

ADD

This chapter covers the following topics:

- Function
- Syntax Description
- Example

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

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

Function

The ADD statement is used to add two or more operands.

Notes:

1. At the time the ADD statement is executed, each operand used in the arithmetic operation must contain a valid value.
2. For additions involving arrays, see also the section *Arithmetic Operations with Arrays*.
3. As for the formats of the operands, see also the section *Performance Considerations for Mixed Formats*.

Syntax Description

Two different structures are possible for this statement.

- Syntax 1
- Syntax 2

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

Syntax 1

ADD [ROUNDED] operand1... TO operand2

Operand Definition Table (Syntax 1):

Operand	Possible Structure				Possible Formats								Referencing Permitted	Dynamic Definition		
<i>operand1</i>	C	S	A	N			N	P	I	F	D	T			yes	no
<i>operand2</i>		S	A	M			N	P	I	F	D	T			yes	yes

Syntax Element Description:

<i>operand1</i>	<i>operand1</i> is the addend.
ROUNDED	If the keyword ROUNDED is used, the result will be rounded. For rules on rounding, see the section <i>Rules for Arithmetic Assignment</i> .
TO <i>operand2</i>	<i>operand2</i> is included in the addition and receives the result of the operation.

Example:

The statement

```
ADD #A(*) TO #B(*) is equivalent to COMPUTE #B(*) := #A(*) + #B(*)
ADD #S TO #R is equivalent to COMPUTE #R := #S + #R
ADD #S #T TO #R is equivalent to COMPUTE #R := #S + #T + #R
ADD #A(*) TO #R is equivalent to COMPUTE #R := #A(*) + #R
```

Syntax 2

ADD [ROUNDED] <i>operand1</i> ... GIVING <i>operand2</i>

Operand Definition Table (Syntax 2):

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

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

Syntax Element Description:

<i>operand1</i>	<i>operand1</i> is the addend.
ROUNDED	If the keyword ROUNDED is used, the result will be rounded. For rules on rounding, see the section <i>Rules for Arithmetic Assignment</i> .
GIVING <i>operand2</i>	<i>operand2</i> receives the result of the operation only and is not included in the addition. If <i>operand2</i> is defined with alphanumeric format, the result will be converted to alphanumeric.

Note:

If Syntax 2 is used, the following applies: Only the (*operand1*) field(s) left of the keyword GIVING are the terms of the addition, the field right of the keyword GIVING (*operand2*) is just used to receive the result value. If just a single (*operand1*) field is supplied, the ADD operation turns into an assignment.

Example:

The statement

```
ADD #S      GIVING #R is equivalent to COMPUTE #R := #S
ADD #S #T   GIVING #R is equivalent to COMPUTE #R := #S + #T
ADD #A(*) 0  GIVING #R is equivalent to COMPUTE #R := #A(*) + 0
            which is a legal operation, due to the rules defined in Arithmetic Operations with Arrays
ADD #A(*)   GIVING #R is equivalent to COMPUTE #R := #A(*)
            which is an illegal operation, due to the rules defined in Assignment Operations with Arrays
```

Example

```
** Example 'ADDEX1': ADD
*****
DEFINE DATA LOCAL
1 #A      (P2)
1 #B      (P1.1)
1 #C      (P1)
1 #DATE   (D)
1 #ARRAY1 (P5/1:4,1:4) INIT (2,*) <5>
1 #ARRAY2 (P5/1:4,1:4) INIT (4,*) <10>
END-DEFINE
*
ADD +5 -2 -1 GIVING #A
WRITE NOTITLE 'ADD +5 -2 -1 GIVING #A' 15X '=' #A
*
ADD .231 3.6 GIVING #B
WRITE          / 'ADD .231 3.6 GIVING #B' 15X '=' #B
*
ADD ROUNDED 2.9 3.8 GIVING #C
WRITE          / 'ADD ROUNDED 2.9 3.8 GIVING #C' 8X '=' #C
*
MOVE *DATX TO #DATE
ADD 7 TO #DATE
WRITE          / 'CURRENT DATE:'          *DATX (DF=L) 13X
                'CURRENT DATE + 7:' #DATE (DF=L)
*
WRITE          / '#ARRAY1 AND #ARRAY2 BEFORE ADDITION'
                / '=' #ARRAY1 (2,*) '=' #ARRAY2 (4,*)
ADD #ARRAY1 (2,*) TO #ARRAY2 (4,*)
WRITE          / '#ARRAY1 AND #ARRAY2 AFTER ADDITION'
                / '=' #ARRAY1 (2,*) '=' #ARRAY2 (4,*)
*
END
```

Output of Program ADDEX1:

```
ADD +5 -2 -1 GIVING #A      #A:    2
ADD .231 3.6 GIVING #B     #B:    3.8
ADD ROUNDED 2.9 3.8 GIVING #C  #C:    7
CURRENT DATE: 2005-01-10      CURRENT DATE + 7: 2005-01-17
```

```
#ARRAY1 AND #ARRAY2 BEFORE ADDITION
#ARRAY1:      5      5      5      5 #ARRAY2:      10      10      10      10

#ARRAY1 AND #ARRAY2 AFTER ADDITION
#ARRAY1:      5      5      5      5 #ARRAY2:      15      15      15      15
```