CEGELEC
INDUSTRIAL CONTROLS

Bit by Bit ANDING, ORING and XORING of 2 Groups of 16 Bit words

ANDGRP T24
ORGRP T25
XORGRP T26

DESCRIPTION

ANDGRP performs logical AND operations, ORGRP logical OR operation, and XORGRP logical XOR operations, upon the bits of two input groups, placing the results into a third group.

These group functions treat a block of adjacent data table locations as a one dimensional array of bits. The block of table locations is referred to as a GROUP. You can refer to any element within a group by its bit number, commencing at bit 0 for the first bit.

Each of these group functions allows you to either:

- perform the logical operation between the bits of two separate groups of the same size, or:
- perform the logical operation between the bits of an input group, taken a 16-bit word at a time, with the same 16-bit value.

FEATURES

- User-specified input and output Group locations.
- Option of logically combining each element of the array with the same 16-bit number.
- User-programmable Group length.

SPECIFICATION

Control language:-

```
          ANDGRP
X---------SPEC.---VALUE----------Q
            T24      Zm
         +-<   >
         |
Y--------+

          ORGRP
X---------SPEC.---VALUE----------Q
            T25      Zm
         +-<   >
         |
Y--------+

          XORGRP
X---------SPEC.---VALUE----------Q
            T26      Zm
         +-<   >
         |
Y--------+
```

Table 1 **Program rung data**

Sym	Location	Use	Range
X	Rung input (upper branch)	Internal GEM80 address of 1st input Group	Addresses of data table locations obtained by using the LOCATE (S20) Special Function
Y	Rung input (lower branch)	Internal GEM80 address of 2nd input Group, or an immediate value if GL is negative	
Zm	Rung location	Start of parameter list (3 locations)	Any write enabled table usable by Special Functions
Q	Rung output	Fault indication	-1 for no errors

Table 2 **Value data**

Sym	Location	Use	Range
EC	Zm	Error code	See Table 3
GL	Zm+1	Group length Negative group length specifies input Y as an immediate value	1 to 4096* data table locations
SA	Zm+2	Internal GEM80 address of output Group	Address of write enabled data table location obtained by using the LOCATE (S20) Special Function

Table 3 Fault conditions

Condition	Result Q	Code in Zm	
		Numeric Value	States ON
PROGRAMMING ERRORS			
Group length invalid	0	3	0 & 1
Invalid Group address (a) 1st Input Group not in table usable by special. (b) 1st input Group extends beyond and of data table area.	0	5	0 & 2
Invalid Group address (a) 2nd input Group not in table usable by Special (b) 2nd input Group extends beyond end of data table area.	0	9	0 & 3
Invalid Group address (a) output Group not in write-enabled table usable by special (b) output Group extends beyond end of data table area.	0	17	0 & 4

PROGRAMMING RULES

1. A AND B-TABLES

On current models of GEM80 Controller that implement Group functions, you are not permitted to use A and B-table locations with the dame number, e.g. you can't have both A7 and B7 in your program. When the Controller compiles your ladder diagram program, it checks it through to make sure that you do not have overlapping A-table and B-table references. If it finds any, it will produce an error message, and your program won't run.

However, the compiler can't do the same sort of checking where you use Group or Array Special functions, as some of the table locations you are using will be implied, and will not appear in your ladder diagram explicitly.

If you want to use a block of A-table locations as an input or output Group or Array, you must therefore be careful that the block of locations you specify does not overlap any B-table locations in use. If it does, the contents of your B-table locations may be affected.

Similarly, if you want to use a block of B-table locations as an input or output Group or Array, you must be careful that the block of locations you specify does not overlap any A-table location in use as the contents of these A-table locations could be affected.

2. FLOATING POINT TABLES

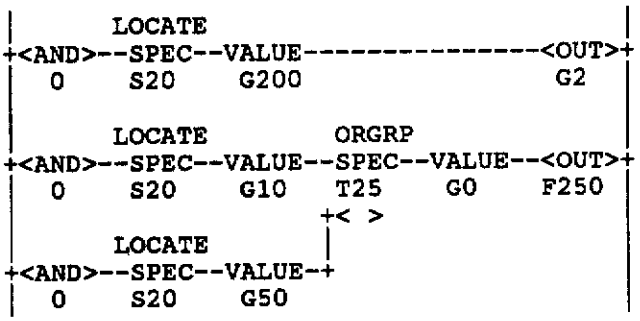
You cannot use ANDGRP, ORGRP or XORGRP with floating point table locations in Q-table. If you try to use them this way, you will get an error message, and your program won't compile.

APPLICATION NOTES

PROGRAM EXAMPLE 1

You want to logically OR the 48 bits contained in three adjacent data table locations G10, G11 and G12 with 48 bits stored in locations G50, G51 and G52. You want to store the respective results in table locations G200, G201 and G202.

To do this, use the following range and associated table values.



G0 = Error code.
G1 = Array length = 3.
G2 = Address of G200 supplied by first rung.
F100 = Fault indication

Typical Results:

Bits 0-15	Input 1 Input 2 Output	G10 1111111111111111 G50 0100111100101001 G200 1111111111111111
Bits 16-31	Input 1 Input 2 Output	G11 1000101010001010 G50 0101000100010010 G201 1101101110011010
Bits 32-47	Input 1 Input 2 Output	G21 0110001001010000 G50 0100000000001000 G202 0110001001011000

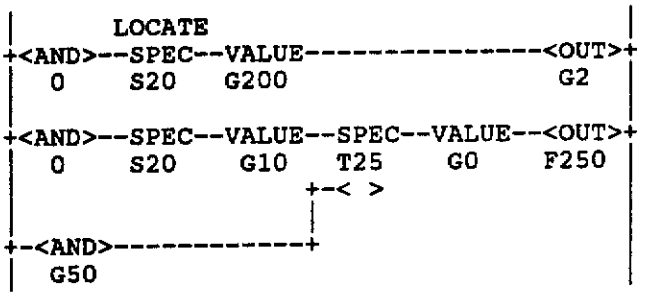
You could express these results in hexadecimal as:

Input 1		Input 2		Input 3	
G10	FFFFH	G50	4F29H	G200	FFFFH
G11	8A8AH	G51	5112H	G201	DB9AH
G12	6250H	G52	4008H	G202	6258H

PROGRAM EXAMPLE 2

You want to logically OR each of the 16 bit words contained in three adjacent data table locations G10, G11 and G12 with 16 bits stored in location G50. You want to store the respective results in table locations G200, G201, G202.

To do this, use the following range and associated table values:



G0 = Error code.
G1 = Array length = -3 (the negative sign indicated that the group input is to be ORed with a common 16-bit word).
G2 = Address of G200 supplied by first rung.
F100 = Fault indication

Typical Results:

Bits 0-15	Input 1	G10 1111111111111111
	Input 2	G50 0000000000001111
	Output	G200 1111111111111111
Bits 16-31	Input 1	G11 1111000000000000
	Input 2	G50 0000000000001111
	Output	G201 1111000000001111
Bits 32-47	Input 1	G21 0000000100000000
	Input 2	G50 0000000000001111
	Output	G202 0000000100001111

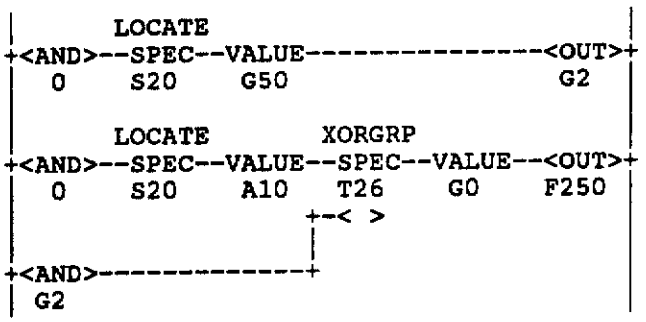
You could express these results in hexadecimal

Input 1		Input 2		Input 3	
G10	FFFFH	G50	000FH	G200	FFFFH
G11	F000H			G201	F00FH
G12	0100H			G202	010FH

PROGRAM EXAMPLE 3

You want to find whether any input signals read into table locations A10, A11 and A12 have changed since the last program scan. To do this, you can logically XOR the 48 bits contained in the three adjacent input data table locations A10, A11 and A12 with 48 bits stored in locations G50, G51, and G52, and then to place the respective results back into table locations G50, G51 and G52 ready for the next scan.

To do this, use the following rungs and associated table values:



G0 = Error code.
G1 = Array length = 3.
G2 = Address of G50 supplied by first rung.
F100 = Fault indication.

Typical Results:

Bits 0-15	Input 1 Input 2 Output	A10 0000000010100001 G50 0000000000100000 G50 0000000010000001
Bits 16-31	Input 1 Input 2 Output	A11 1000101010001010 G51 1000101010000010 G51 0000000000001000
Bits 32-47	Input 1 Input 2 Output	A12 0110001001010000 G52 0110000001010000 G52 0000001000000000

You could express these results in hexadecimal as:

Input 1		Input 2		Input 3	
A10	00A1H	G50	0020H	G50	0081H
A11	8A8AH	G51	8A82H	G51	0008H
A12	6250H	G52	6050H	G52	0200H

CEGELEC
INDUSTRIAL CONTROLS

Kidsgrove, Stoke-on-Trent, ST7 1TW, England
Tel: (0782) 783511 Fax: (0782) 776329

Publication No. T347 Issue 4