

COMPUTE

Structured Mode Syntax

```

{
  { COMPUTE } [ROUNDED] { operand1 [:]= } ... { arithmetic-expression }
  { ASSIGN }   { operand1 [:]= } ... { operand2 }
}
{
  { operand1 := } ... { arithmetic-expression }
  { operand2 }
}
    
```

Reporting Mode Syntax

```

[ { COMPUTE } ] [ROUNDED] { operand1 [:]= } ... { arithmetic-expression }
[ { ASSIGN } ]   { operand1 [:]= } ... { operand2 }
    
```

Operand	Possible Structure			Possible Formats													Referencing Permitted	Dynamic Definition	
Operand1		S	A		M	A	N	P	I	F	B	D	T	L	C	G	O	yes	yes
Operand2	C	S	A		N	A	N	P	I	F	B	D	T	L	C	G	O	yes	no

Related Statements: ADD | SUBTRACT | MULTIPLY | DIVIDE | MOVE

Function

The COMPUTE statement is used to perform an arithmetic or assignment operation.

This statement may be issued in short form by omitting the statement keyword COMPUTE (or ASSIGN). In structured mode, when the statement keyword is omitted, the equal sign (=) must be preceded by a colon (:). However, when the ROUNDED option is used, the statement keyword COMPUTE or ASSIGN must be specified.

For arithmetic operations involving arrays, see also Arithmetic Operations with Arrays in Statement Usage Related Topics.

For more information on data transfer compatibility and the rules for data transfer, see the section Data Transfer.

Result Field - operand1

Operand1 will contain the result of the arithmetic/assignment operation.

For the precision of the result, see Rules for Arithmetic Assignment in Statement Usage Related Topics.

If *operand1* is a database field, the field in the database is not updated.

If *operand1* is a DYNAMIC variable, it is filled with exactly the data and length of *operand2* or the length of the result of the arithmetic-operation, including trailing blanks. The current length of a DYNAMIC variable can be obtained by using the system variable *LENGTH. For general information on DYNAMIC variables, see the section Large and Dynamic Variables/Fields.

If *operand1* is of format C, *operand2* may also be specified as an Attribute Constant.

ROUNDED

If you specify the keyword ROUNDED, the value will be rounded before it is assigned to *operand1*. For information on rounding, see Rules for Arithmetic Assignment in Statement Usage Related Topics.

arithmetic-expression

An arithmetic expression consists of one or more constants, database fields, and user-defined variables.

Natural mathematical functions (which are described in the section System Functions) may also be used as arithmetic operands.

Operands used in an arithmetic expression must be defined with format N, P, I, F, D, or T.

As for the formats of the operands, see also Performance Considerations for Mixed Formats in Statement Usage Related Topics.

The following connecting operators may be used:

Operator	Symbol
Parentheses	()
Exponentiation	**
Multiplication	*
Division	/
Addition	+
Subtraction	-

Each operator should be preceded and followed by at least one blank so as to avoid any conflict with a variable name that contains any of the above characters.

The processing order of arithmetic operations is:

1. Parentheses
2. Exponentiation
3. Multiplication/division (left to right as detected)
4. Addition/subtraction (left to right as detected).

Result Precision of a Division

The precision (number of decimal positions) of the result of a division in a COMPUTE statement is determined by the precision of either the first operand (dividend) or the first result field, whichever is greater.

For a division of integer operands, however, the following applies: For a division of two integer constants, the precision of the result is determined by the precision of the first result field; however, if at least one of the two integer operands is a variable, the result is also of integer format (that is, without decimal positions, regardless of the precision of the result field).

SUBSTRING

If the operands are of alphanumeric or binary format, you may use the SUBSTRING option in the same manner as described for the MOVE statement to assign a part of operand2 to operand1.

Example 1

```

/* EXAMPLE 'ASGEX1S': ASSIGN (STRUCTURED MODE)
DEFINE DATA LOCAL
  1 #A (N3)
  1 #B (A6)
  1 #C (N0.3)
  1 #D (N0.5)
  1 #E (N1.3)
  1 #F (N5)
  1 #G (A25)
  1 #H (A3/1:3)
END-DEFINE
/*****
ASSIGN #A = 5                               WRITE NOTITLE '=' #A
ASSIGN #B = 'ABC'                           WRITE '=' #B
ASSIGN #C = .45                             WRITE '=' #C
ASSIGN #D = #E = -0.12345                   WRITE '=' #D / '=' #E
ASSIGN ROUNDED #F = 199.999                 WRITE '=' #F
#G      := 'HELLO'                          WRITE '=' #G
#H (1) := 'UVW'
#H (3) := 'XYZ'                             WRITE '=' #H (1:3)
END

```

```

#A:      5
#B:     ABC
#C:     .450
#D:    -.12345
#E:    -0.123
#F:     200
#G:    HELLO
#H:    UVW      XYZ

```

Equivalent reporting-mode example: See program ASGEX1R in library SYSEXRM.

Example 2

```

/* EXAMPLE 'CPTEX1S': COMPUTE (STRUCTURED MODE)
/*****
DEFINE DATA LOCAL
1 #I (P2)
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 SALARY (1:2)
1 #A (P4)
1 #B (N3.4)
1 #C (N3.4)
1 #CUM-SALARY (P10)
END-DEFINE
/*****
COMPUTE #A = 3 * 2 + 4 / 2 - 1
WRITE NOTITLE 'COMPUTE #A = 3 * 2 + 4 / 2 - 1' 10X '=' #A
/*****
COMPUTE ROUNDED #B = 3 - 4 / 2 * .89
WRITE 'COMPUTE ROUNDED #B = 3 -4 / 2 * .89' 5X '=' #B
/*****
COMPUTE #C = SQRT (#B)
WRITE 'COMPUTE #C = SQRT (#B)' 18X '=' #C
/*****
LIMIT 1
READ EMPLOY-VIEW BY PERSONNEL-ID STARTING FROM '20017000'
WRITE / 'CURRENT SALARY:' 4X SALARY (1)
  / 'PREVIOUS SALARY:' 4X SALARY (2)
FOR #I = 1 TO 2
  COMPUTE #CUM-SALARY = #CUM-SALARY + SALARY (#I)
END-FOR
WRITE 'CUMULATIVE SALARY:' #CUM-SALARY
END-READ
/*****
END

```

```

COMPUTE #A = 3 * 2 + 4 / 2 - 1      #A:      7
COMPUTE ROUNDED #B = 3 -4 / 2 * .89  #B:     1.2200
COMPUTE #C = SQRT (#B)             #C:     1.1045

CURRENT SALARY:                    34000
PREVIOUS SALARY:                   32300
CUMULATIVE SALARY:                 66300

```

Equivalent reporting-mode example: See program CPTEX1R in library SYSEXRM.