Compare Instructions CMP, EQU, GEQ, GRT, LEQ, LES, LIM, MEQ, NEQ

Using Compare Instructions

The comparison instructions let you compare values using an expression or a specific comparison instruction. Table 3.A lists the available compare instructions.

Table 3.A
Available Compare Instructions

If You Want to:	Use the Instruction:	On Page:
Compare values based on an expression	CMP	3-2
Test whether two values are equal	EQU	3-5
Test whether one value is greater than or equal to a second value	GEQ	3-5
Test whether one value is greater than a second value	GRT	3-6
Test whether one value is less than or equal to a second value	LEQ	3-6
Test whether one value is less than a second value	LES	3-7
Test whether one value is between two other values	LIM	3-7
Pass two values through a mask and test whether they are equal	MEQ	3-9
Test whether one value is not equal to a second value	NEQ	3-10

Important: You can compare values of different data types, such as floating point and integer. You should use BCD and ASCII values for display purposes. If you enter BCD or ASCII values, the processor treats those values as integers. For example, if the value at N7:2 is 10 (decimal) and the value at D9:3 is 10 (BCD), the comparison of N7:2 = D9:3 evaluates as false. The 10 in BCD translates to 0000 0000 0001 0000; the 10 in decimal translates to 0000 0000 0000 1010.

The parameters you enter are program constants or logical addresses of the values you want to compare.

For more information on the operands (and valid data types/values of each operand) used by the instructions discussed in this chapter, see Appendix C.

Using Arithmetic Status Flags

The arithmetic status flags are in word 0 bits 0-3 in the processor status file (S). Monitor these bits if you perform an arithmetic function within the CMP instruction. Table 3.B lists the status bits:

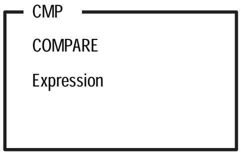
Table 3.B
Arithmetic Status Bits

This Bit:	Description:
S:0/0	Carry (C)
S:0/1	Overflow (V)
S:0/2	Zero (Z)
S:0/3	Sign (S)

Compare (CMP)

The CMP instruction compares values and performs logical comparisons.

Description:



The CMP instruction is an input instruction that performs a comparison on arithmetic operations you specify in the expression. When the processor finds the expression is true, the rung goes true. Otherwise, the rung is false. With Enhanced PLC-5 processors, you can enter multiple operands (complex expression).

The execution time of a CMP instruction is longer than the execution time of one of the other comparison instructions (e.g., GRT, LEQ, etc.). A CMP instruction also uses more words in your program file than the corresponding comparison instruction.

Entering the CMP Expression

The expression defines the operations you want to perform. Define the expression with operators and addresses or program constants. With Enhanced PLC-5 processors, you can enter complex expressions. Table 3.C lists valid operations for an expression; the following list provides guidelines for writing expressions.

- Operators (symbols) define the operations
- Addresses can be direct, indirect, or indexed address(es) (must be word level)
- With Enhanced PLC-5 processors, program constants can be integer or floating-point numbers (if you enter octal values, use a leading &O; if you enter hexadecimal values, use a leading &H; if you enter binary values, use a leading &B)

Table 3.C
Valid Operations for Use in a CMP Expression

Type	Operator	Description	Example Operation
Comparison	=	equal to	if A = B, then ...
	<>	not equal to	if A <> B, then ...
	<	less than	if A < B, then ...
	<=	less than or equal to	if A <= B, then ...
	>	greater than	if A > B, then ...
	>=	greater than or equal to	if A >= B, then ...
Arithmetic	+	add	2 + 3 Enhanced PLC-5 processor: 2 + 3 + 7
	–	subtract	12 – 5
	*	multiply	5 * 2 PLC-5/30, -5/40, -5/60, -5/80: 6 * (5 * 2)
	| (vertical bar)	divide	24 | 6
	–	negate	– N7:0
	SQR	square root	SQR N7:0
	**	exponential (x to the power of y)	10**3 (Enhanced PLC-5 processors only)
Conversion	FRD	convert from BCD to binary	FRD N7:0
	TOD	convert from binary to BCD	TOD N7:0

Determining the Length of an Expression

Enhanced PLC-5 processors support complex instructions (up to a total of 80 characters, including spaces and parentheses). Depending on the operator, the processor inserts characters before/after the operator in your expression to format the expression for easier interpretation. Use Table 3.D to determine the number of characters each operator uses in an expression.

Important: You cannot enter floating point numbers in scientific notation with negative exponents in complex expressions. Instead, use the decimal equivalent or put the number in a floating point file and use the data address in the complex expression.

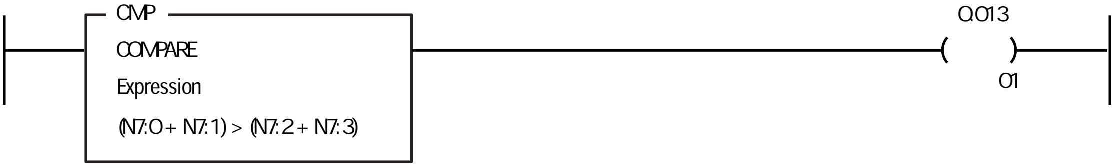
With the CMP instruction, a maximum of 80 characters of the expression can be displayed. If the expression you enter is near this 80 character maximum, when you accept the rung containing the instruction, the processor may expand it beyond 80 characters. When you try to edit the expression, only the first 80 characters are displayed and the rung is displayed as an error rung. The processor does contain the complete expression, however, and the instruction runs properly.

To avoid this display problem, export the processor memory file and make your edits in the PC5 text file. Then import this text file. For more information on importing/exporting processor memory files, see your programming manual.

Table 3.D
Character Lengths for Operators

This Operation:	Using this Operator:	Uses this Number of Characters:
math binary	+, –, *, |	3
	OR, **	4
	AND, XOR	5
math unary	– (negate)	2
	LN	3
	FRD, TOD, DEG, RAD, SQR, NOT, LOG, SIN, COS, TAN, ASN, ACS, ATN	4
comparative	=, <, >	3
	<>, <=, >=	4

Example:

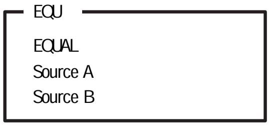


The CMP instruction tells an Enhanced PLC-5 processor: if the sum of the values in N7:0 and N7:1 is greater than the sum of the values in N7:2 and N7:3, set output bit O:013/01. (The total number of characters used in this expressions is 3)

For more information on entering complex expressions, see chapter 4.

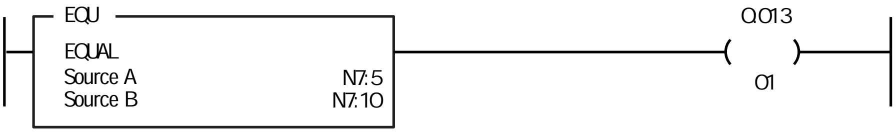
Equal to (EQU)

Description:



Use the EQU instruction to test whether two values are equal. Source A and Source B can either be values or addresses that contain values.

Example:

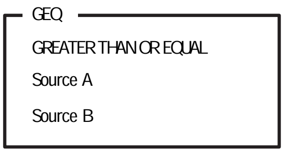


If the value in N7:5 is equal to the value in N7:10, set output bit O:013/01.

Floating point values are rarely absolutely equal. If you need to determine the equality of floating point values, use the LIM instruction (instead of the EQU). For information on the LIM instruction, see page 3-7.

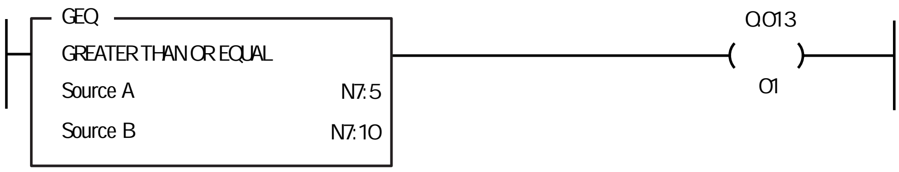
Greater than or Equal to (GEQ)

Description:



Use the GEQ instruction to test whether one value (Source A) is greater than or equal to another value (Source B). Source A and Source B can be values or addresses that contain values.

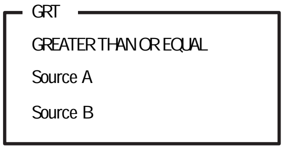
Example:



If the value in N7:5 is greater than or equal to the value in N7:10, set output bit O:013/01.

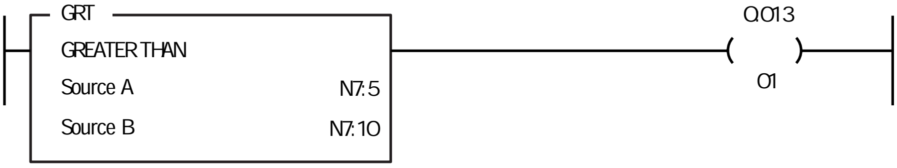
Greater than (GRT)

Description:



Use the GRT instruction to test whether one value (Source A) is greater than another value (Source B). Source A and Source B can either be values or addresses that contain values.

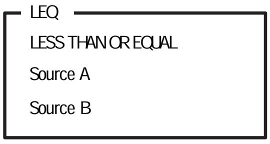
Example:



If the value in N7:5 is greater than the value in N7:10, set output bit O:013/01.

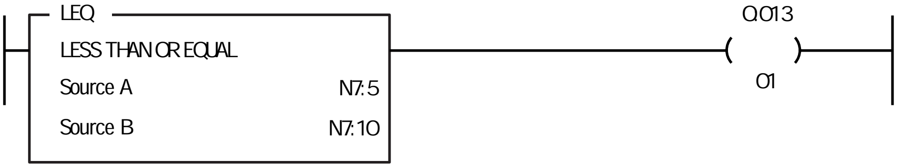
Less than or Equal to (LEQ)

Description:



Use the LEQ instruction to test whether one value (Source A) is less than or equal to another value (Source B). Source A and Source B can either be values or addresses that contain values.

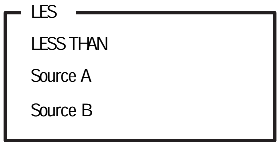
Example:



If the value in N7:5 is less than or equal to the value in N7:10, set output bit O:013/01.

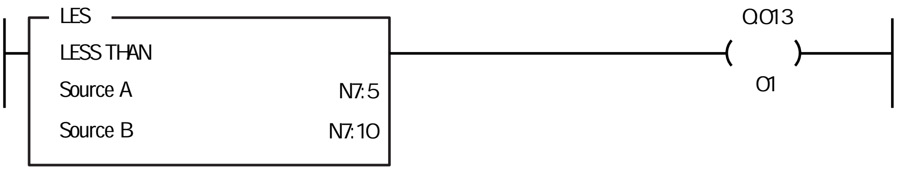
Less than (LES)

Description:



Use the LES instruction to test whether one value (Source A) is less than another value (Source B). Source A and Source B can be values or addresses that contain values.

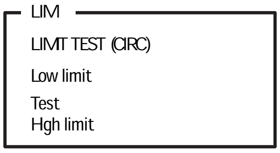
Example:



If the value in N7:5 is less than the value in N7:10, set output bit O:013/01.

Limit Test (LIM)

Description:



The LIM instruction is an input instruction that tests for values inside of or outside of a specified range. The instruction is false until it detects that the test value is within certain limits. Then the instruction goes true. When the instruction detects that the test value goes outside certain limits, it goes false.

You can use the LIM instruction to test if an analog input value is within specified limits.

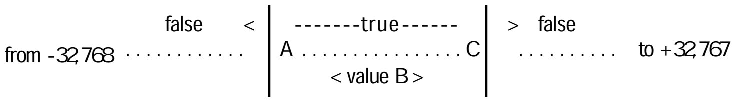
Entering Parameters

To program the LIM instruction, you must provide the processor with the following:

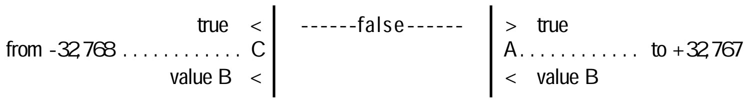
Parameter:	Definition:
Low Limit	a constant or an address from which the instruction reads the lower range of the specified limit range. The address contains an integer or floating-point value.
Test Value	the address that contains the integer or floating-point value you examine to see whether the value is inside or outside the specified limit range.
Hgh Limit	a constant or an address from which the instruction reads the upper range of the specified limit range. The address contains an integer or floating-point value.

LIM Example Using Integer:

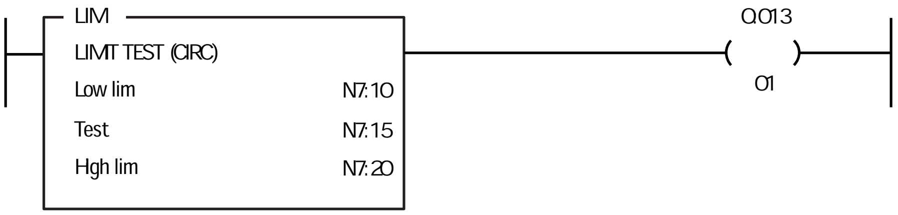
- **If value Low Limit ≤ value High Limit:** When the processor detects that the value of B (Test) is equal to or between limits, the instruction is true; if value Test is outside the limits, the instruction is false.



- **If value Low Limit ≥ value High Limit:** When the processor detects that the value of Test is equal to or outside the limits, the instruction is true; if value Test is between, but not equal to either limit, the instruction is false.



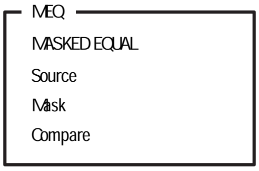
Example (when the Low Limit is less than the High Limit):



If the value in N7:15 is greater than or equal to the value in N7:10 and less than or equal to the value in N7:20, set output bit O:013/01.

Mask Compare Equal to (MEQ)

Description:



The MEQ instruction is an input instruction that compares a value from a source address with data at a compare address, and allows portions of the data to be masked. If the data at the source address matches the data at the compare address bit-by-bit (less masked bits), the instruction is true. The instruction goes false as soon as it detects a mismatch.

You can use the MEQ instruction to extract (for comparison) bit data such as status or control bits from an element that contains bit and word data.

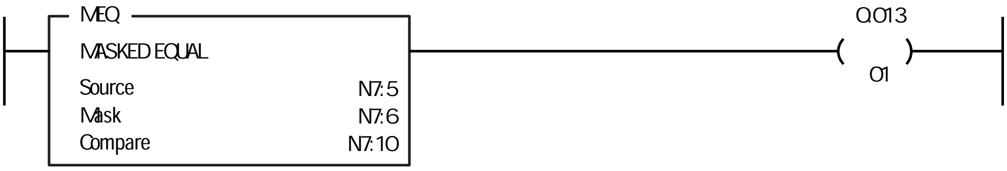
Entering Parameters

To program the MEQ instruction, you must provide the processor with the following:

Parameter:	Definition:
Source	a program constant or data address from which the instruction reads an image of the value. The source remains unchanged.
Mask	specifies which bits to pass or block. A mask passes data when the mask bits are set (1); a mask blocks data when the mask bits are reset (0). The mask must be the same element size (16-bits) as the source and compare address. In order for bits to be compared, you must set (1) mask bits; bits in the compare address that correspond to zeros (0) in the mask are not compared. If you want the ladder program to change the mask value, store the mask at a data address. Otherwise, enter a hexadecimal value for a constant mask value. If you enter a hexadecimal value that starts with a letter (such a F800), enter the value with a leading zero. For example, type `0F800`
Compare	specifies whether you want the ladder program to vary the compare value, or a program constant for a fixed reference. Use 16-bit elements, the same as the source.

Example:

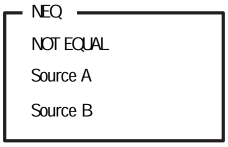
```
Source      01010101 01011111
Mask        11111111 11110000
Compare     01010101 0101xxxx
Result      The instruction is true because
            reference bits xxxx are not compared.
```



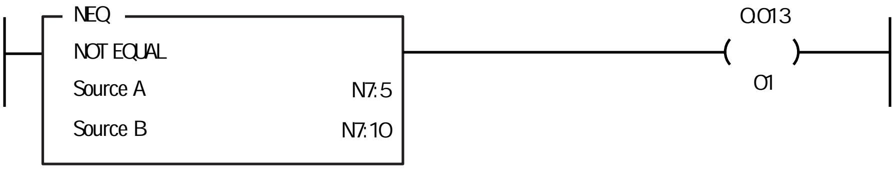
The processor passes the value in N7:5 through the mask in N7:6. It then passes the value in N7:10 through the mask in N7:6. If the two masked values are equal, set output bit O:013/01.

Not Equal to (NEQ)

Description: Use the NEQ instruction to test whether two values are not equal. Source A and Source B can be values or addresses.



Example:



If the value in N7:5 is not equal to the value in N7:10, set output bit O:013/01.