

## **Natural für Großrechner**

### **Systemfunktionen**

Version 4.2.6 für Großrechner

Februar 2010

Dieses Dokument gilt für Natural für Großrechner ab Version 4.2.6 für Großrechner.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Release Notes oder Neuausgaben bekanntgegeben werden.

Copyright © 1979-2010 Software AG, Darmstadt, Deutschland und/oder Software AG USA, Inc., Reston, VA, Vereinigte Staaten von Amerika, und/oder ihre Lizenzgeber..

Der Name Software AG, webMethods und alle Software AG Produktnamen sind entweder Warenzeichen oder eingetragene Warenzeichen der Software AG und/oder der Software AG USA, Inc und/oder ihrer Lizenzgeber. Andere hier erwähnte Unternehmens- und Produktnamen können Warenzeichen ihrer jeweiligen Eigentümer sein.

Die Nutzung dieser Software unterliegt den Lizenzbedingungen der Software AG. Diese Bedingungen sind Bestandteil der Produktdokumentation und befinden sich unter <http://documentation.softwareag.com/legal/> und/oder im Wurzelverzeichnis des lizenzierten Produkts.

Diese Software kann Teile von Drittanbieterprodukten enthalten. Die Hinweise zu den Urheberrechten und Lizenzbedingungen der Drittanbieter entnehmen Sie bitte den "License Texts, Copyright Notices and Disclaimers of Third Party Products". Dieses Dokument

ist Bestandteil der Produktdokumentation und befindet sich unter <http://documentation.softwareag.com/legal/> und/oder im Wurzelverzeichnis des lizenzierten Produkts.

---

## Inhaltsverzeichnis

1 Systemfunktionen .....	1
2 Systemfunktionen für Verarbeitungsschleifen .....	3
Systemfunktionen für Verarbeitungsschleifen benutzen .....	4
AVER(r)(field) .....	6
COUNT(r)(field) .....	6
MAX(r)(field) .....	6
MIN(r)(field) .....	7
NAVER(r)(field) .....	7
NCOUNT(r)(field) .....	7
NMIN(r)(field) .....	8
OLD(r)(field) .....	8
SUM(r)(field) .....	8
TOTAL(r)(field) .....	9
Beispiele .....	9
3 Mathematische Funktionen .....	15
4 Verschiedene Funktionen .....	19
5 POS - Feldidentifikationsfunktion .....	21
6 RET - Returncode-Funktion .....	23
7 SORTKEY - Sort-Key Function .....	25
Stichwortverzeichnis .....	29

---

# 1 Systemfunktionen




---

Diese Dokumentation beschreibt in Natural eingebaute Funktionen, die in Statements benutzt werden können.



**Anmerkung:** Ab Natural Version 6.2 für Windows und UNIX, Version 6.3 für OpenVMS und Version 4.2 für Großrechner haben alle neuen Systemfunktionen einen Stern (\*) als erstes Zeichen, um Namenskonflikte mit zum Beispiel Benutzervariablen in vorhandenen Anwendungen zu vermeiden.

Diese Dokumentation ist in die folgenden Abschnitte untergliedert:

	<b>Systemfunktionen für Verarbeitungsschleifen</b>	Beschreibt Systemfunktionen, die im Zusammenhang mit einer Programmschleife verwendet werden können.
	<b>Mathematische Funktionen</b>	Beschreibt Systemfunktionen, die in arithmetischen Statements und in logischen Bedingungen (Logical Condition Criteria, LCC) unterstützt werden.
	<b>Verschiedene Funktionen</b>	Beschreibt Systemfunktionen zur Feldidentifikation, zum Empfangen des Returncodes eines nicht in Natural geschriebenen Programms, das über ein CALL-Statement aufgerufen wurde, zum Konvertieren von „nicht richtig sortierten“ Zeichen (oder Kombinationen von Zeichen) in andere Zeichen (oder Kombinationen von Zeichen), die vom Sortierprogramm oder Datenbanksystem alphabetisch „richtig sortiert“ werden.

Siehe auch folgende Abschnitte im *Leitfaden zur Programmierung*:

- *Systemfunktionen.*
- *Beispiel für Systemvariablen und Systemfunktionen im Leitfaden zur Programmierung.*





## 2 Systemfunktionen für Verarbeitungsschleifen

---

▪ Systemfunktionen für Verarbeitungsschleifen benutzen .....	4
▪ AVER(r)(field) .....	6
▪ COUNT(r)(field) .....	6
▪ MAX(r)(field) .....	6
▪ MIN(r)(field) .....	7
▪ NAVER(r)(field) .....	7
▪ NCOUNT(r)(field) .....	7
▪ NMIN(r)(field) .....	8
▪ OLD(r)(field) .....	8
▪ SUM(r)(field) .....	8
▪ TOTAL(r)(field) .....	9
▪ Beispiele .....	9

Dieses Kapitel erläutert die Natural-Systemfunktionen, die im Zusammenhang mit einer Programmschleife verwendet werden können.

## Systemfunktionen für Verarbeitungsschleifen benutzen

---

- Spezifikation/Auswertung
- Systemfunktionen im SORT GIVE-Statement
- Arithmetischer Überlauf bei AVER, NAVAR, SUM oder TOTAL
- Statement-Referenzierung (r)

### Spezifikation/Auswertung

Natural-Systemfunktionen können angegeben werden in

- zuweisenden und arithmetischen Statements:

- MOVE
- ASSIGN
- COMPUTE
- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE

- in Eingabe-Ausgabe-Statements:

- DISPLAY
- PRINT
- WRITE

die in einem der folgenden Statement-Blöcke stehen:

- AT BREAK
- AT END OF DATA
- AT END OF PAGE

d.h. für alle Verarbeitungsschleifen in den Statements FIND, READ, HISTOGRAM, SORT oder READ WORK FILE.

Wenn eine Systemfunktion in einem AT END OF PAGE-Statement verwendet wird, muss das betreffende DISPLAY-Statement eine GIVE SYSTEM FUNCTIONS-Klausel enthalten.

Datensätze, die aufgrund einer `WHERE`-Klausel zurückgewiesen werden, werden von einer Systemfunktion nicht ausgewertet.

Wenn mittels Systemfunktionen Datenbankfelder ausgewertet werden, die aus `FIND`-, `READ`-, `HISTOGRAM`- oder `SORT`-Schleifen stammen, die auf verschiedenen Ebenen liegen, werden die Feldwerte jeweils entsprechend ihrer Position in der Verarbeitungsschleifen-Hierarchie verarbeitet. Werte einer äußeren Schleife werden zum Beispiel nur dann verarbeitet, wenn für diese Schleife neue Datenwerte erhalten werden.

Wird eine Benutzervariable vor der ersten Verarbeitungsschleife definiert, wird sie für Systemfunktionen in der Schleife ausgewertet, in der das `AT BREAK`-, `AT END OF DATA`- oder `AT END OF PAGE`-Statement steht; wird eine Benutzervariable innerhalb einer Schleife definiert, wird sie in der gleichen Weise wie ein Datenbankfeld in der derselben Schleife behandelt.

Bei dem selektiven Einsatz von Systemfunktionen zur Auswertung von Benutzervariablen empfiehlt es sich, eine bestimmte Verarbeitungsschleife zu referenzieren (mittels Statement-Label oder Sourcecode-Zeilenummer), um genau festzulegen, in welcher Schleife der Wert der Benutzervariablen ausgewertet werden soll.

### Systemfunktionen im `SORT GIVE`-Statement

Eine Systemfunktion kann auch referenziert werden, nachdem sie in der `GIVE`-Klausel eines `SORT`-Statements ausgewertet wurde.

In diesem Fall muss dem Namen der Systemfunktion bei der Referenzierung ein Stern (\*) vorangestellt werden.

### Arithmetischer Überlauf bei `AVER`, `NAVER`, `SUM` oder `TOTAL`

Ein Feld, das Sie mit den Systemfunktionen `AVER`, `NAVER`, `SUM` oder `TOTAL` verwenden, muss lang genug sein (standardmäßig oder vom Benutzer definiert), um die Summe der Feldwerte aufzunehmen. Falls der addierte Wert die Länge des Feldes überschreitet, erhalten Sie eine Fehlermeldung.

Normalerweise hat die Systemfunktion dieselbe Länge wie das angegebene Feld; ist diese Länge nicht ausreichend, dann verwenden Sie den Parameter `NL` des `SORT GIVE`-Statements, um die Ausgabelänge zu vergrößern:

```
SUM(field)(NL=nn)
```

Dadurch vergrößert sich nicht nur die Ausgabelänge, sondern auch die interne Länge des Feldes.

## Statement-Referenzierung (r)

Statement-Referenzierung kann auch auf bei Systemfunktionen angewendet werden. Weitere Einzelheiten entnehmen Sie dem Unterabschnitt *Datenbankfelder mit der (r)-Notation referenzieren - Notation (r)* im Abschnitt *Benutzervariablen im Natural Leitfadens zur Programmierung*.

Durch Verwendung eines Statement-Labels oder Angabe der Sourcecode-Zeilenummer (r) können Sie bestimmen, in welcher Verarbeitungsschleife die jeweilige Systemfunktion für das betreffende Feld ausgewertet werden soll.

## AVER(r)(field)

---

Format/Länge:	Wie Feld.  Ausnahme: bei einem Feld mit Format N hat $AVER(field)$ Format P (mit derselben Länge wie das Feld).
---------------	---

Diese Systemfunktion enthält den Durchschnittswert (Average) aller Werte des angegebenen Feldes. AVER wird aktualisiert, wenn die Bedingung, unter der AVER angefordert wurde, erfüllt ist.

## COUNT(r)(field)

---

Format/Länge:	P7
---------------	----

Diese Systemfunktion zählt die Durchläufe einer Verarbeitungsschleife. Der Wert von COUNT erhöht sich jedesmal um 1, wenn die Verarbeitungsschleife, in der sich COUNT befindet, durchlaufen wird, und zwar unabhängig vom Wert des mit COUNT angegebenen Feldes.

## MAX(r)(field)

---

Format/Länge:	Wie Feld.
---------------	-----------

Diese Systemfunktion enthält den größten Wert des angegebenen Feldes. MAX wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich MAX befindet, ausgeführt wird.

## MIN(r)(field)

---

Format/Länge:	Wie Feld.
---------------	-----------

Diese Systemfunktion enthält den kleinsten Wert des angegebenen Feldes. MIN wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich MIN befindet, ausgeführt wird.

## NAVER(r)(field)

---

Format/Länge:	Wie Feld.
	Ausnahme: bei einem Feld mit Format N hat NAVER( <i>field</i> ) Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält den Durchschnittswert (Average) aller Werte des angegebenen Feldes, wobei Nullwerte nicht berücksichtigt werden. NAVER wird aktualisiert, wenn die Bedingung, unter der NAVER angefordert wurde, erfüllt ist.

## NCOUNT(r)(field)

---

Format/Länge:	P7
---------------	----

Diese Systemfunktion zählt die Durchläufe einer Verarbeitungsschleife. Der Wert von NCOUNT erhöht sich jedesmal um 1, wenn die Verarbeitungsschleife, in der sich NCOUNT befindet, durchlaufen wird, wobei Durchläufe, bei denen der Wert des angegebenen Feldes Null ist, nicht mitgezählt werden.

Ob das Ergebnis von NCOUNT ein Array oder ein Skalarwert ist, hängt vom Argument (*field*) ab. Die Anzahl der resultierenden Ausprägungen ist dieselbe wie bei Feld.

## NMIN(r)(field)

---

Format/Länge:	Wie Feld.
---------------	-----------

Diese Systemfunktion enthält den kleinsten Wert des angegebenen Feldes, wobei Nullwerte nicht berücksichtigt werden. NMIN wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich NMIN befindet, ausgeführt wird.

## OLD(r)(field)

---

Format/Länge:	Wie Feld.
---------------	-----------

Diese Systemfunktion enthält den „alten“ Wert des angegebenen Feldes, d.h. den Wert, den das Feld vor einem in einer AT BREAK-Bedingung spezifizierten Gruppenwechsel (Wechsel des Feldwertes) bzw. vor einer Seitenende- oder Datenende-Bedingung (END OF PAGE, END OF DATA) hatte.

## SUM(r)(field)

---

Format/Länge:	Wie Feld.
	Ausnahme: bei einem Feld mit Format N hat $SUM(field)$ Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält die Summe aller Werte des angegebenen Feldes. SUM wird jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich SUM befindet, ausgeführt wird. SUM wird nach jedem AT BREAK-Gruppenwechsel wieder auf Null gesetzt, addiert also nur Werte zwischen zwei Gruppenwechseln.

## TOTAL(r)(field)

Format/Länge:	Wie Feld.  Ausnahme: bei einem Feld mit Format N hat TOTAL ( <i>field</i> ) Format P (mit derselben Länge wie das Feld).
---------------	--

Diese Systemfunktion enthält die Gesamtsumme aller Werte des angegebenen Feldes in allen offenen Verarbeitungsschleifen, in denen TOTAL vorkommt.

## Beispiele

- Beispiel 1 – AT BREAK-Statement mit Natural-Systemfunktionen OLD, MIN, AVER, MAX, SUM, COUNT
- Beispiel 2 – AT BREAK-Statement mit Natural-Systemfunktion AVER
- Beispiel 3 – AT END OF DATA-Statement mit Systemfunktionen MAX, MIN, AVER
- Beispiel 4 – AT END OF PAGE-Statement mit Systemfunktion AVER

### Beispiel 1 – AT BREAK-Statement mit Natural-Systemfunktionen OLD, MIN, AVER, MAX, SUM, COUNT

```

** Example 'ATBEX3': AT BREAK (with Natural system functions)
*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 CITY
  2 SALARY (1)
  2 CURR-CODE (1)
END-DEFINE
*
LIMIT 3
READ EMPLOY-VIEW LOGICAL BY CITY = 'SALT LAKE CITY'
  DISPLAY NOTITLE CITY NAME 'SALARY' SALARY(1) 'CURRENCY' CURR-CODE(1)
  /*
  AT BREAK OF CITY
    WRITE / OLD(CITY) (EM=X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X)
      31T '   MINIMUM:' MIN(SALARY(1)) CURR-CODE(1) /
      31T '   AVERAGE:' AVER(SALARY(1)) CURR-CODE(1) /
      31T '   MAXIMUM:' MAX(SALARY(1)) CURR-CODE(1) /
      31T '   SUM:' SUM(SALARY(1)) CURR-CODE(1) /
      35T COUNT(SALARY(1)) 'RECORDS FOUND' /
  END-BREAK
  /*
AT END OF DATA

```

```

WRITE 22T 'TOTAL (ALL RECORDS):'
      T*SALARY TOTAL(SALARY(1))  CURR-CODE(1)
END-ENDDATA
END-READ
*
END

```

Ausgabe des Programms ATBEX3:

CITY	NAME	SALARY	CURRENCY
SALT LAKE CITY	ANDERSON		50000 USD
SALT LAKE CITY	SAMUELSON		24000 USD
S A L T L A K E C I T Y		MINIMUM:	24000 USD
		AVERAGE:	37000 USD
		MAXIMUM:	50000 USD
		SUM:	74000 USD
			2 RECORDS FOUND
SAN DIEGO	GEE		60000 USD
S A N D I E G O		MINIMUM:	60000 USD
		AVERAGE:	60000 USD
		MAXIMUM:	60000 USD
		SUM:	60000 USD
			1 RECORDS FOUND
		TOTAL (ALL RECORDS):	134000 USD

**Beispiel 2 – AT BREAK-Statement mit Natural-Systemfunktion AVER**

```

** Example 'ATBEX4': AT BREAK (with Natural system functions)
*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 CITY
  2 SALARY (2)
*
1 #INC-SALARY (P11)
END-DEFINE
*
LIMIT 4
EMPL. READ EMPLOY-VIEW BY CITY STARTING FROM 'ALBU'
  COMPUTE #INC-SALARY = SALARY (1) + SALARY (2)
  DISPLAY NAME CITY SALARY (1:2) 'CUMULATIVE' #INC-SALARY
  SKIP 1
  /*
AT BREAK CITY

```



```

WRITE NOTITLE
  'AVERAGE:'          T*SALARY (1)  AVER(SALARY(1)) /
  'AVERAGE CUMULATIVE:' T*#INC-SALARY AVER(EMPL.) (#INC-SALARY)
END-BREAK
END-READ
*
END

```

#### Ausgabe des Programms ATBEX4:

NAME	CITY	ANNUAL	CUMULATIVE SALARY
HAMMOND	ALBUQUERQUE	22000 20200	42200
ROLLING	ALBUQUERQUE	34000 31200	65200
FREEMAN	ALBUQUERQUE	34000 31200	65200
LINCOLN	ALBUQUERQUE	41000 37700	78700
AVERAGE:		32750	
AVERAGE CUMULATIVE:			62825

#### Beispiel 3 – AT END OF DATA-Statement mit Systemfunktionen MAX, MIN, AVER

```

** Example 'AEDEXIS': AT END OF DATA
*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 FIRST-NAME
  2 SALARY (1)
  2 CURR-CODE (1)
END-DEFINE
*
LIMIT 5
EMP. FIND EMPLOY-VIEW WITH CITY = 'STUTTGART'
  IF NO RECORDS FOUND
    ENTER
  END-NOREC
  DISPLAY PERSONNEL-ID NAME FIRST-NAME
    SALARY (1) CURR-CODE (1)
/*
AT END OF DATA

```

```

IF *COUNTER (EMP.) = 0
  WRITE 'NO RECORDS FOUND'
  ESCAPE BOTTOM
END-IF
WRITE NOTITLE / 'SALARY STATISTICS:'
              / 7X 'MAXIMUM:' MAX(SALARY(1)) CURR-CODE (1)
              / 7X 'MINIMUM:' MIN(SALARY(1)) CURR-CODE (1)
              / 7X 'AVERAGE:' AVER(SALARY(1)) CURR-CODE (1)

END-ENDDATA
/*
END-FIND
*
END

```

**Ausgabe des Programms AEDEX1S:**

PERSONNEL ID	NAME	FIRST-NAME	ANNUAL SALARY	CURRENCY CODE
11100328	BERGHAUS	ROSE	70800	DM
11100329	BARTHEL	PETER	42000	DM
11300313	AECKERLE	SUSANNE	55200	DM
11300316	KANTE	GABRIELE	61200	DM
11500304	KLUGE	ELKE	49200	DM
SALARY STATISTICS:				
	MAXIMUM:	70800	DM	
	MINIMUM:	42000	DM	
	AVERAGE:	55680	DM	

**Beispiel 4 – AT END OF PAGE-Statement mit Systemfunktion AVER**

```

** Example 'AEPEX1S': AT END OF PAGE (structured mode)
*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 JOB-TITLE
  2 SALARY (1)
  2 CURR-CODE (1)
END-DEFINE
*
FORMAT PS=10
LIMIT 10
READ EMPLOY-VIEW BY PERSONNEL-ID FROM '20017000'
  DISPLAY NOTITLE GIVE SYSTEM FUNCTIONS
    NAME JOB-TITLE 'SALARY' SALARY(1) CURR-CODE (1)
/*
AT END OF PAGE

```

```
WRITE / 28T 'AVERAGE SALARY: ...' AVER(SALARY(1)) CURR-CODE (1)
END-ENDPAGE
END-READ
*
END
```

**Ausgabe des Programms AEPEX1S:**

NAME	CURRENT POSITION	SALARY	CURRENCY CODE
CREMER	ANALYST		34000 USD
MARKUSH	TRAINEE		22000 USD
GEE	MANAGER		39500 USD
KUNEY	DBA		40200 USD
NEEDHAM	PROGRAMMER		32500 USD
JACKSON	PROGRAMMER		33000 USD
	AVERAGE SALARY: ...		33533 USD



# 3 Mathematische Funktionen

In logischen Bedingungen und den Arithmetik-Statements ADD, COMPUTE, DIVIDE, MULTIPLY und SUBTRACT können Sie die folgenden mathematischen Funktionen verwenden:

Funktion	Format/Länge	Ausgegebener Wert
ABS( <i>field</i> )	wie Feld ( <i>field</i> )	Absoluter Wert eines Feldes.
ATN( <i>field</i> )	F8	Arcustangens eines Feldes.
COS( <i>field</i> )	F8	Kosinus eines Feldes. Ist der Wert des Feldes größer oder gleich $10^{17}$ , ist COS( <i>field</i> ) "1".
EXP( <i>field</i> )	F8	Potenz eines Feldes ( <i>e field</i> ) mit der Basis <i>e</i> , wobei <i>e</i> die Basis natürlicher Logarithmen ist.
FRAC( <i>field</i> )	wie Feld ( <i>field</i> )	Bruchteil (hinter dem Komma) eines Feldes.
INT( <i>field</i> )	wie Feld ( <i>field</i> )	Ganzzahliger Teil eines Feldes (Integer).
LOG( <i>field</i> )	F8	Natürlicher Logarithmus eines Feldes.
SGN( <i>field</i> )	wie Feld ( <i>field</i> )	Vorzeichen (Sign) eines Feldes (-1, 0, +1).
SIN( <i>field</i> )	F8	Sinus eines Feldes. Ist der Wert des Feldes größer oder gleich $10^{17}$ , ist SIN( <i>field</i> ) "0".
SQRT( <i>field</i> )	(*)	Quadratwurzel eines Feldes (Square Root). Ein negativer Feldwert wird wie ein positiver behandelt. Auf Großrechnern darf der Feldwert maximal 22 Stellen vor dem Komma haben.
TAN( <i>field</i> )	F8	Tangens eines Feldes. Ist der Wert des Feldes größer oder gleich $10^{17}$ , ist TAN( <i>field</i> ) "0".
VAL( <i>field</i> )	wie Zielfeld ( <i>field</i> )	Numerischer Wert eines alphanumerischen Feldes. Der Wert des Feldes muss ein in alphanumerischer Form (Codepage oder Unicode) dargestellter numerischer Wert sein. Leerstellen vor und nach dem

Funktion	Format/Länge	Ausgegebener Wert
		<p>Komma im Feld werden ignoriert. Komma (Dezimalpunkt) und Vorzeichen werden mitverarbeitet.</p> <p>Wenn das Zielfeld nicht lang genug ist, werden Nachkommastellen abgeschnitten (vgl. <i>Abschneiden und Runden von Feldwerten im Abschnitt Regeln für arithmetische Operationen im Leitfaden zur Programmierung</i>).</p>

\*Diese Funktionen werden wie folgt ausgewertet:

- Wenn das Feld Format/Länge F4 hat, hat  $SQRT(field)$  auch Format/Länge F4.
- Wenn das Feld Format/Länge F8 oder I hat, hat  $SQRT(field)$  Format/Länge F8.
- Wenn das Feld Format N oder P hat, hat  $SQRT(field)$  Format/Länge  $Nn.7$  bzw.  $Pn.7$  (wobei  $n$  automatisch groß genug berechnet wird).

Ein mit einer mathematischen Funktion (außer VAL) verwendetes Feld kann eine Konstante oder ein Skalar sein und muss numerisches, gepackt numerisches, Ganzzahl- oder Gleitkomma-Format (N, P, I oder F) haben.

Ein mit der Funktion VAL verwendetes Feld kann eine Konstante, ein Skalar oder ein Array sein und muss alphanumerisches Format haben.

**Beispiel für mathematische Funktionen:**

```

** Example 'MATHEX': Mathematical functions
*****
DEFINE DATA LOCAL
1 #A      (N2.1) INIT <10>
1 #B      (N2.1) INIT <-6.3>
1 #C      (N2.1) INIT <0>
1 #LOGA   (N2.6)
1 #SQRTA  (N2.6)
1 #TANA   (N2.6)
1 #ABS    (N2.1)
1 #FRAC   (N2.1)
1 #INT    (N2.1)
1 #SGN    (N1)
END-DEFINE
*
COMPUTE #LOGA = LOG(#A)
WRITE NOTITLE '=' #A 5X 'LOG'          40T #LOGA
*
COMPUTE #SQRTA = SQRT(#A)
WRITE          '=' #A 5X 'SQUARE ROOT' 40T #SQRTA
*
COMPUTE #TANA = TAN(#A)
WRITE          '=' #A 5X 'TANGENT'      40T #TANA
*

```

```

COMPUTE #ABS = ABS(#B)
WRITE // '=' #B 5X 'ABSOLUTE' 40T #ABS
*
COMPUTE #FRAC = FRAC(#B)
WRITE // '=' #B 5X 'FRACTIONAL' 40T #FRAC
*
COMPUTE #INT = INT(#B)
WRITE // '=' #B 5X 'INTEGER' 40T #INT
*
COMPUTE #SGN = SGN(#A)
WRITE // '=' #A 5X 'SIGN' 40T #SGN
*
COMPUTE #SGN = SGN(#B)
WRITE // '=' #B 5X 'SIGN' 40T #SGN
*
COMPUTE #SGN = SGN(#C)
WRITE // '=' #C 5X 'SIGN' 40T #SGN
*
END

```

**Ausgabe des Programms MATHEX:**

```

#A: 10.0 LOG 2.302585
#A: 10.0 SQUARE ROOT 3.162277
#A: 10.0 TANGENT 0.648360

#B: -6.3 ABSOLUTE 6.3
#B: -6.3 FRACTIONAL -0.3
#B: -6.3 INTEGER -6.0

#A: 10.0 SIGN 1
#B: -6.3 SIGN -1
#C: 0.0 SIGN 0

```

---



# 4

## Verschiedene Funktionen

---

Folgende Themen werden behandelt:

- **POS - Feldidentifikationsfunktion**
- **RET - Returncode-Funktion**
- **SORTKEY - Sortierschlüssel-Funktion**



# 5 POS - Feldidentifikationsfunktion

---

Format/Länge: I4
------------------

Die Systemfunktion `POS(field-name)` enthält die interne Identifikation des Feldes, dessen Name mit der Systemfunktion angegeben wird.

`POS(field-name)` identifiziert ein bestimmtes Feld, unabhängig von seiner Position in einer Maske (Map). Auch wenn sich die Reihenfolge und Anzahl der Felder in einer Maske ändert, identifiziert `POS(field-name)` nach wie vor eindeutig dasselbe Feld. Damit genügt zum Beispiel ein einziges REINPUT-Statement, um es von der Programmlogik abhängig zu machen, welches Feld MARKiert werden soll.

Beispiel:

```
DECIDE ON FIRST VALUE OF ...
  VALUE ...
    COMPUTE #FIELDX = POS(FIELD1)
  VALUE ...
    COMPUTE #FIELDX = POS(FIELD2)
  ...
END-DECIDE
...
REINPUT ... MARK #FIELDX
```

Wenn das mit `POS` angegebene Feld ein Array ist, muss eine bestimmte Ausprägung angegeben werden; zum Beispiel `POS(FIELDX(5))`. Auf einen Array-Bereich kann `POS` nicht angewendet werden.

## POS und \*CURS-FIELD

`POS(field-name)` kann in Verbindung mit der Natural-Systemvariablen `*CURS-FIELD` dazu verwendet werden, die Ausführung bestimmter Funktionen davon abhängig zu machen, in welchem Feld der Cursor zur Zeit steht.

\*CURS-FIELD enthält die interne Identifikation des Feldes, in dem sich der Cursor zur Zeit befindet; \*CURS-FIELD kann nicht alleine, sondern nur zusammen mit `POS(field-name)` verwendet werden. Sie können beide zusammen dazu benutzen, um zu prüfen, ob sich der Cursor gerade in einem bestimmten Feld befindet, und die weitere Verarbeitung von dieser Bedingung abhängig machen.

Beispiel:

```
IF *CURS-FIELD = POS(FIELDX)
  MOVE *CURS-FIELD TO #FIELDY
END-IF
...
REINPUT ... MARK #FIELDY
```



### Anmerkungen:

1. Die Werte von \*CURS-FIELD und `POS(field-name)` dienen nur zur internen Identifikation der Felder und können nicht für arithmetische Operationen verwendet werden.
2. Der von `POS(field-name)` zurückgegebene Wert für eine Ausprägung eines X-Arrays (ein Array, für das wenigstens eine Dimension als erweiterbar angegeben ist) kann sich ändern, nachdem die Anzahl der Ausprägungen für eine Dimension des Arrays mittels der Statements EXPAND, RESIZE oder REDUCE geändert wurde.
3. Natural RPC: Wenn \*CURS-FIELD und `POS(field-name)` sich auf eine Kontextvariable beziehen, können die daraus resultierenden Informationen nur innerhalb derselben Konversation verwendet werden.
4. In Natural for Ajax-Anwendungen dient \*CURS-FIELD zur Identifikation des Operanden, welcher den Wert des Control darstellt, welches den Eingabefokus hat. Sie können \*CURS-FIELD in Verbindung mit der POS-Funktion benutzen, um eine Prüfung auf das Control, welches den Eingabefokus hat, zu veranlassen und die Verarbeitung in Abhängigkeit von diesem Zustand durchzuführen.

Siehe auch

- *Dialog-Gestaltung, Feld-sensitive Verarbeitung and Einfachere Programmierung im Natural Leitfaden zur Programmierung.*
- *POS22 - Version 2.2 Algorithm for POS System Function in der Parameter-Referenz-Dokumentation.*

# 6 RET - Returncode-Funktion

---

Format/Länge: I4
------------------

Die Systemfunktion `RET(program-name)` kann dazu verwendet werden, den Returncode eines nicht in Natural geschriebenen Programms, das über ein `CALL`-Statement aufgerufen wurde, zu erhalten.

`RET(program-name)` kann in einem `IF`-Statement sowie in den Arithmetik-Statements `ADD`, `COMPUTE`, `DIVIDE`, `MULTIPLY` und `SUBTRACT` verwendet werden.

Beispiel:

```
DEFINE DATA LOCAL
1 #RETURN (I4)
...
END-DEFINE
...
...
CALL 'PROG1'
IF RET('PROG1') > #RETURN
    WRITE 'ERROR OCCURRED IN PROGRAM 1'
END-IF
...
```



# 7 SORTKEY - Sort-Key Function

---

SORTKEY (*character-string*)

Diese Systemfunktion dient zum Konvertieren von „nicht richtig sortierten“ Zeichen (oder Kombinationen von Zeichen) in andere Zeichen (oder Kombinationen von Zeichen), die vom Sortierprogramm oder Datenbanksystem alphabetisch „richtig sortiert“ werden.

Format/Länge: A253

Es gibt in vielen Landessprachen Zeichen (oder Zeichenkombinationen), die von einem Sortierprogramm oder Datenbanksystem nicht in der richtigen alphabetischen Reihenfolge sortiert werden, da die Reihenfolge der Zeichen im vom Computer verwendeten Zeichensatz nicht immer der alphabetischen Reihenfolge der Zeichen entspricht.

Zum Beispiel wird der spanische Buchstabe CH in der Regel von einem Sortierprogramm bzw. Datenbanksystem wie zwei Buchstaben behandelt und zwischen CG und CI einsortiert – gehört aber eigentlich als eigener Buchstabe im spanischen Alphabet zwischen "C" und "D".

Oder es kann sein, dass Kleinbuchstaben und Großbuchstaben entgegen Ihren Wünschen bei der Sortierreihenfolge nicht gleich behandelt werden, dass Ziffern vor Buchstaben sortiert werden (Sie aber wünschen, dass Buchstaben vor Ziffern sortiert werden) oder dass Sonderzeichen (z.B. Bindestriche in Doppelnamen) zu einer unerwünschten Sortierreihenfolge führen.

In solchen Fällen können Sie die Systemfunktion SORTKEY(*character-string*) benutzen. Die von SORTKEY berechneten Werte werden nur als Sortierkriterium benutzt, während die ursprünglichen Werte für die Interaktion mit dem Endbenutzer verwendet werden.

Sie können die SORTKEY-Funktion in einem COMPUTE sowie in einer logischen Bedingung als arithmetischen Operanden verwenden.

Als *character-string* können Sie eine alphanumerische Konstante oder Variable oder eine einzelne Ausprägung eines alphanumerischen Arrays angeben.

Wenn Sie die SORTKEY-Funktion in einem Natural-Programm angeben, wird der User-Exit NATUSK $nn$  aufgerufen — wobei  $nn$  der aktuelle Sprachcode (d.h. der aktuelle Wert der Systemvariablen \*LANGUAGE) ist.

Sie können diesen User-Exit in jeder Programmiersprache, die über eine Standard-CALL-Schnittstelle verfügt, schreiben. Der mit SORTKEY angegebene *character-string* wird an den User-Exit übergeben. Der User-Exit muss so programmiert werden, dass er „falsch sortierte“ Zeichen in dieser Zeichenkette in entsprechende „richtig sortierte“ Zeichen umsetzt. Die umgesetzte Zeichenkette wird dann vom Natural-Programm zur weiteren Verarbeitung verwendet.

Allgemeine Aufruf-Konventionen für externe Programme sind in der Dokumentation zum CALL-Statement erläutert.

Nähere Informationen zu den Aufruf-Konventionen für SORTKEY User-Exits finden Sie im Abschnitt *User Exit for Computation of Sort Keys* in der *Natural Operations Documentation*.

Beispiel:

```
DEFINE DATA LOCAL
1 CUST VIEW OF CUSTOMERFILE
  2 NAME
  2 SORTNAME
END-DEFINE
...
*LANGUAGE := 4
...
REPEAT
  INPUT NAME
  SORTNAME := SORTKEY(NAME)
  STORE CUST
  END TRANSACTION
  ...
END-REPEAT
...
READ CUST BY SORTNAME
  DISPLAY NAME
END-READ
...
```

Angenommen, im obigen Beispiel würden bei mehrmaliger Ausführung des INPUT-Statements nacheinander die Werte "Sanchez", "Sandino" und "Sancinto" eingegeben.

Bei der Zuweisung von SORTKEY(NAME) zu SORTNAME würde der User-Exit NATUSK04 aufgerufen. Dieser müsste so programmiert werden, dass er zunächst alle Kleinbuchstaben in Großbuchstaben umsetzt und dann die Zeichenfolge "CH" in "Cx" umsetzt — wobei  $x$  dem letzten Zeichen im verwendeten Zeichensatz entspricht, also hexadezimal H'FF' (vorausgesetzt dieses letzte Zeichen ist kein druckbares Zeichen).



Es werden sowohl die „eigentlichen“ Namen (NAME) als auch die für die gewünschte Sortierung umgesetzten Namen (SORTNAME) gespeichert. Zum Lesen der Datei wird SORTNAME verwendet. Dann würden bei Ausführung des DISPLAY-Statements die Namen in der richtigen spanischen alphabetischen Reihenfolge ausgegeben:

```
Sancinto  
Sanchez  
Sandino
```



# Stichwortverzeichnis

---

## A

ABS  
Systemfunktion, 15  
ATN  
Systemfunktion, 15

## C

COS  
Systemfunktion, 15

## E

EXP  
Systemfunktion, 15

## F

FRAC  
Systemfunktion, 15

## I

INT  
Systemfunktion, 15

## L

LOG  
Systemfunktion, 15

## S

SGN  
Systemfunktion, 15  
SIN  
Systemfunktion, 15  
SQRT  
Systemfunktion, 15  
Systemfunktionen, 1

## T

TAN  
Systemfunktion, 15

## V

VAL  
Systemfunktion, 15

