SWT
SOFTWARE TOOLS

Deutsches Anwenderhandbuch

Duevej 23
DK-2680 Solrød Strand
Denmark
Phone: **+45) 33 33 05 56**
Mail: **swtools@swtools.com**
www: **www.swtools.com**

Berechnungen und Subfunktionen

22/11/01 / 2022-09-01 008.384

Inhaltsverzeichnis

1. Einleitung

SW-Tools Programmprodukte benutzen für das Berechnungsmodul eine BASIC- ähnliche Sprache. Hiermit können u.a. Feldinhalte geprüft, Berechnungen angestellt und Texte behandelt werden.

	Reservierte Worte	Synonym	Beschreibung
Bedingte Sätze			
	IF		IF Ausdruck ...
	LET		IF Ausdruck LET Ausdruck ...
	ELSE		IF Ausdruck ... ELSE Ausdruck
Block-Sätze			
	BEGIN		Beginnt ein Block
	END		Beendet einen Block
Wort benutzt in Schleifen			
	BREAK		Abbruch der Schleife
	CONTINUE		Gehe zum Ende der Schleife und setze fort
Logische Operatoren			
	NOT		ungleich 0
	AND	UND	und
	OR	ODER	oder
Matematische Operatoren			
	+		Plus
	-		Minus
	*		Multiplizieren
	/		Dividieren
	%		Prozentberechnung
Relations Operatoren			
	=		gleich mit
	>		größer als
	<		kleiner als
	>=		größer als oder gleich mit
	<=		kleiner als oder gleich mit
	<>		ungleich von
Kommentare			
	REM		Volle Kommentarzeile
	/*		Kommentar nach Berechnungsausdruck
Sprünge und Unterprogramme			
	GOTO		Sprung zur Adresse (Label):
	GOSUB		Sprung zum Unterprogramm Adresse (Label):
	RETURN		Rücksprung aus Unterprogramm
	ON..GO..		Bedingter Sprung/

Unterprogrammaufruf

Mit den Befehlen, die diese BASIC-Ähnliche Sprache anbietet, können eine lange Reihe unterschiedlicher Berechnungen definiert werden. Im folgenden werden diese an Hand von Beispielen im einzelnen erläutert.

1.1. Beispiele

1.1.1. IF..ELSE - Bedingte Sätze

Ist der Lieferantensaldo (LE#6) größer 1000 DEM, dann subtrahiere 100 DEM, sonst addiere 47 DEM.

```
IF LE#6 > 1000 LET LE#6 = LE#6 - 100 ELSE LET LE#6 = LE#6 + 47
```

1.1.2. **BEGIN..END** - Blocksätze

Ist der Lieferantensaldo (LE#6) größer 1000 DEM, dann starte den Block, in dem 100 DEM subtrahiert werden, und drucke die Zeile 7.

```
IF LE#6 > 1000 THEN BEGIN
LE#6 = LE#6 - 100
PRINT(7)
END
```

D.h., alle Berechnungszeilen zwischen BEGIN und END werden nur dann ausgeführt, wenn die angegebene Bedingung erfüllt ist.

1.1.3. START/END...NEXT...REPEAT - Schleifen

Die unten gezeigten Schleifen lesen alle Lieferanten im Intervall 111-999. Ist der Saldo kleiner 1000 DEM, wird der entsprechende Lieferant nicht weiter behandelt.

```
START(LE),"111"
END(LE),"999"
NEXT(LE)
IF LE#6 < 1000 CONTINUE
REM *** Bearbeitung von eine Lieferant ***
REPEAT(LE)
```

Alle Lieferanten im Intervall 111-999 werden gelesen. Wird ein Lieferant mit einem Saldo größer 10.000 DEM gelesen, wird die Schleife abgebrochen.

```
START(LE),"111"
END(LE),"999"
NEXT(LE)
IF LE#6 > 10000 BREAK /* Saldo > 10000 --> Schleife abbrechen
REPEAT(LE)
IF LE#6 > 10000 ....
```

1.1.4. **NOT, AND, OR** - Logische Operatoren

```
IF NOT VA#5 LET VA#5=#DD
```

Ist das letzte Kaufdatum gleich 0, dann setze das Kaufdatum gleich Tagesdatum. Dieser Satz entspricht

```
IF VA#5=0 LET VA#5=#DD
```

Sofern der Kaufpreis ungleich 0 UND das letzte Kaufdatum ungleich 0 ist, dann drucke die Zeile 5.

```
IF VA#4<>0 AND VA#5<>0 PRINT(5)
```

Sofern der Kaufpreis ungleich 0 ODER das letzte Kaufdatum ungleich 0 ist, dann drucke die Zeile 5.

```
IF VA#4<> OR VA#5<>0 PRINT(5)
```

1.1.5. REM, /* - Kommentare

```
REM *** Dieser Report wurde von SW-Tools ApS entwickelt ***
REM *** den 07.09.1997
IF LE#6 > 1000 LET LE#6 = LE#6 - 100 /* Korrigiere Saldo
```

1.1.6. **GOTO** Sprung zur Adresse (Label)

Mit Hilfe von GOTO kann eine bestimmte Adresse (Label) angesprungen werden, typisch in Abhängigkeit eines Feldwertes. Das Label, definiert mit einem NAMEN, bestimmt die Adresse des Sprunges. Im unteren Beispiel wird die Zeile 7 drei mal ausgedruckt.

```
        #30 = 0                 /* Lösche Zähler
WIEDER:                         /* Label für späteren Sprung
        PRINT(7)                /* Drucke
        #30 = #30 + 1           /* Setze Zähler + 1
        IF #30 < 3 GOTO WIEDER  /* Durchlaufe die Schleife 3 x
```

1.1.6.1. **ON**...GOTO/GOSUB - Bedingter Sprung/Unterprogrammaufruf

Mit ON kann, abhängig von einem Feldwert, eine bestimmte Adresse angesprungen werden. ON kann zusammen mit GOTO und GOSUB benutzt werden

```
        #30 = 0
        ON #7 GOTO EINS,ZWEI,EINS
        #30 = #30 + 1          /* #30 wird = 3 wenn Feld 7 nicht = 1,2 oder 3
ZWEI:   #30 = #30 + 1          /* #30 wird = 2 wenn Feld 7 = 2
EINS:   #30 = #30 + 1          /* #30 wird = 1 wenn Feld 7 = 1 oder 3
```

1.1.7. **GOSUB** Unterprogrammaufruf

Soll eine Berechnung mehrere Male ausgeführt werden, kann mit Vorteil folgendes unterprogramm, das mit einem LABEL beginnt, geschrieben werden. Das Programm wird mit GOSUB aufgerufen.

```
       #30 = 0                  /* Lösche Zähler
       GOSUB DRU                /* Rufe Unterprogramm
       GOSUB DRU                /* Rufe Unterprogramm nochmals
       RETURN                   /* Beende normale Berchnungen
  DRU: #30 = #30 + 1            /* Routine DRU, Feld 30 +1
       PRINT(7)                 /* Drucke Zeile 7
       RETURN                   /* Rücksprung aus Unterprogramm
```

1.1.7.1. **RETURN** - Rücksprung aus einem Unterprogramm

Mit RETURN wird ein Unterprogramm beendet, und man fährt mit den Berechnungen an der Stelle fort, an der man das Unterprogramm aufgerufen hat. Sie auch später eine RETURN-Funktion, mit der ein bestimmter Wert (Resultat) übergeben wird.

1.2. Felder

1.2.1. #xx og kk#xx - Dateifelder

Man bestimmt eine Feld in einer Datei mit seiner Feldnummer:

```
     #xx = Feldnr. xx in der Hauptdatei
   kk#xx = Feldnr. xx in der Datei kk.
```

Beachten Sie, daß kk, KK, Kk, kK auf unterschiedliche Sätze in der Datei KK verweist. Normalerweise soll man nur Kleinbuchstaben verwenden, also kk.

1.2.1.1. #xx(von,bis) - Teile eines Feldes

Teile eines Feldes gibt man z.B. mit kk#xx(von,bis) an und können in rein numerischen als auch alphanumerischen Feldern (=Text) benutzt werden.

```
#30 = #2(3,4)           /* Feld 30 = Zeichen 3 und 4 von Feld 2
```

In Textfeldern, und nur in solchen, kann ein Teil eines Feldes gleich einem bestimmten Text gesetzt werden:

```
#2="Sorenco und Sohn KG"
#2(9,14)="xx"           /* Feld 2 wird = "Sorenco xx KG"
```

1.2.1.2. #xx(Nr) - Tabellenfelder

Tabellenfelder werden mit z.B. kk#xx(Nr), wobei Nr von 0 aufwärts geht, angesprochen.
Ein Feld kann in der Datenbank als Tabellenfeld definiert sein, wobei dann die Formatangabe eine Nummer enthält, z.B. 20(003). Hier gibt 003 an, daß das Feld drei weitere Elemente besitzt, bzw. daß drei zusammenhängende Felder (also wie eine Tabelle) in den Berechnungen behandelt werden sollen. Beachten Sie, daß Freifelder mit Hilfe der Formatangabe auch als Tabellenfelder definiert werden können.
Als Beispiel hierfür kann die Demo-Lieferantendatei gelten, wobei die Adressblöcke #2, #3 und #4 auch als Tabelle #2(0), #2(1) und #2(2) aufgefaßt werden können.

```
#30 = #2(#31)          /* Freifeld 31 bestimmt Adr.Zeilen 0,1 oder 2
PRINT(7)               /* Anschließender Ausdruck
#2(#31)="xx"           /* Wird = "xx" gesetzt
```

Überschreitet man die maximale Nummer für eine Tabelle, können sehr mystische Werte erscheinen, z.B. wenn man im obigen Fall 2#(4) angeben würde.

1.2.1.3. Konvertierung zwischen Zahl- und Textfeldern

Man hat die Möglichkeit, ein numerisches Feld gleich einem Textfeld zu setzen, z.B. #30 = #2, und anschließend mit dem Zahlenwert des Textes Berechnungen anstellen. Sehen Sie auch später die Funktionen NUMBER und NUMS.
Setzt man ein Textfeld gleich einem numerischen Feld, z.B. #2 = #30, wird ein Text mit einer variablen Länge, abhängig vom Wert in Feld 30, gebildet, z.B. "123". Normalerweise benutzt man jedoch #2 = #30 USING "#####", um das Format des Textfeldes zu bestimmen (siehe auch Funktion USING).

1.2.2. **SY#xx** - Systemfelder

Systemfelder sind Sonderfelder, die in der Pseudodatei SY definiert sind, und immer zur Verfügung stehen müssen. Einige dieser Systemfelder werden im folgenden beschrieben. Für eine komplette Übersicht verweisen wir auf die Beschreibung der SY-Datei.
Ein Systemfeld kann entweder mit der Feldnummer (SY#1) oder mit dem Kürzel (#DD), das voran im Feldnamen angegeben ist, angesprochen werden. Bestimmte Systemfelder sind mit einer Datei verbunden und müssen deshalb den entsprechenden Dateinamen mitenthalten, z.B. kk#RECNO, wobei kk die jeweilige Datei angibt.

1.2.2.1. **#DD, #PD** - Tagesdatum und per Datum

Eingabe zu Beginn einer Liste (99.99.99).

1.2.2.2. **#PP** - Seitennummer

Wird automatisch bei Seitenwechsel zugeteilt (9999).

1.2.2.3. **#SN** - Systemname

Kann benutzt werden, wenn mehrere Systeme, d.h. mehrere Unternehmen/Datei- Systeme, installiert sind. Beachten Sie auch die Felder #SU für den Namen eines Subsystems und #CN für einen Firmennamen.

1.2.2.4. #OK - Resultat des Lesens einer Datei

Unmittelbar nach dem Lesen einer Datei kann der Wert in #OK abgefragt werden. Dieses Feld hat den Wert 0, wenn alles in Ordnung war, ungleich 0 bei aufge- tretendem Fehler.

1.2.2.5. **#UN** Anwender Name

Sie können #UN benutzen, um den Anwendernamen für diesen PC, der im Lizenz Modul eingegeben wurde, zu erhalten.

1.2.2.6. **#LIN** Zeilennummer und #LOF Anzahl Zeilen

#LIN enthält die aktuelle Zeilennummer und #LOF die Anzahl Zeilen des aktuellen Formulars.

1.2.2.7. **#LEVEL** - Jetzige Summenebene

Mit #LEVEL können Berechnungen/Ausdrücke abhängig von der jeweiligen Summenebene gesteuert werden.

1.2.2.8. kk#RECNO - Zuletzt gelesene Satznummer der Datei kk

Sofern das verwendete Datenbanksystem mit Satznummern arbeitet, kann die zuletzt gelesene Satznummer der Datei kk in kk#RECNO gefunden werden. Siehe auch kk#NUMBER (relative Satznummer) und kk#FILENAME.

1.2.3. **WW#xx** - Freifelder (Arbeitsfelder)

Einem Programm werden bei Neudefinition 40 Freifelder zugewiesen. Diesen Felder muß bei erster Benutzung Name und Format zugeteilt werden. Später können diese Felder umdefiniert werden.
Die Feldnummer liegen im Anschluß an die Feldnummern der Hauptdatei, werden aber unter den Bezeichnungen WW#1,WW#2,.... gespeichert. Hierdurch können bei späterer Erweiterung die Freifelder automatisch neu numeriert werden.
Die Anzahl der Freifelder kann in IQ und DATAMASTER über Programmparameter bestimmt werde. In RAPGEN kann durch Angabe einer höheren Freifeldnummer die Anzahl der Freifelder vergrößert werden.

1.2.3.1. #Dntext - Eingabedaten

In RAPGEN bewirkt ein Freifeldname, der mit #Dn beginnt, die Eingabe von Daten (bis zu 7) zu Beginn einer Liste.

1.2.3.2. **#Ptext** - Bildfelder

Ein Freifeld mit dem Namen #P.... definiert ein Textfeld, das auf ein Bild verweist.

2. Arithmetische Funktionen

In diesem Abschnitt werden unterschiedliche Funktionen für Zahlenbehandlung, wie z.B. Auf-/Abrundungen, Potenzierungen usw. behandelt.

2.1. **ABS** - Absoluter Wert einer Zahl

Zahl ABS(Zahl *par1*)
Parameter: *par1* : Zahl, die in einen absoluten Wert konvertiert werden soll.
Beschreibung: Die Funktion retourniert den absoluten Wert von Parameter *par1*. D.h. den positiven Wert ohne Vorzeichen.
Rückgabe: Absoluter Wer einer Zahl.
Siehe auch: SGN
Beispiel: #1 = ABS(-123.45) */* Feld #1 enthält anschließend den Wert 123.45*

2.2. **FNH** - Aufrunden, keine Dezimalstellen

Zahl FNH(Zahl *par1*)
Parameter: *par1* : Angabe einer Zahl (mit Dezimalstellen)
Beschreibung: Die Funktionen wird zum Aufrunden einer Zahl mit Dezimalstellen in eine Zahl ohne Dezimalstellen benutzt.
Rückgabe: Aufgerundete Zahl ohne Dezimalstellen.
Siehe auch: FNR, RUN
Beispiel: #1 = FNR(1234.56) */* Feld #1 enthält anschließend den Wert 1235*

2.3. FNR - Aufrunden, 2 Dezimalstellen

Zahl FNR(Zahl *par1*)
Parameter: *par1* : Angabe einer Zahl (mit Dezimalstellen)
Beschreibung: Mit Hilfe dieser Funktion kann eine Zahl mit mehr als zwei Dezimalstellen in eine Zahl mit zwei Dezimalstellen aufgerundet werden. Der Listgenerator rundet immer das Rechenergebnis auf die Anzahl Dezimalstellen auf, die für das Resultatfeld im Format vorgegeben sind. Die Funktionen FNR/FNH können benutzt werden, falls man eine andere Form des Aufrundens wünscht.
Aufrundungen können generell geändert werden. Mit Hilfe der Funktion RUND kann definiert werden:

Rückgabe: Der aufgerundete Wert.
Siehe auch: FNH, RUN, RUND
Beispiel: #1 = FNR(123.456) */* Feld #1 beinhaltet anschließend den Wert 123.46*

2.4. **FRA** - Aussondern des Dezimalwertes einer Zahl

Zahl FRA(Zahl *par1*)
Parameter: *par1* : Angabe der Zahl, deren Dezimalwert ausgesondert werden soll.
Beschreibung: Die Funktion sondert den Dezimalwert einer Zahl aus und gibt diesen zurück.
Rückgabe: Dezimalwert als 0.<dezimalwert>.
Siehe auch: FNH, FNR, RUN
Beispiel: #1=FRA(123.456) /* *Ergibt 0.456* , #1=FRA(-12.345) /* *Ergibt -0.345*

2.5. INT - Ganzzahlwert einer Zahl

Zahl INT(Zahl *par1*)
Parameter: *par1* : Angabe der Zahl
Beschreibung: Die Funktion retourniert den Ganzzahlwert einer Zahl, d.h. den Wert ohne Dezimalstellen.
Rückgabe: Ganzzahlwert.
Siehe auch: FRA
Beispiel: #1=INT(1234.56) /* *Ergibt 1234* , #1=INT(-12.345) /* *Ergibt -13*

2.6. **NOT** - Logische Negation

Zahl NOT(Zahl *par1*)
Parameter: *par1* : Angabe der Zahl
Beschreibung: Die Funktion retourniert 1 wenn *par1* gleich 0, 0 wenn *par1* ungleich 0.
Rückgabe: 0 oder 1.
Siehe auch: SGN
Beispiel: NOT(1) */* ist 0*

2.7. **POW** - Potenzierung

Zahl POW(Zahl *par1*, Zahl *par2*)

par2 : Angabe des Exponenten
Beschreibung: Die Funktion potenziert die Zahl *par1* mit dem Exponenten *par2*
Rückgabe: Potenzierter Wert.
Siehe auch: SQR
Beispiel: #1=POW(8,3) */* *Ergibt 512 (8*8*8)* , #1=POW(4,0.5) */* *Ergibt 2*

2.8. RUN - Aufrunden auf x Dezimalstellen

Zahl RUN(Zahl *par1*, Zahl *par2*)

par2 : Anzahl der Dezimalstellen, auf die gerundet werden soll
Beschreibung: Diese Funktion rundet eine Zahl auf die angegebene Anzahl Dezimalstellen auf.
Rückgabe: Aufgerundete Zahl.
Siehe auch: FNH, FNR, INT
Beispiel: #1=RUN(-123.4567,3) */* Feld 1 enthält anschließend den Wert -123.457*

2.9. RUND - Definition der FNR Auf-/Abrundung

Zahl RUND(Zahl *par1*, Zahl *par2*)

par2 : Anzahl Dezimalstellen, auf die auf-/abgerundet werden soll, z.B. 2
Beschreibung: Die RUND-Funktion definiert, wie die FNR-Funktion auf-/abrunden soll. Wenn *par1* positiv ist, wird aufgerundet, wenn negativ, wird abgerundet.
Rückgabe: Keine.
Siehe auch: FNR
Beispiel:

```
RUND(-25,2)      /* Abrunden auf 25 in 2 Dezimalstellen
RUND(5,2)        /* Aufrunden auf 5 in 2 Dezimalstellen
RUND(1,3)        /* Aufrunden auf 3 Dezimalstellen
RUND(1,2)        /* Normale FNR-Funktion
```

2.10. SGN - Untersuchen, ob Zahl negativ, Null oder positiv ist

Zahl SGN(Zahl *par1*)
Parameter: *par1* : Angabe der Zahl
Beschreibung: Die Funktion untersucht, ob eine Zahl negativ, gleich Null oder positiv ist.
Rückgabe:

```
  -1          Zahl ist negativ
   0          Zahl ist gleich Null
   1          Zahl ist positiv
```

Siehe auch: INT, NOT
Beispiel: #1=SGN(-123.45) */* Feld #1 enthält anschließend den Wert -1.*

2.11. **SQR** - Berechnen der Quadratwurzel einer Zahl

Zahl SQR(Zahl *par1*)
Parameter: *par1* : Angabe der Zahl, deren Quadratwurzel berechnet werden soll.
Beschreibung: Die Funktion berechnet die Quadratwurzel der Zahl in *par1*.
Rückgabe: Quadratwurzel.
Siehe auch: POW
Beispiel: #1=SQR(4) */* *Ergibt 2*

3. Textfunktionen

Dieser Abschnitt beschreibt die Funktionen für generelle Textänderungen und Editierung von numerischen Werten in Text.

3.1. CONV - Erstatten von Zeichen in einem Text

Text CONV(Text *par1*, Text *par2*, Text *par3*)

par3 : Angabe der neuen Zeichen, die eingesetzt werden sollen
Beschreibung: Die Funktion prüft jedes Zeichen im Text *par1*. Sofern ein Zeichen einem in *par2* engegebenen Zeichen entspricht, wird dieses mit dem neuen Zeichen in *par3* erstattet. Wenn Parameter 1 "abc" und Parameter 2 "ABC" beinhaltet, erstattet diese Funktion das Zeichen a mit A, b mit B und c mit C.
Rückgabe: Text, in dem die angegebenen Zeichen erstattet wurden.
Siehe auch: LOWER, SMAA, UPPER
Beispiel: #1 = CONV("hans", "hn", "lr") */* *ergibt "lars"*

3.2. **EDIT** - Editieren einer Ganzzahl

Text EDIT(Zahl *par1*, Text *par2*)

par2 : USING Editierungsmaske
Beschreibung: Die EDIT-Funktion konvertiert eine Ganzzahl in ein Textfeld, wobei die angegebene USING-Maske das Aussehen des Textes bestimmt.
Rückgabe: Editierter Text.
Siehe auch: NUMBER, USING
Beispiel:

```
#1 = EDIT(-123,"&&&,&&")        /* ergibt "001,23"
#1 = EDIT(123,"##&-#&&&")       /* ergibt "  0- 123"
#1 = EDIT(123,"fx.# og ##")     /* ergibt "fx.1 og 23"
```

3.3. **FIND** - Finden einer Zeichenfolge in einem Textfeld

Zahl FIND(Text *par1*, Text *par2*, Zahl *par3*, Zahl *par4*, Zahl *par5*)

Beschreibung: Die Funktion sucht nach der Zeichenfolge *par1* im Text *par2*. Bei Parameter müssen in " " angegeben werden.
Rückgabe: Retourniert -1, wenn die Zeichenfolge nicht gefunden wurde, sonst eine positive Zahl, die der Position im Text entspricht, beginnend mit 1.
Siehe auch:
Beispiel:

```
#1 = "Dies ist ein Text"
#2 = FIND("ein", #1)         /* Feld #2 enthält anschließend den Wert 10.
```

3.4. **LEN** - Länge eines Textes

Zahl LEN(Text *par1*)
Parameter: *par1* : Angabe des Textes
Beschreibung: Die Funktion berechnet die Länge eines Textes.
Rückgabe: Länge des Textes.
Siehe auch: SPOFF
Beispiel:

```
#1 = "SW-Tools ApS"
#2 = LEN(#1)           /* retourniert die Länge des Textes
```

Feld #2 enthält anschließend den Wert 12.

3.5. LOWER - Konvertieren eines Textes in Kleinbuchstaben

Text LOWER(Text *par1*)
Parameter: *par1* : Angabe des Textes, der konvertiert werden soll
Beschreibung: Die Funktion konvertiert einen Text in Kleinbuchstaben.
Rückgabe: Text in Kleinbuchstaben.
Siehe auch: CONV, SMAA, UPPER
Beispiel:

```
#1 = "DIES IST ein TEST XYZ"
#2 = LOWER(#1)   /* Feld #2 enthält anschließend "dies ist ein test xyz"
```

3.6. NAME - Aussondern in Vor- und Nachname

Text NAME(Text *par1*, Zahl *par2*)

Beschreibung: Die Funktion sondert, soweit möglich, Vor- und Nachname in einem gegebenen Textfeld aus und retourniert diesen entsprechend *par2*. Dieser Wert kann z.B. nachher für Sortierungen benutzt werden.
Hierzu wird die SSV-Textdatei WORDS.GER benutzt, in der jede Zeile besondere Ausdrücke wie Hr, Frau, GmbH u.ä. sowie eventuelle Erstattungen (Herr;Hr) für diese enthält.
Rückgabe: Name entsprechend *par2*.
Siehe auch: SMAA, SOGE
Beispiel:

```
#1 = NAME("HR CHRIS HANSON",0)    /* ergibt "HANSON, CHRIS Hr."
#1 = NAME("OLSEN, MICHAEL",1)     /* ergibt "MICHAEL OLSEN"
```

3.7. NUMBER - Konvertierung 'mystischer' Zahlen

Zahl NUMBER(Text *par1*)
Parameter: *par1* : Text, der eine Zahl enthält
Beschreibung: Die NUMBER Funktion extrahiert eine Zahl aus einem Textfeld, auch wenn zwischen den Ziffern Buchstaben oder Zeichen gefunden werden.
Rückgabe: Die gefundene Ganzzahl (es werden keine Dezimalstellen gebildet).
Siehe auch: EDIT, NUMS, USING
Beispiel:

```
#1=NUMBER("33)33 05 56") /* Telefonnummer in Zahl 33330556 konvertieren
#1=NUMBER("31/03-1997")  /* Das Datum wird in die Zahl 31031997 konvertiert
#1=NUMBER("ab1cd2&3.4")  /* Ergibt 1234
```

3.8. NUMS - Konvertierung eines Textfeldes in eine Zahl

Zahl NUMS(Text *par1*)
Parameter: *par1* : Textfeld, das eine Zahl enthält
Beschreibung: In einer Berechnungszeile wie #1=#2, wobei #1 ein numerisches und #2 ein alphanumerisches Feld ist, wird eine eventuelle Zahl in #2 von einem Text in einen numerischen Wert konvertiert. Das gleiche Resultat kann mit #1=NUMS(#2) erreicht werden, was jedoch nicht zwingend ist.
Möchte man mit den Zahlenwerten eines Textfeldes rechnen, wie z.B. #1=#2+#2, muß #1=NUMS(#2)+NUMS(#3) benutzt werden, da hiermit erreicht wird, daß die alphanumerische Zahl in einem Textfeld vor der Berechnung in einen numerischen Wert konvertiert wird.
Rückgabe: Numerischer Wert des Textfeldes. Das Dezimalkomma muß als Punkt angegeben sein.
Siehe auch: NUMBER
Beispiel: #1 = NUMS("ab111") + NUMS("222,22 test") + NUMS("333.33")
Feld 1 enthält anschließend die Summe der Textfelder, also = 555.33

3.9. PACK - Packen einer Zahl

Text PACK(Text *par1*, Zahl *par2*)

par2 : 0, z.Zt. nicht benutzt, reserviert für späteren Gebrauch
Beschreibung: Entspricht dem 8870 BASIC CALL 60,A$,B$ wie B$=PACK(A$)
Rückgabe: Der gepackte Wert des Feldes.
Siehe auch: UNPACK
Beispiel: #1 = PACK(#2) */* #1 = #2 gepackt.*

3.10. SMAA - Konvertiert einen Text in Klein-Großbuchstaben - Namen

Text SMAA(Text *par1*)
Parameter: *par1* : Angabe des Textes, der konvertiert werden soll
Beschreibung: Die Funktion konvertiert den Text in *par1* in Groß- und Kleinbuchstaben, d.h. der erste Buchstabe eines jeden Wortes wird in einen Großbuchstabe, alle anderen in Kleinbuchstaben konvertiert. Hierbei wird die SSV-Textdatei mit Hinblick auf Rechtschreibung und Erstattungen geprüft, z.B. im Zusammenhang mit Hr., GmbH, GmbH & Co KG. geprüft.
Beachten Sie bitte, daß die SMAA in DATAMASTER auch für die Online-Konvertierung von Namenfeldern benutzt werden kann.
Rückgabe: Der konvertierte Text.
Siehe auch: CONV, LOWER, NAME, UPPER
Beispiel:

```
#1 = SMAA("MICHAEL OLSEN")  /* ergibt "Michael Olsen"
#1 = SMAA("SORENCO GMBH")   /* ergibt "Sorenco GmbH"
```

3.11. SOGE - Bildung eines Suchbegriffes aus dem Adressfeld

Text SOGE(Text *par1*, Zahl *par2*)

par2 : Länge des Namenteils im Ergebnis.
Beschreibung: Aus einer angegebenen Adresse werden Straße und Nummer isoliert und anschließend in einem Feld neu zusammengestellt, wobei jetzt der Straßenname die in *par2* enthaltene Länge erhält. Diese Funktion kann z.B. im Zusammenhang mit Sortierungen oder Suchen verwendet werden.
Rückgabe: Straßenname mit der Länge *par2*, gefolgt von einer 4stelligen Hausnummer.
Siehe auch: LOWER, NAME, SMAA, UPPER
Beispiel:

```
#1 = SOGE("Baumweg 3",10)          /* ergibt "Baumweg_____3"
#1 = SOGE("27, Rue de Saute",8)    /* ergibt "RuedeSau__27"
```

3.12. SPOFF - Entfernen Voran- und nachgestellter Leerstellen in einem Text

Text SPOFF(Text *par1*, Bitflag *par2*)

Beschreibung: Die Funktion entfernt alle voran- und nachgestellten Leerstellen in einem Text. Leerstellen innerhalb eines Textes werden auf jeweils eine Position reduziert.
Rückgabe: Der behandelte Text.
Siehe auch: LEN
Beispiel:

```
#1="   Dies    ist    ein Text   "
#2=SPOFF(#1)          /* Feld #2 enthält anschließend "Dies ist ein Text".
```

3.13. UNPACK - Entpacken einer Zahl

Text UNPACK(Text *par1*, Zahl *par2*)

par2 : 0, wird z.Zt. nicht verwendet
Beschreibung: Entspricht dem 8870 BASIC CALL 61,A$,B$ wie B$=UNPACK(A$)
Rückgabe: Der entpackte Wert des Feldes.
Siehe auch: PACK
Beispiel: #1=UNPACK(#2) */* #1 ergibt #2 entpackt.*

3.14. UPPER - Konvertieren eines Textes in Großbuchstaben

Text UPPER(Text *par1*)
Parameter: *par1* : Angabe des Textes, der konvertiert werden soll
Beschreibung: Die Funktion konvertiert einen Text in Großbuchstaben.
Rückgabe: Der konvertierte Text.
Siehe auch: CONV, LOWER, SMAA
Beispiel:

```
#1="Dies ist ein Test"
#2=UPPER(#1)     /* Feld #2 enthält anschließend "DIES IST EIN TEST"
```

3.15. USING - Editieren einer Zahl

Text USING(Zahl *par1*, Text *par2*)

par1 : USING Maske für die Editierung
Beschreibung: Die USING-Funktion konvertiert eine Zahl in ein Textfeld, wobei die angegebene USING Maske das Format des Textes bestimmt.
Die Funktion kann auch mit der Syntax: Textfeld = Zahl USING "maske" benutzt werden.
Rückgabe: Der editierte Text.
Siehe auch: EDIT
Beispiel:

```
#1 = USING(-123,"&&&,&&")            /* ergibt "001,23"
#1 = USING(123.45,"#####")           /* ergibt "__123"
#1 = USING(1234.56,"###,###.##")     /* ergibt "__1,234,56"
#1 = 123.45 USING "#####"            /* ergibt "__123"
```

4. Prüfziffern und Gültigkeitsprüfungen

Dieser Abschnitt behandelt die Funktionen für Prüfziffern und Gültigkeitsprüfungen von numerischen und alphanumerischen Werten (Text und Zahlen).

4.1. CCODE - Feld Prüftext (DATAMASTER Checkcodetext)

Text CCODE(Text *par1*, Feld *par2*)

par2 : Feldreferenz mit Definition des Checkcode, z.B. "7", "#7", "va#7", "va07"
Beschreibung: Diese Funktion findet die Checkdefinitionen für der Feld *par2*. Die hierzu angegebene Text für der Wert *par1* wird retourniert.
Rückgabe: Der Checktext.
Siehe auch: VALID, VALCH
Beispiel: #1 = CCODE(9,"va#7") */* ergibt "Special"*

4.2. **CHECK** - OCR Prüfung

Text CHECK(Text *par1*)
Parameter: *par1* : Angabe einer Zahl (Nummer), z.B. Kundennr.
Beschreibung: Die Funktion behandelt einen num. Wert und retourniert einen Text, der den num. Wert mit der zugehörigen OCR-Prüfziffer enthält.
#47=CHECK (#19) bewirkt die Bildung einer OCR-Prüfziffer Modulo 10 mit der Gewichtung 212121.. für das Textfeld #19.
CHECK("123456789012345") retourniert also einen Text mit einem zusätzlichen Zeichen: "1234567890123452".
Rückgabe: Text plus OCR-Prüfziffer.
Siehe auch: CHEX
Beispiel: #1 = CHECK("33330556") */* ergibt "333305563"*

4.3. CHEX - Modulo 11 Prüfziffer

Text CHEX(Text *par1*, Text *par2*)

par2 : Gewichtung für die Berechnung der Prüfziffer (2 Ziffern per zu prüfendes Zeichen).
Beschreibung: #47=CHEX (#15, "01020304") berechnet wie die CHECK Routine eine Prüfziffer und fügt diese am Ende des Textfeldes der zu prüfenden Zahl hinzu. Die Prüfziffer wird Modulo 11, mit der Gewichtung 01,02,03,04 usw. entspr. 2. Parameter berechnet.
Rückgabe: Text plus Prüfziffer.
Siehe auch: CHECK
Beispiel: #2=CHEX("330556", "010203040506") */* *ergibt "3305569"*

4.4. VALCH - Prüfung eines Textes auf Gültigkeit

Zahl VALCH(Text *par1*, Text *par2*)

par2 : Angabe der Prüfvorschriften (zugelassene Intervalle, getrennt mit Komma)
Beschreibung: Die Funktion prüft, ob der Text in *par1* den in *par2* angegebenen Texten entspricht. Die Texte in *par2* müssen durch Komma getrennt sein.
Rückgabe: Retourniert 0, wenn *par1* nicht innerhalb der in *par2* angegebenen Grenzen liegt.
Siehe auch: CCODE, VALID
Beispiel: #1=VALCH("Chris", "Anne,Chris,Ole,Michael") */* Feld #1 enthält anschließend den Wert 2.*

4.5. VALID - Prüfen, ob eine Zahl innerhalb angegebener Grenzwerte liegt

Zahl VALID(Zahl *par1*, Zahl *par2*, Zahl *par3*)

. **Beschreibung:** Die Funktion prüft den Wert in *par1* entsprechend den in *par2* angegebenen Grenzwerten. Die Syntax für die Grenzwerte ist wie folgt:
"1,2,8-10,12" D.h. die Werte 1, 2, 8 bis 10 und 12 sind zugelassen.
"-1,2,8-10,12" Ein Minus voran bedeutet, daß die entsprechenden Werte nicht zugelassen sind.

```
#20="1-3,8-12"
VALID(15,#20,1)
```

ändert den Wertebereich de Feldes #20 und fügt den Wert 15 ein: "1-3,8-12,15"
Rückgabe: 0, wenn Wert in *par1* nicht entsprechend *par2* zugelassen ist.
Siehe auch: CCODE, VALCH
Beispiel: #1 = VALID(9, "1,2,8-10,12")
Feld #1 enthält anschließen den Wert 3 (der zu prüfende Wert wurde im 3. Element im im 2. Parameter gefunden).

5. Datum Funktionen

In diesem Abschnitt wird die Behandlung eines Datums und das Rechnen mit einem Datum beschrieben.

5.1. **DATE** - Datum JJJJMMTT

Zahl DATE()
Rückgabe: Das aktuelle Datum in der Form JJJJMMTT.

5.2. DATECALC - Berechnung eines Datums

Datum DATECALC(Datum *par1*, Zahl *par2*, Zahl *par3*, Zahl *par4*, Zahl *par5*)

par5 : Angabe des Tages (TT) oder Tage
Beschreibung: Mit dieser Funktion kann ein bestimmtes Datum gesetzt bzw. berechnet (Addieren/ Subtrahieren) werden. Wird der *par2* gleich 0 gesetzt, kann das Datum mit Hilfe der Parameter *par3-par5* gesetzt werden. Sofern die Parameter 3, 4 und 5 ungleich Null sind, wird der Parameter 1 nicht berücksichtigt. Möchte man jedoch nur den Monat bestimmen, wird vom Datum in *par1* ausgegangen, und der Monat entsprechend *par4* gesetzt.

Rückgabe: Das berechnete Datum in der Form JJJJMMTT.
Siehe auch: DAY, FNA, FNB, FND, FNU, FNV, FNY, MONTH, WDAY, WORKD
Beispiel:

```
#1=DATECALC(0, 0, 1997, 10, 16)   /* Datum = 16.Oktober 1997 (19971016)
#1=DATECALC(19970101, 1, 0, 2, 0) /* Addiere 2 Monate zum Datum (19970301)
#1=DATECALC(19971016, 2, 1, 2, 3) /* Subtrahiere 1 Jahr, 2 Mn. und 3 Tage
                                     vom Datum (19960813)
```

5.3. DAY - Beschreibung des Datums in Textform

Text DAY(Datum *par1*)
Parameter: *par1* : Angabe des Datum in der Form JJJJMMTT
Beschreibung: Die Funktion bildet einen Text, der das Datum als: <?> <Wochentag>, den <Tag>. <Monatsbezeichnung> <Jahr>
beschreibt. Sofern der Tag ein Feiertag ist, wird das erste Zeichen <?> gleich * gesetzt. Handelt es sich um einen halben Feiertag, wird das entspr. Zeichen gleich / gesetzt. Es wird der gleiche Kalender wie in WORKD beschrieben benutzt.
Rückgabe: Das Datum als Text.
Siehe auch: DATECALC, FNA, FNB, FND, FNU, FNV, MONTH, WDAY, WORKD
Beispiel: #1 = DAY(19931016) */* Bilde den Datumtext für 19931016*
Feld #1 enthält anschließend den Text "* Sonnabend, den 16 Oktober 1993"

5.4. FNA - Umrechnen eines Datums in Anzahl Tage ab Jahr 0

Zahl FNA(Datum *par1*, Zahl *par2*)

Beschreibung: Die Funktion rechnet das angegebene Datum um in die Anzahl der Tage, die seit dem Jahr 0 verstrichen sind. Die Funktion kann u.a. dazu benutzt werden, den Unterschied in Tagen zwischen zwei Daten zu berechnen.
Rückgabe: Anzahl Tage ab Jahr 0.
Siehe auch: FNB, FND, FNU, FNV, DATECALC, DAY, MONTH, WDAY, WORKD
Beispiel:

```
#1 = 19931215           /* Datum = 15. Dezember 1993
#2 = FNA(#1)            /* Wieviele Tage seit Jahr 0 ?
#3 = #2 - FNA(19931202) /* Wieviele Tage seit dem 2. Dez. 1993 ?
```

Feld #2 enthält anschließend den Wert 728277 und Feld #3 den Wert 13.

5.5. FNB - Umrechnen von Anzahl Tagen seit Jahr 0 in ein Datum

Datum FNB(Zahl *par1*, Zahl *par2*)

Beschreibung: Die Funktion rechnet die Anzahl Tage seit Jahr 0 um in ein gültiges Datum.
Rückgabe: Datum in der Form JJJJMMTT.
Siehe auch: DATECALC, DAY, FNA, FND, FNU, FNV, MONTH, WDAY, WORKD
Beispiel:

```
#1 = FNA(19931215) /* Rechne das Dato 15. Dez. 1993 um
#2 = FNB(#1 + 9)   /* addiere 9 Tage und rechne um in JJJJMMTT
```

Das Feld #2 enthält anschließend den Wert 19931224, also 24. Dez. 1993.

5.6. FND - Wenden eines Datums

Datum FND(Datum *par1*)
Parameter: *par1* : Angabe des Datums TTMMJJ
Beschreibung: Diese Funktion ist für Systeme, die das Datum mit nur 6 Ziffern speichern, von Bedeutung. Um eine mehr lesefreundliche Datumsangabe zu erhalten, oder mit Hinblick auf Sortierungen kann es notwendig sein, das Datum zu wenden, da z.B.

970101 ist größer als 961231, aber 311296 ist größer als 10197

Es ist also notwendig FND(#7) anstelle von #7 zu benutzen, wenn Feld #7 ein Datumfeld in der Form TTMMJJ ist.
Rückgabe: Datum in der Form TTMMJJ oder JJMMTT.
Siehe auch: DATECALC, DAY, FNA, FNO, FNU, FNV, FNY, MONTH, WDAY, WORKD
Beispiel:

```
#1 = FND(310395)            /* ergibt 950331
#1 = FND(950331)            /* ergibt 310395
#1 = FND(19950331)          /* ergibt 310395
```

5.7. FNE - Umrechnen eines Datums in eine Monatsnummer

Zahl FNE(Datum *par1*)
Parameter: *par1* : Angabe des Datums in der Form JJJJMMTT oder JJMMTT
Beschreibung: Die Funktion rechnet ein Datum (Jahr+Monat) um in Anzahl Monate seit Jahrhunderwechsel, und kann u.a. dazu verwendet werden, Monatsintervalle zu berechnen.
Rückgabe: Anzahl Monate (seit Jahrhundertwechsel)
Siehe auch: DATECALC, DAY, FNA, FNB, FND, FNV, MONTH, WDAY, WORKD
Beispiel: #1 = FNE(19950331) */* *ergibt 1143 = 95*12+3*

5.8. FNF - Umrechnen eines Datums in Tagesnummer, 360 Tage pr. Jahr

Zahl FNF(Datum *par1*)
Parameter: *par1* : Angabe des Datums in der Form JJJJMMTT oder JJMMTT
Beschreibung: Die Funktion rechnet ein Datum in Anzahl Tage (ab Jahr 0) um. Es wird mit 360 Tagen per Jahr gerechnet (wie in FNA(datum,360).
Rückgabe: Anzahl Tage ab Jahr 0.
Siehe auch: FNA
Beispiel:

```
#1 = FNF(19950331) /* ergibt 718290
#1 = FNF(950331)   /* ergibt 34290
```

5.9. FNO - Konvertieren eines Datums in TTMMJJ

Datum FNO(Datum *par1*)
Parameter: *par1* : Datum in der Form TTMMJJ, JJMMTT oder JJJJMMTT
Beschreibung: Unabhängig, wie das Datum gespeichert wurde, wird dieses für z.B. eine Ausgabe in TTMMJJ gewendet und retourniert.
Rückgabe: TTMMJJ
Siehe auch: FND, FNY
Beispiel:

```
#1 = FNO(310395)        /* ergibt 310395
#1 = FNO(950331)        /* ergibt 310395
#1 = FNO(19950331)      /* ergibt 310395
```

5.10. FNU - Umrechnen eines Datums in einen Wochentag

Zahl FNU(Datum *par1*)
Parameter: *par1* : Datum in der Form JJJJMMTT
Beschreibung: Die Funktion bestimmt den Wochentag eines bestimmten Datums.

Siehe auch: DATECALC, DAY, FNA, FNB, FND, FNV, MONTH, WDAY, WORKD
Beispiel: #1 = FNU(19931215) */* Welcher Wochentag ist der 15. Dez. 1993 ?*
Feld #1 enthält anschließend den Wert 4 (=Mittwoch).

5.11. FNV - Umrechnen eines Datums in Wochennummer / Wochennummer in Datum

Zahl FNV(Zahl *par1*)
Parameter: *par1* : Angabe des Datums als JJJJMMTT, oder Wochennummer als JJJJWW
Beschreibung: Die Funktion konvertiert ein Datum in eine Wochennummer (JJJJWW), sofern *par1* ein Datum enthält. Enthält *par1* jedoch eine Wochennummer (JJJJWW), berechnet die Funktion das entsprechende Datum. Gleiche Funktion wie WEEK(Datum).
Rückgabe: Die Zahl JJJJWW, wobei WW = Wochennummer, oder Datum JJJJMMTT.
Siehe auch: DATECALC, DAY, FNA, FNB, FND, FNU, MONTH, WDAY, WEEK , WORKD
Beispiel:

```
#1 = FNV(19931016)    /* Berechne Wochennr. für 16. Okt. 1993
#2 = FNV(#1)          /* Berechne letzten Sonntag für Wochennr. 41
```

Feld #1 enthält anschließend den Wert 199341, also Woche Nr. 41. Feld #2 enthält das Datum 19931010.

5.12. **FNY** - Konvertieren eines Datums in JJJJMMTT

Datum FNY(Datum *par1*)
Parameter: *par1* : Datum in der Form TTMMJJ, JJMMTT oder JJJJMMTT
Beschreibung: Unabhängig davon, wie das Datum gespeichert ist, wird das Datum in der Form JJJJMMTT retourniert, und kann anschließend für Berechnungen verwendet werden.
Rückgabe: JJJJMMTT
Siehe auch: FND, FNO
Beispiel:

```
#1 = FNY(310395)        /* ergibt 19950331
#1 = FNY(950331)        /* ergibt 19950331
#1 = FNY(19950331)      /* ergibt 19950331
```

5.13. **MONTH** - Konvertieren des Monats in Textform

Text MONTH(Datum *par1*)
Parameter: *par1* : Angabe des Datums in der Form JJJJMMTT
Beschreibung: Die Funktion konvertiert einen angegebenen Monat in Textform.
Rückgabe: Name des Monats.
Siehe auch: DATECALC, DAY, FNA, FNB, FND, FNU, FNV, WDAY, WORKD
Beispiel: #1 = MONTH(19931016) */* *16101993*
Feld #1 enthält anschließend den Wert "Oktober".

5.14. **TIME** - Uhrzeit SSMMSS

Zahl TIME()
Rückgabe: Aktuelle Uhrzeit in der Form SSMMSS (SS = Stunden, MM = Minuten , SS = Sekunden).

5.15. WDAY - Konvertieren eines Datums in einen Wochentag (Text)

Text WDAY(Datum *par1*)
Parameter: *par1* : Datum in der Form JJJJMMTT
Beschreibung: Die Funktion bildet ausgehend von dem angegebenen Datum einen Text für den entsprechenden Wochentag als: <?> Wochentag
Handelt es sich um einen Feiertag, ist das erste Zeichen <?> gleich *, bei einem halben Feiertag gleich /. Es wird der Kalender wie in WORKD beschrieben verwendet.
Rückgabe: Wochentag als Text.
Siehe auch: DATECALC, FNA, FNB, FND, FNU, FNV, MONTH, WDAY, WORKD
Beispiel: #1 = WDAY(19931016) */* *Feld #1 enthält anschließend den Text "*Sonnabend"*

5.16. **WEEK** - Umrechnen eines Datums in Wochennummer oder Wochennummer in Datum

Zahl WEEK(Zahl *par1*)
Parameter: *par1* : Angabe des Datums in der Form JJJJMMTT, oder Wochennummer in der Form JJJJWW
Beschreibung: Die Funktion konvertiert ein Datum in eine Wochennummer JJJJWW, sofern *par1* ein Datum enthält. Enthält *par1* eine Wochennummer, wird diese in ein Datum konvertiert, das dem letzten Sonntag vor der angegebenen Wochennummer entspricht. Gleiche Funktion wie FNV(Datum).
Rückgabe: Retourniert eine Zahl JJJJWW, wobei WW = Wochennummer, oder ein Datum JJJJMMTT
Siehe auch: FNV
Beispiel:

```
#1 = WEEK(19931016)    /* Berechne Wochennr. für 16. Oktober 1993
#2 = WEEK(#1)          /* Berechne letzten Sonntag vor Woche Nr. 41
```

Feld #1 enthält anschließend den Wert 199341, also Woche Nr. 41 in 1993. Feld #2 enthält das Datum 19931010.

5.17. **WORKD** - Berechnen der Anzahl Arbeitstage zwischen zwei Daten

Zahl WORKD(Datum *par1*, Datum *par2*)

par2 : Angabe eines Datums in der Form JJJJMMTT
Beschreibung: Die Funktion berechnet die Anzahl der Arbeitstage zwischen den zwei Datumsangaben.
#47 = WORKD (#15, #PD) berechnet die Anzahl der Arbeitstage zwischen Datum in Feld 15 und dem eingegebenen 'per Datum'.
Zuerst werden die Anzahl Tage zwischen den beiden Daten berechnet. Anschließend werden alle Sonnabende und Sonntage abgezogen. Jetzt wird im internen Kalender nachgeschlagen, um alle eventuellen Feiertage, die nicht auf einen Sonnabend oder Sonntag fallen, abzuziehen.
Der interne Kalender kann, falls notwendig, individuell angepaßt werden. Die Funktion basiert ihre Berechnungen auf der Datei RAPDAY.GER. Diese Datei ist eine SSV Textdatei, in der jede Zeile einen Feiertag in der Form JJJJMMTT enthält. Handelt es sich um einen halben Feiertag, wird der Wert 50 (=50%) angehängt, z.B. 19960605;50.
Rückgabe: Anzahl Arbeitstage zwischen zwei Daten.
Siehe auch: DATECALC, FNA, FNB, FND, FNU, FNV, MONTH, WDAY, WORKD
Beispiel: #1 = WORKD(19930420, 19930430) */* Feld #1 enthält anschließend den Wert 19.*

6. Behandlung mehrerer Felder gleichzeitig

In diesem Abschnitt wird die Behandlung mehrerer Felder gleichzeitig (Feldergruppen) beschrieben, hierunter speziell die Funktion LET.

6.1. LET - Berechnung von Feldern in Gruppen

Zahl LET(Felder *par1*)
Parameter: *par1* : Angabe eines oder mehrerer Felder
Beschreibung: Die Funktion wird zur Bearbeitung mehrerer Felder in Gruppen verwendet. Die Felder können mit den folgenden Ausdrücken behandelt/berechnet werden. Felder **XX** Konstante/Feld, wobei **xx** folgende Bedeutung haben kann:

Operator	**Beschreibung**
=	Setze Felder gleich mit
+=	Addiere zu Felder
-=	Subtrahiere von Feldern
*=	Multipliziere Felder mit
/=	Dividiere Felder mit
%=	Setze Felder gleich mit dem Divisionsrest

Rückgabe: Retourniert den Wert 0, wenn Berechnung fehlerfrei abgeschlossen wurde.
Siehe auch: CLEAR, ZERO
Beispiel:

LET-Ausdruck	**Funktion**
LET("#1-10=12")	Feld 1 bis 10 wird gleich 12 gesetzt
LET("#20,25=3,7")	#20=3 und #25=7
LET("#20-25=le#1-10")	Feld 20-25 wird gleich Feld 1-6 der Datei le gesetzt
LET("#20-25=le#1-2")	#20=#22=#24=le#1, #21=#23=#25=le#2
LET("le#1,3,va#7=#1,ku#3")	Mehrere Dateien können benutzt werden
LET("#20-25+=1")	Addiere 1 auf die Felder 20-25

6.1.1. LET - Zuordnen von Werten (IQ)

Zahl LET(Felder *par1*)
Parameter: *par1* : Angabe eines oder mehrerer Felder
Beschreibung: Die LET Anweisung wurde erweitert, um auch Werte in Felder anderer Programme abzustellen. Es kann jetzt auch bei Transaktionsabfragen die LET Anweisung über den Zeilenindex gesteuert werden.
Rückgabe: Retourniert den Wert 0, wenn Berechnung fehlerfrei abgeschlossen wurde.
Siehe auch:
Beispiel:

Letanweisung	**Funktion**
LET (20.#1-3=#1-3)	Felder 1-3 des Programms 20 = Felder des aktuellen Programms
LET (#1-3=20.#4-6)	Felder 1-3 des aktuellen Programms = Felder 4-6 des Programms 20
LET (#10=#3.4)	Feld 10 = Feld 3 in Transaktionszeile 4

6.1.2. LET - Anlage neuer Dateien (RAP)

Zahl LET(Felder *par1*)
Parameter: *par1* : Angabe eines oder mehrerer Felder
Beschreibung: Die LET Funktion kann zur Anlage neuer Dateien benutzt werden.
Rückgabe: Retourniert den Wert 0, wenn Berechnung fehlerfrei abgeschlossen wurde.
Siehe auch: INSERT, UPDATE, *Rapgen Handbuch*
Beispiel:

Letanweisung	**Funktion**
LET(aa=#1-3,87,le#2)	Definiere Datei aa, Schlüssel=aa#1, Type=1.Datenbanktreiber
LET(aa=#1-3,6K,15D)	Schlüssel aa#4 und aa#5 (Duplikate)
LET(aa=#1-3,6,15:2,NP)	Schlüssel aa#2 und rel. Satznr. (Duplikate)
LET(aa=#1-3),12000	12000 Datensätze (Standard sind 1000 Sätze)
LET(aa=#1-3),-1	Datei wird jedesmal neu angelegt
LET(aa=#1-3),1000,xnet	Datei ist eine XNET Datei
LET(aa=#1-3) -acc	Zugriffsdatei, wird jedesmal neu angelegt
LET(07/aa=#1-3),25	Die LU kann für Basic Dateien angegeben werden

6.2. CLEAR - Löschen aller Felder in einer Datei (RAP)

Zahl CLEAR(Datei *par1*)
Parameter: *par1* : Angabe des Dateikürzels
Beschreibung: Die Funktion löscht alle Felder in der angegebenen Datei.
Rückgabe: Retourniert den Wert, wenn das Löschen fehlerfrei durchgeführt wurde.
Siehe auch: ZERO
Beispiel:

```
UPDATE(1)           /* Datei wird ajourgeführt
CLEAR(VA)           /* Löschen aller Felder in der Artikeldatei
VA#1 = "1234"       /* Artikelnummer
INSERT(VA)          /* Schreibe neuen Satz in die Artikeldatei
```

Es wird ein neuer Satz in die Artikeldatei geschrieben. Alle Felder, mit Ausnahme der Artikelnummer, sind gelöscht.

6.3. CLRFLAG- Setzen von Feldoptionen (IQ)

CLRFLAG(Felder *par1*, Zahl *par2*, Zahl *par3*)

Beschreibung: Jedem Bildschirmfeld sind Parameter zugeordnet, die die Verwendung definieren. Mit der SETFLAG Funktion werden dies Parameter gesetzt, mit CLRFLAG gelöscht. Siehe SETFLAG.
Siehe auch: SETFLAG, GETFLAG
Beispiel: CLRFLAG("#12,44",7,0)

6.4. COLOR - Bestimmen der Hintergrundsfarbe für eine Feldgruppe

COLOR(Felder *par1*, FarbeRot *par2*, FarbeGrün *par3*, FarbeBlau *par4*)

par4 : Blauer Farbenwert (0-255)
Beschreibung: Die Hintergrundsfarbe für die angegebenen Felder wird entsprechend RGB Wert gesetzt. d.h. die Flächen werden in der angegebenen Farbe gezeigt.
Rückgabe: Keine
Siehe auch: COLORF
Beispiel:

```
COLOR("#3-4",255,0,0)   /* Feld 3 und 4 erhalten eine rote Box
COLOR("#3-4",-1)        /* Keine Hintergrundsfarbe
```

6.5. COLORF - Bestimmen der Vordergrundsfarbe für eine Feldgruppe

COLORF(Felder *par1*, FarbeRot *par2*, FarbeGrün *par3*, FarbeBlau *par4*)

par4 : Blauer Farbenwert (0-255)
Beschreibung: Die Vordergrundsfarbe für die angegebenen Felder wird entsprechend RGB Wert gesetzt. d.h. die Texte werden in der angegebenen Farbe gezeigt.
Rückgabe: Keine
Siehe auch: COLOR
Beispiel: COLORF("#3-4",0,0,255) */* Texte/Zahlen in Feld 3 und 4 werden in blauer Farbe gezeigt.*

6.6. DIALOG - Funktion für zusätzliche Eingabe

Zahl DIALOG(Felder *par1*)
Parameter: *Par1*: Felder, die im Dialog angezeigt werden sollen;
Beschreibung: Die DIALOG Funktion gibt dem Anwender die Möglichkeit, Dialogfenster mit ausgewählten Feldern an jeder Stelle der Listverarbeitung oder einem IQ Programm zu definieren.
DIALOG("#1,7-8,le#3") definiert einen Dialog mit einem bestimmten Feld. Die Beschreibung für dieses Feld, wenn vorhanden, wird gleichzeitig als fließende Eingabehilfe verwendet.
Folgende Optionen können bezüglich eines Feldes gewählt werden:

```
Lxxxx   Zeile    (Dialog Einheit)
Pxxxx   Position (Dialog Einheit)
Hxxxx   Höhe     (Dialog Einheit)
Wxxxx   Breite   (Dialog Einheit)
N       Kein vorangestellter Text
N1      Addiere Feldnummer zu vorangestelltem Text
N2      Anzeige des Textes über dem Feld
C       COMBOBOX, Feldprüfungsdefinitionen gezeigt als Wert
O       LISTBOX, Feldprüfungsdefinitionen gezeigt als Wert
:xx     Springe auf nächste Spalte, Feldzeile xx
+xx     Springe xx Feldzeilen abwärts
```

.
Rückgabewert: OK=0, ANNULIEREN=1
Siehe auch: PARAMS
Beispiel:

```
DIALOG("#1-3,11")             /* Dialog mit einem gegebenen Feld
```

6.7. GETFLAG- Erlaubt die Abfrage der Feldattribute (IQ)

Zahl GETFLAG(Felder *par1*, Zahl *par2*, Zahl *par3*)

Beschreibung: Jedes Bildschirmfeld ist mit Parametern (Bits) beschrieben, die deren Gebrauch definieren. Die Funktion SETFLAG kann benutzt werden um die Feldattribute zu setzen, CLRFLAG entfernt sie wieder. Die Funktion GETFLAG kann benutzt werden um die Feldattribute zu lesen.
Rückgabe: Keine
Siehe auch: SETFLAG, CLRFLAG
Beispiel: GETFLAG("#12,44",7,0)

6.8. SETFLAG- Setzen der Optionen für ein Feld (IQ)

SETFLAG(Felder *par1*, Bitflag *par2*, Zahl *par3*)

Beschreibung: Jedem Feld auf dem Bildschirm sind eine Reihe von Parametern zugeordnet. Die Funktion SETFLAG wird genutzt, um diese Kennungen (flags) zu setzen, CLRFLAG, um die Kennungen zu löschen.
Normaleweise soll für *Par3* Type nur 0 angegeben sein.
Rückgabe: Keine
Siehe auch: GETFLAG, CLRFLAG
Beispiel: SETFLAG("#12,44",7,0)

6.9. ZERO - Löschen einer Anzahl von Feldern

ZERO(Felder *par1*)
Parameter: *par1* : Feldangabe
Beschreibung: Die angegebenen Felder werden gelöscht. ZERO hat die gleiche Funktion wie LET.
Rückgabe: Keine
Siehe auch: LET, CLEAR
Beispiel: ZERO("3,19") */* Löschen der Felder 3 und 19.*

7. Kontrollfunktionen für Listen

Dieser Abschnitt beschreibt die Funktionen, die zur Steuerung und Kontrolle von Berechnungen und Ausdrücken in einer Liste in RAPGEN benutzt werden können.
Die Funktionen CHAIN, MESS und RETURN können auch in IQ und DATAMASTER verwendet werden. Die übrigen Funktionen sind ohne Bedeutung für Bildschirmprogramme.

7.1. CHAIN - Start der nächsten Liste oder eines neuen Programms (RAP)

Zahl CHAIN()

par3 : Leer oder Index,Summenebene,Firmanr
Beschreibung: CHAIN(7) startet die Liste Nr. 7 unmittelbar nach Abschluß dieser Liste. Hierbei werden die gleichen Startparameter wie für die aktuelle Liste benutzt.
CHAIN(7,",310395,-,9999",",1") setzt das 'per Datum' gleich 310395, Startvon bleibt leer, Stopbei gleich 9999 und unterstes Summenniveau gleich 1. Die übrigen Parameter bleiben unverändert.
CHAIN(2007) startet die Liste Nr. 7 im Untersystem 2.
CHAIN(-1,"c:/windows/write.exe") startet das aktuelle (Windows-)Programm.
Bei jedem CHAIN(nr) wird eine neue lfd. Nummer (von 1 aufwärts) zugeteilt. Wurde die Liste aus dem Menü gestartet, erhält diese die lfd. Nummer 0. Mit #20=CHAIN(), wobei CHAIN ohne Parameter benutzt wird, kann diese Nummer gelesen werden. Hiermit kann eine Liste mehrere Male gestartet werden, z.B. wenn mann eine bestimmte Anzahl Kopien benötigt.
CHAIN("c:/windows/write.exe") kann in IQ/DATAMASTER Programmen verwendet werden, um ein neues WINDOWS-Programm zu starten.
Rückgabe: CHAIN() retourniert die aktuelle lfd. Nummer.
Siehe auch: EXIT, CHAINR
Beispiel:

```
#20=CHAIN()       /* Dies ist die Liste Nr. 7
IF #20<3 CHAIN(7) /* Die gleiche Liste wird 4 x nacheinander gestartet.
```

7.1.1. CHAINR - Direkte Verkettung eines Programms bzw. externen Kommandos (RAP)

par3 : Leer oder Index,Summenebene,Firmanr
Beschreibung: Das CHAIN Kommando wird immer unter ZULETZT abgesetzt, d.h. das nächste Programm wird gestartet, wenn das aktuelle beendet ist.
Benutzen Sie CHAINR anstelle von CHAIN, um das aktuelle Programm abzubrechen und unmittelbar ein neues Programm aufzurufen und zu starten.
Rückgabe: Keine
Siehe auch: EXIT , CHAIN
Beispiel: CHAINR(-1,"Notepad") */* Notepad aktivieren*

7.1.2. CHAIN - Verkettung eines IQ Programmes bzw. externes Kommando (IQ)

CHAIN(text *par1*, text *par2*)

par2 : Schlüßel für übertragene Satz
Beschreibung: Aktiviert eine Programmnummer bzw. eine Windows Kommandofolge
Rückgabe: Keine
Siehe auch: EXIT, ISACTIVE, WAIT
Beispiel:

```
   CHAIN("20")     startet das Programm 20
   CHAIN("+5")     startet und aktiviert das Programm 5
   CHAIN(">5")     startet das Programm 5, der aktuelle Satz wird nicht
übertragen
   CHAIN("$5")     startet und aktiviert das Programm 5, und wartet, bis
                   dieses abgeschlossen ist
   CHAIN("+5",#1) startet das Programm 5, das den Satz entsprechend
                   in Feld 1 liest

   #20="notepad"
   #20="command.com /C edit myfile.txt"
   CHAIN(#20)          startet das spezifische Windows Programm

   CHAIN("rapwin &")  & als letztes Zeichen bedeutet, daß IQ nicht gestoppt
                        wird während das neu gestartete Programm abläuft
```

7.2. **WAIT** - Parkt das Programm (IQ)

WAIT(programm *par1*)
Parameter: *par1* : Programmnummer
Beschreibung: Parkt das entsprechende Programm. Wenn das aufgerufene Programm beendet wird. (s. EXIT) werden die Berechnungen weiter ausgeführt.
Rückgabe: Keine.
Siehe auch: CHAIN , EXIT
Beispiel: WAIT(20) */* Warten für Programm 20*

7.3. COMPILE - Kompilieren einer Liste (RAP)

COMPILE(Zahl *par1*)

Diese Funktion kann nur verwendet werden, wenn ein C-Compiler installiert und RAPGEN mit einer Kompilierungslizenz erworben wurde.
Beschreibung: Um die Wahl 'COMPILE' im 'Parametermenü' zu umgehen, jedesmal wenn eine Liste nach Änderungen gestartet werden soll, kann die Kompilierung in einer Berechnungszeile festgelegt werden.
Siehe auch: INSTALL
Beispiel: COMPILE /* *Liste wird kompiliert*

7.4. EXIT - Beenden einer Liste (RAP)

Zahl EXIT(Zahl *par1*)

Beschreibung: Die Funktion beendet eine Liste bzw. den aktuellen Durchlauf (Sortieren/Drucken).
Siehe auch: CHAIN, CHAINR, MESS
Beispiel:

```
READ(le)              /* Lesen der Lieferantendaten
IF #OK THEN BEGIN     /* Beenden, wenn Lieferant nicht gefunden wird
#12="Lieferant ", le#1, " nicht gefunden:"
MESS(#12)
EXIT(2)
END
```

7.4.1. EXIT - Schließt ein IQ Programm (IQ)

EXIT(Zahl *par1*)
Parameter: *par1* : Programmnummer zu Schließen
Beschreibung: EXIT(0) schließt das aktuelle IQ- Programm.
Rückgabe: Keine
Siehe auch: CHAIN , MESS, WAIT
Beispiel:

```
   EXIT(20) schließt das Programm 20, wenn es geöffnet ist, 1020 gibt Subsystem
1
   EXIT(-1) schließt das Programm Auswahlmenü.
   EXIT(-2) schließt das Feld Auswahlmenü innerhalb eines Programms.
   EXIT(-3) schließt alles und verläßt IQ.
```

7.5. KEYS - Start/Stop Angaben (RAP)

Zahl KEYS()

par2 : Eventueller fester Name für .KEY Definitionsdatei
Beschreibung: Mit der KEYS-Funktion kann eine Liste über mehrere Start/Stop Intervalle, die als Zeilen in einer Textdatei definiert sind, ablaufen. KEYS ersetzt hiermit die Angabe von START/STOP (eventuell auch INDEX) bei Start der Liste.
Die KEYS-Datei kann mit einem einfachen Texteditor angelegt werden, und kann z.B. folgende Zeilen enthalten:

```
0001
1000-1999
0005-0099,0200,0155-0157
2:205-271
47/2000-2500
```

In jeder Zeile können Einzelschlüssel oder Intervalle für eine Ausgabe der Liste angegeben werden. Mit 2: wird angegeben, daß der Index 2 benutzt werden soll. 47/ gibt eine Berechnungscode an, der mit z.B. #20=KEYS() gelesen und anschließend für die Berechnung benutzt werden kann.
Benutzt man KEYS(), erhält man eine gesammelte Liste mit allen angegebenen Intervallen. KEYS(1) bewirkt, daß für jede Zeile in der KEYS-Datei eine selbständige Liste ausgegeben wird. ENDSUM-Funktionen kann eventuell im Zusammenhang mit mit einer Gesamtsumme benutzt werden.
Um eine Liste in dieser Weise steuern zu können, muß KEYS nicht unbedingt in der Liste selbst angegeben sein. Bei Start einer Liste kann in START VON angegeben werden:

```
(aa)                          Start mit KEYS-Datei aa
(1000,1100-1200,0004          Durchlauf dieser Intervalle
```

Wird keine Path/Extension für die KEYS-Datei angegeben, wird davon ausgegangen, daß sich diese in dem normalen List-Directory, mit der Extension .KEY, befindet, also z.B. c:/rapfil/rap/aa.key
Rückgabe: KEYS() retourniert den Berechnungscode (47 von 47/111-222) für das aktuelle Intervall.
Siehe auch: ENDSUM, INDEX
Beispiel:

```
KEYS(0,"c:/mydir/enfil.min")  /* Liste wird über diese Datei gesteuert
#20=KEYS()                    /* Ein Berechnungscode wird gelesen
```

7.6. INDEX - Angabe des Index und Start/Stopwertes für Listen (RAP)

Zahl INDEX(Index *par1*, Text *par2*, Text *par3*)

par3 : Angabe des Wertes, den der Anwender für 'Stop bei' eingibt
Beschreibung: Mit dieser Funktion wird der Index und das Start/Stop-Intervall für eine Liste angegeben. In *par1* wird der entsprechende Index für die Hauptdatei bestimmt, d.h. der Ordnungsbegriff, mit dem die Datei gelesen werden soll. In *par2* und *par3* wird ein eventuelles Start/Stop Intervall angegeben.
Gibt man Start/Stop mit einem + voran ein, werden diese Werte zu vom Anwender zu Beginn der Liste eingegebenen Start/Stop Werten hinzuaddiert.
INDEX(-2) erlaubt die entsprechende Datei über den zweiten Index absteigend zu lesen.
Rückgabe: Retourniert den Index, mit dem die Hauptdatei gelesen werden soll.
Siehe auch: KEYS
Beispiel: INDEX(2,"D","D") */* Hauptdatei = KU (Demo-Währungskurse)*
Im Beispiel wird der Index 2 für die Liste bestimmt, so daß die Währungskurse sortiert nach Währungsbezeichnung, und nicht nach Währungscode, gelesen werden. Ferner werden nur die Sätze gelesen, deren Währungsbezeichnung mit 'D' beginnt.
INDEX(1,"+02","+02") */* Drucke 024711, wenn 4711 eingegeben wird*

7.7. LTOT - Bestimmen des niedrigsten Summenebene (RAP)

Zahl LTOT(Niveau *par1*)
Parameter: *par1* : Angabe des niedrigsten Summenebene für die Liste
Beschreibung:
Sofern *par1* >= 0 bestimmt dieser Wert das niedrigste Summenebene. Das niedrigste Summenebene kann auch beim Start die Liste eingegeben sein.
Rückgabe: Retourniert des niedrigsten Summenebene für die Liste.
Siehe auch: MTOT
Beispiel: LTOT(1) */* Nur Summen wird gedruckt - keine Zeilen.*

7.8. MTOT - Bestimmen des maximalen Summenebenen (RAP)

Zahl MTOT(Niveau *par1*)
Parameter: *par1* : Gibt des maximalen Summenebenen an
Beschreibung: Die Funktion bestimmt des maximalen Summenebenen für eine Liste. Sofern *par1* gleich 0 ist, werden keine Summen ausgegeben.
Rückgabe: Retourniert des maximale Summenebene für die Liste.
Siehe auch: LTOT
Beispiel: MTOT(1) */* Keine unsinnigen Endsummen*

7.9. MESS - Ausgabe einer Mitteilung auf dem Bildschirm

Zahl MESS(Text *par1*)
Parameter: *par1* : Mitteilung, die auf dem Bildschirm ausgegeben werden soll
Beschreibung: MESS blendet eine Mittelungsfenster auf dem Bildschirm ein. Abhängig vom letzten Zeichen im Text werden folgende Symbole und Knöpfe gezeigt:

Text	**Symbol**	**Knöpfe**	**Defaultknopf**
Text	Info	OK	OK
Text?	!	OK, CANCEL	CANCEL
Text??	?	YES, NO, CANCEL	YES
Text!	!	YES, NO	YES
Text!!	STOP	OK	OK
Text?!	STOP	OK, CAN	OK

Rückgabe: 0=OK oder JA, 1=NO, -1=CANCEL
Siehe auch: EXIT
Beispiel:

```
#1=MESS("Stoppen der Liste ?")
IF #1=0 EXIT(0)              /* Beende Liste
```

7.10. NOPAS - Kein Password/Kennwort für diese Liste (RAP)

NOPAS()
Parameter: Keine
Beschreibung: Die Funktion entfernt den Password/Kennwort-Schutz für diese Liste. Normalerweise ist eine Liste, in der Daten geändert werden, durch das Password CARE geschützt. Mit NOPAS() oder PAS() kann dieses Password entfernt oder geändert werden.
Siehe auch: PAS, UPDATE
Beispiel:

```
UPDATE(1)
NOPAS()       /* Kein Password für diese Liste
```

7.11. **PAS** - Bestimmen des Passwords/Kennwortes (RAP)

Zahl PAS(Text *par1*)
Parameter: *par1* : Angabe des gewünschten Passwords/Kennwortes
Beschreibung: Die Funktion bestimmt das Password/Kennwort, das bei Start diese Liste vom Anwender eingegeben werden muß.
Siehe auch: NOPAS
Beispiel: PAS("SWTOOLS") */* Password = SWTOOLS*

7.12. **PARAMS** - Funktion für zusätzlichen Listen Startparameter (RAP)

PARAMS(Felder *par1*)
Parameter: *Par1:* Felder, die im Dialog angezeigt werden sollen
Beschreibung: PARAMS("#1,7C,6O,le#3") ist eine Variante der Funktion DIALOG, wobei hier die entsprechenden Eingaben bei Start des Programms, und nicht bei Ablauf vorgenommen werden.
Benutzen Sie PARAMS, wird im Startbild ein zusätzlicher Schalter <Extra Parameter> eingeblendet.
Rückgabewert: Keine.
Siehe auch: DIALOG
Beispiel:

```
PARAMS("#1-3,11")              /* Dialog mit gegebenen Feldern
```

7.13. RETURN - Rückgabe aus Berechnungen

Zahl RETURN(Zahl *par1*)
Parameter: *par1* : Angabe des Codes, der retourniert werden soll
Beschreibung: Die Funktion wird bei Abschluß einer Berechnung (z.B. Lesen) eines bestimmten Satzes der Hauptdatei benutzt. Wird kein Parameter angegeben bzw. ist *par1* = 0, werden die definierten Zeilen für diesen Satz ausgegeben. Wird z.B. eine 1 retourniert, wird der Satz nicht behandelt, also auch nicht ausgeschrieben.
Rückgabe: Keine
Siehe auch: GOSUB
Beispiel: IF LE#6 < 1000 RETURN(1) */* Keine Ausgabe, wenn Saldo < 1000*

7.14. SORTKEY - Bereitstellung eines extra Sortierbegriffes (RAP)

Zahl SORTKEY(Datei *par1*)
Parameter: *par1* : 0, -1 oder Datei-ID
Beschreibung: Unter bestimmten Umständen kann es wünschenswert sein, eine Liste so sortieren zu können, daß ein Satz mehrere male auftritt. Hier kann es sich zum Beispiel um eine Artikelliste handeln, in der jeder Artikel sowohl unter dem normalen, wie auch unter dem alternativen Lieferanten auftreten soll. In diesem Falle wird nach einem Freifeld sortiert, das anschließend berechnet wird. Ein extra Sortierbegriff wird immer dann eingesetzt, wenn die Funktion SORTKEY aufgerufen wird.
Mit dieser Funktion können mehrere Dateien miteinander gemischt werden. Die Sortierdatei beinhaltet eine Zahl, die normalerweise auf einen Satz in der Hauptdatei der Liste verweist. Mit SORTKEY(le) wird jetzt ein Schlüssel eingefügt, der auf die Datei le verweist. Mit z.B. #20=SORTKEY(-1) kann abgelesen werden, welche der Dateien z.Zt. als Hauptdatei arbeitet. Hierüber können die Berechnungen gesteuert werden.
Rückgabe: Nummer der Hauptdatei, normalerweise 1.
Siehe auch: MERGE
Beispiel:

```
#11=#9                  /* Sortierfreifeld = Alt. Lieferant
IF #11<>0 SORTKEY(0)    /* Extra Sortierbegriff
#11=#6                  /* Normales Sortieren entspr. Lieferant
```

7.15. SORTWORK - Anwendung einer bestimmten Sortierdatei (RAP)

SORTWORK(Zahl *par1*)
Parameter: *par1* : Nummer der Sortierdatei
Beschreibung: Zum Sortieren richtet RAPGEN Hilfsdateien mit den Namen c:/tmp/SIN00000.000 und c:/tmp/SUD00000.000 ein, wobei c:/tmp/ die normale Adresse einer solchen Datei ist. Diese Dateien werden nach einem Sortierdurchlauf nicht gelöscht, da man bei Start der nächsten Liste durch Angabe von

START BEI: SORT oder SORTD

Sortierzeit vermeiden kann. Es wird dann die gleiche Sortierung wie beim letzten Durchlauf benutzt. Wird diese Vorgehensweise häufig benutzt, kann das Löschen dadurch verhindert werden, daß man eine Nummer mitgibt, z.B. SORTWORK(47), also c:/tmp/SIN00000.047 og c:/tmp/SUD00000.047
Rückgabe: Keine
Siehe auch:
Beispiel: SORTWORK(47)

7.16. **WANN** - WANN soll eine Rechenoperation ausgeführt werden (RAP)

WANN(Zahl *par1*, Zahl *par2*)

Beschreibung: Dieses Kommando wird benutzt, wenn mehrere Möglichkeiten der Berechnung bestehen, z.B. mehrmaliger Durchlauf eines Programmen bzw. Programmteiles.

8. Drucker steurung

In diesem Abschnitt wird die Drucker Funktionen beschrieben.

8.1. COPIES- Anzahl der Kopien (RAP)

COPIES(Zahl *par1*, Drucker *par2*)

par2 : Drucker nummer
Beschreibung: COPIES(1) gibt Ihnen eine extra Kopie des Ausdruckes. Es können maximal 30 Kopien angefordert werden. Beachten Sie bitte, daß für alle Kopien Platz in der Spool-Datei vorhanden sein muß.
COPIES(1,7) gibt Ihnen eine extra Kopie des Ausdruckes auf dem Drucker Nr. 7. Beachten Sie bitte, daß ein ungewollter Seitenwechsel auftreten kann, wenn der angegebene Kopiedrucker ein kleinere Seitenformat verwendet.
Rückgabe: Keine
Siehe auch: PRINTER
Beispiel: COPIES(1) */* Druck Zweimal*

8.2. **PAGE** - Angabe der Layout-Seite (RAP)

Zahl PAGE(Zahl *par1*)
Parameter: *par1* : Angabe der gewünschten Layout-Seite
Beschreibung: Normalerweise wird für die Ausgabe einer Liste die Seite 0 der Liste für das Layout benutzt. Soll die Liste jedoch in z.B. einer anderen Sprache gedruckt werden, kann eine andere Layout-Seite in dieser Funktion bestimmt werden. Es können bis zu 9 Layout-Seiten definiert sein. Diese Seiten können über das 'Datei'-Menü (Seiten-Layout) angesprochen und geändert werden.
Rückgabe: Layout-Seite, die für diese Liste benutzt wird.
Siehe auch: PRINT
Beispiel:

```
PAGE(le#5)     /* Wechsle Layout-Seite auf Sprache des Lieferanten
PRINT(1-10)    /* Drucke Text
```

8.3. PRINT - Druckvorschriften für eine Liste (RAP)

Zahl PRINT(Text *par1*)
Parameter: *par1* : Bestimmt die Zeile(n), die gedruckt werden soll(en).
Beschreibung: Mit Hilfe dieser Funktion können bestimme Zeilen in einer Liste gedruckt werden. Weiterhin kann hiermit angegeben werden, welche Zeilen z.B. bei Seitenwechsel oder Summenbildungen gedruckt werden sollen. Die Syntax für die Funktion ist im folgenden beschrieben:

Funktion	**Beschreibung**
PRINT(1-10)	Drucke Zeile 1 bis 10
PRINT(1,+2,2)	Drucke Zeile 1, gefolgt von 2 Leerzeilen, anschließend Zeile 2
PRINT(1,:60,2)	Drucke Zeile 1, positioniere auf Zeile 60 und drucke Zeile 2
PRINT(:1003,1,3)	Positioniert auf die 3. letzte Zeile (der Seite) und drucke Zeile 1 und 3
PRINT(1-10,:1,20)	Drucke Zeile 1 bis 10, gefolgt von Seitenwechsel, drucke Zeile 20
PRINT(*H)	Zeilen, die mit H= definiert sind, sollen gedruckt werden

Druckanweisungen für u.a. Seitenwechsel:

Funktion	**Beschreibung**
PRINT(H=1-4)	Bei Seitenwechsel Drucken der Zeilen 1-4
PRINT(L=8)	Transaktionszeile soll in Zeile 8 gedruckt werden
PRINT(T=10)	Summenzeile ist gleich Zeile 10
PRINT(D=9)	Überschrift für READH ist gleich Zeile 9
PRINT(B=:1002,17)	Als Fußtext auf jeder Seite soll Zeile 17 gedruckt werden
PRINT(N=3,:1,1-4)	Neue Seite ab 3 Zeilen von Seitenende, Überschrift auf Zeile 1-4
PRINT(A=10)	Zeile 10 soll vor einem Gesamtblock gedruckt werden
PRINT(C=11)	Zeile 11 soll nach einem Gesamtblock gedruckt werden

Beachten Sie, daß ein TextFeld in einem Druckerkommando als

```
#11-"1-4,15"
PRINT(#11)
```

benutzt werden kann.
PRINT(>2) Druckausgaben auf Drucker 2 geleiten, sehen PRINTER.
Rückgabe: Keine
Siehe auch: PAGE, PRINTER
Beispiel: PRINT(:60,1-10) /* *Positioniere auf Zeile 60 und drucke Zeile 1-10*

8.3.1. **PRINT** - Druckausgabe Kontrolle (RAP.)

PRINT(text *par1*)
Parameter: *par1* : Option=Wert
Beschreibung: Das PRINT Kommando wurde mit der Syntax PRINT(xx=Wert yy) erweitert, wobei xx,Wert und yy Werte haben können:

	Funktion	**Beschreibung**
xx=	ml	Linke Randbreite
	mr	Rechte Randbreite
	mt	Kopfhöhe
	mb	Fusshöhe
	eh	Leerzeile
	ce	Schließen des Listfensters bei Beendigung
	fh	Standard Zeichengrösse für alle Zeilen
	cd	Schließen des Druckerdokumentes und Neustart
yy=	cm	Centimeter
	in	Zoll
	pt	Punkte
	<keine>	Gerätepixels

8.3.2. **PRINT(?=** - Abfrage der Druckereinstellung (RAP.)

PRINT(?= text *par1*)

Beschreibung:
Auch das PRINT Kommando wurde um eine Abfragefunktion erweitert, um Informationen von der internen Druckerroutine (Printhandler) erhalten zu können.
Der Rückgabewert yy mach eine Angabe in pixels, ausgenommen wenn xx=5,8,9,15 oder 16 ist.

8.4. PRINT(LAB= - Etikett Funktion (RAP)

PRINT(LAB=Text *par1*, Text *par2*, Text *par3*, Text *par4*, Text *par5*, Text *par6*)

par6 : Kopien

Beschreibung: Breite und Höhe eines Etiketts kann in Zentimeter oder Inch angegeben werden. Es gilt folgende Syntax:

```
7cm gleich 7 Centimetres
2in gleich 2 Inches
```

Das unten angeführte Beispiel druckt Etiketten von links nach rechts, und zwar 21 Etiketten per Seite (3 Spalten, 7 Zeilen), mit der Höhe und Breite von je 7 cm. Es werden 2 Kopien gedruckt.

Rückgabewert: Keiner

Siehe auch: PRINT

Beispiel:

```
ZUERST
PRINT(LAB=1,3,7,7cm,7cm,2) /* Etikettdruck
NORMAL
```

8.5. PRINTER- - Druckerwahl (RAP.)

PRINTER(Drucker *par1*)
Parameter: *par1* : Druckernummer
Beschreibung: Diese Funktion wird im Zusammenhang mit dem Druckerdialog benutzt. Um einen default Drucker für eine Liste zu bestimmen, kann folgende Berechnungszeile eingefügt werden:
Rückgabe: Keine
Siehe auch: COPIES, PRINT
Beispiel: PRINTER(7) */* default Drucker für diese Liste ist der Drucker Nr. 7*

8.5.1. PRINTER - Ausgabe auf mehreren Drucker gleichzeitig (RAP)

PRINTER(Zahl *par1*, Drucker *par2*)
Parameter: *par1* : Druckernummer *par2* : DruckerID
Beschreibung: PRINTER(2,7) öffnet einen zweiten Drucker (definiert als Drucker Nr. 7). Es erfolgt keine Druckausgabe, bis

PRINT(>2)

in einer Berechnungszeile auftritt. Anschließend werden alle Druckausgaben auf den angegebenen Drucker geleitet. PRINT(>1) dirigiert den Druck wieder um auf den Standarddrucker.
Die Seitennummerierung ist individuell für jeden Drucker, und können auch im Papierformat differieren. Es können maximal bis zu 30 gleichzeitige Drucker angegeben werden.
Rückgabe: Keine
Siehe auch: COPIES, PRINT
Beispiel: PRINTER(2,7) */* zweiten Drucker Nr. 7 öffnen.*

8.6. PRTTOTAL - Manuelle Steuerung der Summenausgabe (RAP)

PRTTOTAL(Niveau *par1*)
Parameter: *par1* : Nummer der Summenebene
Beschreibung: Der Listgenerator gibt normalerweise eine Zwischensumme aus, wenn sich der entsprechende Sortierungsbegriff ändert. Mit PRTTOTAL kann diese Automatik ausgeschaltet werden.
Rückgabe: Keine
Siehe auch: ENDSUM
Beispiel:

```
IF #7=1 PRTTOTAL(1)             /* Zwischensumme, wenn Feld 7 = 1
SIDST
PRTTOTAL(2)                     /* Gesamtsumme zuletzt
```

8.7. SCRPRT - Nochmaliger Aufruf der Bildschirmausgabe (IQ)

SCRPRT(Dateiname *par1*)
Parameter: *Par1*: Name der Datei, die am Bildschirm aufgerufen werden soll
Beschreibung: SCRPRT("dateiname") aktiviert den Bildschirmdrucker für die Anzeige der Datei "dateiname".
Rückgabewert: Keiner
Siehe auch: PRINT
Beispiel:

```
SCRPRT("c:/w/ab.cde")      /* Anzeigen dieser Datei auf dem Bildschirmdrucker
```

9. Lesen einer Datei

In diesem Abschnitt wird die READ-Funktion zum Lesen von sekundären Dateien, sowie die START/STOP/REPEAT Funktionen zum Lesen mehrerer Sätze beschrieben.
Die Grundprinzipien für Dateiverbindungen sind im RAPGEN-Handbuch, Abschnitt 'Mehrere Dateien', beschrieben.

9.1. READ - Lesen eines Satzes in einer Datei

Zahl READ(Datei *par1*, Index *par2*), Verweis *par3*

par3 : Angabe eines evt. Verweises, der nicht als Standard besteht
Beschreibung: Die Funktion liest einen Satz in einer Datei.
READ(le) liest die Datei le mit Hilfe des Standardverweises, der in dem Data-Dictionary definiert ist.
READ(le),#9 liest die Datei le mit dem Schlüssel #9 für den Index 1, unabhängig davon, ob ein Standardverweis definiert ist.
READ(va.02),#6 liest die Datei va mit Feld 6 als Schlüssel für Index 2, unabhängig davon, ob ein Standardverweis definiert ist.
READ(le),"1",#9(3,4),#7 bildet als Schlüssel eine Kombination aus der Konstanten "1", Feld 9 Position 3-4 und Feld 7.
READ(le.00),#6 liest die Datei le mit der Satznummer (Index 0), die in Feld 6 angegeben ist.
Rückgabe: 0, wenn der Satz gelesen wurde.
Siehe auch: START, NEXT, REPEAT, END , PRIOR , READR, READX
Beispiel: READ(le) */* Lies Lieferantendatei*

9.2. READH - Lesen eines Satzes und eventuelles Drucken der Überschriftszeile

Zahl READH(Datei *par1*, Index *par2*), Verweis *par3*

par3 : Angabe eines evt. Verweises, der nicht als Standard besteht
Beschreibung: Die Funktion liest einen Satz in der angegebenen Datei (wie READ). Bei Wechsel des Sortierbegriffes in der Hauptdatei (z.B. Lieferantennummer), wird die Überschriftszeile gedruckt, die für diesen READH angegeben ist.
Rückgabe: Wert 0, wenn Satz ordnungsgemäß gelesen wurde.
Siehe auch: READ
Beispiel: READH(le) /* *Lese Lieferanten mit eventueller Überschriftszeile*

9.3. READR - Lesen eines Satzes mit einer bestimmten Satznummer

Zahl READR(Datei *par1*), Verweis *par2*

par2 : Angabe eines evt. Verweises, der nicht als Standard besteht
Beschreibung: Die Funktion liest einen durch eine Satznummer angegebenen Satz in einer Datei. READR kann nur in Datenbanksystemen verwendet werden, die eine Satznummer zum Lesen verwenden, und ist nur aus Kompatibilitätsgründen zu früheren Versionen hier aufgeführt.
READ(le.00),#6 entspricht READR(le),#6
Siehe auch: READ, READX

9.4. READX - Lesen eines Satzes mit einer relativen Statznummer

Zahl READX(Datei *par1*), Verweis *par2*

par2 : Angabe eines evt. Verweises, der nicht als Standard besteht
Beschreibung: Die Funktion liest einen durch eine relative Satznummer angegebenen Satz in einer Datei. READX kann nur in Datenbanksystemen verwendet werden, die eine Satznummer zum Lesen verwenden, und ist nur aus Kompatibilitätsgründen zu früheren Versionen hier aufgeführt.
READ(le.00),#6+N entspricht READX(le),#6
Siehe auch: READ, READR

9.5. START - Bestimmen des Indexes und des Intervalls für eine Datei

Zahl START(Datei *par1*, Index *par2*), Verweis *par3*

par3 : Angabe eines evt. Verweises, der nicht als Standard besteht
Beschreibung: Die Funktion bereitet die NEXT-Funktion vor, indem hier das Intervall der Schlüssel, die verwendet werden sollen, definiert wird. Es können hierbei die Standardverweise zwischen Dateien verwendet werden. Man kann auch individuelle Schlüssel, wie unter READ beschrieben, verwenden.
Bei START einer Liste wird der Schlüssel normalerweise nicht vollständig angegeben. Nachfolgendes Lesen mit NEXT findet jedoch alle Sätze, deren Schlüsselbeginn mit dem in START angegebenen Teilschlüssel übereinstimmt.
Rückgabe: 0, wenn Intervall bestimmt wurde.
Siehe auch: READ, NEXT, REPEAT, END , PRIOR
Beispiel:

```
#47=0                    /* Lösche Summenfeld
START(va)                /* Start Lesen Artikeldatei
NEXT(va)                 /* Lese nächsten Satz
#47=#47+va#3             /* Summiert aller Artikeln
REPEAT(va)               /* Weiter, bis keine weiteren Artikel
```

9.6. NEXT - Lese nächsten Satz im Intervall

Zahl NEXT(Datei *par1*)
Parameter: *par1* : Angabe der Datei-ID
Beschreibung: Die Funktion wird im Zusammenhang mit START/NEXT/REPEAT Schleifen benutzt. Mit den Funktionen START(0) und END(0) wird das Intervall für die Schleife definiert. NEXT liest einen Satz in der Datei. Wenn die Zeile REPEAT ausgeführt wird, wird ein weiterer Satz mit NEXT gelesen, solange weitere Sätze innerhalb des angegebenen Intervalls zu lesen sind.
Rückgabe: Retourniert 0, solange weitere Sätze gelesen werden sollen.
Siehe auch: READ, START, REPEAT, END , PRIOR
Beispiel:

```
PRINT                   /* Übernehmen der Drucksteuerung
PRINT(4,6,5)            /* Drucke Lieferantendaten
START(va)               /* Start Lesen der Artikelsätze
NEXT(va)                /* Lese nächsten Artikelsatz
PRINT(7)                /* Drucke Artikelzeile
REPEAT(va)              /* Weiter, bis keine weiteren Artikel
```

9.7. REPEAT - Wiederhole Lesen nach NEXT

Zahl REPEAT(Datei *par1*)
Parameter: *par1* : Angabe der Datei-ID
Beschreibung: Die Funktion wird im Zusammenhang mit START/NEXT/REPEAT Schleifen benutzt. Mit den Funktionen START(0) und END(0) wird das Intervall für die Schleife definiert. NEXT liest einen Satz in der Datei. Wenn die Zeile REPEAT ausgeführt wird, wird ein weiterer Satz mit NEXT gelesen, solange weitere Sätze innerhalb des angegebenen Intervalls zu lesen sind.
Rückgabe: Keine
Siehe auch: START, NEXT PRIOR
Beispiel:

```
#47=0                    /* Lösche Summenfeld
START(va)                /* Start Lesen Artikelsätze
NEXT(va)                 /* Lese nächsten Artikelsatz
#47=#47+va#3             /* Bilde Summe
REPEAT(va)               /* Weiter, bis keine weiteren Artikel
```

9.8. GETKEY - Lesen des aktuellen Schlüssels

Text GETKEY(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: GETKEY liest den Indexbegriff für den zuletzt gelesenen Datensatz in der Datei *par1* Die Funktion wird speziell im Zusammenhang mit Datenbanksystemen, in denen der Indexbegriff notwendigerweise nicht als Feld vorhanden ist, benutzt.
Rückgabe: Aktueller Schlüssel in Textformat.
Siehe auch:
Beispiel: #20 = GETKEY(va)

9.9. **END** - Bestimme Ende des Intervalles nach START

Zahl END(Datei *par1*), Verweis *par2*

par2 : Angabe des größten Schlüssels innerhalb des Intervalles
Beschreibung: Mit der Funktion START werden erster und letzter Schlüssel in einem Intervall gleich gesetzt. Normalerweise gibt man END nur dann an, wenn man ein von START unterschiedlich gesetztes Intervallen brauchen will.
Rückgabe: Keine
Siehe auch: READ, START, REPEAT, NEXT , PRIOR
Beispiel:

```
UPDATE(1)                  /* Löschen der Arbeitsdatei xx
START(xx),"0000"           /* Start Lesen der Datei bei Dateianfang
END(xx),"9999"             /* Lesen bis Dateiende
NEXT(xx)                   /* Lese nächsten Datensatz
DELETE(xx)                 /* Lösche alle
REPEAT(xx)                 /* Weiter, bis Dateiinhalt gelöscht ist
```

9.10. **PRIOR** - Lesen des vorangegangenen Datensatzes im Intervall

Zahl PRIOR(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: Wie NEXT, doch wird nicht der nächste, sondern der vorhergehende Satz gelesen. NB: Nicht alle Datenbanksysteme unterstützen Lesen in umgekehrter Reihenfolge.
Rückgabe: Wert 0, solange weitere Sätze im Intervall.
Siehe auch: READ, START, REPEAT, NEXT , END
Beispiel:

```
PRINT                       /* Übernahme der Drucksteuerung
#47=0                       /* Lösche Zähler
START(va)                   /* Start Lesen Artikelsatz
PRIOR(va)                   /* Lesen vorhergehenden Artikelsatz
#47=#47+1                   /* Zähler + 1
IF #47=1 PRINT(4,6,5)       /* Drucke Liefer.kopf erstes mal
PRINT(7)                    /* Drucke Artikelzeilen in umgekehrter Folge
REPEAT(va)                  /* Weiter, bei keine weiteren Artikelsätze
IF #47>0 PRINT(8)           /* Drucke Fuß, wenn Anzahl Artikel ungleich 0
```

9.11. SPEED- Optimierung der READ Strategie

SPEED()
Parameter: Keine
Beschreibung: Mit der SPEED() Funktion kann die READ Strategie optimiert werden. Ein Satz wird nicht erneut gelesen, wenn der gleiche Schlüssel auftritt. In diesem Falle wird der entsprechende Satz aus dem Speicher übernommen. Seien Sie bitte vorsichtig mit dieser Funktion, wenn es sich um Listen handelt, in die während der Verarbeitung geschrieben wird.
Rückgabe: Keine
Siehe auch: READ
Beispiel: SPEED() */* Read optimierung*

10. Schreiben in Dateien

In diesem Anschnitt werden die verschiedenen Möglichkeiten, in eine Datei zu schreiben, beschrieben. Um diese Funktionen anwenden zu können, muß bei Installation eine Zulassung zum Schreiben gegeben sein, das Datenbanksystem muß das Schreiben in Dateien unterstützen, und schließlich muß auch der Anwender (über die Anwenderkennung) eine Schreibzulassung besitzen.
Ein Programm, das in eine oder mehrere Dateien schreibt, sollte immer ordentlich getestet sein. Es ist die

Verantwortung des Anwenders, daß Änderungen in einer Datei korrekt vorgenommen werden.

10.1. UPDATE - Zulassung zum Schreiben in Dateien

Zahl UPDATE(Zahl *par1*, Felder *par2*)

par2 : Zugelassende Felder.
Beschreibung: UPDATE(1) muß in eine Liste, in der Datensätze geschrieben werden sollen, vor dem ersten Schreibbefehl eingesetzt werden.
Die Update-Funktion wurde mit Feldparametern erweitert.

```
UPDATE(1,"va#6")    /* erlaubt den Update nur für das Feld va#6
UPDATE(1,"le#3-4") /* Parameter muß für jede Datei einzeln gesetzt werden
UPDATE(0)           /* kann jetzt im DATAMASTER genutzt werden
```

Rückgabe: Keine
Siehe auch: DELETE, INSERT, REWRITE, WRITE, NOPAS
Beispiel:

```
UPDATE(1)        /* Schreibzulassung
NOPAS()          /* Keine Password
#6=#6+10         /* Berechnung neuer Werte
REWRITE(le)      /* Schreib Datensatz zurück (Lieferentendatei)
```

10.2. REWRITE - Schreibe einen gelesenen Satz in die Datei zurück

Zahl REWRITE(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: Die Funktion schreibt einen gelesenen Satz in die eingegebene Datei zurück. Der Satz muß vorher aus dieser Datei gelesen worden sein. Wenn ein Datenbanksystem es zuläßt, kann der Wert in dem Indexfeld geändert werden. Damit die Funktion aktiv werden kann, muß vorher ein UPDATE(1) ausgeführt worden sein.
Rückgabe: Wert 0, wenn der Satz ordnungsgemäß zurückgeschrieben wurde.
Siehe auch: DELETE, INSERT, WRITE, NOPAS, UPDATE
Beispiel:

```
UPDATE(1)        /* Schreibzulassung
NOPAS()          /* Kein Password
NACH             /* Nur nach Selektieren
#6=#6+10         /* Berechnung eines neuen Wertes
REWRITE(le)      /* Zurückschreiben in die Lieferantendatei
```

10.3. INSERT - Einfügen eines neuen Satzes in eine Datei

Zahl INSERT(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: Die Funktion fügt einen neuen Satz in eine Datei ein. ALLE Felder im einzufügenden Satz müssen einen zulässigen Wert beinhalten, bevor der Funktion INSERT ausgeführt wird. Damit die Funktion aktiv werden kann, muß vorher ein UPDATE(1) ausgeführt worden sein.
Rückgabe: Wert 0, wenn der Satz ordnungsgemäß eingefügt wurde.
Siehe auch: DELETE, REWRITE, WRITE, NOPAS, UPDATE , CLEAR, LET
Beispiel:

```
UPDATE(1)              /* Schreibzulassung
NOPAS()                /* Kein password
CLEAR(le)              /* Löschen aller Felder (Lieferant)
LET("le#1,3=#7,17")    /* Einsetzen der Werte
INSERT(le)             /* Einfügen neuen Satz in Lief.datei
```

10.4. DELETE - Löschen eines Satzes in einer Datei

Zahl DELETE(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: Die Funktion löscht einen Satz in der angegebenen Datei. Der Satz muß vor dem Löschen gelesen worden sein. Damit die Funktion aktiv werden kann, muß vorher ein UPDATE(1) ausgeführt worden sein.
Rückgabe: Wert 0, wenn der Satz ordnungsgemäß gelöscht wurde.
Siehe auch: INSERT, REWRITE, WRITE, NOPAS, UPDATE
Beispiel:

```
UPDATE(1)               /* Schreibzulassung
NOPAS()                 /* Kein Password
NACH                    /* Nur nach Selektieren
DELETE(va)              /* Satz löschen (Artikeldatei)
```

10.5. WRITE - Zurückschreiben oder Einfügen eines Satzes in eine Datei

Zahl WRITE(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: Die Funktion schreiben einen Satz zurück bzw. fügt einen neuen Satz in eine Datei ein. Sofern mit dem letzten READ ein Satz gelesen wurde, entspricht diese Funktion der Funktion REWRITE(), sonst einem INSERT(). Damit die Funktion aktiv werden kann, muß vorher ein UPDATE(1) ausgeführt worden sein.
Rückgabe: Wert 0, wenn Funktion ordnungsgemäß ausgeführt wurde.
Siehe auch: INSERT, REWRITE, DELETE, NOPAS, UPDATE
Beispiel:

```
UPDATE(1)               /* Schreibzulassung
READ(le),#6             /* Lesen eine Satz
IF #OK THEN BEGIN       /* Wenn Satz nicht vorhanden
le#1=#6                 /* Berechne Lieferantennummer
le#2="Neu Name"         /* und Namen
END
le#6=le#6+#3            /* Setze Felder in Lieferantensatz
WRITE(le)               /* Einfügen oder rückschreiben des Satzes
```

11. Export / Import externer Dateien

Dieser Abschnitt beschreibt die Funktionen für Import/Export Von Daten von/in fremde Dateien.

11.1. EXPORT - Export von Daten in eine Textdatei

Zahl EXPORT(Felder *par1*, Dateiname *par2*, Text *par3*, Text *par4**6, Text *par5*, Text *par6**6)

Beschreibung:
EXPORT exportiert Daten in eine Textdatei. Mit Hilfe dieser Funktion können u.a. Daten in andere Systeme überführt werden, z.B. in Textverarbeitungs- oder Tabellenkalkulationssysteme.
Die Felder, die in *par1* angegeben werden, müssen als Text angegeben werden, also z.B. "#1-99".
Gibt man für den physischen Dateiname in *par2* keine Adresse (path) an, wird dieser in TMP abgelegt. Fehlt die Extension-Angabe, wird .OUT eingesetzt. Wird kein Dateiname angegeben, verwendet das System den Namen der entsprechenden Liste, also z.B. c:\tmp\DM1007.OUT für die Liste Nr. 7.
Mit *par3* und *par5* kann die Statzlänge und die Zeilenaufteilung der Datei gesteuert werden.
par4 wird normalerweise nur im Zusammenhang mit festen Satzlängen für die Überführung und Mainframe-Systeme benutzt.
par6 besteht aus 6 Zeichen, die das Format einer durch Komma separierten Datei bestimmen. Beachten Sie, daß das Zeichen " als \" (zwei Zeichen) angegeben werden muß.
Standardmäßig werden alle alphanumerischen Felder als "xxxx" geschrieben, wobei auftretende " in ' konvertiert werden. Die numerischen Felder werden als 99.99 geschrieben, wobei das Zeichen . einen Dezimalpunkt angibt. Alle Felder werden durch Komma getrennt.
Die Asciitextdatei kann nun mit EXPORT("CLOSE") geschlossen werden. Dies kann nützlich sein wenn man während der Laufzeit sich diese Datei ansehen möchte.(siehe CHAIN)
Rückgabe: Keine
Siehe auch: IMPORT
Beispiel:

```
NACH                            /* NACH Selektion
EXPORT("#1-6","le.csv")         /* Alle Felder werden exportiert (CSV)
EXPORT("#1-6","le.csv","","","","--,\"'.") Wie oben
```

Ausgehend von obigem Beispiel beinhaltet die Datei le.csv folgende Daten:

```
"100","HUMBER LTD.","HUMBER STREET 223","4711 COPENHAGEN S",,123.25
"102","AX & AX LTD.","SEA PARK ROAD 43","2100 COPENHAGEN",,25000
"105","WEBB'S SUPPLIERS LTD.","EAST STREET 373","4711 COPENHAGEN F",,500
```

Beispiel:

```
EXPORT("#1-6","le.ssv","000001","","","--;. ,")  /* Alle Felder als SSV
```

Ausgehend von obigen Beispiel beinhaltet die Datei le.ssv folgende Daten:

```
SW-Tools
100;HUMBER LTD.;HUMBER STREET 223;4711 COPENHAGEN S;;123,25
```

Beispiel:

```
EXPORT("#1-2,5-6","a","-80","1")  /* Fester Länge, ohne crlf
```

11.2. IMPORT - Import von Daten aus einer Textdatei (RAP)

IMPORT(Felder *par1*, Dateiname *par2*, Text *par3*, Text *par4**6, Text *par5*, Text *par6**6)

Beschreibung: Die Funktionen importiert Daten aus einer Textdatei.
Die Felder, die in *par1* angegeben werden, müssen in " " gesetzt werden. In Zusammenhang mit dieser Funktion können einfache Berechnungen mitgegeben werden, z.B. IMPORT("#1-5,+6","a")

Funktion	Beschreibung
+	Addiere zu Feld
-	Subtrahiere von Feld
&	Überspringe Feld
=	Setze Feld gleich mit
:xx	Springe auf Satzposition xx

Der Dateiname, der in *par2* angegeben wird, kann eine path enthalten, z.B. "c:/export/le.csv".
Rückgabe: Keine
Siehe auch: EXPORT, IMPOCONT, IMPONEXT, IMPOTHIS
Beispiel:

```
                              /* Hauptdatei der Liste ist le (Lieferanten)
   UPDATE(1)                  /* Schreibzulassung
   NOPAS()                    /* Kein Password
   IMPORT("#1-6","le.csv")    /* Lese einen Satz aus der Textdatei (CSV)
   READ(LE),#1                /* Prüfe, ob Lieferant vorhanden
   LE#6=LE#6+#6               /* Addiere letzten Betrag zu Saldo
   IF #OK LET("LE#1-6=#1-6")  /* Wenn nicht vorhanden, setze alle Felder
   WRITE(LE)                  /* Schreibe LE Lieferant
```

Beispiel:

```
   IMPORT("#1-6","le.ssv","","","","--;- -")  /* Import von (SSV) Textdatei
```

11.2.1. **IMPOCONT** - Fortsetzen mit IMPORT (RAP)

IMPOCONT(Felder *par1*)
Parameter: *par1* : Der Felder, die durch die einzulesene Textdatei gebildet werden sollen
Beschreibung: IMPOCONT setzt mit dem Import weiterer Felder des Satzes, der zuletzt mit IMPORT gelesen wurde, und ab der Position, die als letzte mit IMPORT behandelt wurde, fort. Diese Funktion kann typisch dort Verwendung finden, wo zu Satzbeginn unterschiedliche Feldinhalte definiert werden.
Siehe auch: IMPORT, IMPONEXT, IMPOTHIS

11.2.2. **IMPONEXT** - Import des nächsten Satzes (RAP)

IMPONEXT(Felder *par1*)
Parameter: *par1* : Der Felder, die durch die einzulesene Textdatei gebildet werden sollen
Beschreibung: IMPONEXT liest den nächsten Satz und importiert dessen Felder. Kann z.B. benutzt werden, wenn in einem Satz angegeben wird, daß der/die folgenden Satz/Sätze zusammengehören.
Siehe auch: IMPORT, IMPOCONT, IMPOTHIS

11.2.3. **IMPOTHIS** - Import dieses Satzes noch mal (RAP)

IMPOTHIS(Felder *par1*)
Parameter: *par1* : Der Felder, die durch die einzulesene Textdatei gebildet werden sollen
Beschreibung: IMPOTHIS importiert den gleichen Satz (wie bei zuletzt ausgeführten IMPORT) nochmals. Kann z.B. verwendet werden, wenn zu Beginn eines importierten Satzes unterschiedliche Feldinhalte definiert werden.
Siehe auch: IMPORT, IMPOCONT, IMPONEXT

11.3. FTP - File Transfer Prozessor

Zahl FTP(Zahl *par1*, Text *par2*)

Par2: FTP command
Beschreibung: Die FTP Funktion wurde eingeführt, um dem geübten Anwender die Möglichkeit zu geben. Dateien in eine Liste, basierend auf einer SSV Datei mit Dateinamen, zu übertragen. Für die vollständige Beschreibung aller Befehle verweisen wir auf das entsprechende FTP Handbuch. Beachten Sie, daß Freifelder für einen Befehl verwendet werden können, und daß die 32-Bit Version lange Dateinamen unterstützt.
Das Beispiel zeigt einen Transfer einer Datei vom Quattro System mit dem Befehl QUATTRO. Der Transfer erfolgt mit Headerblock. XQUAT bewirkt einen Transfer, in dem die zusätzlichen FTP Informationen von der übertragenen Datei entfernt werden.
Rückgabewert: Für OPEN: FTP Handle, Andere : FTP Fehlerkode, 0=OK
Beispiel:

```
#10=FTP(0,"open 200.0.0.9")          /* Verbindung zum Server
#11=FTP(#10,"user cms mypas")        /* Anwender cms, Password mypas
#11=FTP(#10,"binary")                /* Wechsel in binäre Übertragung
#11=FTP(#10,"quattro")               /* Wechsel in Quattro Backup Modus
#11=FTP(#10,"get /X.BASIC/0/AFIL c:/mydir/myfil") /* Lesen der Datei
if #11<>0 FTP(#10,"error")           /* Anzeige Fehlermitteilung
#11=FTP(#10,"xquat c:/mydir/myfil")  /* Konvertierung von Quattro
#11=FTP(#10,"quit")                  /* Ende
```

12. Mehrere Firmen und Verflechtung von Dateien

Die in diesem Abschnitt beschriebenen Funktionen können verwendet werden, wenn das System mehrerer Firmen beinhaltet, wobei die Dateien für diese Firmen in jeweils eigenen Datenbanken bzw. Tabellen liegen, und wenn man mehrere Dateien mit gleicher Definition miteinander verflechten will.

12.1. ACCESS- Prüfung, ob die Datei vorhanden ist

Zahl ACCESS(Dateiname *par1*)
Parameter: *par1* : Dateiname
Beschreibung: Prüft, ob die gegebene Datei vorhanden ist (Rückgabewert 0).
Rückgabe: 0 ob die Datei vorhanden ist.
Siehe auch: OPEN
Beispiel: IF ACCESS("datei.ssv")=0 MESS("Ok ? ")

12.2. **COMNO** - Firmenidentifikation (FirmaID)

Text COMNO(Datei *par1*)
Parameter: *par1* : Leer oder Datei-ID
Beschreibung: Die Funktion retourniert die FirmaID für die aktuelle Datei (für die Hauptdatei, wenn nicht anderes angegeben wurde).
Rückgabe: FirmaID
Siehe auch: OPCOM
Beispiel: #1 = COMNO() */* Lese aktuelle FirmaID, z.B. "001".*

12.3. **ENDSUM** - Extra Endsumme bei Listen mit mehreren Hauptdateien

ENDSUM()
Parameter: Keine
Beschreibung: In einer Liste, die wiederum aus mehreren selbständigen Listen besteht (definiert durch die Funktionen MERGE oder OPCOM), kann eine Endsumme, die die einzelnen Listen zusammenfaßt, mit ENDSUM() als Berechnungszeile eingefügt werden.
Die Systemfelder #CO und #CN werden als *** bei ENDSUM ausgegeben.
Rückgabe: Keine
Siehe auch: KEYS, MERGE , OPCOM
Beispiel: ENDSUM() */* Drucke extra Endsumme am Schluß*

12.4. FILENAME - Aktueller physischer Name einer Datei

Text FILENAME(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: FILENAME retourniert den physischen Name der Datei, die mit Datei eröffnet wurde.
Rückgabe: Physischer Dateiname
Siehe auch: OPEN
Beispiel: #1 = FILENAME(va) /* *ergibt "c:/rapfil/ssv/isa/va.ssv"*

12.5. OPEN - Eröffnen einer Datei mit einem gegebenen Namen

Zahl OPEN(Datei *par1*, Dateiname *par2*, Schnittstelle *par3*)

par3 : 0 oder Datenbank Interface Nummer
Beschreibung: Mit dieser Funktion kann eine bestimmte Datei anstelle einer bereits offenen Datei geöffnet werden. Die bereits offene Datei wird geschlossen.
Kann die angegebene Datei nicht gefunden werden, bzw. tritt ein anderer Fehler in diesem Zusammenhang auf, wird eine entsprechende Fehlermitteilung ausgegeben.
Sofern *par3* angegeben wird, wird die Datei entsprechend dem Datenbanktyp, der in BASIS.SSV für diese Datei definiert ist, geöffnet.
Rückgabe: 0=OK, <>0=Fehler
Siehe auch: ACCESS, FILENAME, MERGE, OPCOM
Beispiel:

```
ZUERST
OPEN(va,"c:/swtools/demo/va.ssv")    /* Benutze diese Datei
OPEN(va,#50)                         /* Angabe der Datei bei Start
```

12.5.1. OPEN - Zwischenzeitliches Schließen einer Datei

OPEN(Datei *par1*, Konstant *par2*)

par2 : "-"
Beschreibung: Dateien können zwischenzeitlich geschlossen werden, um anderen (CHAIN) Programmen den Dateizugriff zu erlauben. Beachten Sie bitte, daß die Hauptdatei nicht in dieser Weise geschlossen werden darf.
Rückgabe: 0=Ok, <>0=Fehler.
Siehe auch: ACCESS, FILENAME, MERGE, OPCOM
Beispiel:

```
OPEN("ku","-")       /* Zwischenzeitliches Schließen der Datei ku
CHAIN("command.com /c edit ku.ssv") /* editieren der Datei ku.ssv
OPEN("ku","+")       /* Wiedereröffnung der Datei
```

12.6. MERGE - Verflechten mehrerer Dateien in einer Liste (RAP)

Zahl MERGE(Datei *par1*, Dateiname *par2*, Schnittstelle *par3*)

par3 : 0 oder Datenbank Interface Nummer

Beschreibung: Mit Hilfe dieser Funktion können mehrere Hauptdateien in einer Liste verflochten werden. Man kann entweder eine durch seine Datei-ID ansprechen (wenn diese als selbständige Datei definiert ist), oder wie in OPEN den physischen Dateinamen angeben. Die benutzten Dateien müssen die gleiche Struktur besitzen.
Kann eine angegebene Datei nicht gefunden werden, bzw. tritt ein anderer Fehler in diesem Zusammenhang auf, wird eine entsprechende Fehlermitteilung ausgegeben.
Sofern *par3* angegeben wird, wird die Datei entsprechend dem Datenbanktyp, der in BASIS.SSV für diese Datei definiert ist, geöffnet.
Eine Liste, in der MERGE benutzt wird, sollte normalerweise sortiert sein. Wird MERGE ohne Sortierung verwendet, erhält man zuerst eine Liste der eigentlichen Hauptdatei und anschließend je eine der einzuflechtenden Dateien. Die ENDSUM Funktion kann zur Bildung von Gesamtendsummen benutzt werden.
Wird MERGE ohne Angabe von Parametern benutzt, wird eine sog. Mergenummer retourniert, nämlich 1 für die Hauptdatei, 2 für die erste Verflechtung, 3 für die zweite Verflechtung usw.

Ohne Parameter: MERGENUMMER, ab 1 aufsteigend

Siehe auch: ENDSUM, OPCOM, OPEN

Beispiel:

```
MERGE(0,"c:/swtools/demo/va.ssv")   /* Verflachte mit dieser Datei
MERGE(le)                           /* und mit le Datei
#12=MERGE()                         /* Lese Mergenummer 1,2 oder 3
```

12.7. OPCOM - Öffnen von Dateien in mehreren Firmen

Zahl OPCOM()

par3 : 0 oder Datenbank Interface Nummer
Beschreibung:
Die OPCOM Funktion erlaubt den Zugriff auf mehrere Firmen in einer Liste.
Man kann eine Liste mit so aufbauen, daß diese für jede Firma einzeln durchlaufen wird. Hierfür muß die Berechnungszeile OPCOM ("111,777-888") oder OPCOM(#50) eingefügt werden, wobei #50 als Eingabefeld bei Start der Liste benutzt wird. Die Liste kann z.B. mit Gesamtendsummen oder mit Sortierungen, bei denen z.B. Informationen verschiedener Firmen gesammelt werden, erweitert werden.
Die Systemfelder #CO und #CN können für die Ausgabe der FirmenID/Firmenname in einer Überschrift verwendet werden.
In einer Artikelliste, definiert auf der Datei va, können die Artikelinformationen einer anderen Firma mit Hilfe von OPCOM(va,"555") und READH gelesen werden. Hiermit erreicht man, daß in z.B. va#8 der Bestand der aktuellen Firma, in va#7 der Bestand der Firma 555 berechnet wird.
Eine Liste, die auf einer Statistikdatei basiert, kann durch Angabe der Firmennummer in einem Statistiksatz die Dateien dieser Firma mit OPCOM(0,#47) öffnen.
Sofern *par3* angegeben wird, wird die Datei entsprechend dem Datenbanktyp, der in BASIS.SSV für diese Datei definiert ist, geöffnet.
Wird OPCOM ohne Parameter aufgerufen, wird die aktuelle Firmennummer retourniert.

Ohne Parameter: FirmenID
Siehe auch: COMNO, ENDSUM, MERGE, OPEN
Beispiel:

```
OPCOM("001,777-888")            /* Liste für diese Firmen
OPCOM("*")                      /* Liste für alle Firmen
OPCOM(va,"123")                 /* Artikeldatei Firma 123
OPCOM(0,"777")                  /* Firma 777 für alle außer Hauptdatei
OPCOM(-1,"888")                 /* Firma 888 für alle Dateien

OPCOM(#50)                      /* Eingabe Firmen bei Start
```

13. DATAMASTER Funktionen

Diese Funktionen stehen ausschließlich für DATAMASTER zur Verfügung, und können nicht in Listprogrammen verwendet werden. Einige Funktionen können jedoch auch in IQ benutzt werden. In diesem Falle ist dies gesondert bei der jeweiligen Funktion angegeben.

13.1. **DISABLE**- Keine Eingabe in diesem Programm (IQ)

DISABLE(programmnr *par1*)
Parameter: *par1* : Programnummer.
Beschreibung: Alle Eingaben in dieser Programmnummer werden übergangen.
Rückgabe: Keine.
Siehe auch: ENABLE , FOCUS
Beispiel: DISABLE(20)

13.2. **DISP** - Anzeigen geänderter Felder (IQ)

DISP(Felder *par1*)
Parameter: *par1* : "" oder Felder
Beschreibung: Diese Funktion dient dazu, Felder, die am Bildschirm angezeigt werden, und in einer Berechnung verändert werden, mit ihrem geänderten Wert am Bildschirm zu zeigen. Wenn DISP ausgelassen wird, ist es unsicher, wann die geänderten Werte auf dem Bildschirm erscheinen.
Der DISP() Befehl aktualisiert alle auf dem Bildschirm befindlichen Felder oder nur bestimmte Felder (DISP("#1,4"))
Rückgabe: Keine
Siehe auch:
Beispiel: DISP()

13.3. DOFUNCTION - Ausführen externer Funktionen (IQ)

DOFUNCTION(Funktion *par1*, text *par2*, Programnr *par3*)

par3 : Eventuel Programnr
Beschreibung: DOFUNCTION sendet die Nachricht <Funktionnr> zum laufenden IQ-Programm oder zum geöffneten <Programm>. Ein Schlüssel kann an das aufgerufene Programm weitergegeben werden. Die Liste der verfügbaren Funktionsnummern findet man in der Berechnungslistbox bei der Anwahl 'Wahl von Funktionen'
Rückgabe: Keine.
Siehe auch: CHAIN, PLSNEXT, TRANSMIT
Beispiel:

```
DOFUNCTION(505,#1,20) /* Programm 20 liest die Satz mit Schlüssel #1
DOFUNCTION(550)       /* vergrößert den aktuellen Bildschirm
```

13.4. ENABLE- Erlaubt die Eingabe für ein Programm (IQ)

ENABLE(programmnr *par1*)
Parameter: *par1* : Programmnummer
Beschreibung: Erlaubt alle Eingaben für die gegebene Programmnummer.
Rückgabe: Keine.
Siehe auch: DISABLE , FOCUS
Beispiel: ENABLE(20) */* Erlaubt Eingaben für programm 20*

13.5. **FOCUS** - Aktiviert das Programm (IQ)

FOCUS(Programmnr *par1*)
Parameter: *par1* : Programmnummer.
Beschreibung: Aktiviert die Eingabe und setzt den Fokus auf das angegebene Programm.
Rückgabe: Keine.
Siehe auch: DISABLE, ENABLE
Beispiel: FOCUS(20) */* Programm 20 wird Aktiviert*

13.6. FUNC - Bestimmen des Änderungsmodus (IQ)

Zahl FUNC(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: Anhand der Eingaben durch den Anwender bestimmt DATAMASTER, ob das Schreiben/ Zurückschreiben eines bestimmten Satzes notwendig ist, und in gegebenem Falle, wie dies auszuführen ist. FUNC wird in Schreibroutinen benutzt, um den Änderungsmodus abzulesen.
Rückgabe:

Wert	Funktion
0	Keine Schreiben notwendig
1	Ein bestehender Satz soll geändert werden
2	Ein neuer Satz soll eingefügt werden
3	Ein bestehender Satz soll gelöscht werden

Siehe auch: SETUPD, ON
Beispiel:

```
ON FUNC(cu) GOSUB MAINWRT,MAININS,MAINDEL
IF FUNC(va)!=3 LET #27=#27+va#4
```

13.7. GETINFO - Lesen zusätzlicher Programminformationen (IQ/DM)

Zahl GETINFO(Zahl *par1*, Text *par2*)

par2 : Feldverweis

Beschreibung: Diese Funktion erlaubt Ihnen, besondere Informationen von einem IQ/DM Programm zu lesen. Typ 0 und 1 retournieren die eindeutige ID eines Fensters. Dieser Wert kann von einer anderen Funktion benutzt werden, um dieses Fenster ansprechen zu können. Ein Beispiel hierzu finden Sie im Handbuch OLE.

Bei Typ 2 bis 5 wird ein Feldverweis in *par2* verlangt. Beispiel: Um die Startspalte für das Artikelfeld Nr. 7 zu erhalten, muß *par2* "va#7" enthalten. Die Koordinaten für ein Feld werden geben die aktuelle Größe des Feldes an, wie es in IQ/DM definiert wurde. Wünschen Sie die aktuellen Koordinaten eines Feldes entsprechend dem aktuellen Skalierungsfaktor, z.B. Zoom ein/aus, müssen Sie den Typ 6 bis 9 benutzen.

Typ 2-9: Feldkoordinaten. Der Wert kann in einem Feld mit dem Format 9,T2 stehen.

Beispiel:

```
GETINFO(0)            /* Lesen der ID für IQ Programmfenster
GETINFO(2,"va#7");    /* Lesen der x Startkoordinate für va Feld 7
```

13.8. **HELP** - Anzeigebox mit dem Hilfetext zum Feld (IQ)

HELP(Feld *par1*)
Parameter: *par1* : Feldreference
Beschreibung: HELP(#31) zeigt eine Nachrichtenbox mit dem Hilfetext für das angegebene Feld.
Siehe auch: MESS
Beispiel: HELP("#31")

13.9. ISACTIVE - Prüft ob ein Programm aktiv ist (IQ)

Zahl ISACTIVE(Programmnr *par1*)
Parameter: *par1* : Programmnummer
Beschreibung: Prüft ob ein Programm aktiv ist.
Rückgabe: Gibt den Wert 1 zurück wenn <Programm> aktiv ist, sonst 0.
Siehe auch: CHAIN, EXIT, WAIT
Beispiel: IF ISACTIVE(20)=0 CHAIN(20) */* Start Programm 20 ob nicht aktiv*

13.10. KEYON - Schaltet das Schlüsseleingabefeld EIN/AUS (IQ)

KEYON(Zahl *par1*)

Beschreibung: KEYON(0) löscht das Schlüsseleingabefeld, (1) reaktiviert es.
Rückgabe: Keine.
Siehe auch:
Beispiel: KEYON(0) */* Schlüsseleingabefeld löschen*

13.11. **LINE** - Lesen bzw. Setzen der aktuellen Zeilennummer (IQ/DM)

Zahl LINE(Zahl *par1*)
Parameter: *par1* : Art der Information
Beschreibung: Die Funktion liest bzw. setzt die Zeilennummer in IQ/DM. Die Zeilennummer ist der Zähler für Zeilen in einem Programm, und ist definiert als **va#1-6l** oder **le#1-6/va#1-6**.
Wenn *par1* gleich 0 ist, retourniert die Funktion die augenblicklich aktive Zeile.
Wenn *par1* gleich -1 ist, wird die Anzahl der für dieses Programm definierten Zeilen zurückgegeben. Wurde das Programm definiert als **va#1-61,t5**, ist der Rückgabewert **5**.
Ist *par1* größer als 0, wird die aktive Zeile gleich *par1* gesetzt.
Rückgabewert: Zeilennummer oder 0 (wenn Setzen der Zeilennummer)
Beispiel:

```
#20=LINE()  /* Lesen der augenblicklich aktiven Zeilennummer
```

13.12. LOOP - Aufruf einer Routine für alle Sätze im Zeilenpuffer (IQ)

LOOP(Label *par1*)
Parameter: *par1* : Label (Name der Routine, die aufgerufen werden soll)
Beschreibung: Jeder Satz, der in einem Listprogramm gelesen wird, wird mit den Ergebnissen eventueller Berechnungen (nicht jedoch globale Arbeitsfelder) im Zeilenpuffer abgelegt.
Zum Schreiben in einem solchen Programm wird LOOP zum Schreiben aller Zeilen verwendet. Ebenso wird LOOP zum nochmaligen Berechnen von SUM benutzt.
Rückgabe: Keine
Siehe auch: GOSUB, ON
Beispiel:

```
LOOP(MAIN)       /* Schreiben in Hauptdatei
LOOP(TRANS)      /* Schreiben der Transaktionszeilen
LOOP(SUMIT)      /* Neuberechnung von SUM
LOOP(TRANSDEF)   /* Änderung des Schlüsselwertes für alle Transaktionen
```

13.13. **MENUCH** - Setzen des Menükennzeichens (IQ)

MENUCH(Menunr *par1*)
Parameter: *par1* : Menünummern
Beschreibung: Wechselt zum entsprechenden Menü und ermöglicht ein Ändern der Menüparameter.
Rückgabe: Keine.
Siehe auch: MENUUPD, MENUS
Beispiel: MENUCH("31-32") */* Wechselt die IQ Aktualisierungsmenü*

13.14. MENUS - Änderung von Menüs (IQ)

MENUS(Menunr *par1*)
Parameter: *par1* : -xxx=Deaktivieren, +xxx=Aktivieren der Menünummern xxx.

Menünummer	Funktion
1/11	Einfügen neuen Satz in Hauptdatei/Transaktionsdatei
2/12	Änderung eines Satzes in Hauptdatei/Transaktionsdatei
3/13	Löschen eines Satzes in Hauptdatei/Transaktionsdatei
4/14	Superindex für Hauptdatei/Transaktionsdatei
5/15	Selektionen in Hauptdatei/Transaktionsdatei
6/16	Superindex Feldwahl für Hauptdatei/Transaktionsdatei
20	Suchen in Liste, muß mit Eingabe übereinstimmen
21/22/23/24/25	Transaktionen, Nächste/Vorhergehende/Erste/Letzte/Richtung
26	Anzeige des Schlüssels beim Suchen
27	'Case sensitive' Suchen
31/32	Ändern andere/geändert von anderen
41/42/43/44	Hauptdatei, Nächste/Vorhergehende/Erste/Letzte
51/52/53	Berechnungen/Änderungen Layout/Speichern
54/55	Parameter Menüs
61/62/63/64	Neues Programm, Lösche Programm, Drucke Programm, Starte Programm
100-149	Index fest / Index Menüs
999	Aktivieren aller Punkte

Beschreibung: MENUS kann in sowohl IQ und DATAMASTER Programmen benutzt werden, um nicht benötigte Menüpunkte auszuschalten.
MENUS kann auch bei Aufruf von IQ aus Windows mit den -m+xxx oder -m-xxx Parameter aktiviert werden. Speziell ist dies notwendig, wenn in Berechnungen für ein Programm, in dem diese Funktion nicht aktiviert ist, geändert werden soll, z.B. C:\SWTOOLS\IQWIN -m999
Rückgabe: Keine
Siehe auch: MENUCH, MENUUPD
Beispiel: MENUS("-51-55") */* Deaktivieren von Programmänderungen*

13.15. MENUUPD - Hinzufügen/Überwachung des Menüs (IQ)

MENUUPD(Menunr *par1*, Zahl *par2*, Zahl *par3*)

par3 : Text
Beschreibung: Manuelles Hinzufügen zum bzw. Kontrolle des Menüs.
MENUUPD(1,2000"Mein &eignes Menü") erweitert das Menü 1 mit der Funktion 2000.
Bei Wahl dieses neuen Menüpunktes wird die Funktion mit dem Label FU2000: ausgeführt.
Rückgabe: Keine.
Siehe auch: MENUCH, MENUS
Beispiel: MENUUPD(1,2000"Mein &eignes Menü") */* erweitert das Menü 1 mit der Funktion 2000.*

13.16. NEXTFLD - Sprung auf Eingabefeld (IQ)

NEXTFLD(Feld *par1*)
Parameter: *par1* : Feld, in das die nächste Eingabe erfolgen soll
Beschreibung: Möchte man die feste Feldreihenfolge für Eingaben in Abhängigkeit von Berechnungen ändern, kann hierfür die Funktion NEXTFLD benutzt werden.
Benutzung der Funktion NEXTFLD erweitert mit Programnummer und Zeilenummer.
Rückgabe: Keine
Siehe auch: NEXTFLDSEQ, SEQ
Beispiel:

```
IF #4<#3 NEXTFLD(#3)
NEXTFLD("#10")        /* setzt das nächste Eingabefeld auf Feld 10
NEXTFLD("#10.2")      /* springt zu Feld 10 in Zeile 2
NEXTFLD("5.#10")      /* springt zu Programm 5 mit Feld 10
```

13.17. NEXTFLDSEQ - Sprung zum Eingabefeld in Reihenfolge (IQ)

NEXTFLDSEQ(Zahl *par1*, Zahl *par2*)

par2 : Feldnummer
Beschreibung: Springt zu einem bestimmten Feld in einer vorgegebenen Eingabereihenfolge.
Rückgabe: Keine.
Siehe auch: SEQ , NEXTFLD
Beispiel: NEXTFLDSEQ(2,1) */* Springt zum ersten Feld in der Eingabereihenfolge Nr.2*

13.18. OBJECTADDSTRING - Hinzufügen einer Text zu einem Objekt (IQ)

OBJECTADDSTRING(Feld *Par1*, Text *Par2*, Text *Par3*)

Par3: Text, der als Index verwendet werden soll
Beschreibung: Die Funktion fügt einen Text in ein Objekt. Das Ergebnis dieser Funktion ist von dem Objekttyp abhängig. Bitte beachten Sie die folgenden Regeln:

Objekt	Bedeutung
BUTTON	Text, der in einem Button gezeigt werden soll
COMBOBOX	Hinzufügen eines neuen Elements in die Liste
EDITBOX	Einfügen eines Textes in das Edit-Fenster, wenn der Schalter für multiple gesetzt wird der Text angefügt.
LISTBOX	Hinzufügen eines neuen Elements in die Liste

Der Parameter *par3* wird nur benutzt, wenn es sich um das Objekt COMBOBOX oder LISTBOX handelt. Der Parameter muß den normalen Feldwert enthalten.
Rückgabe: Keine
Siehe auch: OBJECTCLEAR
Beispiel: OBJECTADDSTRING("va#7",gr#2,gr#1) */* Anzeige Name und benutzt.Nr. als Index*

13.19. OBJECTCLEAR- Lösche Inhalt eines Objektes (IQ)

OBJECTCLEAR(Feld *par1*)
Parameter: *Par1* : Feld im Formular, z.B. va#7
Beschreibung: Die Funktion löscht den Inhalt eines Objektes.
Rückgabe: Keiner
Siehe auch: OBJECTADDSTRING
Beispiel:

```
   OBJECTCLEAR("va#7")   /* Löschen aller früheren Werte
   START(gr),""          /* Lesen aller Werte der Artikelgruppendatei
   NEXT(gr)
      OBJECTADDSTRING("va#7",gr#2,gr#1) /* Anzeige Name und benutzt.Nr. als
Index
   REPEAT(gr)
```

13.20. OBJECTGETSTRING- Lesen des Index eines in einem Objekt gewählten Feldes (IQ/DM)

Text OBJECTGETSTRING(Feld *par1*)
Parameter: *par1* : Feld im Formular, z.B. va#7
Beschreibung: Die Funktion liest den normalen Wert eines Feldes in einer Combobox/Listfenster. Der Rückgabewert entspricht *par3* in der Funktion OBJECTADDSTRING.
Rückgabewert: Normaler (Index) Wert des gewählten Feldes.
Siehe auch: OBJECTADDSTRING
Beispiel:

```
#20=OBJECTGETSTRING("va#6")  /* Lesen der aktuellen Lieferantennummer
```

13.21. PLSNEXT - Vorbereiten und Lesen der Hauptdatei (IQ)

PLSNEXT(Zahl *par1*, text *par2*, Zahl *par3*,)

Beschreibung: Vorbereiten und Lesen der Hauptdatei entspr. dem gegebenen Modus. Diese Funktion wird u.a. von den Menüs und den Funktionen Seite vor/zurück benutzt. Ist die Inputkennung (flag) gesetzt, wird ein Satzschlüssel benutzt. Im anderen Falle erfolgt das Lesen als nächster/vorhergehender Satz.
Rückgabe: Keine.
Siehe auch: DOFUNCTION, TRANSMIT
Beispiel: PLSNEXT(0,#1,1) */* liest den nächsten Satz mit Feld #1 als Schlüssel*

13.22. SEQ - Änderung der Eingabefolge (IQ)

SEQ(Zahl *par1*, Felder *par2*)

par2 : Feldangabe für die neue Eingabefolge
Beschreibung: Mit Hilfe dieser Funktion wird die Eingabefolge geändert.
Rückgabe: Keine
Siehe auch: NEXTFLD, NEXTFLDSEQ
Beispiel:

```
SEQ(2,"va#2-3,5")          /* Setze normale Änderungssequenz
IF #7=1 SEQ(2,"va#4,3")    /* Speziell für diese Artikelgruppe
```

13.23. SETUPD - Markierung eine Datei/Zeile für schreiben (IQ)

SETUPD(Datei *par1*)
Parameter: *par1* : Datei-ID
Beschreibung: Beim Änderung die Werte von bestimmte 'kritische' (Schlüssel) Felder in der Hauptdatei kann daß notwendig sein, alle Postierungen für schreiben zu markieren. Nicht nur Zeilen daß beim Input veränderte sein soll geschrieben werden.
Rückgabe: Keine
Siehe auch: LOOP
Beispiel: SETUPD(va)

13.24. SHOW-
Anschalten/Abschalten/Anzeige/Ausblenden eines Feldes (IQ/DM)

Zahl SHOW(Feld *par1*, Zahl *par2*)

3 = Ausblenden
Beschreibung: Diese Funktion erlaubt Ihnen, ein Feld an- oder abzuschalten, und das Feld anzuzeigen oder auszublenden.
Rückgabewert: Keiner
Beispiel:

```
SHOW("va#7",1)        /* Abschalten des Feldes va#7
```

13.25. SUPER - Vorbereiten des Suchens mit Superindex (IQ)

SUPER(DateiId *par1*) , Text *par2*

par2 : Schlüßel
Beschreibung: Die SUPER Funktion initialisiert den NEXT read für die Verwendung des Superindex.
Rückgabe: Keine.
Siehe auch: NEXT, START
Beispiel:

```
SUPER(va),#21      /* NEXT benutzt Superindex Suchen für Text in #21
NEXT(va)           /* Muß unmittelbar folgen, um den Satz zu lesen
SUPER(va)          /* Superindex wird abgeschaltet
SUPER(va),"#1-3"   /* Superindex Felder zeigen auf die Felder 1-3
```

13.26. TRANSMIT- Update von andere IQ Programme (IQ)

TRANSMIT(Zahl *par1*, text *par2*, text *par3*)

par3 : Eventuel Verknüpfung
Beschreibung: Übertragung des aktuellen Satzes in ein oder mehrere Programme, wobei die automatische Verknüpfung bzw. eine angegebene Verknüpfung benutzt wird.

```
Progid= ""         Senden an alle anderen
        "20"       Senden nur an Progr. 20, wenn dieses aktiv ist
        "le"       Senden an alle, die die Datei le als Hauptdatei benutzen
Connection=""      Benutzen automatischer Verknüpfungen
        "1,2P"     Benutzen von Feld 1 und 2 (gepackt) als Verknüpfung
        "va.01.6" Benutzen von va als Übertragungsdatei zu anderen Programmen
```

Rückgabe: Keine.
Siehe auch: PLSNEXT, DOFUNCTION
Beispiel: TRANSMIT(0,"","") */* Update aller Programme, nützen die automatische Verknüpfungen*

13.27. TRANSSEL- IQ Transaktionszeile-selektionen (IQ)

TRANSEL(text *par1*, Zahl *par2*)

Beschreibung: Durchsuche eine eventuelle Eingabe und definiere Transaktionszeile-selektionen, wenn die Eingabe Formeln wie #15>0 enthält. Wird benutzt, wenn Pfeile im Schlüsselfeld auftreten.
Rückgabe: Keine.
Siehe auch:
Beispiel: TRANSSEL("#15>20",1) */* Selektion definieren*

14. SYSTEM Funktionen

Diese Funktionen sind für spezielle Programme entwickelt, in denen man direkten Zugriff auf Dateien und Dateiadressen benötigt.

14.1. DEBUG- Aktivieren des DEBUG Fensters (IQ)

DEBUG(Zahl *par1*)

Beschreibung: DEBUG(1) öffnet das DEBUG Fenster, in dem alle berechneten Ausdrücke und die zugehörigen Programmnummern/Labels bei Ausführung aufgelistet werden.
Das Fenster wird beim Verlassen von IQ geschlossen, oder mit DEBUG(0).
Rückgabe: Keine
Siehe auch: WIF, WIFS
Beispiel: DEBUG(0) */* Debug aktivieren*

14.2. EXEC- Ausführen eines Alphastrings als Befehlszeile

EXEC(text *par1*Programnr *par2*)

par2 : **(IQ/DM)**Eventuel Programm nummer
Beschreibung:

```
#20="#2=17"
EXEC(#20)
```

führt den in Feld 20 gespeicherten Alphastring aus.
Wenn Freifelder in der Funktion EXEC benutzt werden, muß man die WW#nn Feldnummer aus der Programmdokumentation entnehmen.
Generell werden die Strings in den EXEC Funktionen nicht überprüft. Bei der Programmausführung kann es daher aufgrund der C- Syntax zu Problemen führen. Es wird ausdrücklich darauf hingewiesen, daß in den EXEC Anweisungen keine Funktionsaufrufe sein dürfen.
Ein Punkt sollte besonders beim RAPGEN beachtet werden: #15=2 setzt Feld 15 gleich 2 auch wenn es als IF #15=2 LET #16=3 genutzt wird. Man muß aufgrund der C- Syntax das Gleichheitszeichen verdoppeln: IF (#15==2) LET #16=3
IQ: EXEC(#20,15) schaltet zum aktiven Programm 15 und führt die angegebene Berechnung aus.
Rückgabe: Keine
Siehe auch:
Beispiel: EXEC(#20) */* Berechnung beim Start von eine Liste eingegeben*

14.3. GETFLD- Setzen SY Struktur Zeigern (IQ)

GETFLD(text *par1*)
Parameter: *par1* : Feltspecifikation
Beschreibung: Diese Funktion setzt die Systemvariablen (SY#..) zu zeigen auf die genannten Feld. Die Felddefinitionen kann dann abgefragt und geändert werden.
Rückgabe: Keine

14.4. INSTALL- Externe Funktionen integrieren

INSTALL(text *par1*, text *par2*, text *par3*, text *par4*)

par4 : Eventuel eigene Funktionsname
Beschreibung: Programmierer können selbstgeschriebene Funktionen (dll) in den IQ integrieren.

ACHTUNG: Falscher Gebrauch dieser Funktion kann zum Systemabsturz führen.

Rückgabe: Keine
Siehe auch:
Beispiel:

```
INSTALL("a.dll","b","3,[ss]")
   aktiviert #20=B(#21) von a.dll, #20 und #21 sind Variablen vom Typ short

INSTALL("some.dll","aname","3,[sC1]", "FUNNY")
   aktiviert #30=FUNNY(#31,#32)als Funktion aname der Datei some.dll gibt die
   Werte im #30 vom Typ short, die Parameter im #31 als Zeiger vom Typ char,
   #32 vom Typ long.
```

14.5. SYSPAR - Lesen der Systemparameter

Text SYSPAR(Zahl *par1*)

Beschreibung: SYSPAR liest entsprechend obigen Nummern die Systemparameter.
Rückgabe: Systemparameter
Siehe auch: SYSPARSET
Beispiel: #1 = SYSPAR(4) */* Lese TMP Adresse*

14.6. SYSPARSET - Setzen eines neuen Wertes für Systemparameter

SYSPARSET(Zahl *par1*, Text *par2*)

par2 : Neuer Wert für diesen Systemparameter
Beschreibung: SYSPARSET setzt einen neuen Wert für den angegebenen Systemparameter.
Rückgabe: Keine
Siehe auch: SYSPAR
Beispiel: SYSPARSET(4,"c:/meintmp/") */* Setze neue TMP Adresse*

14.7. USERINFO - Lesen der Anwenderinformation

Text USERINFO(Zahl *par1*)

17=Anwenderdefiniert
Beschreibung: Die Funktion liest die gewünschte Information.
Die Zahl in *par1* verweist auf die Feldnummer in der Systemdatei US, in der die Felder 11-17 individuell für jede Installation definiert werden können.
Rückgabewert: Zeichenkette mit der gewünschten Information
Beispiel:

```
#11=USERINFO(6)  /* Lese 1. Bemerkung
```

14.8. **WIF** - Testdruck (IQ)

WIF(text *par1*)
Parameter: *par1* : Text
Beschreibung: WIF ermöglicht einen Aktionsbezogenen Testdruck in einem textdatei ohne das Bildschirm Layout zu verändern (c:/wif)
Rückgabe: Keine
Siehe auch: WIFS, DEBUG
Beispiel: WIF("Ich bin hier") /* *Testdruch von Text*

14.9. **WIF**- Testdruck (RAP)

WIF(text *par1* , text *par2*)

. **Beschreibung:** WIF ermöglicht Testdruck in c:/wif.
Rückgabe: Keine
Siehe auch: WIFS, DEBUG
Beispiel: WIF("Feld ist %s.",#2) /* *Testdruch*

14.10. **WIFS**- Testdruck von Feldinhalt (IQ)

WIFS(Felder *par1*)
Parameter: *par1* : Felder zu drucken
Beschreibung: WIFS druckt den Feldinhalt der in dem Parameter angegebenen Felder in die Datei c:/wif.
Rückgabe: Keine
Siehe auch: WIF, DEBUG
Beispiel: WIFS("va#1-3,le#2") /* *Testdruck von Feldwerte*

Index