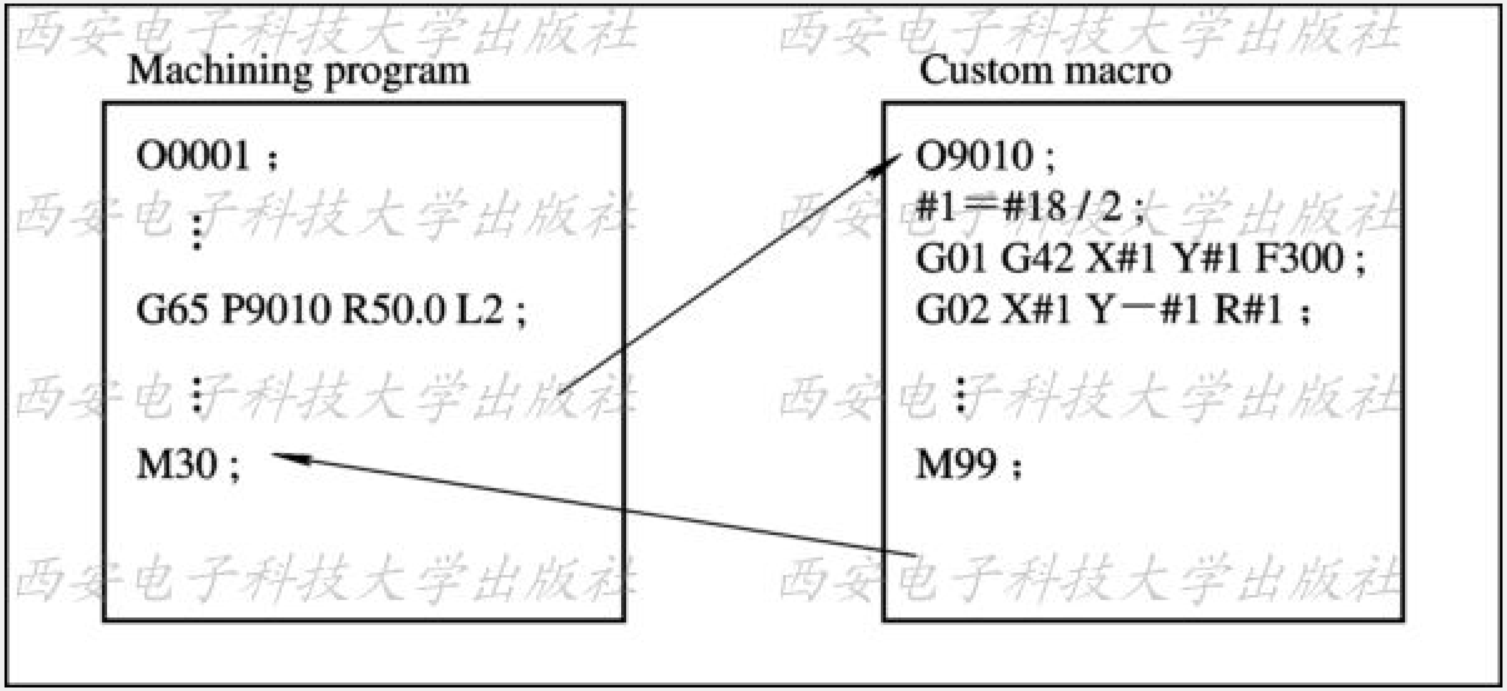
15 CUSTOM MACRO

BACK

Although subprograms are useful for repeating the same operation, the custom macro function also allows use of variables, arithmetic and logic operations, and conditional branches for easy development of general programs such as pocketing and user-defined canned cycles. A machining program can call a custom macro with a simple command, just like a subprogram.



```
O0001 ;
  ⋮
G65 P9010 R50.0 L2 ;
  ⋮
M30 ;
```

```
O9010 ;
#1=#18 / 2 ;
G01 G42 X#1 Y#1 F300 ;
G02 X#1 Y-#1 R#1 ;
  ⋮
M99 ;
```

Fig.15.0 Custom macro

15.1 VARIABLES

An ordinary machining program specifies a G code and the travel distance directly with a numeric value; examples are G100 and X100.0.

With a custom macro, numeric values can be specified directly or using a variable number. When a variable number is used, the variable value can be changed by a program or using operations on the MDI panel.

```
＃1=＃2+100 ;
G01 X＃1 F300 ;
```

Explanation

· **Variable representation**

When specifying a variable, specify a number sign (＃) followed by a variable number. Personal computers allow a name to be assigned to a variable, but this capability is not available for custom macros.

［Example］ ＃1

An expression can be used to specify a variable number. In such a case, the expression must be enclosed in brackets.

［Example］ ＃［＃1+＃2-12］

· **Types of variables**

Variables are classified into four types by variable number.

Table 15.1 Types of variables

Variable number	Type of variable	Function
#0	Always null	This variable is always null. No value can be assigned to this variable.
#1 -#33	Local variables	Local variables can only be used within a macro to hold data such as the results of operations. When the power is turned off, local variables are initialized to null. When a macro is called, arguments are assigned to local variables.
#100 -#149 (#199) #500 -#531 (#999)	Common variables	Common variables can be shared among different macro programs. When the power is turned off, variables #100 to #149 are initialized to null. Variables #500 to #531 hold data even when the power is turned off. As an option, common variables #150 to #199 and #532 to #999 are also available. However, when these values are using.
#1000 -	System variables	System variables are used to read and write a variety of NC data items such as the current position and tool compensation values.

NOTE

Common variables ＃150 to ＃199 and ＃532 to ＃999 are optional.

· **Range of variable values**

Local and common variables can have value 0 or a value in the following ranges:

-10E47 to -10E-29

10E-29 to 10E47

If the result of calculation turns out to be invalid, an P/S alarm No.111 is issued.

· **Omission of the decimal point**

When a variable value is defined in a program, the decimal point can be omitted.

［Example］

When ＃1=123; is defined, the actual value of variable ＃1 is 123.000.

· Referencing variables

To reference the value of a variable in a program, specify a word address followed by the variable number. When an expression is used to specify a variable, enclose the expression in brackets.

［Example］ G01X［＃1+＃2］F＃3;

A referenced variable value is automatically rounded according to the least input increment of the address.

［Example］

When G00X＃1; is executed on a 1/1000 mm CNC with 12.3456 assigned to variable ＃1, the actual command is interpreted as G00X12.346; .

To reverse the sign of a referenced variable value, prefix a minus sign (-) to ＃.

［Example］ G00X-＃1;

When an undefined variable is referenced, the variable is ignored up to an address word.

［Example］

When the value of variable ＃1 is 0, and the value of variable ＃2 is null, execution of G00X＃1Y＃2; results in G00X0; .

· Common custom macro variables for tow paths (twopath control)

For twopath control, macro variables are provided for each path. Some common variables, however, can be used for both paths, by setting parameters No.6036 and 6037 accordingly.

· **Undefined variable**

When the value of a variable is not defined, such a variable is referred to as a “null” variable. Variable ＃0 is always a null variable. It cannot be written in, but it can be read.

Ⅰ. Quotation

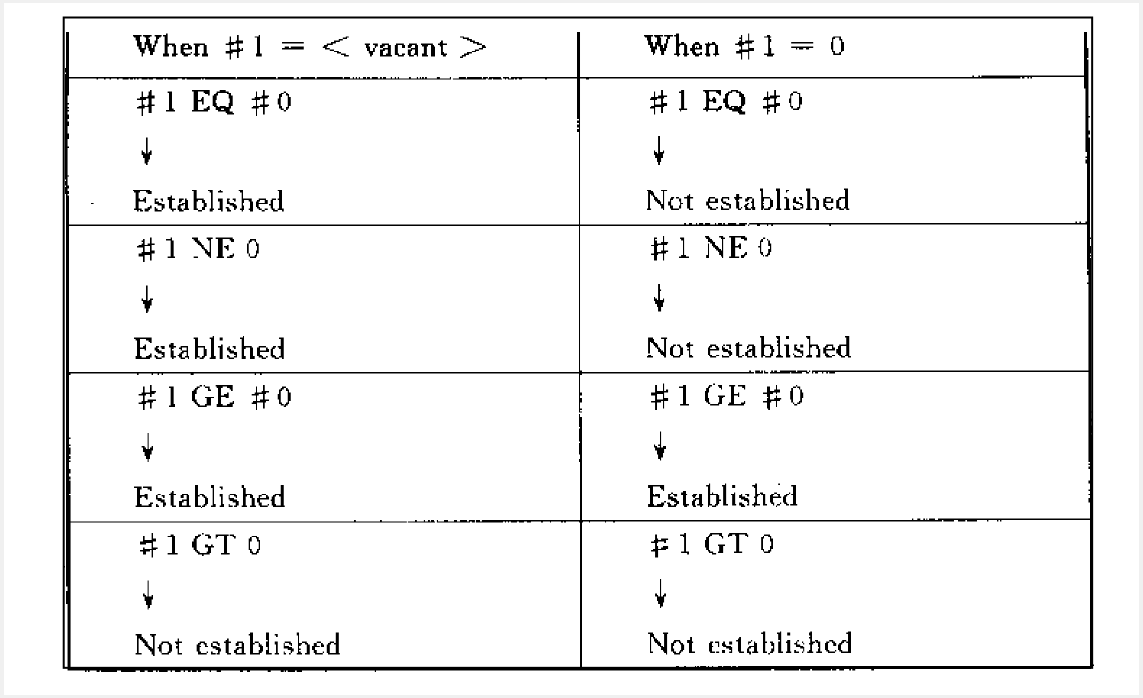
When an undefined variable is quotated, the address itself is also ignored.

［Example］ When ＃1 is 0 and ＃2 is null, the result of executing G00 X＃1 Y＃2; will be the same as when G00 X0; is executed.

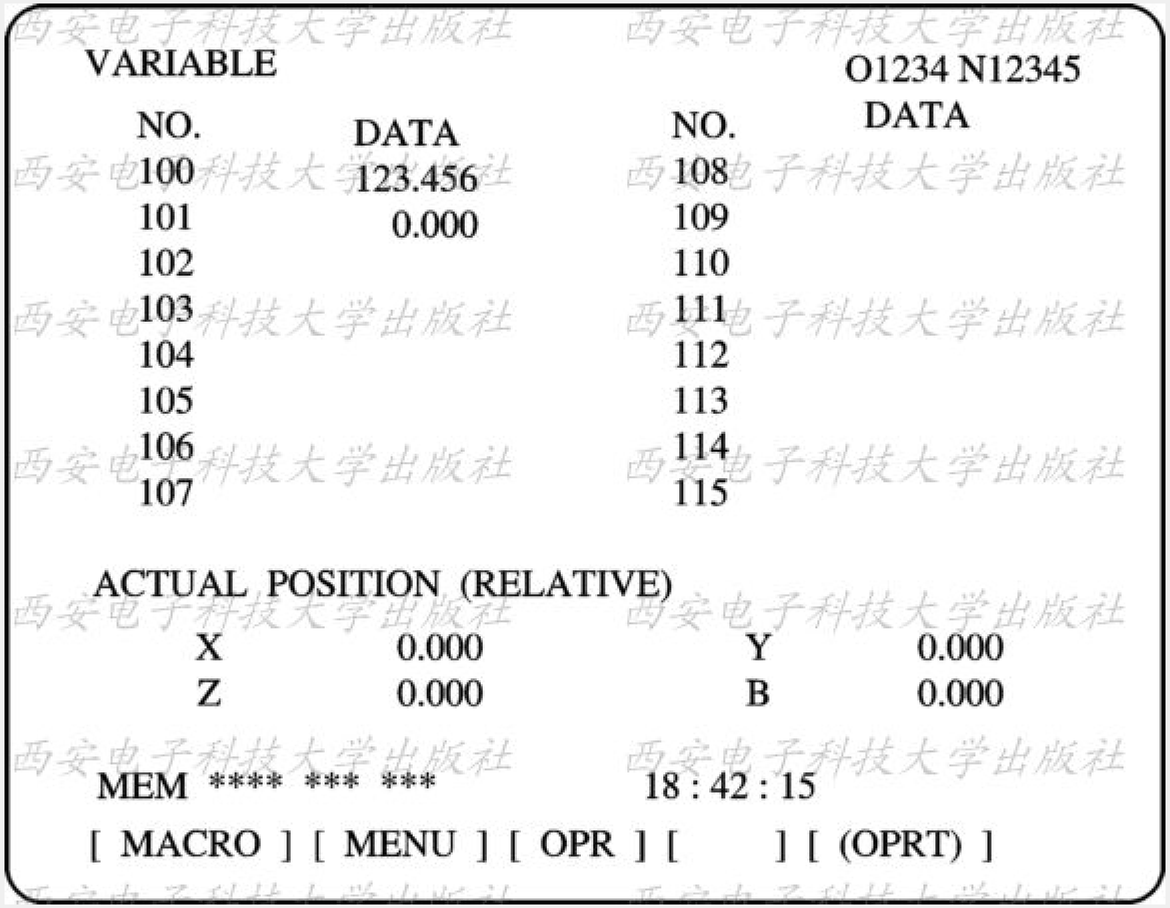
Ⅱ. Operation <vacant> is the same as 0 except when replaced by <vacant>

When #1 = < vacant >	When #1 = 0
#2 = #1 ↓ #2 = < vacant >	#2 = #1 ↓ #2 = 0
#2 = #1 * 5 ↓ #2 = 0	#2 = #1 * 5 ↓ #2 = 0
#2 = #1 + #1 ↓ #2 = 0	#2 = #1 + #1 ↓ #2 = 0

Ⅲ. Conditional expressions <vacant> differs from 0 only for EQ and NE



When #1 = < vacant >	When #1 = 0
#1 EQ #0 ↓ Established	#1 EQ #0 ↓ Not established
#1 NE 0 ↓ Established	#1 NE 0 ↓ Not established
#1 GE #0 ↓ Established	#1 GE #0 ↓ Established
#1 GT 0 ↓ Not established	#1 GT 0 ↓ Not established



```
VARIABLE                                              O1234 N12345
   NO.          DATA                    NO.         DATA
   100          123.456                 108
   101            0.000                 109
   102                                  110
   103                                  111
   104                                  112
   105                                  113
   106                                  114
   107                                  115

ACTUAL  POSITION  (RELATIVE)
      X          0.000                  Y          0.000
      Z          0.000                  B          0.000

MEM  ****  ***  ***                18 : 42 : 15
[ MACRO ] [ MENU ] [ OPR ] [       ] [ (OPRT) ]
```

Fig.15.1 Variables on CRT page

(i) When the value of a variable is blank, the variable is null.

(ii) The mark ********* indicates an overflow (when the absolute value of a variable is greater than 99999999) or an underflow (when the absolute value of a variable is less than 0.0000001).

Limitations

Program numbers, sequence numbers, and optional block skip numbers cannot be referenced using variables.

［Example］ Variables cannot be used in the following ways:

```
O＃1;
/＃2G00X100.0;
N＃3Y200.0;
```

15.2 SYSTEM VARIABLES

System variables can be used to read and write internal NC data such as tool compensation values and current position data. Note, however, that some system variables can only be read. System variables are essential for automation and general purpose program development.

Explanations

· **Interface signals**

Signals can be exchanged between the programmable machine controller (PMC) and custom macros.

Table 15.2 System variables for interface signals

Variable number	Function
#1000 − #1015 #1032	A 16-bit signal can be sent from the PMC to a custom macro. Variables #1000 to #1015 are used to read a signal bit by bit. Variable #1032 is used to read all 16 bits of a signal at one time.
#1100 − #1115 #1132	A 16-bit signal can be sent from a custom macro to the PMC. Variables #1100 to #1115 are used to write a signal bit by bit. Variable #1132 is used to write all 16 bits of a signal at one time.
#1133	Variable #1133 is used to write all 32 bits of a signal at one time from a custom macro to the PMC. Note, that values from −99999999 to +99999999 can be used for #1133.

For detailed information, refer to the connection manual.

· **Tool compensation values**

Tool compensation values can be read and written using system variables. Usable variable numbers depend on the number of compensation pairs, whether a distinction is made between geometric compensation and wear compensation, and whether a distinction is made between tool length compensation and cutter compensation. When the number of compensation pairs is not greater than 200, variables ＃2001 to ＃2400 can also be used.

Table 15.3　System variables for tool compensation memory A

Compensation number	System variable
1	#10001 (#2001)
⋮	⋮
200	#10200 (#2200)
⋮	⋮
999	#10999

Table 15.4　System variables for tool compensation memory B

Compensation number	Geometry compensation	Wear compensation
1	#11001 (#2201)	#10001 (#2001)
⋮	⋮	⋮
200	#11200 (#2400)	#10200 (#2200)
⋮	⋮	⋮
999	#11999	#10999

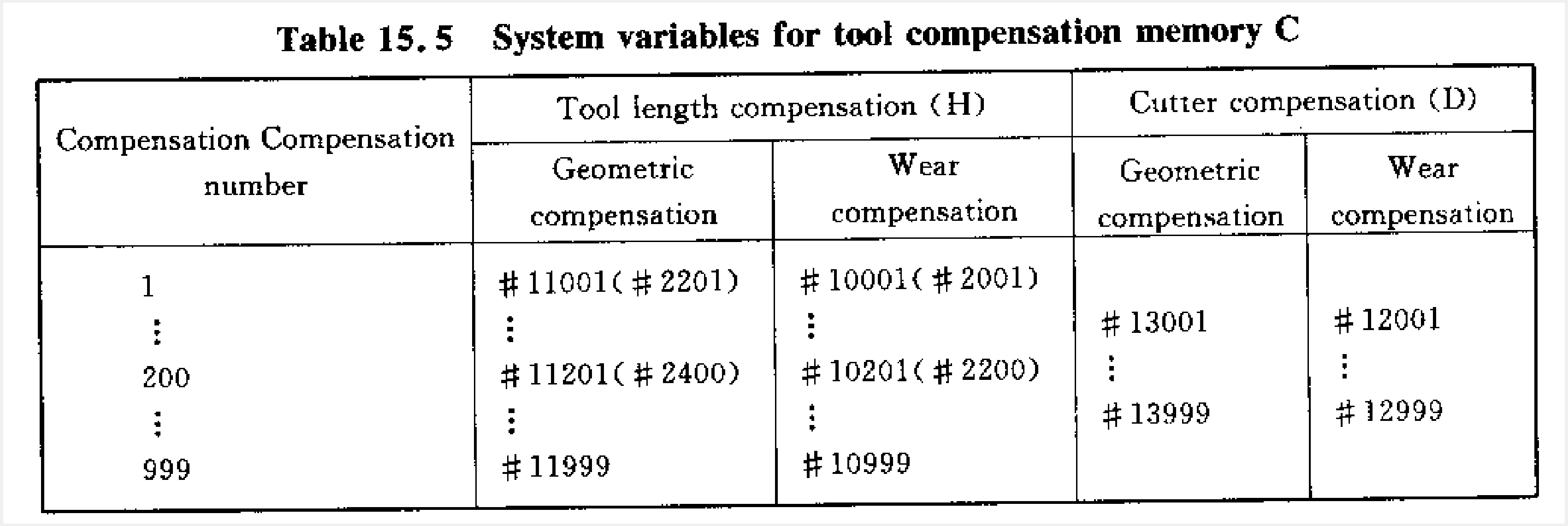


Table 15.5 System variables for tool compensation memory C

Compensation Compensation number	Tool length compensation (H)		Cutter compensation (D)	
	Geometric compensation	Wear compensation	Geometric compensation	Wear compensation
1	#11001(#2201)	#10001(#2001)	#13001	#12001
⋮	⋮	⋮	⋮	⋮
200	#11201(#2400)	#10201(#2200)		
⋮	⋮	⋮		
999	#11999	#10999	#13999	#12999

· Macro alarms

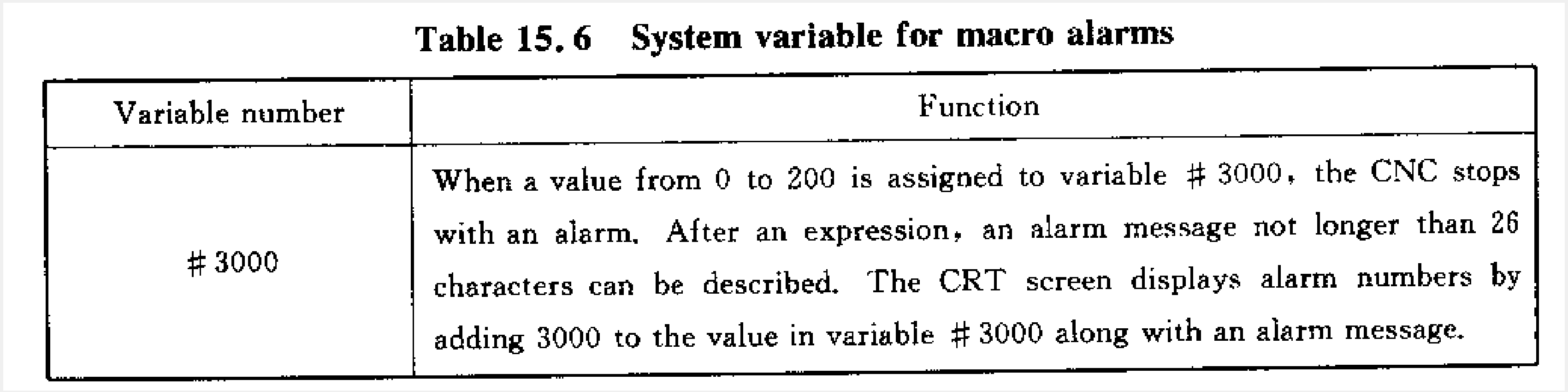


Table 15.6 System variable for macro alarms

Variable number	Function
#3000	When a value from 0 to 200 is assigned to variable #3000, the CNC stops with an alarm. After an expression, an alarm message not longer than 26 characters can be described. The CRT screen displays alarm numbers by adding 3000 to the value in variable #3000 along with an alarm message.

· **Time information**

［Example］　＃3000=1(TOOL NOT FOUND);

The alarm screen displays “3001 TOOL NOT FOUND.”

Time information can be read and written.

Table 15.7 System variables for time information

Variable number	Function
#3001	This variable functions as a timer that counts in 1-millisecond increments at all times. When the power is turned on, the value of this variable is reset to 0. When 2147483648 milliseconds is reached, the value of this timer returns to 0.
#3002	This variable functions as a timer that counts in 1-hour increments when the cycle start lamp is on. This timer preserves its value even when the power is turned off. When 9544.371767 hours is reached, the value of this timer returns to 0.
#3011	This variable can be used to read the current date (year/month/day). Year/month/day information is converted to an apparent decimal number. For example, September 28, 1994 is represented as 19940928.
#3012	This variable can be used to read the current time (hours/minutes/seconds). Hours/minutes/seconds information is converted to an apparent decimal number. For example, 34 minutes and 56 seconds after 3 p.m. is represented as 153456.

15.3 ARITHMETIC AND LOGIC OPERATION

The operations listed in Table 15.8 can be performed on variables. The expression to the right of the operator can contain constants and/or variables combined by a function or operator. Variables #j and #k in an expression can be replaced with a constant. Variables on the left can also be replaced with an expression.

Table 15.8 Arithmetic and logic operation

Function	Format	Remarks
Definition	#i=#j	
Sum	#i=#j+#k;	
Difference	#i=#j−#k;	
Product	#i=#j*#k;	
Quotient	#i=#j/#k;	
Sine	#i=SIN[#j];	An angle is specified in degrees. 90 degrees and 30 minutes is represented as 90.5 degrees.
Cosine	#i=COS[#j];	
Tangent	#i=TAN[#j];	
Arctangent	#i=ATAN[#j]/[#k];	
Square root	#i=SQRT[#j];	
Absolute value	#i=ABS[#j];	
Rounding off	#i=ROUND[#j];	
Rounding down	#i=FIX[#j];	
Rounding up	#i=FUP[#j];	
OR	#i=#j OR #k;	A logical operation is performed on binary numbers bit by bit.
XOR	#i=#j XOR #k;	
AND	#i=#j AND #k;	
Conversion from BCD to BIN	#i=BIN[#j];	Used for signal exchange to and from the PMC.
Conversion from BIN to BCD	#i=BCD[#j];	

Explanations

· **Angle units**

The units of angles used with the SIN, COS, TAN, and ATAN functions are degrees. For example, 90 degrees and 30 minutes is represented as 90.5 degrees.

· **ATAN function**

After the ATAN function, specify the lengths of two sides separated by a slash. A result is found where 0-result-360.

［Example］ When ＃1=ATAN［1］/［-1］, the value of ＃1 is 135.0.

· **ROUND function**

(ⅰ) When the ROUND function is included in an arithmetic or logic operation command, IF statement, or WHILE statement, the ROUND function rounds off at the first decimal place.

［Example］ When ＃1=ROUND［＃2］; is executed where ＃2 holds 1.2345, the value of variable ＃1 is 1.0.

(ⅱ) When the ROUND function is used in NC statement addresses, the ROUND function
rounds off the specified value according to the least input increment of the address.

［Example］ Creation of a drilling program that cuts according to the values of variables ＃1 and ＃2, then returns to the original position.

Suppose that the increment system is 1/1000 mm, variable ＃1 holds 1.2345, and variable ＃2 holds 2.3456. Then,

G00 G91 X-＃1;　　　Moves 1.235 mm

G01 X-＃2 F300;　Moves 2.346 mm

G00 X［＃1+＃2］;　Since 1.2345 + 2.3456 = 3.5801, the travel distance is 3.580, which does not return the tool to the original position

This difference comes from whether addition is performed before or after rounding off.

G00X-［ROUND［＃1］+ROUND［＃2］］ must be specified to return the tool to the original position.

· **Rounding up and down to an integer**

With CNC, when the absolute value of the integer produced by an operation on a number is greater than the absolute value of the original number, such an operation is referred to as rounding up to an integer.

Conversely, when the absolute value of the integer produced by an operation on a number is less than the absolute value of the original number, such an operation is referred to as rounding down to an integer.

Be particularly careful when handling negative numbers.

［Example］ Suppose that ＃1=1.2 and ＃2=-1.2.

When ＃3=FUP［＃1］ is executed, 2.0 is assigned to ＃3.

When ＃3=FIX［＃1］ is executed, 1.0 is assigned to ＃3.

When ＃3=FUP［＃2］ is executed, -2.0 is assigned to ＃3.

When ＃3=FIX［＃2］ is executed, -1.0 is assigned to ＃3.

· **Abbreviations of arithmetic and logic operation commands**

When a function is specified in a program, the first two characters of the function name can be used to specify the function.

［Example］

ROUND → RO

FIX → FI

· **Priority of operations**

Ⅰ. Functions

Ⅱ. Operations such as multiplication and division (*, /, AND)

Ⅲ. Operations such as addition and subtraction (+, -, OR, XOR)

$$\text{(Example)}\ \#1=\underbrace{\#2+\underbrace{\#3*\underbrace{\text{SIN}[\#4]}_{\boxed{1}}}_{\boxed{2}}}_{\boxed{3}};$$

$\boxed{1}$, $\boxed{2}$ and $\boxed{3}$ indicate the order of operations.

Fig.15.3(a) Priority of operations

· **Bracket nesting**

Brackets are used to change the order of operations. Brackets can be used to a depth of five levels including the brackets used to enclose a function.

When a depth of five levels is exceeded, P/S alarm No.118 occurs.

(Example) $$\#1=\underbrace{\mathrm{SIN}[\underbrace{\underbrace{[\underbrace{[\underbrace{\#2+\#3}_{\boxed{1}}]*\#4}_{\boxed{2}}+\#5]}_{\boxed{3}}*\#6}_{\boxed{4}}]}_{\boxed{5}};$$

$\boxed{1}$ to $\boxed{5}$ indicate the order of operations.

Fig.15.3(b) Bracket nesting