Addition, Subtraction, Multiplication and Division of two Arrays of Numbers

ADDARAY T12
SUBARAY T13
MULARAY T14
DIVARAY T15

DESCRIPTION

The ADDARAY, SUBARAY, MULARAY and DIVARAY Special Functions perform the same integer arithmetic operations as the standard functions ADD and SUB and the Special Functions MULT (T10) and DIV (T11), but on one-dimensional arrays of numbers rather than on single numbers.

The ADDARAY, SUBARAY, MULARAY and DIVARAY Special Functions treat a block of adjacent data table locations as a one dimensional array of numerical values. You can refer to any element within an array using an index that commences at 0 for the first word.

The ADDARAY function allows you to either:
- add the elements from two input arrays of the same size, and place their sums in a third array, or:
- add the same value to all the elements of an input array, placing the resulting sums in an output array.

Similarly, the SUBARAY function allows you to either:
- subtract the elements of one input array from the elements of a second input array of the same size, and place the corresponding results in a third array, or:
- subtract the same value from all the elements of an input array, placing the results in an output array.

The MULARAY function allows you to either:
- multiply together the elements of two input arrays of the same size, and place the corresponding products in a third array, or:
- multiply all the elements of an input array by the same value, placing the products in an output array.

Similarly, the DIVARAY function allows you to either:
- divide the elements of one input array by the corresponding elements of a second array of the same size, and place the corresponding quotients in a third array, or:
- divide all the elements of an input array by the same value, placing the quotients in an output array.

With DIVARAY, there is also an option for you to store the remainders from the divisions in a fourth array when you need to.

All the array Special Functions covered by this Data Sheet produce results that are:
- rounded to the nearest integer;
- limited to be within the range ±32767 if they would otherwise overflow.

Where results are limited, an array Special Function indicates this by setting the appropriate error code, and specifying the position of the first overflow on the rung output.

FEATURES

- User-specified Input and output Array locations.
- Option of operating on all locations of the input array.
- User-programmable Array length.
- Limiting of results.

SPECIFICATION

Control language:

```
         ADDARAY
X -------SPEC.---VALUE-------- Q
           T12      Zm
         +-<   >
         |
Y -------+


         SUBARAY
X -------SPEC.---VALUE-------- Q
           T13      Zm
         +-<   >
         |
Y -------+


         MULARAY
X -------SPEC.---VALUE-------- Q
           T14      Zm
         +-<   >
         |
Y -------+


         DIVARAY
X -------SPEC.---VALUE-------- Q
           T15      Zm
         +-<   >
         |
Y -------+
```

Table 1 Program rung data

Sym	Location	Use	Range
X	Rung input (upper branch)	Internal GEM 80 address of 1st input Array	Addresses of data table locations obtained by LOCATE (S20) Special Function
Y	Rung input (lower branch)	Internal GEM 80 address of 2nd input Array or immediate value	
Zm	Value location	Start of parameter list (3 locations) (4 locations for DIVARAY only)	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	Array length. Negative array length specifies input Y as an immediate value	1 to 4096 data table locations
SA	Zm + 2	Internal GEM 80 address of output Array	Address of write-enabled data table location obtained by using the LOCATE (S20) Special Function
*RA	Zm + 3	Internal GEM 80 address of optional remainder Array	0 for no remainders; as SA above if you require remainders

*DIVARAY only

Table 3 Fault conditions

Condition	Result Q	Code in Zm Numeric Value	Code in Zm States ON
PROGRAMMING ERRORS			
Array length invalid.	0	3	0 & 1
Invalid array address: (a) 1st input Array not in table usable by Special Functions. (b) 1st input Array extends beyond end of data table area.	0	5	0 & 2
Invalid Array address (a) 2nd input Array not in table usable by Special Functions. (b) 2nd input Array extends beyond end of data table area.	0	9	0 & 3
Invalid Array address: (a) output Array not in write-enabled table usable by Special Functions; (b) output Array extends beyond end of data table area.	0	17	0 & 4
(DIVARAY only) Invalid Array address: (a) remainder Array not in write-enabled table usable by Special Functions; (b) remainder Array extends beyond end of data table area.	0	33	0 & 5
(DIVARAY only) Divide by zero error. Q = position of 1st error.	word number	1025	0 & 10

Table 3 Fault conditions (continued)

Condition	Result Q	Code in Zm Numeric Value	Code in Zm States ON
STATUS			
Operation causes positive overflow (limit to 32767). Q = position of 1st error.	word number	257	0 & 8
Operation causes negative overflow (limit to –32767). Q = position of 1st error.	word number	513	0 & 9

PROGRAMMING RULES

1. A and B-tables

On current models of GEM 80 Controller that implement Array functions, you are not permitted to use A and B-table locations with the same 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 haven't got any 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 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 Array, you must be careful that the block of locations you specify does not overlap any A-table locations in use as the contents of these A-table locations could be affected.

2. Floating Point Tables

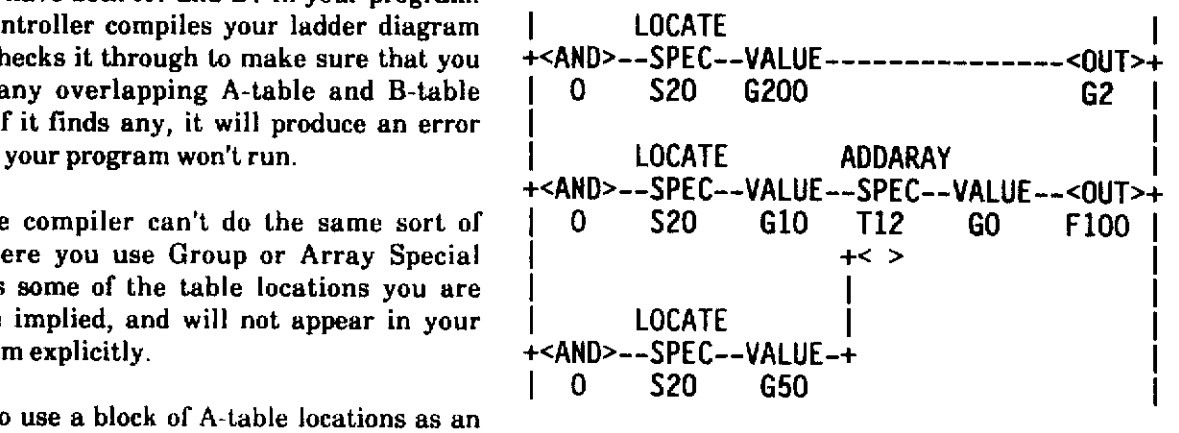
You cannot use ADDARAY, SUBARAY, MULARAY or DIVARAY with floating point table locations in Q-table. If you try to use them this way, you will get an error message, and your program won't compile.

APPLICATION NOTES

ADDARAY Example 1

You have a block of three adjacent data table locations G10, G11 and G12. You want to add the values contained in these locations to the values in data table locations G50, G51 and G52, and to store the respective results in data locations G200, G201 and G202.

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



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

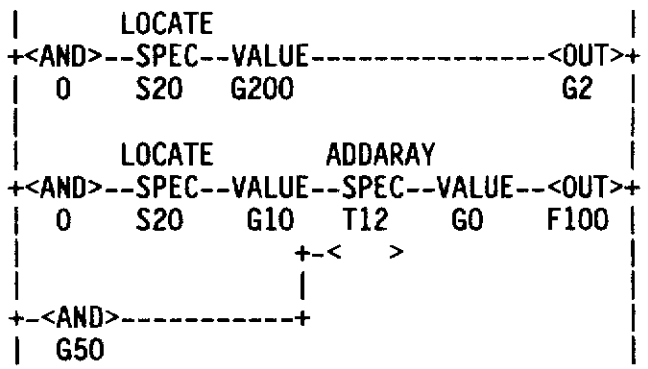
Typical Results:

Input 1		Input 2		Output	
G10	300	G50	-712	G200	-412
G11	-3000	G51	8453	G201	5453
G12	2909	G52	6000	G202	8909

ADDARAY Example 2

You have a block of three adjacent data table locations G10, G11 and G12. This time, however, you want to add the value in data table location G50 to each of the values contained in these locations, and to store the respective results in data locations G200, G201 and G202.

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



G0 = Error code.
G1 = Array length = -3
(negative sign indicates that you want to add a common value to all elements in the input array).
G2 = Address of G200 supplied by first rung.
F100 = Fault indication.

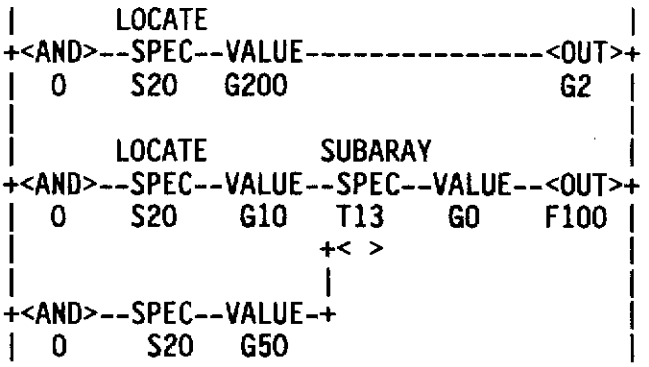
Typical Results:

Input 1		Input 2 (common)		Output	
G10	300	G50	-712	G200	-412
G11	-3000			G201	-3712
G12	2909			G202	2197

SUBARAY Example 1

You have a block of three adjacent data table locations G10, G11 and G12. You want to subtract the values in data table locations G50, G51 and G52 from the values contained in these locations, and to store the respective results in data locations G200, G201 and G202.

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



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

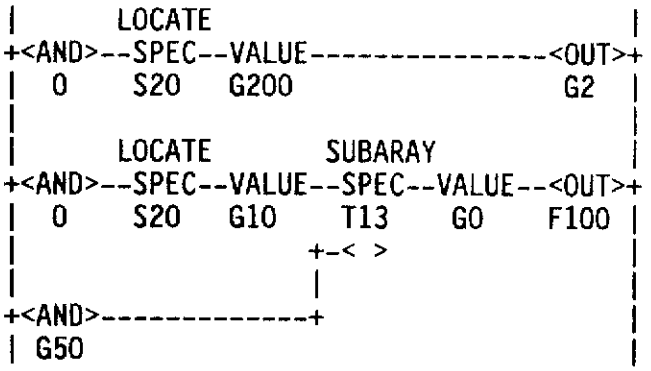
Typical Results:

Input 1		Input 2		Output	
G10	300	G50	-712	G200	1012
G11	-3000	G51	8453	G201	-11453
G12	2909	G52	6000	G202	-3091

SUBARAY Example 2

You have a block of three adjacent data table locations G10, G11 and G12. This time, however, you want to subtract the value in data table location G50 from each of the values contained in these locations, and to store the respective results in data locations G200, G201 and G202.

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



G0 = Error code
G1 = Array length = -3
(negative sign indicates that you want to subtract a common value from all elements in the input array).
G2 = Address of G200 supplied by first rung
F100 = Fault indication

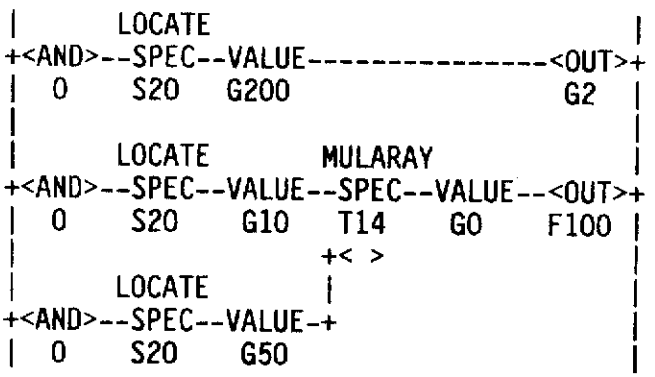
Typical Results:

Input 1		Input 2 (common)		Output	
G10	300	G50	-712	G200	1012
G11	-3000			G201	-2288
G12	2909			G202	3621

MULARAY Example 1

You have a block of three adjacent data table locations G10, G11 and G12. You want the values contained in these locations to be multiplied by the values in data table locations G50, G51 and G52, and to store the respective results in data locations G200, G201 and G202.

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



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

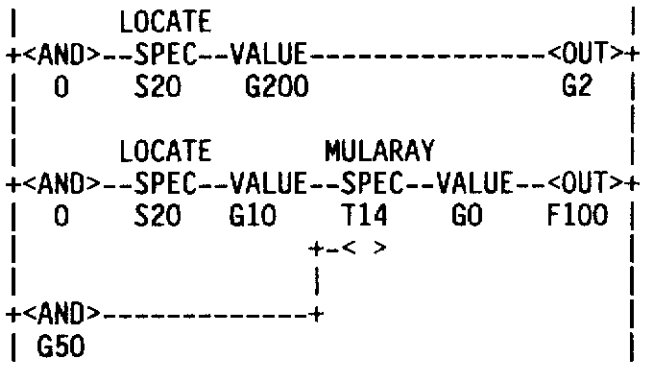
Typical Results:

Input 1		Input 2		Output	
G10	300	G50	3	G200	900
G11	-3000	G51	-5	G201	15000
G12	2	G52	-20	G202	-40

MULARAY Example 2

You have a block of three adjacent data table locations G10, G11 and G12. This time, however, you want the values contained in these locations to be multiplied by the same value, contained in data table location G50, and you want to store the results in data locations G200, G201 and G202.

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



G0 = Error code
G1 = Array length = -3
(negative sign indicates that you want to multiply all elements in the input array by a common value).
G2 = Address of G200 supplied by first rung
F100 = Fault indication

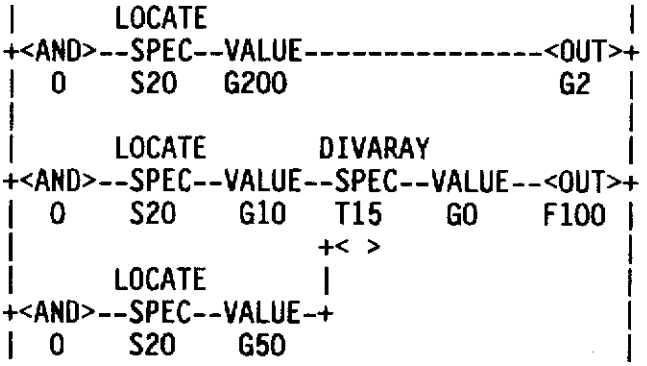
Typical Results:

Input 1		Input 2 (common)		Output	
G10	300	G50	3	G200	900
G11	-3000			G201	-9000
G12	2			G202	6

DIVARAY Example 1

You have a block of three adjacent data table locations G10, G11 and G12. You want the values contained in these locations to be divided by the values in data table locations G50, G51 and G52, and to store the respective results in data locations G200, G201 and G202. Also, you don't need the remainders from these divisions for use anywhere else in your program.

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



G0 = Error code.
G1 = Array length = 3.
G2 = Address of G200 supplied by first rung.
G3 = 0 (you don't want any remainder array).
F100 = Fault indication.

Typical Results:

Input 1		Input 2		Output	
G10	300	G50	-45	G200	-6
G11	-3000	G51	15	G201	-200
G12	2909	G52	5678	G202	0

INDUSTRIAL CONTROLS

Kidsgrove, Stoke-on-Trent, ST7 1TW, England
Tel: (0782) 783511 Fax: (0782) 776329 Telex: 36293/4 CICKID G
Registered in England No. 2259095