MULTIPLY

This chapter covers the following topics:

- Function
- Syntax Description
- Example

Related Statements: ADD | COMPRESS | COMPUTE | DIVIDE | EXAMINE | MOVE | MOVE ALL | RESET | SEPARATE | SUBTRACT

Belongs to Function Group: *Arithmetic and Data Movement Operations*

Function

The `MULTIPLY` statement is used to multiply two operands. Depending on the syntax used, the result of the multiplication may be stored in *operand1* or *operand3*.

If a database field is used as the result field, the multiplication results in an update only to the internal value of the field as used within the program. The value for the field in the database remains unchanged.

For multiplications involving arrays, see also *Rules for Arithmetic Assignments*, *Arithmetic Operations with Arrays* (in the *Programming Guide*).

Syntax Description

Two different structures are possible for this statement.

- Syntax 1 - MULTIPLY without GIVING Clause
- Syntax 2 - MULTIPLY with GIVING Clause

Syntax 1 - MULTIPLY without GIVING Clause

If Syntax 1 used, the result of the multiplication may be stored in *operand1*.

`MULTIPLY` [**`ROUNDED`**] *operand1* **`BY`** *operand2*

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Operand Definition Table (Syntax 1):

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand1		S	A		M		N	P	I	F								yes	no
operand2	C	S	A		N		N	P	I	F								yes	no

Syntax Element Description (Syntax 1):

operand1* BY *operand2	*operand1* is the multiplicand, *operand2* is the multiplier. As the `GIVING` clause is not used, the result is stored in *operand1*, hence the statement is equivalent to: `<oper1> := <oper1> * <oper2>`
ROUNDED	If you specify the keyword `ROUNDED`, the value will be rounded before it is assigned to *operand1* or *operand3*. For information on rounding, see *Rules for Arithmetic Assignment, Field Truncation and Field Rounding* (in the *Programming Guide*).

Syntax 2 - MULTIPLY with GIVING Clause

If Syntax 2 used, the result of the multiplication may be stored in *operand3*.

`MULTIPLY` [**`ROUNDED`**] *operand1* **`BY`** *operand2* **`GIVING`** *operand3*

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Operand Definition Table (Syntax 2):

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand1	C	S	A		M			N	P	I	F							yes	no
operand2	C	S	A		N			N	P	I	F							yes	no
operand3		S	A		M	A	U	N	P	I	F	B*		T				yes	yes

* Format B of *operand3* may be used only with a length of less than or equal to 4.

Syntax Element Description (Syntax 2):

operand1* BY *operand2* GIVING *operand3	*operand1* is the multiplicand, *operand2* is the multiplier. As the `GIVING` clause) is used, *operand1* will not be modified and the result will be stored in *operand3*, hence the statement is equivalent to: `<oper3> := <oper1> * <oper2>` If *operand1* is a numeric constant, the `GIVING` clause is required.
ROUNDED	If you specify the keyword `ROUNDED`, the value will be rounded before it is assigned to *operand1* or *operand3*. For information on rounding, see *Rules for Arithmetic Assignment*, *Field Truncation and Field Rounding* (in the *Programming Guide*).

Example

```
** Example 'MULEX1': MULTIPLY
************************************************************************
DEFINE DATA LOCAL
1 #A      (N3) INIT <20>
1 #B      (N5)
1 #C      (N3.1)
1 #D      (N2)
1 #ARRAY1 (N5/1:4,1:4) INIT (2,*) <5>
1 #ARRAY2 (N5/1:4,1:4) INIT (4,*) <10>
END-DEFINE
*
MULTIPLY #A BY 3
WRITE NOTITLE 'MULTIPLY #A BY 3'           25X '=' #A
*
MULTIPLY #A BY 3 GIVING #B
WRITE 'MULTIPLY #A BY 3 GIVING #B'         15X '=' #B
*
MULTIPLY ROUNDED 3 BY 3.5 GIVING #C
WRITE 'MULTIPLY ROUNDED 3 BY 3.5 GIVING #C'  6X '=' #C
*
MULTIPLY 3 BY -4 GIVING #D
WRITE 'MULTIPLY  3 BY -4 GIVING #D'        14X '=' #D
*
MULTIPLY -3 BY -4 GIVING #D
WRITE 'MULTIPLY -3 BY -4 GIVING #D'        14X '=' #D
*
MULTIPLY  3 BY  0 GIVING #D
WRITE 'MULTIPLY  3 BY  0 GIVING #D'        14X '=' #D
*
WRITE / '=' #ARRAY1 (2,*) '=' #ARRAY2 (4,*)
MULTIPLY #ARRAY1 (2,*) BY #ARRAY2 (4,*)
WRITE / 'MULTIPLY #ARRAY1 (2,*) BY #ARRAY2 (4,*)'
      / '=' #ARRAY1 (2,*) '=' #ARRAY2 (4,*)
*
END
```

Output of Program MULEX1:

```
MULTIPLY #A BY 3                        #A:   60
MULTIPLY #A BY 3 GIVING #B              #B:    180
MULTIPLY ROUNDED 3 BY 3.5 GIVING #C     #C:   10.5
MULTIPLY  3 BY -4 GIVING #D             #D: -12
MULTIPLY -3 BY -4 GIVING #D             #D:  12
MULTIPLY  3 BY  0 GIVING #D             #D:   0

#ARRAY1:      5      5      5      5 #ARRAY2:     10     10     10     10

MULTIPLY #ARRAY1 (2,*) BY #ARRAY2 (4,*)
#ARRAY1:     50     50     50     50 #ARRAY2:     10     10     10     10
```