ADD

Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiel

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

Gehört zur Funktionsgruppe: *Arithmetische Funktionen und Datenzuweisungen*

Funktion

Das ADD-Statement wird benutzt, um zwei oder mehr Operanden zu addieren.

Anmerkungen:

1. Zu dem Zeitpunkt, zu dem das ADD-Statement ausgeführt wird, muss jeder bei der arithmetischen Operation benutzte Operand einen gültigen Wert enthalten.
2. Zu Additionen mit Arrays, siehe auch den Abschnitt *Arithmetische Operationen mit Arrays* im *Leitfaden zur Programmierung*.
3. Zum Format der Operanden, siehe auch den Abschnitt *Formatwahl im Hinblick auf die Verarbeitungszeit* im *Leitfaden zur Programmierung*.

Syntax-Beschreibung

Mit dem ADD-Statement können Sie zwei oder mehrere Operanden addieren.

- Syntax 1
- Syntax 2

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Syntax 1

ADD [**ROUNDED**] *operand1*... **TO** *operand2*

Operanden-Definitionstabelle (Syntax 1):

Operand	Mögliche Struktur						Mögliche Formate													Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N				N	P	I	F		D	T					ja	nein
operand2		S	A		M				N	P	I	F		D	T					ja	ja

Syntax-Element-Beschreibung:

operand1	*operand1* ist der Addend (zweiter Summand).
ROUNDED	Wünschen Sie das Ergebnis gerundet, geben Sie das Schlüsselwort ROUNDED an. Die für das Runden gültigen Regeln finden Sie unter *Regeln für arithmetische Operationen* im *Leitfaden zur Programmierung*.
TO *operand2*	*operand2* wird in die Addition einbezogen und enthält anschließend das Ergebnis der Operation.

Beispiel:

Das Statement

```
ADD #A(*) TO #B(*)  ist gleichbedeutend mit  COMPUTE #B(*) := #A(*) + #B(*)
ADD #S    TO #R     ist gleichbedeutend mit  COMPUTE #R    := #S + #R
ADD #S #T TO #R     ist gleichbedeutend mit  COMPUTE #R    := #S + #T + #R
ADD #A(*) TO #R     ist gleichbedeutend mit  COMPUTE #R    := #A(*) + #R
```

Syntax 2

ADD [**ROUNDED**] *operand1***... GIVING** *operand2*

Operanden-Definitionstabelle (Syntax 2):

Operand	Mögliche Struktur						Mögliche Formate													Referenzierung erlaubt	Dynam. Definition
operand1	C	S	A		N				N	P	I	F		D	T					ja	nein
operand2		S	A		M		A	U	N	P	I	F	B*	D	T					ja	ja

* Format B von *operand3* kann nur mit einer Länge von kleiner gleich 4 verwendet werden.

Syntax-Element-Beschreibung:

operand1	*operand1* ist der Addend (zweiter Summand).
ROUNDED	Wünschen Sie das Ergebnis gerundet, geben Sie das Schlüsselwort ROUNDED an. Die für das Runden gültigen Regeln finden Sie unter *Regeln für arithmetische Operationen* im *Leitfaden zur Programmierung*.
GIVING ***operand2***	*operand2* erhält nur das Ergebnis der Operation und wird nicht in die Addition einbezogen. Wird *operand2* mit alphanumerischem Format definiert, dann wird das Ergebnis in alphanumerisches Format umgewandet.

Anmerkung:
Bei Verwendung von Syntax 2 gilt Folgendes: Das Feld bzw. die Felder (*operand1*) links vom Schlüsselwort GIVING sind die Terme der Addition, das Feld rechts von GIVING (*operand2*) wird nur zum Empfang des Ergebnisses genutzt. Wird nur ein einzelnes Feld (*operand1*) geliefert, dann wird aus der ADD-Operation in eine Zuweisung.

Beispiel:

Das Statement

```
ADD #S       GIVING #R  ist gleichbedeutend mit  COMPUTE #R := #S
ADD #S #T    GIVING #R  ist gleichbedeutend mit  COMPUTE #R := #S + #T
ADD #A(*) 0  GIVING #R  ist gleichbedeutend mit  COMPUTE #R := #A(*) + 0
         Dies ist eine zulässige Operation aufgrund der Regeln im Abschnitt Arithmetische Operationen mit Arrays
ADD #A(*)    GIVING #R  ist gleichbedeutend mit  COMPUTE #R := #A(*)
         Dies ist eine unzulässige Operation aufgrund der Regeln im Abschnitt Zuweisungen bei Arrays
```

Beispiel

```
** 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
```

Ausgabe des Programms 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
```