DIVIDE

This chapter covers the following topics:

- Function
- Syntax Description
- Example

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

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

Function

The `DIVIDE` statement is used to divide two operands.

Note:
Concerning Division by Zero: If an attempt is made to use a divisor (*operand1*) which is zero, either an error message or a result equal to zero will be returned; this depends on the setting of the session parameter `ZD` (described in the *Parameter Reference* documentation).

Syntax Description

Different structures are possible for this statement.

- Syntax 1 - DIVIDE without GIVING Clause
- Syntax 2 - DIVIDE Statement with GIVING Clause
- Syntax 3 - DIVIDE with REMAINDER Option

Syntax 1 - DIVIDE without GIVING Clause

```
DIVIDE [ROUNDED] operand1 INTO operand2
```

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

Operand Definition Table:

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

Syntax Element Description:

operand1* INTO *operand2	**Operands:** *operand1* is the divisor, *operand2* is the dividend. The result is stored in *operand2* (result field), hence the statement is equivalent to: `<oper2> := <oper2> / <oper1>` The result field may be a database field or a user-defined variable. If *operand2* is a constant or a non-modifiable Natural system variable, the `GIVING` clause is required. The number of decimal positions for the result of the division is evaluated from the result field (that is, *operand2*).
ROUNDED	If you specify the keyword `ROUNDED`, the result will be rounded.

Syntax 2 - DIVIDE Statement with GIVING Clause

```
DIVIDE [ROUNDED] operand1 INTO operand2 [GIVING operand3]
```

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

Operand Definition Table:

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

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

Syntax Element Description:

operand1 **INTO** ***operand2*** **GIVING** ***operand3***	**Operands:** *operand1* is the divisor, *operand2* is the dividend, the result is stored in *operand3*, hence the statement is equivalent to: `<oper3> := <oper2> / <oper1>` If a database field is used as the result field, the division only results in an update to the internal value of the field as used within the program. The value for the field in the database remains unchanged. The number of decimal positions for the result of the division is evaluated from the result field (that is, *operand3*). For the precision of the result, see also *Rules for Arithmetic Assignments*, *Precision of Results for Arithmetic Operations* (in the *Programming Guide*).
ROUNDED	If you specify the keyword `ROUNDED`, the result will be rounded.

Syntax 3 - DIVIDE with REMAINDER Option

DIVIDE *operand1* **INTO** *operand2* [**GIVING** *operand3*] **REMAINDER** *operand4*

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

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand1	C	S	A		N			N	P	I								yes	no
operand2	C	S	A		N			N	P	I								yes	no
operand3		S	A			A	U	N	P	I	F	B*						yes	yes
operand4		S	A			A	U	N	P	I	F	B*		T				yes	yes

* Format B of *operand3* and *operand4* may be used only with a length of less than or equal to 4.

Syntax Element Description:

operand1	**Divisor:** *operand1* is the divisor; that is, the number or quantity by which the dividend is to be divided to produce the quotient.
operand2	**Result Field:** If the `GIVING` clause is not used, the result is stored in *operand2*. The result field may be a database field or a user-defined variable. If *operand2* is a constant or a non-modifiable Natural system variable, the `GIVING` clause is required.
ROUNDED	If you specify the keyword `ROUNDED`, the result will be rounded.
GIVING *operand3*	If the keyword `GIVING` is used, *operand2* will not be modified and the result will be stored in *operand3*. If a database field is used as the result field, the division only results in an update to the internal value of the field as used within the program. The value for the field in the database remains unchanged. The number of decimal positions for the result of the division is evaluated from the result field (that is, *operand2* if no `GIVING` clause is used, or *operand3* if the `GIVING` clause is used). For the precision of the result, see also *Rules for Arithmetic Assignments*, *Precision of Results for Arithmetic Operations* (in the *Programming Guide*).
REMAINDER *operand4*	If the keyword `REMAINDER` is specified, the remainder of the division will be placed into the specified field (*operand4*). If `GIVING` *and* `REMAINDER` are used, none of the four operands may be an array range. Internally, the remainder is computed as follows: 1. The quotient of the division of *operand1* into *operand2* is computed. 2. The quotient is multiplied by *operand1*. 3. The product of this multiplication is subtracted from *operand2*. 4. The result of this subtraction is assigned to *operand4*. For each of these steps, the rules described under *Precision of Results for Arithmetic Operations* (in the *Programming Guide*) apply.

Example

```
** Example 'DIVEX1': DIVIDE
************************************************************************
DEFINE DATA LOCAL
1 #A (N7) INIT <20>
```

```
1 #B (N7)
1 #C (N3.2)
1 #D (N1)
1 #E (N1) INIT <3>
1 #F (N1)
END-DEFINE
*
DIVIDE 5 INTO #A
WRITE NOTITLE 'DIVIDE 5 INTO #A' 20X '=' #A
*
RESET INITIAL #A
DIVIDE 5 INTO #A GIVING #B
WRITE 'DIVIDE 5 INTO #A GIVING #B' 10X '=' #B
*
DIVIDE 3 INTO 3.10 GIVING #C
WRITE 'DIVIDE 3 INTO 3.10 GIVING #C' 8X '=' #C
*
DIVIDE 3 INTO 3.1 GIVING #D
WRITE 'DIVIDE 3 INTO 3.1 GIVING #D' 9X '=' #D
*
DIVIDE 2 INTO #E REMAINDER #F
WRITE 'DIVIDE 2 INTO #E REMAINDER #F' 7X '=' #E '=' #F
*
END
```

Output of Program DIVEX1:

```
DIVIDE 5 INTO #A                    #A:        4
DIVIDE 5 INTO #A GIVING #B          #B:        4
DIVIDE 3 INTO 3.10 GIVING #C        #C:    1.03
DIVIDE 3 INTO 3.1 GIVING #D         #D:  1
DIVIDE 2 INTO #E REMAINDER #F       #E:  1 #F:  1
```