Designation and function of <blank> is classified in the following two versions A and B. The control is set for either version. Versions A and B cannot be changed.

		Version A	Version B
1	Concept of "#0"	. No conception of #0. . Commanding #0 causes alarm.	. #0 defined as variables of <blank> . Commanding #0 at the left-hand side of the equation.
2	Variable <blank> is commanded in the replacement equation.	. Where #2 is <blank> , command #3=#2; means #3=0.	. Where #2 is <blank> command #3=#2; means #3= <blank>.
3	Variable <blank> is commanded in the part program.	. Where #2 is <blank>, command 600 x #2; is equivalent to command 600 x 0;	. Where #2 is <blank> command 600 x #2; is equivalent to 600; (Adress is Ignored.)
4	Variable <blank> is commanded in the condition of EQ and NE.	. Where #2 is <blank>, #3 is 0 ① Condition "IF #3 EQ #2" is established ② Condition "IF #3 NE #2" is not established.	. Where #2 is <blank> #3 is 0 ① Condition "IF #3 EQ #2" is not established. ② Condition "IF #3 EQ #2 is established.
5	Others	#3=#[#0+#0] #3=#2 * #0; #3=#0 + #0; #3=#0/#0; #3=5*#0; In these commands, #3=0, #3=2-#0; means #3=2 #3=5/#0; causes alarm. Blank in the replacement described above is treated as "0." . Condition IF#3 GE#2 is established when #2 and #3 are <blank> , or #2 is 0 and #3 is <blank> . . Condition IF #3LT #2 is not established when #2 and #3 are <blank> , or #2 is <blank>, and #3=0.	

2.11.5 OPERATION COMMANDS

Various operations can be performed between variables and between variables and constants. The operation expression is represented in the form of #i = ⟨expression⟩, in which ⟨expression⟩ is a general arithmetic operational expression produced by combining variables and constants with operators and functions. The available opearations and functions are as follows. Instead of #j and #k, constants may be used.

(1) Variable Definition and Replacement

#i = #j ··· definition, replacement.
#i = #[#j + #k] ...

(2) Add-Type Operations

#i = #j + #k	···	Sum.
#i = #j - #k	···	Difference.
#i = #j OR #k	···	Logical sum (for each of 32 bits).
#i = #j XOR #k	···	Exclusive logical sum (for each of 32 bits).

(3) Multiply-Type Operations

#i = #j * #k	···	Product.
#i = #j / #k	···	Quotient.
#i = #j AND #k	···	Logical product (for each of 32 bits).

Note: In OR, XOR, or AND operation, the variable value (or constant) is converted into the binary 32-bit equivalent and the operation is performed on each bit.

(4) Functions

#i = SIN [#j]	···	Sine (in degrees).
#i = COS [#j]	···	Cosine (in degrees).
#i = TAN [#j]	···	Tangent (in degrees).
#i = ATAN [#j/#k]	···	Arctangent (in degrees).
#i = SQRT [#j]	···	Square root.
#i = ABS [#j]	···	Absolute value.
#i = BIN [#j]	···	Convert from BCD.
#i - BCD [#j]	···	Convert into BCD.
#i = ROUND [#j]	···	Produce integer by rounding,
#i = FIX [#j]	···	Truncate the fractions.
#i = FUP [#j]	···	Raise the fractions to a unit.