 => communication interface did not have initialization<br>-6 => CCD device program is range<br>-7 => CCD device type setting wrong<br>-101 => TCP/IP initialization fail<br>-102 => TCP/IP string format error<br>-103 => TCP/IP did not connect<br>-104 => TCP/IP send string location=empty<br>-105 => TCP/IP send file fail<br>-106 => TCP/IP communication brake<br>-107 => TCP/IP receive file overtime<br>-108 => TCP/IP set up channel fail<br>-109 => TCP/IP switch IP fail<br>-110 => TCP/IP connect to Server fail<br>-201 => RS232 initialization fail<br>-202 => RS232 COM PORT setting fail<br>-203 => RS232 COM1 PORT not for INT, check P#40007=4<br>-204 => RS232 COM2 PORT not for INT, check P#40008=4<br>-205 => RS232 Transmission speed setting error<br>-206 => RS232 Transmission bit setting error<br>-207 => RS232 Transmission even/odd check setting error<br>-208 => RS232 Transmission stop bit setting error<br>-209 => RS232 send string location=empty<br>-210 => RS232 send file fail<br>-211 => RS232 receive file overtime<br>-212 => RS232 receive file too big<br>-213 => RS232 receive fil fail</td></tr>
<tr><td>Example</td><td>Example 1, use TCP/IP to connect with CCD：<br>W_CCD(0,100,"192.168.0.10,8500");  /* IP=192.168.0.10,PORT=8500 */</td></tr>
</table>

| | |
|---|---|
| | W_CCD(0,1,"T1");  /* wait CCD reply data */<br>#1 = PASER_CCD(0,1);  /* analyze CCD reply data，get 1 column value */<br>#2 = PASER_CCD(0,2);  /* analyze CCD reply data，get 2 column value */<br>…<br>#20 = PASER_CCD(0,20);  /* analyzeCCDReplydata，get 20 column value */<br><br>**Example2**, use RS232 to connect with CCD：<br>W_CCD(0,101,"COM1,9600,812");  /* COM1,BAUD=9600,BITS=8,PARITY=Even,STOP=2 */<br>W_CCD(0,1,"T1");  /* wait CCD reply data */<br>#1 = PASER_CCD(0,1);  /* analyze CCD reply data, get 1 column value */<br>#2 = PASER_CCD(0,2);  /* analyzeCCD reply data, get 2 column value e */<br>…<br>#20 = PASER_CCD(0,20);  /* analyze CCD reply data, get 20 column value */<br><br>**Example3**, switch CCD program number：<br>W_CCD(0,100,"192.168.0.10,8500");  /* IP=192.168.0.10,PORT=8500 */<br>W_CCD(0,2,2);  /* switch CCD program number 2 */<br>W_CCD(0,1,"T1");  /* wait CCD reply data */<br>#1 = PASER_CCD(0,1);  /* analyze CCD reply data, get 1 column value */<br>#2 = PASER_CCD(0,2);  /* analyze CCD reply data, get 2 column value e */<br>…<br>#20 = PASER_CCD(0,20);  /* analyze CCD reply data, get 20 column value */ |

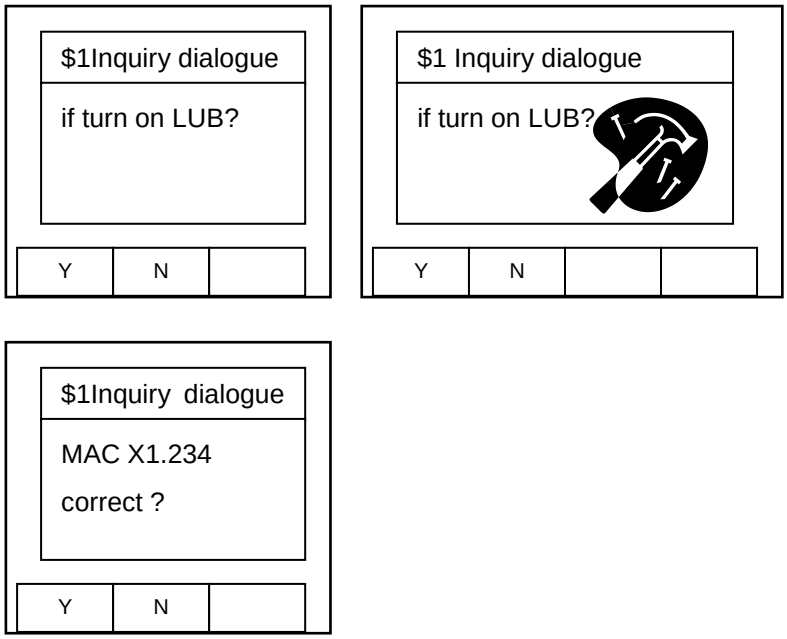
| | PASER_CCD(PATH,ITEM) | Get CCD reply data column value | R |
|---|---|---|---|
| Description | **This function will be ignored when preview & R-restart**<br>● PATH =>Path No., value range 0 ~6, unit：N/A.<br>0：Current path, 1~6：First path~6th path.<br>● ITEM =>Appoint CCD reply data column, value range1 ~20, unit：N/A | | |
| Reply | Get appointed CCD reply data column value, unit：N/A<br>Alarm：<br>-1 => over range | | |
| Example | Refer to W_CCD function example | | |

### 8.13 MACRO dialogue function

| | MSG_OK("TITLE","TEXT","FILE") | Message Dialogue | W |
|---|---|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)；This function will be ignored when preview & R-restart**<br>● TITLE =>Dialogue title, empty is allowed<br>● TEXT =>Dialogue message<br>● FILE =>Dialogue with graph name, empty is allowed<br>If above arguments string has MACRO variables, system will replace it to be the value automatically. Refer to 5.5 string variables replace value chapter. | | |
| Reply | Alarm：<br>-1 => Function arguments format error<br>-2 =>Title is over string length limit<br>-3 => Did not input dialogue message<br>-4 => Graph name is over string length limit | | |
| Example | **Example1**：<br>MSG_OK("Dialogue message","welcome to use our product",""); /* Dialogue did not have graph */<br>**Example2**：<br>MSG_OK("Dialogue message","welcome to use our product ","PIC.JPG"); /* Dialogue has graph */<br>**Example3**：<br>#2= R_MACH_COOR(0,1); /* if this path MACHINE coordinate first axis=1.234mm, #2Value=1.234 */<br>MSG_OK("Dialogue message ","MACHINE coordinateX#2correct!",""); /* show ”MACHINE coordinate X1.234 correct !” */ | | |


Example1 Example2 Example3

| | MSG_YES("TITLE","TEXT","FILE") Inquiry Dialogue R |
|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)；This function will be ignored when preview & R-restart**<br>● TITLE =>Dialogue title, empty is allowed<br>● TEXT =>Dialogue message<br>● FILE =>Dialogue with graph name, empty is allowed<br>If above arguments string has MACRO variables, system will replace it to be the value automatically. Refer to 5.5 string variables replace value chapter. |
| Reply | Read dialogue input button(0:N,1:Y)<br>Alarm：<br>-1 => Function arguments format error<br>-2 =>Title is over string length limit<br>-3 => Did not input dialogue message<br>-4 => Graph name is over string length limit |
| Example | **Example1**：<br>#1 = MSG_YES("Inquiry dialogue","if turn on LUB?",""); /* Dialogue did not have graph */<br>**Example2**：<br>#1 = MSG_YES("Inquiry dialogue "," if turn on LUB?","PIC.JPG"); /* Dialogue has graph */<br>**Example3**：<br>#2= R_MACH_COOR(0,1); /* if this path MACHINE coordinate first axis=1.234mm, #2Value=1.234 */<br>#1 = MSG_YES("Inquiry dialogue ","MACHINE coordinateX#2 correct?",""); /* show” MACHINE coordinate X1.234 correct ?” */ |

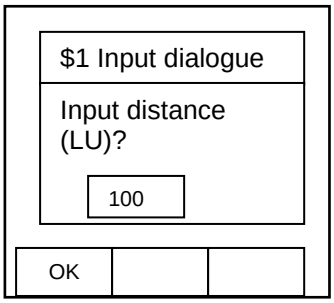
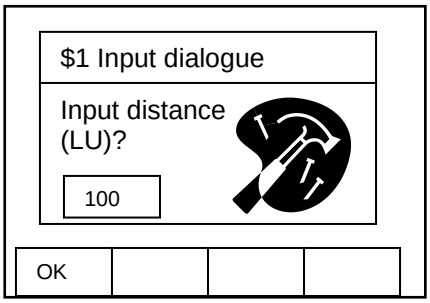
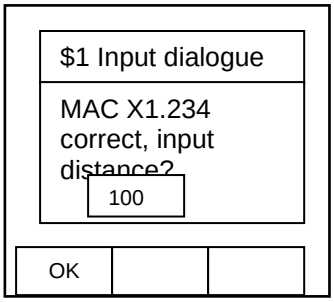



Example1 Example2

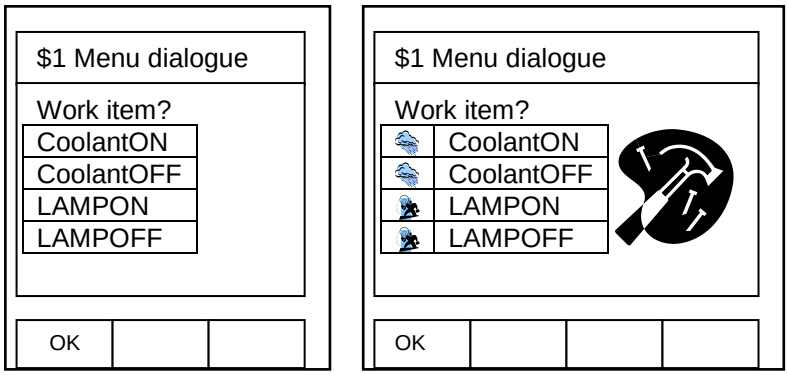
Example3

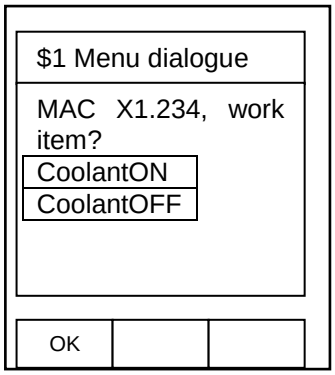
Result：When pressing”Y”button#1=1, when pressing ”N”button#1=0

| | INPUT("TITLE","TEXT","FILE",MIN,MAX,DEF) Input Value Dialogue R |
|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)；This function will be ignored when preview & R-restart**<br>● TITLE =>Dialogue title, empty is allowed<br>● TEXT =>Dialogue message<br>● FILE =>Dialogue with graph name, empty is allowed<br>● MIN =>Input Min.value, value range-2147483648 ~ 2147483647, unit：N/A<br>● MIN => Input Max.value, value range-2147483648 ~ 2147483647, unit：N/A<br>● DEF => Input default value, value range-2147483648 ~ 2147483647, unit：N/A<br>If above arguments string has MACRO variables, system will replace it to be the value automatically. Refer to 5.5 string variables replace value chapter. |
| Reply | Read dialogue input value, value range-2147483648 ~ 2147483647, unit：N/A<br>Alarm：<br>-1 => Function arguments format error<br>-2 =>Title is over string length limit<br>-3 => Did not input dialogue message<br>-4 => Graph name is over string length limit<br>-5 => Input value is over range |
| Example | **Example1**：<br>#1 = INPUT("Input dialogue","input moving distance(LU)?","",0,500,100); /* dialogue did not |

| | |
|---|---|
| | have graph */<br>**Example2**：<br>#1 = INPUT("Input dialogue","input moving distance(LU)?","PIC.JPG",0,500,100); /* dialogue has graph */<br>**Example3**：<br>#2= R_MACH_COOR(0,1); /* if this path MACHINE coordinate first axis=1.234mm, #2Value=1.234 */<br>#1 = INPUT("Input dialogue","MACHINE coordinateX#2 correct,input distance?","",0,500,100); /* show ” MACHINE coordinateX1.234 correct,input distance?” */<br>  <br>  <br>  <br>Example1 Example2<br>Example3<br>Result：After inputing, press”OK”, #1=100。 |

| | MENU("TITLE","TEXT","FILE",DEF) Menu Dialogue R |
|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)；This function will be ignored when preview & R-restart**<br>● TITLE =>Dialogue title, empty is allowed<br>● TEXT =>Dialogue message<br>● FILE =>Dialogue with graph name, empty is allowed<br>● DEF =>Input menu default value, value range1 ~ 10, unit：N/A<br>If above arguments string has MACRO variables, system will replace it to be the value automatically. Refer to 5.5 string variables replace value chapter. |
| Reply | Menu dialogue reply value item 1~10, value range1~10, unit：N/A<br>Alarm： |

| | |
|---|---|
| | -1 => Function arguments format error<br>-2 =>Title is over string length limit<br>-3 => Did not input dialogue message<br>-4 => Graph name is over string length limit<br>-5 => Did not have any list<br>-6 => Menu default value is over item numbers |
| Example | **Example1**：<br>MENU_ADD("Turn oncoolant",""); /* item did not have graph */<br>MENU_ADD("Turn offcoolant",""); /* item did not have graph */<br>MENU_ADD("Turn onworking lamp",""); /* item did not have graph */<br>MENU_ADD("Turn offworking lamp",""); /* item did not have graph */<br>#1 = MENU("Menu dialogue ","choose work item?","",1); /*Dialogue did not have graph */<br>**Example2**：<br>MENU_ADD("Turn oncoolant","WATER.JPG"); /* item has graph */<br>MENU_ADD("Turn offcoolant","WATER.JPG"); /* item has graph */<br>MENU_ADD("Turn onworking lamp","LIGHT.JPG"); /* item has graph */<br>MENU_ADD("Turn offworking lamp","LIGHT.JPG"); /* item has graph */<br>#1 = MENU("Menu dialogue "," choose work item?","PIC.JPG",1); /* Dialogue has graph */<br>**Example3**：<br>MENU_ADD("Turn oncoolant",""); /* item did not have graph */<br>MENU_ADD("Turn offcoolant",""); /* item did not have graph */<br>#2= R_MACH_COOR(0,1); /* if this path MACHINE coordinate first axis=1.234mm, #2Value=1.234 */<br>#1 = MENU("Menu dialogue ","MACHINE coordinateX#2, choose work item?"," ",1); /*Dialogue did not have graph */<br>  |

| | |
|---|---|
| |  <br>Example1 Example2<br>Example3<br>Result：when choosing item=Turn off coolant, #1=2 |

| | MENU_ADD("TEXT","FILE") | Menu Dialogue Setting | W |
|---|---|---|---|
| Description | **This function will be ignored when preview & R-restart**<br>This function is to add menu dialogue content, every call will add one menu item, maximum can be 10 items. When calling MENU(…)function, this new adding menu by this path will be cleared after poping up dialogue.<br>● TEXT =>Menu item<br>● FILE =>graph name, empty is allowed | | |
| Reply | Alarm：<br>-1 => Function arguments format error<br>-2 =>Title is over string length limit<br>-3 => Did not input dialogue message<br>-4 => Graph name is over string length limit<br>-5 => Menu item is over range | | |
| Example | Refer to MENU function example | | |

| | PLOT_ADD("FILE","LEGEND") | Curve Data File Name Setting | W |
|---|---|---|---|
| Description | **This function will be ignored when preview & R-restart**<br>This function is to add vurve data file name, every call will add one curve data, maximum one can be up to 10 datas. After calling PLOT_SHOW(…)function, pop up will clear all curve data names.<br>● FILE =>Curve data file name<br>● LEGEND =>Curve data description | | |
| Reply | Alarm：<br>-1 => function arguments format error<br>-2 => Did not input menu content<br>-3 => Curve data file name over string length limit | | |

| | |
|---|---|
| | -4 => Curve data string over string length limit<br>-5=> Curve data amount over range |
| Example | Refer to PLOT_SHOW function example |

| | PLOT_SHOW("TITLE","TEXT","XLB","YLB") 曲線圖框 W |
|---|---|
| Description | **This function will stop interpretation automatically.(Type Ⅰ)；this function will be ignoredat graph preview and R-restart.**<br>● TITLE =>frame title, empty is allowed<br>● TEXT => frame message, empty is allowed<br>● XLB => frame X axis tag, empty is allowed<br>● YLB => frame Y axis tag, empty is allowed<br>If above arguments string has MACRO variables, system will replace it to be the value automatically. Refer to 5.5 string variables replace value chapter. |
| Reply | Alarm：<br>-1 => function arguments input format error<br>-2 => frame message is over string length limit<br>-3 => frame X axis tag is over string length limit<br>-4 => frame Y axis tag is over string length limit<br>-5 => did not input curve data |
| Example | PLOT_ADD("DATA1.TXT","data1"); /* 1 data */<br>PLOT_ADD("DATA2.TXT","data2"); /* 2 data */<br>PLOT_SHOW("Curve frame","voltage data","time(ms)", "Votage(mV)"); |

### 8.14 MACRO write function

| | OPEN("FILE") Open file W |
|---|---|
| Description | ● FILE =>write file name.<br>Open file, when open file command is valid, PRINT command can be used. Multi-part program can not be opened at the same time, only one file at one time.<br>If above arguments string has MACRO variables, system will replace it to be the value automatically. Refer to 5.5 string variables replace value chapter. |
| Reply | Alarm：<br>-1 => function arguments input format error.<br>-2 => Do not input file name.<br>-3 => File name is over string length limitation.<br>-4 => Do not close previous file.<br>-5 => open file fail. |
| Example | **Usage Example 1**： |

| | |
|---|---|
| | @15=10<br>@20=2<br>#3=1<br>#20=9<br>OPEN("A12345");<br>PRINT("G0 Y@15 X4"); /* export to file result：G0 Y10 X4 */<br>PRINT("G0 Y@15 X#3"); /* export to file result：G0 Y10 X1 */<br>PRINT("X100 Z10 F200"); /* export to file result：X100 Z10 F200 */<br>PRINT("G0 Y(@15+15.5) X4"); /* export to file result：G0 Y(10+15.5) X4 */<br>PRINT("G0 Y#1 X4"); /* export to file result：G0 Y X4 */<br>CLOSE();<br>M30<br>**Usage Example 2**：<br>@15=10<br>@20=2<br>#3=1<br>#20=9<br>OPEN("A12345");<br>PRINT("G0 X4 Y(\#15+#20)"); /* export to file result：G0 X4 Y(#15+9) */<br>PRINT("G0 X\@20 Y\#15"); /* export to file result：G0 X@20 Y#15 */<br>PRINT("G0 X(@20+\#3) Y(\#30+@20)"); /* export to file result：G0 X(2+#3) Y(#30+2) */<br>PRINT("G0 X(\@15/\#3) Y(15+\@20)"); /* export to file result：G0 X(@15/#3) Y(15+@20) */<br>PRINT("G0 X(\#3+5) Y(\#30+#3)"); /* export to file result：G0 X(#3+5) Y(#30+1) */<br>PRINT("G0 X(\@15*5.45) Z(\#15-\#3)"); /* export to file result：G0 X(@15*5.45) Z(#15-#3) */<br>PRINT("G0 Y(\#15+@20) Z(\#15/\@20)"); /* export to file result：G0 Y(#15+2) Z(#15/@20) */<br>CLOSE();<br>M30 |

| | CLOSE() | Close file | W |
|---|---|---|---|
| Description | Close the file that was opened by OPEN command. File will be closed automatically when part program closes or system resets. After file closed, PRINT command is invalid. | | |
| Reply | N/A | | |
| Example | Refer to open function example | | |

| | PRINT("STRING") | Write file string | W |
|---|---|---|---|
| Description | **This function will stop interpretation automatically(Type Ⅰ)；function will be ignored when simulation and program restart.**<br>STRING =>Any string, if above arguments string has MACRO variables, system will replace it to | | |

| | |
|---|---|
| | be the value automatically. Refer to 5.5 string variables replace value chapter. |
| Reply | Alarm：<br>-1 => function arguments input format error.<br>-2 => do not open file. |
| Example | Refer to open function example |

# 9 Variables

This chapter is to explain whole series of INT mode.

Version： Milling machine INT mode **mill_int_Ver03.01.01**

## 9.1 Variables range menu

| No. | Name | MACRO | HMI | Memory |
|---|---|---|---|---|
| #00 | NULL | Read only | Read only | N |
| #01~#199 | Local variables | Read/write | Read only | N |
| @0000 | NULL | Read only | Read only | N |
| @0001~999 | End-user global variables | Read/write | Read/write | Y |
| @1000~1999 | System global variables | Only system MACRO Read/write | By user authority Read/write | N |
| @5000~5999 | Shared path end-user global variables | Read/write | Read/write | Y |
| @6000~6999 | Shared path system global variables | Only system MACRO Read/write | By user authority Read/write | N |

## 9.2 Number range classification

(1). Local Variables：

#00：always empty.

#01 ~ #199：read/write

For every level's program, there are 199 local variables. If you cancel this level's program, variables will be canceled. But if you press RESET, it will go back to main program level, and main level's local variables was determined by P#50004.1. If you press RESET, you can not cancel this. But if you reboot the controller, every level's local variables will be canceled.

(2). Global Variables：

All levels of the program shared this common variable at single path.

@0000：Always empty.

@0001 ~ @0999：User global variables range, only suitable for its path, value clear will be determined by P# 50005.1 whether booting or pressing RESET.

@1000 ~ @1999：System global variables range, only suitable for its path system MACRO, value clear will be determined by pressing RESET.

@5000 ~ @5999：User global variables range, suitable for all paths, value clear will be determined by P#

50006 whether booting or pressing RESET.

@6000 ~ @6999：System global variables range, suitable for all path system MACRO, value clear will be determined by pressing RESET.

### 9.3 Local variables and global variables BIT Function Usage Description

You may need to have a lot of tags to help MACRO procedure judge, this tag will use a lot of local variables,therefore we provide local/global variable BIT to help you solve this problem, example as below：

- BIT usage format =#i.j, also in the i=variables number range；j=2 digits，Value range=00~31
- #1.00 =>Means#1 Bit0
- #1.03 => Means #1 Bit3
- #1.3or #1.30=> Means #1 Bit30
- #1.32=>Bit uses over 32 apponts, system will show alarm：MACRO variables numbers is over range.
- #(1+0.31)=># Parentheses do not support Bit, left side means #1
- #1=123 => Left side means #1 appointed value 123
- #1.00=123 =>Bit operation, right side is bigger than 0=True. Left side means #1 appointed value 1
- #1.02=456=> Bit operation, right side is bigger than 0=True. Left side means #1 appointed value 4
- #3 = #1.02 =>#3 appointed =#1 Bit2. Left side means #3 appointed value 1
- #3.04 = #1.02 =>#3 Bit4 appointed =#1 Bit 2. Left side means#3 appointed value 16
- #1= R_REG(5) =>If R[5]=123. Left side means#1appointed value 123
- #1.00= R_REG(5) => If R[5]=123. Left side means #1 appointed value 1
- #1.02= R_REG(5) => If R[5]=123. Left side means #1 appointed value 4
- #3 = R_REG_BIT(5,2)  => If R[5]=5, #3 appointed=R[5] Bit2. Left side means #3 appointed value1
- #3.04 = R_REG_BIT(5,2) => If R[5]=5, #3 Bit4 appointed =R[5] Bit2. Left side means #3 appointed value 16

### 9.4 Note

Betweem "/*" and "*/" , those input will be canceled.

Example：

```
/* test1 */;
G00 X10. /* test 2 */;
/* test3 */ G01 Y20.;
G01 X10. Y20.; /* test
```

### 9.5 Variables to value function in string

Version：M**ill_int_Ver03.01.34**

When editing program or MACRO, you may need sting, ex：G65 “string” or：ALARM("string"). If you want tomix MACRO variables in this string(value of global variable’@’, local variable’#’), you can use this function to show some MACRO calculating value or system info in it. It’s easy for user to understand program or MACRO current situation from string, example as below：

#1= R_MACH_COOR(0,1);  /* If this path MACHINE coordinate 1 axis= 1.234mm, #1 value=1.234 */
ALARM("MACHINE coordinate X#1mm");  /* Alarm 610000, string：” MACHINE coordinate X1.234mm” */

From above example, you can see the sting is within” “, and #1 is MACRO variables. When running to ALARM MACRO command, system will send alarm, and show string” MACRO coordinate X1.234 correct”, #1 will change to be value automatically.

**Valid digits：**

MACRO variables with () and numbers can appoint valid digits for MACRO variables to change to be value, rules are：

- When this is integer value, it will add a zero in front of this value.
  EX：When #1=12, string will be
  ” MACHINE coordinate X#1(4)mm”  =>  ” MACHINE coordinate X0012 mm”
  EX：When #1=1234567, if the value is over valid digits, it will not do any changes.
  ” MACHINE coordinate X#1(4)mm”  =>  ” MACHINE coordinate X1234567mm”
- When this is not integer value, it will add zero after this value to be a valid value.
  EX：When #1=12.34, string will be
  ” MACHINE coordinate X#1(4) mm”  =>  ” MACHINE coordinate X12.3400 mm”
  EX：When #1=12.345678, string will be
  ” MACHINE coordinate X#1(4) mm”  =>  ” MACHINE coordinate X12.3456 mm”

**Stop changing digit：**

You can add ’\’ with MACRO variables to make it stop changing value.

EX：When #1=12.345678, string will be

”MACRO calculating result \#1 = #1”  =>  ” MACRO calculating result #1 = 12.345678”

# 10 System M Code

This chapter is to explain whole series of INT mode.

Version： M**ill_int_ Ver03.01.56**

### 10.1 INT use M code

| Program Command | Description | System M Code |
|---|---|---|
| SYS_SUB_CALL P_<br>SYS_SUB_CALL “string”<br>SYS_SUB_CALL “string” P_ | Call system sub routine<br>P：calling file name number(O+ P_No)<br>“String”：any sting<br>“String”P_：calling file name(string+4digitsP_NO) | 99900001 |
| AXIS_FREE | Free path axis | 99900003 |
| PATTERN_BEGIN | Pattern begin | 99900004 |
| PATTERN_END K_ D_ E_ P_ | Pattern end<br>K：If there is argument, it means relative MACRO layers. If not, it’s current layer.<br>D：Appoint pattern start letters<br>E：Appoint starting letters’ number<br>P：Appoint insert file’s number(sys_single_pattern+P_No) | 99900005 |
| DIS_HMIPLOT_ON | HMI ON | 99900006 |
| DIS_HMIPLOT_OFF | HMI OFF | 99900007 |
| MUTE_ON | Mute ON | 99900008 |
| MUTE_OFF | Mute OFF | 99900009 |
| RESTART_OK | Restart OK | 99900010 |
| RES_RESIDUAL | Run return to rest SBK manually | 99900012 |
| HIDE_SHIFT_ON | Shift OFF, make G51,G51.1,G68,G41,G42 invalid. Usually use on changing tool MACRO command which did not need to shift. | 99900013 |
| HIDE_SHIFT_OFF | Shift ON | 99900014 |
| SBK_OFF | SBK OFF | 99900015 |
| SBK_ON | SBK ON | 99900016 |
| FIX_RAPID_OR_OFF | Fixed RAPID OVERRIDE 100% OFF | 99900017 |
| FIX_RAPID_OR_ON | Fixed RAPID OVERRIDE 100% ON | 99900018 |
| FIX_CUT_OR_OFF | Fixed CUT OVERRIDE 100% OFF | 99900019 |
| FIX_CUT_OR_ON | Fixed CUT OVERRIDE 100% ON | 99900020 |
| PRIORITY_OFF | PRIORITY OFF. Lower priority when meeting stop INT in a path or SBK full | 99900021 |

| | | |
|---|---|---|
| PRIORITY_ON | PRIORITY ON | 99900022 |

### 10.2 Kernel system M code

| Part program command | Description | System M Code |
|---|---|---|
| PRO_STOP | Notify OP stop( the same to M0) | -3 |
| OPT_STOP | Notify OP selective stop(the same to M1) | -4 |
| MDI_FIN | Notify OP, MDI end | -5 |
| INS_MACRO_FIN | Notify OP, PLC insert part program end | -12 |
| PROG_END | Program end, system reset | -13 |
| INTO_MREADY | Program end, NC ready | -14 |
| MDI_TO_RY | MDI ready | -24 |
| MDI_TO_BS | MDI stop | -25 |
| FIX_AUTOMODE_ON | Fixed mode selection is auto mode | -26 |
| FIX_AUTOMODE_OFF | Return mode control by R bit | -27 |

# 11 Appendix

## 11.1 Milling machine system MACRO list

| System MACRO File Name | FunctionDescription |
|---|---|
| sys_func_mdi1 | MDI MACRO |
| sys_func_mr1 | Retun MACRO manually with movement |
| sys_func_mr_residual | Retun MACRO manually without movement |
| sys_func_prog_restart1 | Program Restart Call System MACRO |
| sys_func_cycle_cancel | G80 Call System MACRO automatically after canceling cycle |
| sys_modal_g73 | G73 Rapid Peck Drilling Cycle |
| sys_modal_g74 | G74 Left-Handed Screw Thread Tapping Cycle |
| sys_modal_g76 | G76 Fine Boring Cycle |
| sys_modal_g81 | G81 Drilling Cycle/Spot Boring |
| sys_modal_g82 | G82 Drilling Cycle/Counter Boring |
| sys_modal_g83 | G83 Peck Drilling Cycle |
| sys_modal_g84 | G84 Right-Handed Screw Thread Tapping Cycle |
| sys_modal_g85 | G85 Reaming Cycle |
| sys_modal_g86 | G86 Boring Cycle |
| sys_modal_g87 | G87 Back Boring Cycle |
| sys_modal_g88 | G88 Boring Cycle |
| sys_modal_g89 | G89 Reaming Cycle |
| sys_macro_g10 | G10 Data Input Setting |
| sys_macro_g27 | G27 Return to Origin Check |
| sys_macro_g28 | G28 Return to the First Reference Point |
| sys_macro_g29 | G29 Return from the First Reference Point 00 |
| sys_macro_g30 | G30 Auto Return to the 2nd, 3rd and 4th Reference Points |
| sys_macro_g31 | G31 Skip Signal Abort Block |
| sys_macro_g53 | G53 Rapid Positioning of Machine Coordinate System |
| sys_macro_g92 | G92 Coordinate Setting 00 |
| sys_macro_m0 | Program stop |
| sys_macro_m1 | Optional stop |
| sys_macro_m2 | End of program |
| sys_macro_m30 | Program rewind |
| sys_macro_m2000 | Waiting M code |
| sys_macro_m2010 | Wait for free axis command |