SWT
SOFTWARE TOOLS

français manuel d'utilisation

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

Calculs and sous-fonctions

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

Table des matières

1. Introduction

La syntaxe pour les calculs dans RAPGEN est basée sur une sorte de langage de programmation de BASIC. Ce langage vous permet de tester les valeurs de champ, les instructions arithmétiques, le traitement de texte, etc.

	Mots réservés	synonyme	Description
Instruction conditionnelle			
	IF		expression SI ...
	LET		expression SI expression LAISSE ...
	ELSE		expression SI ... expression SINON
Instruction de bloc			
	BEGIN		commencer un bloc
	END		terminer un bloc
Control loop flow			
	BREAK		interrompre loop
	CONTINUE		continuer loop
Opérateurs logiques			
	NOT		ne pas égal à 0
	AND		et
	OR		ou
Opérateurs arithmétiques			
	+		addition
	-		soustraction
	*		multiplier
	/		diviser
	%		pourcentage
Opérateurs relationels			
	=		équal à
	>		supérieur à
	<		less than
	>=		supérieur ou égal à
	<=		inférieur ou égal à
	<>		ne pas égal à
Commentaires			
	REM		Ligne de commentaire complete
	/*		Commentaire d'après les instructions de calcul
Saut et sous-routines			
	GOTO		va jusqu'à l'étiquette:
	GOSUB		exécuter l'étiquette de sous-routine:
	RETURN		retourner à partir de la sous-routine
	ON..GO..		Saut conditionnel/l'appel de routine

De plus, ce langage vous permet d'utiliser un ensemble de termes différentes de calcul pour effectuer des instructions. Dans la section suivante, nous allons voir quelques exemples :

1.1. Exemples

1.1.1. **SI. SINON** - Instructions conditionnels

Si le solde de fournisseur (LE#6) est supérieur à 1000 soustrairez le solde de 100 sinon additionnez 47 au solde.

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

1.1.2. **BEGIN.END** Instructions de bloc

Si le solde de fournisseur (LE#6) est supérieur à 1000, commencez le bloque de façon que 100 soit soustrais du solde et que la ligne 7 soit imprimée.

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

Ainsi, toutes les lignes de calcul qui se trouvent entre BEGIN et END sont seulement exécutées dans le cas où la condition serait remplie.

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

Les boucles suivantes lisent tous les fournisseurs qui se trouvent dans l'intervalle de 111 à 999. Si le solde est inférieur à 1000, le fournisseur ne sera pas traité.

```
START(LE),"111"
END(LE),"999"
NEXT(LE)
IF LE#6 < 1000 CONTINUE /* skip suppliers with a balance < 1000
REM *** process suppliers ***
REPEAT(LE)
```

Les boucles suivantes lisent tous les fournisseurs qui se trouvent dans l'intervalle de 111 à 999. Lorsque le solde est supérieur à 10000, la boucle s'arrête.

```
START(LE),"111"
END(LE),"999"
NEXT(LE)
IF LE#6 > 10000 BREAK  /* break loop if balance > 10000
REPEAT(LE)
IF LE#6 > 10000 ....   /* supplier found with balance > 10000
```

1.1.4. **NOT, AND, OR** - Opérateurs logiques

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

Si la dernière date d'achat est égale à 0, mettrez la date d'achat à ce jour. Cette instruction corresponde à :

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

Si le prix d'achat n'est pas égal à 0 ET la date d'achat n'est pas égale à 0, la ligne 5 sera imprimée sur le rapport.

```
IF VA#4<>0 AND VA#5<>0 PRINT(5) /* if cost and date set print line
```

Si le prix d'achat n'est égal à 0 OU la dernière date d'achat n'est pas égale à 0, La ligne 5 sera imprimée sur le rapport.

```
IF VA#4<> OR VA#5<>0 PRINT(5) /* Si if cost or date not equals 0 print line 5
```

1.1.5. REM, /* - Commentaires

```
REM *** this report is developed by SW-Tools ApS ***
REM *** date. 07.09.1997
IF LE#6 > 1000 LET LE#6 = LE#6 - 100 /* Ajust the balance
```

1.1.6. **GOTO** Sauter vers l'étiquette

A l'aide de l'instruction GO TO, vous pouvez vous déplacer dans les calculs typiquement en fontion de la valeur d'un champ. Une étiquette définie avec le 'NOM:' décide l'endroit vers lequel vous pouvez sauter. L'exemple montré au-dessous de la ligne 7 est imprimé trois fois.

```
        #30 = 0                 /* Zero counter
AGAIN:                          /* Label for later jump
        PRINT(7)                /* Do something
        #30 = #30 + 1           /* Count
        IF #30 < 3 GOTO AGAIN   /* Do it three times
```

1.1.6.1. **ON**...GOTO/GOSUB - Saut conditionnel et Appel de sous-routine

A l'aide de ON, vous pouvez, indépendamment de la valeur, sauter vers une étiquette donnée. Vous pouvez utiliser ON en liaison avec GOTO et GOSUB.

```
        #30 = 0
        ON #7 GOTO ONE,TWO,ONE
        #30 = #30 + 1        /* #30 becomes 3 if field 7 not equals 1,2 or 3
   TWO: #30 = #30 + 1        /* #30 becomes 2 if field 7 equals 2
   ONE: #30 = #30 + 1        /* #30 becomes 1 if field 7 equals 1 or 3
```

1.1.7. **GOSUB** Appel aux sous-routines

Si vous voulez exécuter les calculs à plusieurs reprises, vous pouvez écrire ces lignes en tant que sous-routine commencée par 'l'étiquette:' et appelée à l'aide GOSUB

```
        #30 = 0                     /* Zero counter
        GOSUB DOIT                  /* Call the routine
        GOSUB DOIT                  /* Call the routine again
        RETURN                      /* End the normal calculations
  DOIT: #30 = #30 + 1               /* Routine DOIT, count up field 30
        PRINT(7)                    /* And print line 7
        RETURN                      /* Return from the subroutine
```

1.1.7.1. **RETURN** Retour à partir d'une sous-routine

A l'aide de la commande RETURN, vous pouvez terminer une sous-routine de telle sorte que les calculs continuent à partir de l'endroit d'où vous l'appelez. Voir aussi la section consacrée à la fonction RETURN et à la manière selon laquelle vous pouvez terminez les calculs (à l'aide d'une valeur retour, par exemple).

1.2. Champs

1.2.1. #xx or kk#xx - Champs de fichier

Vous pouvez faire référence à un champ d'un fichier en mettant :

```
     #xx = fieldnumber xx from the mainfile
   kk#xx = fieldnumber xx from the file kk
```

Il est à noter que kk, KK, Kk et kK font référence à des enregistrements de types différents du fichier KK. Normalement, vous devez utiliser les lettres en minuscules (kk, par exemple).

1.2.1.1. #xx(from,to) - Partie des champs

Les parties des champs sont indiquées en tant que kk#xx(De, à), Vous pouvez utiliser cette syntaxe tant pour les champs numériques que pour les champs alphanumériques.

```
#30 = #2(3,4)      /* Field 30 becomes character 3 thru 4 of field 2
```

Avec les champs de texte alphanumériques, vous pouvez aussi indiquer qu'une partie d'un champ doit correspondre à un texte :

```
#2="Sorenco and Son Ltd."
#2(9,15)="xx"      /* Field 2 becomes "Sorenco xx Ltd."
```

1.2.1.2. #xx(no) - Champs de table

Vous pouvez faire référence aux champs de table en mettant kk#xx(no). C'est à dire les numéros à partir de 0 et les suivants
Vous pouvez définir un champ dans la base de données en tant qu'un champ de table si le format contient, par exemple, 20(003). C'est à dire 3 éléments supplémentaires ou un ensemble des champs en continu ayant le même format que vous voulez traiter en tant qu'une table dans les calculs. Il est à noter que les champs libres peuvent également être définis en tant que champs de table à l'aide de la spécification de format.
Par exemple, les démons. fichiers de fournisseur dans lesquels le nom de bloc #2, #3 et #4 peut aussi être utilisé en tant qu'une table #2(0), #2(1) et #2(2)

```
#30 = #2(#31)              /* Freefield 31 specifies name line 0,1 or 2
PRINT(7)                   /* Which is printed
#2(#31)="xx"               /* And set to "xx"
```

Il est à noter que des valeurs mystérieuses peuvent apparaître lorsque vous dépassez le nombre maximal de tables possible en mettant #2(4).

1.2.1.3. Conversion entre les champs numériques et les champs de texte

Vous pouvez indiquer un champ numérique = un champ de texte, par exemple, #30 = #2 et puis . vous pouvez continuer d'effectuer vos calculs. Vous pouvez utiliser les fonctions NUMBER et NUMS pour effectuer des conversions plus avancées. Voir la section consacrée à ces conversions.
En mettant un champ de texte = un champ numérique, par exemple, #2 = #30, une chaîne de texte avec une longueur de variable est créée en fonction du numéro dans le champ 30, par exemple "123". Normalement, on frappe l'instruction #2 = #30 USING "#####" pour indiquer la présentation du champ de texte. Veuillez vous reporter au chapitre consacré à la fonction USING.

1.2.2. **SY#xx** - Champs de système

Les champs de système sont des champs spéciaux définis dans le pseudo fichier SY et toujours présents. Une partie des champs de système est présenté dans les paragraphes suivants. Pour obtenir une vue d'ensemble complète, veuillez vous reporter à votre définition actuelle du fichier SY.
Vous pouvez faire référence à un champ de système en utilisant le numéro SY#1 ou l'abréviation sous forme de deux caractères #DD placée au début du nom de champ. Certains champs de système sont associés à un fichier. Afin de pouvoir faire référence à ces champs, frappez l'abréviation kk# par exemple kk#RECNO

1.2.2.1. **#DD, #PD** - Date d'aujourd'hui et valeur de ce jour

Elles sont indiquées lors du démarrage d'un rapport.(99.99.99).

1.2.2.2. **#PP** - Numéro de page

Il est inscrit automatiquement lors de l'appel de page, (9999).

1.2.2.3. **#SN** - Nom de système

Il est utilisé si RAPGEN possède des systèmes multiples, par exemple, de différentes sociétés ou un ensemble complet de fichier. Veuillez noter les champs #SU pour le nom de sous-système et #CN pour le nom d'entreprise

1.2.2.4. **#OK** - Résultat de la lecture d'un fichier

Après la lecture d'un fichier, vous pouvez frapper #OK. Ce champ aura la valeur 0 si la lecture d'un enregistrement est effectuée. Tout autre indique une erreur.

1.2.2.5. **#UN** Nom de l'utilisateur

Pour obtenir le nom de l'utilisateur de votre PC, utilisez #UN entré dans le module de licence.

1.2.2.6. **#LIN** Numéros de ligne et #LOF Nombre de lignes sur la page

#LIN contient la ligne d'impression actuelle, et #LOF le nombre des lignes sur la page.

1.2.2.7. **#LEVEL** - Niveau de total actuel

A l'aide de #LEVEL, vous pouvez contrôler les calculs et l'impression en fonction du niveau de sous-total. Pour plus d'informations concernant cette fonction, veillez vous reporter au manuel d'utilisation de RAGPEN.

1.2.2.8. kk#RECNO - Dernier numéro d'enregistrement utilisé à partir du fichier kk

Si le système de base de données est associé aux numéros d'enregistrements, le dernier enregistrement lu pour le fichier kk peut être recherché dans kk#RECNO. Veuillez noter les champs kk#NUMBER pour le numéro d'enregistrement correspondant et le NOM de Fichier kk#

1.2.3. **WW#xx** - Champs libres (Champs de travail)

Lors de la création d'un programme, ce programme reçoit 40 champs de travail. D'abord, ces champs doivent être définis avec un nom et un format. Vous pouvez les redéfinir ultérieurement en double-cliquant sur le champ.
Les numéros de champ sont placés après les numéros de champ du fichier maître. Ces champs seront stockés en tant que WW#1,WW#2,.... Par ceci, l'extention du fichier maître plus tard entraîne une numération automatique des champs libres dans tous les programmes.
Le nombre des champs libre peut être ajusté dans IQ/DATAMASTER à l'aide de la fonction de paramètres de programme et dans RAPGEN en utilisant un numéro supérieur. Par ceci, le nombre des champs libres sont automatiquement élargi.

1.2.3.1. **#Dntexte** - Entrée de données

Dans RAPGEN un nom de champ libre qui commence par #Dn entraîne l'entrée de données à partir de 1 à 7 lors du démarrage du rapport.

1.2.3.2. **#Ptexte** - Champs d'images

Un nom de champ libre qui commence par #P et qui est défini en tant que champ de texte fait une référence à une image.

2. Fonctions arithmétiques

Dans la section suivante, nous allons voir les fonctions pour le traitement de taux (l'arrondissement et l'évelation à une puissance, par exemple).

2.1. **ABS** - Valeur absolue d'un nombre

Le nombre ABS(nombre *par1*)
Paramètres: *par1* : Nombre à convertir en une valeur absolue
Description: Cette fonction retournera la valeur absolue du paramètre *par1*. Par exemple, la valeur positive sans marque.
Valeur retour: La valeur positive.
Voir aussi: SGN
Exemple: #1 = ABS(-123.45) /* *Champ #1 contient la valeur 123.45*

2.2. **FNH** - Arrondissement du nombre sans des décimales

nombre FNH(number *par1*)
Paramètres: *par1* : indique un nombre(y compris des décimales)
Description: Cette fonction vous permet d'arrondir un nombre contenant des décimales à un nombre sans des décimales..
Valeur retour: Ce nombre sans décimales.
Voir aussi FNR, RUN
Exemple: #1 = FNH(1234.56) /* Champ #1 contient la valeur 1235

2.3. **FNR** - Arrondir un nombre à deux décimales

nombre FNR(nombre *par1*)
Paramètres: *par1* : indique un nombre (avec des décimales)
Description: Cette fonction vous permet d'arrondir un nombre avec plus de deux décimales à un nombre avec deux décimales. RAPGEN arrondit toujours le résultat d'un calcul à ce nombre de décimales que le champ calculé contient. Cependant, vous pouvez utiliser les fonctions FNH/FNR si vous souhaitez utiliser un différent type d'arrondissement.
A l'aide de la fonction ARRONDI vous pouvez définir :

Valeur return: Le nombre arrondi.
Voir aussi: FNH, RUN, RUND
Exemple: #1 = FNR(123.456) */* Champ #1 contient la valeur 123.46*

2.4. **FRA** - Calculer la valeur décimale d'un nombre

nombre FRA(nombre *par1*)
Paramètres: *par1* : le nombre (sans décimales)
Description: Cette fonction exécute le tri de la valeur décimale d'un nombre et puis elle la retournera.
Valeur return: La valeur décimal notée 0.<valeur décimale>.
Voir aussi: FNH, FNR, RUN
Exemple: #1=FRA(123.456) /* *fait 0.456* , #1=FRA(-12.345) /* *fait -0.345*

2.5. **INT** - Valeur de nombre complète d'un nombre

nombre INT(nombre *par1*)
Paramètres: *par1* : un nombre
Description: Cette fonction retournera la valeur de nombre complète d'un nombre, c'est à dire la valeur la plus proche et la plus base sans décimales.
Valeur retour: La valeur de nombre complète.
Voir aussi: FRA
Exemple: #1=INT(1234.56) /* *fait 1234* , #1=INT(-12.34) /* *fait -13*

2.6. **NOT** - Négation logique

nombre NOT(nombre *par1*)
Paramètres : *par1* : un nombre
Description: Cette fonction retournera 1 si par1 est égal à zéro, et 0 si par1 n'est pas égal à zéro.
Valeur retour: 0 ou 1.
Voir aussi: SGN
Exemple: NOT(1) *est 0*

2.7. **POW** - Elévation à une puissance

nombre POW(nombre *par1*, nombre *par2*)

par2 : l'exposant
Description: Cette fonction élève le nombre *par1* au carré *par2*.
Valeur retour : Puissance.
Valeur retour: SQR
Exemple: #1=POW(8,3) /* *fait 512 (8*8*8)* , #1=POW(4,0.5) /* *fait 2*

2.8. RUN - Arrondir à X décimales

nombre RUN(nombre *par1*, nombre *par2*)

par2 : Nombre des décimales à arrondir
Description: La fonction RUN arrondit le nombre donné aux décimales indiquées.
Valeur retour: Le nombre arrondi.
Voir aussi: FNH, FNR, INT
Exemple: #1=RUN(-123.4567,3) /* *Champ 1 prendra la valeur -123.457*

2.9. RUND - Définition de la fonction d'arrondissement FNR

nombre RUND(nombre *par1*, nombre *par2*)

par2 : Le nombre des décimales à arrondir, par exemple 2
Description: La fonction RUND définit la manière selon laquelle la fonction FNR effectue l'arrondissement. Si par1 est positive, FNR arrondira en PLUS, si par1 est négative, FNR arrondira en MOINS.
Valeur retour: Aucune.
Voir aussi: FNR
Exemple:

```
RUND(-25,2)      /* Round DOWN to nearest 25 pence with 2 decimals
RUND(5,2)        /* Round UP to nearest 5 pence
RUND(1,3)        /* Round to 3 decimals
RUND(1,2)        /* FNR function will work as default
```

2.10. SGN - Contrôler si le nombre est négatif, zéro ou positif

nombre SGN(nombre *par1*)
Paramètres: *par1* : définit un nombre
Description: Cette fonction contrôlera si le nombre est négatif, zéro ou positif.
Valeur retour:

```
-1       The nombre is negative
 0       The nombre is zero
 1       The nombre is positive
```

Voir aussi: INT, NOT
Exemple: #1=SGN(-123.45) /* *Champ #1 prendra la valeur -1.*

2.11. **SQR** - Calculer la racine carrée du nombre

nombre SQR(nombre *par1*)
Paramètres: *par1* : le nombre auquel vous souhaitez prendre la racine carrée.
Description: Cette fonction calcule la racine carrée du nombre dans *par1*.
Valeur retour: La racine carrée.
Voir aussi: POW
Exemple: #1=SQR(4) */* fait 2*

3. Fonctions de texte

Dans la section suivante, nous allons voir la manière selon laquelle les textes et les numéros dans le texte sont modifiés et convertis.

3.1. CONV - Modification des caractères dans un texte

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

par3 : les nouveaux caractères à insérer
Description: Cette fonction contrôlera chaque caractère dans le texte *par1*. Si le caractère est égal à un des caractères donnés dans *par2* il sera remplacé par le nouveau caractère présenté dans *par3*. Si le paramètre 1 contient "abc" et le paramètre 2 le texte "ABC", cette fonction remplacera a par A, et b par B ainsi que c par C.
Valeur retour: Le texte dans lequel les caractères voulus sont convertis.
Voir aussi: LOWER, SMAA, UPPER
Exemple: #1 = CONV("hans", "hn", "lr") */* donnera "lars"*

3.2. **EDIT** - Edition d'un nombre complet

texte EDIT(nombre *par1*, texte *par2*)

par2 : USING masque pour l'édition
Description: La fonction EDIT convertit un nombre complet en un texte de champ. Le masque USING détermine la présentation du texte à l'écran.
Valeur retour: Le champ de texte édité.
Voir aussi: NUMBER, USING
Exemple:

```
#1 = EDIT(-123,"&&&,&&")        /* Returns "001,23"
#1 = EDIT(123,"##&-#&&&")       /* Returns "  0- 123"
#1 = EDIT(123,"eg.# and ##")    /* Returns "eg.1 and 23"
```

3.3. **FIND** - Rechercher un texte dans le champ de texte

nombre FIND(texte *par1*, texte *par2*, nombre *par3*, nombre *par4*, nombre *par5*)

Description: La fonction recherchera des textes *par1* dans le texte *par2*.Il faut mettre les deux paramètres entre guillemets "".
Valeur retour: Retournera -1 si le texte n'est pas trouvé ou un nombre positif qui correspond à la position selon laquelle le texte a été trouvé. (à partir de 1).
Voir aussi:
Exemple:

```
#1 = "This is a texte"
#2 = FIND("te", #1)          /* Field #2 contains the value 11.
```

3.4. **LEN** - Taille d'un texte

nombre LEN(texte *par1*)
Paramètres: *par1* : définit un texte
Description: Cette fonction calcule la taille d'un texte.
Valeur retour: La taille du texte.
Voir aussi: SPOFF
Exemple:

```
#1 = "SW-Tools ApS"
#2 = LEN(#1)        /* returns the length of the texte
```

Champ #2 prendra la valeur 12, car il y a 12 caractères dans #1.

3.5. LOWER - Convertir les lettres d'un texte en minuscules

texte LOWER(texte *par1*)
Paramètres: *par1* : définit un texte à convertir
Description: Cette fonction convertit un texte en minuscules, par exemple toutes les lettres de A à Z seront converties en des lettres à partir de a à z.
Valeur retour: Le texte converti.
Voir aussi: CONV, SMAA, UPPER
Exemple:

```
#1 = "THIS is a TEST"
#2 = LOWER(#1)   /* Field #2 then contains the texte "this is a test"
```

3.6. NAME - Extraction du prénom et nom de famille

texte NAME(texte *par1*, tal *par2*)

Description: Cette fonction extraira les prénoms et le nom de famille à partir d'un champ de texte indiqué et puis elle retrouvera ces noms selon *par2*. Cette valeur peut, par exemple, être utilisée pour faire le tri.
Pour ce faire, utilisez le fichier texte de SSV WORDS.ENG. Chaque ligne contient un mot spécial comme Mme, Mlle, M., et éventuellement un remplacement (Monsieur ;M.)
Valeur retour: Le nom selon *par2*.
Voir aussi: SMAA, SOGE
Exemple:

```
#1 = NAME("MR CHRIS HANSON",0)    /* Gives "HANSON, CHRIS Mr."
#1 = NAME("OLSEN, MICHAEL",1)     /* Gives "MICHAEL OLSEN"
```

3.7. NUMBER - Conversion des nombres 'Mystérieuses'

nombre NUMBER(texte *par1*)
Paramètres: *par1* : Un champ de texte contenant un nombre
Description: Cette fonction NUMBER prend une valeur à partir du champ de texte même s'il existe des caractères entre les chiffres.
Valeur retour: Le nombre complet recherché, aucunes décimales sont retournées.
Voir aussi: EDIT, NUMS, USING
Exemple:

```
#1=NUMBER("33)33 05 56")  /* Covert phonenumber to value 33330556
#1=NUMBER("31/03-1997")   /* A date is converted to the value 31031997
#1=NUMBER("ab1cd2&3.4")   /* Returns 1234
```

3.8. NUMS - Conversion du champ de texte en nombre

nombre NUMS(texte *par1*)
Paramètres: *par1* : Un texte contenant un nombre
Description: Dans une ligne, par exemple, #1=#2 dans laquelle #1 est numérique et #2 représente un champ de texte, un nombre éventuel dans le champ 2 sera converti automatiquement en nombre. Vous obtiendrez le même résultat en utilisant #1=NUMS(#2) mais NUMS est optionnel.
Cependant, si vous souhaitez effectuer des calculs en utilisant la valeur des champs de texte, par exemple, #1=#2+#3, il faut utiliser #1=NUMS(#2)+NUMS(#3) pour indiquer que les textes doivent d'abord être convertis en nombre.
Valeur retour: La valeur numérique du champ de texte. Le point décimal doit être indiqué comme (point)
Voir aussi: NUMBER
Exemple: #1 = NUMS("aa111") + NUMS("222,22 test") + NUMS("333.33")
Le champ 1 donne le total des nombres dans les champs de texte = 555.33

3.9. **PACK** - Compression d'un nombre

texte PACK(texte *par1*, nombre *par2*)

par2 : 0, non utilisé, réservé pour le type de progiciel futur
Description: 8870 - basic call 60,A$,B$ le même que B$=PACK(A$)
Valeur retour: La valeur compressée du champ.
Voir aussi: UNPACK
Exemple: #1=PACK(#2) */* #1 becomes the packed value of #2*

3.10. SMAA - Convertir les lettres du texte en minuscules et en majuscules - noms

texte SMAA(texte *par1*)
Paramètres: *par1* : Le texte à convertir
Description: Cette fonction met les lettres du texte dans *par1* en minuscules et en majuscules. Par exemple, la première lettre dans chaque mot sera convertie en majuscule, tandis que les autres lettres sont converties en minuscules. Le fichier texte de SSV WORDS.ENG sera contrôlé pour le premier et dernier mot donné, par exemple, M. ou SARL.
Veuillez noter que la fonction SMAA peut également être utilisée dans DATAMASTER pour la conversion des noms d'entrée de champs.
Valeur retour: Le texte converti.
Voir aussi: CONV, LOWER, NAME, UPPER
Exemple:

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

3.11. SOGE - Création d'une clé de recherche à partir d'une adresse de champ

texte SOGE(texte *par1*, nombre *par2*)

Description: Selon l'adresse notée, le nom de la rue et le numéro de rue seront isolés. Ils sont placés ensemble pour créer un champ avec un nom de la rue d'une longueur fixe *par2* suivi par un numéro de rue. Vous pouvez, par exemple, utiliser ce champ pour faire le tri et la recherche.
Valeur retour: Un nom de la rue avec la longueur *par2* suivi par un numéro de rue de 4 chiffres.
Voir aussi: LOWER, NAME, SMAA, UPPER
Exemple:

```
#1 = SOGE("MAIN STREET 3",12)      /* Gives "MAINSTREET____3"
#1 = SOGE("27, Rue de Saute",8)    /* Gives "Rue deSau__27"
```

3.12. SPOFF - Supprimer des blancs placés au début ou à la fin du texte

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

Description: Cette fonction supprimera tous les blancs qui sont placés au début ou à la fin du texte. De plus, tous les endroits blancs seront seulement réduits à un caractère blanc.
Valeur retour: Le texte converti.
Voir aussi: LEN
Exemple:

```
#1="   This     is    a texte    "
#2=SPOFF(#1)       Field #2 then contains the value "This is a texte".
```

3.13. UNPACK - Un nombre décompressé

texte UNPACK(texte *par1*, nombre *par2*)

par2 : 0, non utilisé, réservé au type de progiciel futur
Description: 8870 - basic call 61,A$,B$ est le même que B$=UNPACK(A$)
Valeur retour: La valeur décompressée du champ.
Voir aussi: PACK
Exemple: #1=UNPACK(#2) */* #1 est la valeur décompressée #2*

3.14. UPPER - Convertir les lettres du texte en minuscules

texte UPPER(texte *par1*)
Paramètres: *par1* : définit un texte à convertir
Description: Cette fonction convertit des lettres d'un texte en majuscules, par exemple, toutes les lettres de a à z qui seront converties en des lettres de A à Z.
Valeur retour: Le texte converti.
Voir aussi: CONV, LOWER, SMAA
Exemple:

```
#1="This is a test"
#2=UPPER(#1)       /* Field #2 then contains the texte "THIS IS A TEST"
```

3.15. **USING** - Edition d'un nombre

texte USING(nombre *par1*, texte *par2*)

par2 : USING masque pour éditer
Description: La fonction USING convertit un nombre en un champ de texte. Le masque noté USING détermine la présentation du texte à l'écran.
Il faut faire appel à la fonction en utilisant la syntaxe spéciale de programmation de BASIC : textefield = nombre USING "masque"
Valeur retour: Le champ de texte édité.
Voir aussi: EDIT
Exemple:

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

4. Checkchiffre et validation

Dans la section suivante, nous allons voir les fonctions pour le calcul de checkchiffre, la validation des textes et des nombres.

4.1. **CCODE** - Champ checktexte (DATAMASTER checkcodetexte)

texte CCODE(texte *par1*, field *par2*)

par2 : Le nombre de champ qui contient la définition de contrôle "7", "#7", "va#7", "va07"
Description: Cette fonction lit la définition de champ à partir de la base de données du champ *par2* et recherchera les codes de contrôle pour celui-ci. Ce texte associé à la valeur *par1* sera retourné.
Valeur retour: Le texte de contrôle. Blanc indique ne pas permis, "-" indique qu'aucun contrôle n'est défini.
Voir aussi: VALID, VALCH
Exemple: #1 = CCODE(9,"va#7") */*donnera "Special"*

4.2. **CHECK** - Contrôle OCR

texte CHECK(texte *par1*)
Paramètres: *par1* : un numéro, par exemple, un numéro de client
Description: Cette fonction traitera un numéro et retrouvera un texte contenant un numéro de contrôle OCR
#47=CHECK (#19) calculera le module 10 du chiffre de contrôle OCR avec les poids 212121...pour le champ de texte #19 et additionnera cela au dernier chiffre.
CHECK("123456789012345") retournera un texte auquel est additionné un caractère "1234567890123452".
Valeur retour: Le texte et le chiffre de contrôle OCR.
Voir aussi: CHEX
Exemple: #1 = CHECK("33330556") */* donnera "333305563"*

4.3. **CHEX** - Contrôle de Module 11

texte CHEX(texte *par1*, texte *par1*)

par2 : le poids pour le calcul du chiffre de contrôle, 2 chiffres pour chaque entrée de caractère.
Description: #47=CHEX (#15,"01020304") calculera selon le même principe que CHECK un chiffre de contrôle et additionnera celui-ci au champ retourné.
Ce chiffre calculera le module 11 avec les poids 01, 02, 03, 04 selon le deuxième paramètre. Chaque paramètre avec deux chiffres correspond au poids d'un chiffre dans le champ.
Valeur retour: Le texte et le chiffre de contrôle.
Voir aussi: CHECK
Exemple: #2=CHEX("330556", "010203040506") */* donnera "3305569"*

4.4. VALCH - Contrôler si le texte recherché se trouve dans la chaîne de validation

nombre VALCH(texte *par1*, texte *par2*)

par2 : La valeur permise séparé par une virgule **Description:** Cette fonction contrôlera si *par1* se trouve parmi les valeurs notées dans *par2*. Il faut que toutes les valeurs notées dans *par2*.soient séparées par, (une virgule).
Valeur retour: Retournera 0 si *par1* n'est pas trouvé *par2*.
Voir aussi: CCODE, VALID
Exemple: #1=VALCH("Chris", "Anne,Nette,Chris,Ole,Michael") */*champ #1 contiendra la valeur 2.*

4.5. VALID - Contrôler si le nombre se trouve parmi les valeurs notées et permises.

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

. **Description:** Cette fonction contrôlera si la valeur notée dans *par1* est permise en contrôlant les valeurs permises dans *par2*.La syntaxe pour *par2* sera:
"1,2,8-10,12", c'est à dire que les valeurs 1, 2, 8 à 10 et 12 seront permises
"-1,2,8-10,12" En indiquant la marque moins au début de la ligne, les valeurs suivantes NE seront PAS permises

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

change l'intervalle dans le champ de texte #20 en insérant 15 de telle façon que #20 devienne: "1-3,8-12,15"
Valeur retour: Retournera 0 si *par1* ne sera pas trouvé dans *par2*.
Voir aussi: CCODE, VALCH
Exemple: #1 = VALID(9, "1,2,8-10,12")
Ainsi, le champ #1 prendra la valeur 3 comme si cette valeur avait été trouvé dans le troisième intervalle.

5. Fonctions de date

Dans ce chapitre nous allons décrire le calcul de date.

5.1. **DATE** - Date YYYYMMDD

nombre DATE()
Valeur retour: Le date actuelle sous la forme YYYYMMDD.

5.2. DATECALC - Calcul d'une date

Date DATECALC(Date *par1*, nombre *par2*, nombre *par3*, nombre *par4*, nombre *par5*)

par5 : jour(s) JJ

Description:

Cette fonction vous permettra d'indiquer une date, d'additionner ou de soustraire. En mettant *par2* 0, vous pouvez indiquer la date en utilisant les paramètres *par3-par5*. En mettant 3, 4 et 5, le paramètre 1 ne sera pas traité. Si vous voulez uniquement indiquer le mois, cette fonction utilisera la date donnée dans*par1* et modifiera le mois pour le mois donné dans *par4*.

Valeur retour: La date calculée sous la forme YYYYMMDD.

Voir aussi: DAY, FNA, FNB, FND, FNU, FNV, FNY,MONTH, WDAY, WORKD

Exemple:

```
#1=DATECALC(0, 0, 1997, 10, 16)   /* set the date 16.october 1997 (19971016)
#1=DATECALC(19970101, 1, 0, 2, 0) /* add 2 months to the date (19970301)
#1=DATECALC(19971016, 2, 1, 2, 3) /* subtract 1 year, 2 months and
                                     3 days from the date (19960813)
```

5.3. DAY - Description d'une date sous la forme de texte

texte DAY(Date *par1*)
Paramètres: *par1* : une date sous la forme YYYYMMDD
Description: Cette fonction crée une chaîne de texte qui décrit la date en tant que : <?> <jour de la semaine> le. <jour> <mois> <année>
Si le jour est un 'jour de fête' , le premier symbole représenté sera *, S'il s'agit seulement d'un demi jour de fête /, sinon blanc. On utilisera le même agenda électronique comme déjà décrit pour WORKD.
Valeur retour: Chaîne de texte contenant la date.
Voir aussi: DATECALC, FNA, FNB, FND, FNU, FNV, MONTH, WDAY, WORKD
Exemple: #1 = JOUR19931016) */* *créer une chaîne de texte pour le 16 octobre 1993*
Champ #1 contiendra la valeur "*Samedi le 16 octobre 1993"

5.4. FNA - Convertir la date en un nombre de jour à partir de l'an 0

nombre FNA(Date *par1*, nombre *par2*)

Description: Cette fonction convertira la date donnée en ce nombre de jours à partir de l'an 0. Elle peut, par exemple, effectuer des calculs sur la différence entre deux dates
Valeur retour: Nombre de jours depuis l'an 0.
Voir aussi: FNB, FND, FNU, FNV, DATECALC, DAY, MONTH, WDAY, WORKD
Exemple:

```
#1 = 19931215           /* the date 15. december 1993
#2 = FNA(#1)            /* how many days since 0 ?
#3 = #2 - FNA(19931202) /* how many days since 2. december ?
```

Le champ #2 contiendra la valeur 728277 et le champ #3 la valeur 13

5.5. FNB - Convertir le nombre de date à partir de l'an 0 en date

Date FNB(nombre *par1*, nombre *par2*)

Description: Cette fonction convertira un nombre de jours à partir de l'an 0 en une date AAAAMMJJ. C'est à dire qu'un nombre retourné à l'aide de la fonction FNA() peut être donné en tant que paramètre pour cette fonction qui entraîne le retour d'une date correcte.
Valeur retour: La valeur retour sera une date sous la forme AAAAMMJJ.
Voir aussi: DATECALC, DAY, FNA, FND, FNU, FNV, MONTH, WDAY, WORKD
Exemple:

```
#1 = FNA(19931215) /* convert the date 15. december 1993
#2 = FNB(#1 + 9)   /* add 9 days and convert to date YYYYMMDD
```

Le champ #2 contiendra la valeur 19931224, par exemple le. 24. décembre 1993

5.6. FND - Conversion de date

Date FND(Date *par1*)
Paramètres: *par1* : définit une date sous la forme AAAAMMJJ
Description: Cette fonction permet de convertir des dates à partir d'un format en un autre. Normalement, elle sera utilisée lors de l'exécution de tri ou de sélections. Par exemple

970101 is greater than 961231 but 311296 is greater than 010197

Vous verrez qu'il sera nécessaire d'utiliser la fonction FND (#7) au lieu de #7 si le champ 7 est un champ de date sous la forme JJMMYY
Valeur retour: La valeur retour est une date sous les formes AAMMJJ ou JJMMAA.
Voir aussi: DATECALC, DAY, FNA, FNO, FNU, FNV, FNY, MONTH, WDAY, WORKD
Exemple:

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

5.7. FNE - Convertir la date en numéro de mois.

nombre FNE(Date *par1*)
Paramètres: *par1* : une date sous la forme AAAAMMJJ ou AAMMJJ
Description: Vous pouvez utiliser cette fonction pour effectuer des calculs des intervalles du mois.
Valeur retour: Cette fonction calculera le numéro de mois comme ANNEE*12 + MOIS (AA*12+MM)
Voir aussi: DATECALC, DAY, FNA, FNB, FND, FNV, MONTH, WDAY, WORKD
Exemple: #1 = FNE(19950331) */* gives 1143 = 95*12+03*

5.8. FNF - Convertir la date en numéro de jour, 360 jours par an

nombre FNF(Date *par1*)
Paramètres: *par1* : la date sous la forme AAAAMMJJ ou AAMMJJ
Description: Cette fonction convertira la date en un nombre de date à partir de l'an donné en utilisant 360 jours par an, selon le même procédé que FNA(date,360)
Valeur retour: Nombre des jours à partir de l'an 0.
Voir aussi: FNA
Exemple:

```
#1 = FNF(19950331) /* gives 1718290
#1 = FNF(950331)   /* gives 34290
```

5.9. **FNO** - Convertir la date en JJMMAA

Date FNO(Date *par1*)
Paramètres: *par1* : Date sous la forme JJMMAA, AAMMJJ ou AAAAMMJJ
Description: Indépendamment de la manière selon laquelle les dates sont tournées dans le champ d'entrée, la forme JJMMAA. est retournée. Vous pouvez l'utiliser pour la sortie.
Valeur retour: DDMMYY
Voir aussi: FND, FNY
Exemple:

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

5.10. FNU - Convertir la date en jour de la semaine

nombre FNU(Date *par1*)
Paramètres: *par1* : une date sous la forme AAAAMMJJ
Description: Cette fonction permet de calculer le jour d'une date.

Voir aussi: DATECALC, DAY, FNA, FNB, FND, FNV, MONTH, WDAY, WORKD
Exemple: #1 = FNU(19931215) */*quel jour sera le 15. décembre 1993 ?*
Le champ #1 contiendra la valeur (=Mercredi)

5.11. FNV - Convertir la date en numéro de semaine ou le numéro de semaine en date

nombre FNV(nombre *par1*)
Paramètres: *par1* : définit une date sous la forme AAAAMMJJ ou un numéro de semaine sous la forme AAAASS
Description: Cette fonction convertira une date en numéro de semaine AAAASS, si *par1* est une date. Si *par1* est un numéro de semaine AAAASS, cette fonction retrouvera la date qui correspond au dernier dimanche avant la semaine donnée. Le même que SEMAINE(date)
Valeur retour: Retournera un nombre AAAASS, dans lequel AAAA = année et SS= numéro de semaine, ou une date sous forme de AAAAMMJJ.
Voir aussi: DATECALC, DAY, FNA, FNB, FND, FNU, MONTH, WDAY, WEEK, WORKD
Exemple:

```
#1 = FNV(19931016) /* calculate weeknombre of the date 16. oktober 1993
#2 = FNV(#1)       /* calculate the last sunday before weeknombre 41
```

Ainsi, le champ #1 contient la valeur 199341, correspondante au numéro de semaine 41. Le champ #2contient la date 19931010.

5.12. **FNY** - Convertir la date en AAAAMMJJ

Date FNY(Date *par1*)
Paramètres: *par1* : Date sous la forme JJMMAA, AAMMJJ ou AAAAMMJJ
Description: Indépendamment de la manière selon laquelle les dates tournent dans le champ d'entrée, la date sera retournée sous la forme AAAAMMJJ. Vous pouvez l'utiliser dans vos calculs.
Valeur retour: AAAAMMJJ
Voir aussi: FND, FNO
Exemple:

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

5.13. **MONTH** - Description d'un mois sous forme de texte

texte MONTH(Date *par1*)
Paramètres: *par1* : définit une date sous la forme AAAAMMJJ
Description: Cette fonction généra un texte qui correspond au nom du mois voulu.
Valeur retour: Retournera le nom du mois.
Voir aussi: DATECALC, DAY, FNA, FNB, FND, FNU, FNV, WDAY, WORKD
Exemple: #1 = MONTH(19931016) */* date le 16 octobre 1993*
Ainsi, le champ #1contiendra la valeur "octobre".

5.14. **TIME** - Le temps actuel TTMMSS

nombre TIME()
Valeur retour: Le temps actuel sous la forme TTMMSS.

5.15. **WDAY** - Description du jour de la semaine pour la date

texte WDAY(Date *par1*)
Paramètres: *par1* : définit une date sous la forme AAAAMMJJ
Description: Cette fonction généra un texte qui décrit la date sous la forme: <?> jour de la semaine
Si la date est un jour de fête, le premier symbole présenté sera * . Si la date est un demi jour de fête /, sinon blanche. Le même agenda électronique comme déjà décrit pour WORKD sera utilisé.
Valeur retour: Un texte contenant le jour.
Voir aussi: DATECALC, FNA, FNB, FND, FNU, FNV, MONTH, WDAY, WORKD
Exemple: Si #1 = WDAY(19931016) */* *le champ #1 contiendra la valeur "*Samedi"*

5.16. WEEK - Convertir la date en numéro de semaine ou le numéro de semaine en date

nombre WEEK(nombre *par1*)
Paramètres: *par1* : définit une date sous la forme AAAAMMJJ, ou un numéro de semaine sous la forme AAAASS
Description: Cette fonction convertira une date en numéro de semaine AAAASS, si *par1* est une date. Si *par1* est un numéro de semaine AAAASS, cette fonction retournera une date qui correspond au dernier dimanche qui se trouve avant la semaine indiquée. Le même que FNV(date)
Valeur retour: Retournera un nombre AAAASS, dans lequel AAAA = année et SS= numéro de semaine, ou une date AAAAMMJJ.
Voir aussi: FNV
Exemple:

```
#1 = WEEK(19931016) /* calculate weeknombre of the date 16. oktober 1993
#2 = WEEK(#1)       /* calculate the last sunday before weeknombre 41
```

Le champ #1 contiendra la valeur 199341 correspondante au numéro de semaine 41. Le champ #2 contiendra la date 19931010.

5.17. **WORKD** - Calculer le nombre des jours de travail entre deux dates

nombre WORKD(Date *par1*, Date *par2*)

par2 : définit une date sous la forme AAAAMMJJ
Description: Cette fonction calculera le nombre des jours de travail entre deux dates.
#47 = WORKD (#15,#PD) Calculera le nombre des jours de travail actuels à partir de la date donnée dans le champ15 à la date entrée 'par jour '.
Cette fonction commencera par calculer le nombre des jours entre deux dates. Tous les samedis et dimanches seront déduits. Ensuite, les fonctions rechercheront à partir du agenda instauré où les jours de fête se trouvent et puis déduiront les demis jours de fête s'ils se trouvent dans l'intervalle donné
Cet agenda électronique instauré peut éventuellement être ajusté individuellement. Cette fonction utilise le fichier RAPDAY.ENG. Ce fichier est un fichier texte SSV dans lequel chaque ligne contient un jour de fête sous la forme AAAAMMJJ Les demi jours de fête suivront sous la forme, par exemple, 19960630;50. C'est à dire que le deuxième champ donnera le pourcentage.
Valeur retour: Retournera le nombre des jours de travail entre deux dates.
Voir aussi: DATECALC, FNA, FNB, FND, FNU, FNV, MONTH, WDAY, WORKD
Exemple: #1 = WORKD(19930420, 19930430) /* *Champ #1 contiendra la valeur 19.*

6. Traitement des champs multiples

Dans ce chapitre nous allons décrire les fonctions pour le traitement des champs multiples, spécialement la fonction LET.

6.1. LET - Calcul des champs multiples simultanément

nombre LET(fields *par1*)
Paramètres: *par1* : définit un ou plusieurs champs
Description: Cette fonction vous permettra de traiter un ou plusieurs champs simultanément. Vous pouvez calculer ces champs à l'aide des expressions suivantes : champ **XX** constant/champ, dans lequel **XX** peut être

Operator	Meaning
=	set fields equal to
+=	add value to the fields
-=	subtract value from the fields
*=	multiply fields with the value
/=	divide fields with the value
%=	set fields to the mod. value from the division
&=	perform logical and operation on fields
|=	perform logical or operation on fields

Valeur retour: Retournera 0 si le calcul est effectué correctement.
Voir aussi: CLEAR, ZERO
Exemple:

Letexpression	Function
LET("#1-10=12")	Field 1 to 10 is set equal to 12
LET("#20,25=3,7")	#20=3 and #25=7
LET("#20-25=le#1-10")	Field 20-25 is set to the file le field 1-6
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")	Several files may be mixed
LET("#20-25+=1")	Add 1 to all the fields 20-25

6.1.1. LET - Instaurer les champs égaux dans les programmes d'IQ (IQ)

nombre LET(fields *par1*)
Paramètres: *par1* : définit un ou plusieurs champs
Description: L'élargissement de la fonction LET vous permettra de travailler entre plusieurs programmes et entre les lignes dans un programme de transaction.
Valeur retour: Retournera 0 si le calcul est effectué correctement
Voir aussi:
Exemple:

Letexpression	**Function**
LET (20.#1-3=#1-3)	Sets field 1-3 for program 20 = this program #1-3
LET (#1-3=20.#4-6)	Sets field 1-3 in this program to #4-6 from program 20
LET (#10=#3.4)	Sets field 10 equal to field 3 from line 4

6.1.2. LET - Création des nouveaux fichiers (RAP)

nombre LET(fields *par1*)
Paramètres: *par1* : un ou plusieurs champs
Description: La fonction LET peut être utilisée pour construire des nouveaux fichiers.
Valeur retour: Retournera 0 si le calcul est effectué correctement.
Voir aussi: INSERT, UPDATE, *Manuel de Rapgen*
Exemple:

Letexpression	**Function**
LET (aa=#1-3,87,le#2)	Define file aa, key=aa#1, type=1.database driver
LET (aa=#1-3,6K,15D)	Keys aa#4 and aa#5 (duplicates)
LET (aa=#1-3,6,15:2,NP)	Keys aa#2 and rel.recno (duplicates)
LET (aa=#1-3),12000	12000 records (default is 1000 if needed)
LET (aa=#1-3),-1	File should be builded eachtime
LET (aa=#1-3),1000,xnet	File is a XNET file
LET (aa=#1-3) -acc	File is an access file, build always
LET (07/aa=#1-3),25	Lu may be given for basic files

6.2. CLEAR - Garnir des zéros tous les champs dans un fichier (RAP)

nombre CLEAR(file *par1*)
Paramètres: *par1* : l'abréviation à deux caractère du fichier
Description: Cette fonction garnit à zéro tous les champs d'un fichier.
Valeur retour: Retournera 0 si OK.
Voir aussi: ZERO
Exemple:

```
UPDATE(1)       /* the report updates the file
CLEAR(VA)       /* zero all fields from article file
VA#1 = "1234"   /* article nombre
INSERT(VA)      /* insert new record in article file
```

Cet exemple insère un nouveau enregistrement dans le fichier d'article. En raison de la fonction CLEAR(), tous les autres champs à part le numéro d'article sont garnis à zéro.

6.3. CLRFLAG - Désactiver les paramètres des champs (IQ)

CLRFLAG(champs *par1*, nombre *par2*, nombre *par3*)

Description: Chaque champ à l'écran est associé à un nombre des paramètres (bits) définissant la manière selon laquelle les champs sont utilisés. Vous pouvez utiliser la fonction SETFLAG pour instaurer les drapeaux, et la fonction CLRFLAG pour les désactiver.
Voir aussi: SETFLAG, GETFLAG
Exemple: CLRFLAG("#12,44",7,0)

6.4. COLOR - Instaurer la couleur de fond de boîte pour un nombre des champs

COLOR(champs *par1*, CouleurRouge *par2*, CouleurVerte *par3*, Couleur Bleue *par4*)

par4 : Valeur de couleur bleu (0-255)
Description: La couleur de fond pour les champs indiquées est instaurée en tant que valeur RGB, c'est à dire que la boîte du champ sera remplie avec la couleur donnée
Valeur retour: Aucune
Voir aussi: COLORF
Exemple:

```
COLOR("#3-4",255,0,0)      /* Field 3 and 4 becomes a red box around
COLOR("#3-4",-1)           /* No background color for the fields
```

6.5. COLORF - Instaurer la couleur pour l'élément graphique (premier plan) pour un nombre des champs

COLOR(Champs *par1*, CouleurRouge *par2*, CouleurVerte *par3*, CouleurBleue *par4*)

par4 : Valeur de couleur bleue (0-255)
Description: La couleur de l'élément graphique(premier plan) pour un champ donnée est instaurée en tant valeur RGB, c'est à dire que le texte dans le champ sera présenté dans la couleur indiqué,
Valeur retour: Aucune
Voir aussi: COLOR
Exemple: COLORF("#3-4",0,0,255) */* Field 3 and 4 are printed in blue*

6.6. DIALOG - Dialogue supplémentaire d'entrée de données

Nombre DIALOG(Fields *par1*)
Paramètres: *Par1*: Champs affichés dans le dialogue
Description: La fonction DIALOG vous permettra de présenter les boîtes de dialogue avec les champs voulus, à un instant déterminé dans le rapport ou dans un programme d'IQ, par exemple en clinquant sur un champ.
DIALOG("#1,7-8,le#3") définit un dialogue avec les champs indiqués. La documentation des champs est utilisée en tant que aide en ligne qui apparaît quand vous déplacez le pointeur de la souris sur les noms de champ.
Avec les numéros de champ vous pouvez indiquer un ou plusieurs options

```
Lxxxx   Line     (dialog units)
Pxxxx   Position (dialog units)
Hxxxx   Height    (dialog units)
Wxxxx   Width    (dialog units)
N       No leading texte
N1      Add fieldnombre to leading texte
N2      Display leadingtexte above field instead of left of field
C       COMBOBOX, Field check definitions shown as values
O       LISTBOX, Field check definitions shown as values
:xx     Skip to next column fieldline xx
+xx     Skip xx fieldlines down
```

.
Valeur retour: OK=0, CANCEL=1
Voir aussi: PARAMS
Exemple:

```
DIALOG("#1-3,11")            /* Make a dialog with the given fields
```

6.7. GETFLAG- Lire les paramètres d'un champ (IQ)

nombre GETFLAG(champ *par1*, nombre *par2*, nombre *par3*)

Description: Chaque champ à l'écran est lié à un nombre des paramètres (bits) définissant la manière selon laquelle les champs sont utilisés.
La fonction SETFLAG vous permettra d'instaurer ces paramètres et la fonction CLRFLAG de les désactiver. La fonction GETFLAG vous permettra de lire ces drapeaux.
Valeur retour: Aucune
Voir aussi: SETFLAG, CLRFLAG
Exemple: GETFLAG("#12,44",7,0)

6.8. SETFLAG- Instaurer les paramètres des champs sur l'écran (IQ)

SETFLAG(fields *par1*, Bitflag *par2*, nombre *par3*)

Description: Chaque champ sur l'écran est associé à un nombre des paramètres (bits) définissant la manière selon laquelle les champs sont utilisés. La fonction SETFLAG vous permettra d'instaurer ces paramètres et la fonction CLRFLAG de les désactiver. La fonction GETFLAG vous permettra de lire ces drapeaux.
Pour le type de paramètre 3, vous devez toujours indiquer 0
Valeur retour: Aucune
Voira aussi: GETFLAG, CLRFLAG
Exemple: SETFLAG("#12,44",7,0)

6.9. ZERO - Garnir à zéro un nombre des champs

ZERO(fields *par1*)
Paramètres: *par1* : Spécification de champ
Description: Les champs indiqués seront garnis à zéro. La fonction ZERO fonctionne selon le même procédé que LET.
Valeur retour: None
Voir aussi: LET, CLEAR
Exemple: ZERO("3,19") */* Zeroes field 3 and field 19*

7. Contrôle de rapport

Dans la section suivante, nous allons décrire les fonctions pour le contrôle du débit dans les calculs et des copies d'un rapport dans RAPGEN. Les fonctions CHAIN, MESS et RETURN peuvent également être utilisées dans IQ et dans DATAMASTER. Les autres fonctions ne seront pas importants pour les programmes de l'écran.

7.1. CHAIN - Démarrage du rapport suivant ou un autre programme (RAP)

nombre CHAIN()

par3 : Blanc ou Index, niveau de total, numéro de l'entreprise
Description: Une fois le rapport terminé, CHAIN(7) commencera le numéro de rapport 7. Lors du démarrage, les mêmes paramètres de départ pour ce rapport sont utilisés.
CHAIN(7,",310395,-,9999","1") mettra par date en tant que 310395, la clé de démarrage en tant que rien, la clé de fin en tant que 9999 et le niveau le plus bas en tant que 1. Les autres paramètres de départ ne sont pas changés.
CHAIN(2007) commencera le numéro de rapport 7 dans le sous-système 2.
CHAIN(-1,"c:/windows/write.exe") commencera ce programme de (windows).
Chaque fois que vous exécutez CHAIN un nouveau nombre d'exécution est indiqué à partir de 1 et les suivants. Un rapport, commencé à partir du menu, aura le numéro d'exécution 0. #20=CHAIN() où CHAIN sont utilisées sans des paramètres, vous pouvez rechercher ce numéro de telle façon que vous puissiez exécuter un rapport plusieurs fois, par exemple imprimer un nombre des copies.
CHAIN("c:/windows/write.exe") peut être utilisée dans les programmes IQ/DATAMASTER pour démarrer un programme de windows
Valeur retour: CHAIN() retournera le numéro d'exécution actuel.
Voir aussi: EXIT , CHAINR
Exemple:

```
#20=CHAIN()          /* This is report nombre 7.
IF #20<3 CHAIN(7)    /* The same report will be started 4 times.
```

7.1.1. CHAINR - Commercer un programme ou un contrôle externe directement (RAP)

CHAINR(nombre *par1*, texte *par2*, texte *par3*)

par3 : Blanc ou Index, niveau de total, numéro de l'entreprise
Description: La commande CHAIN est toujours placée à la fin du rapport. C'est à dire que le programme suivant commencera d'abord après que l'exécution a eu lieu.
Nous vous conseillons d'utiliser CHAINR au lieu de CHAIN pour interrompre le programme en cours d'exécution et pour activer un autre programme immédiatement.
Valeur retour: None
Voir aussi: EXIT , CHAIN
Exemple: CHAINR(-1,"Notepad") */* Commencer l'ardoise immédiatement*

7.1.2. CHAIN - Démarrer un programme IQ ou une commande externe (IQ)

CHAIN(texte *par1*, texte *par2*)

par2 : Clé optionnelle pour la lecture d'un enregistrement
Description: Activer un numéro de programme ou une chaîne de commandes
Valeur retour: Aucune
Voir aussi: EXIT, ISACTIVE, WAIT
Exemple:

```
CHAIN ("20")    starts program 20.
CHAIN ("+5")    starts program 5 and activates this.
CHAIN (">5")    starts program 5, the current record will not be transmitted
CHAIN ("$5")    starts program 5, activates it and waits until this finishes.
CHAIN ("+5",#1) starts program 5 which will read a record using #1

#20="notepad"
#20="command.com /C edit myfile.txt"
CHAIN(#20)        starts the specified windows program
CHAIN("rapwin &") & as last character lets IQ continue
                    while the newstarted program is running.
```

7.2. **WAIT** - Attendre jusqu'à ce que le programme soit fini (IQ)

WAIT(programno *par1*)
Paramètres: *par1* : Numéro de programme
Description: Attendez jusqu'à ce que le programme soit fini(Voir EXIT).Les calcules continuent à être effectuer quand le deuxième fenêtre de programme est fermée.
Valeur retour: Aucune.
Voir aussi: CHAIN , EXIT
Exemple: WAIT(20) */* Ne continuer pas avant que le programme soit prêt*

7.3. COMPILE - Compiler un rapport (RAP)

COMPILE(nombre *par1*)

Conditions : Vous pouvez seulement utiliser cette fonction si vous possédez un compilateur C sur votre PC ou si vous avez acheté RAPGEN avec la licence de compilation.
Description: Au lieu de choisir 'Compiler' à partir du menu 'Paramètres' chaque fois que le rapport est démarré après avoir effectuer des modifications, vous pouvez déterminer ceci à l'aide des calculs.
Voir aussi: INSTALL
Exemple: COMPILE */* Le rapport sera compilé*

7.4. EXIT - Finir le rapport (RAP)

nombre EXIT(nombre *par1*)

Description: Cette fonction terminera le rapport ou le parcours actuel. (tri/imprimée).
Valeur retour: None
Voir aussi: CHAIN , CHAINR , MESS
Exemple:

```
READ(le) /* Read supplier data
IF #OK THEN BEGIN /* terminate the report if supplier not found
#12="Supplier ", le#1, " not found:"
MESS(#12)
EXIT(0)
END
```

7.4.1. EXIT - Fermer le programme IQ ou la fenêtre (IQ)

EXIT(nombre *par1*)
Paramètres: *par1* : numéro de programme à fermer
Description: EXIT(0) fermera le programme d'IQ actuel.
Valeur retour: None
Voir aussi: CHAIN , MESS, WAIT
Exemple:

```
EXIT(20) closes program 20 if this is open, 1020 gives subsystem 1.
EXIT(-1) closes the program selection window.
EXIT(-2) closes the field   selection window.
EXIT(-3) closes and exits all IQ.
```

7.5. KEYS - Indications de démarrage et d'arrêt (RAP)

nombre KEYS()

par2 : Eventuellement nom fixe sur la définition de fichier .KEY
Description: A l'aide de la fonction KEYS, vous pouvez exécuter un rapport dans des intervalles de démarrage et d'arrêt définit en tant que lignes dans un fichier texte. Par ceci, KEYS remplacera l'indication de démarrage et d'arrêt lors du démarrage et éventuellement l'INDEX.
Vous pouvez créer le fichier KEYS avec un simple éditeur-texte qui permet d'introduire éventuellemet :

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

Chaque ligne peut contenir un clé unique ou des intervalles pour l'impression. Avec 2: vous indiquez l'exécution via index 2, avec 47/ vous indiquez un codet de calcul qui peut être recherché à l'aide de par exemple #20=KEYS() qui sera utilisé dans les calculs.
Si vous utilisez KEYS(0), vous recevrez une liste des tous les enregistrements spéciaux dans le fichier clés. KEYS(1) entraîne une liste isolée de chaque ligne dans le fichier clé. De plus, la fonction ENDSUM peut éventuellement être utilisée pour recevoir un grand total
Il ne sera pas absolument nécessaire d'utiliser KEYS dans le rapport qui doit être commandé de cette manière. Lors du démarrage rapport, dans COMMENCER A PARTIR DE, vous pouvez indiquer :

```
(aa)                          Start with keysfile aa
(1000,1100-1200,0004          Run over these key ranges
```

Si aucun chemin ou extension ne sera indiqué pour le fichier clé, celui-ci sera placé dans le répertoire de rapport avec l'extension KEY, par exemple c:/rapfil/rap/aa.key
Valeur retour: KEYS() retournera le codet de calcul (47 of 47/111-222) pour l'intervalle actuel.
Voir aussi: ENDSUM, INDEX
Exemple:

```
KEYS(0,"c:/mydir/enfil.min")  /* The report is controlled from this file.
#20=KEYS()                    /* A calculation code is read.
```

7.6. INDEX - Instaurer l'index et la valeur de démarrage et d'arrêt pour le rapport (RAP)

nombre INDEX(index *par1*, texte *par2*, texte *par3*)

par3 : la valeur entrée par l'utilisateur normalement dans le champ d'arrêt à
Description: Cette fonction vous permettra d'indiquer de manière fixe l'index et l'intervalle d'arrêt et de démarrage pour un rapport. Si *par1* >= 1, l'index pour le fichier maître du rapport sera inscrit. C'est à dire que la lecture des enregistrements du fichier sera effectuée en suivant un ordre préétabli. Si *par2* contient quelque chose, cette fonction inscrira l'intervalle de démarrage pour *par3*.
Si les paramètres de démarrage et d'arrêt contient le premier caractère en tant que plus (+), la valeur indiquée sera mise devant ce que l'utilisateur a inscrit lors du démarrage du rapport.
INDEX(-2) verrouille le rapport afin d'utiliser l2, suivant l'ordre de tri descendant. Le driver de la base de données supportera la lecture effectuée en suivant un ordre descendant.
Valeur retour: Retournera l'index que le fichier maître utilise.
Voir aussi: KEYS
Exemple: INDEX(2,"D","D") */* Le fichier principal du rapport est KU (fichier de devise)*
Cette exemple maintient l'index 2 pour le rapport de telle façon que les devises soient lues en suivant l'ordre du nom de devise et pas le codet de devise. De plus, c'est seulement les enregistrements qui commence par la lettre "D" qui seront lus.
INDEX(1,"+02","+02") */* imprimer 024711l quand 4711* est entré

7.7. LTOT - Niveau de total le plus bas (RAP)

nombre LTOT(level *par1*)
Paramètres: *par1* : indiquera le niveau de total le plus bas voulu du rapport
Description: Si *par1* >= 0 , cette fonction inscrit le niveau de total le plus bas pour le rapport. Ce niveau correspond au niveau donné lors du démarrage d'un rapport.
Valeur retour: Retournera le niveau le plus bas du rapport.
Voir aussi: MTOT
Exemple: LTOT(1) */*imprimer uniquement des totaux, supprimer toutes les lignes singulières*

7.8. **MTOT** - Niveau de total maximal (RAP)

nombre MTOT(level *par1*)
Paramètres: *par1* : Le niveau de total maximum d'un rapport
Description: Cette fonction verrouille le niveau de total maximal d'un rapport. Si *par1* est égal à 0, le rapport n'imprimera pas des totaux.
Valeur retour: Retournera le niveau de total maximal.
Voir aussi: LTOT
Exemple: MTOT(1) */* un grande total sans importance sera superposé*

7.9. MESS - Afficher message à l'écran

nombre MESS(texte *par1*)
Paramètres: *par1* : le message à afficher
Description: MESS affichera un texte dans la boîte de message de Windows. Les symboles et les boutons suivants sont présentes en fonction du dernier caractère présenté dans le texte :

texte	**Symbol**	**Buttons**	**Defaultbutton**
texte	Info	OK	OK
texte?	!	OK, CANCEL	CANCEL
texte??	?	YES, NO, CANCEL	YES
texte!	!	YES, NO	YES
texte!!	STOP	OK	OK
texte?!	STOP	OK, CAN	OK

Valeur retour: 0=OK or YES, 1=NO, -1=CANCEL
Voir aussi: EXIT
Exemple:

```
#1=MESS("Stop the report !")
IF #1=0 EXIT(0)               /* exit the report
```

7.10. NOPAS - Aucun mot de passe et nom utilisateur sur le rapport (RAP)

NOPAS()
Paramètres: Aucunes
Description: Cette fonction enlèvera la protection en tant que mot de passe du rapport. Normalement, un rapport exécutant une mise à jour aura le mot de passe CARE. En utilisant NOPAS() ou PAS(), ce mot de passe peut être supprimé ou remplacé par un autre.
Voir aussi: PAS, UPDATE
Exemple:

```
UPDATE(1)
NOPAS()              /* no password on this report
```

7.11. **PAS** - Instaurer le mot de passe et le nom utilisateur (RAP)

nombre PAS(texte *par1*)
Paramètres: *par1* : le mot de passe voulu ou le nom utilisateur
Description: Cette fonction vous permettra d'insérer un mot de passe fixe sur le rapport. Vous devez indiquer ce mot de passe lors du démarrage du rapport.
Voir aussi: NOPAS
Exemple: PAS("SWTOOLS") */* indiquer mot de passe en tant que SWTOOLS*

7.12. **PARAMS** - Paramètres de démarrage supplémentaires (RAP)

PARAMS(Fields *par1*)
Paramètres: *Par1:* Champs à afficher dans le dialogue de démarrage
Description: PARAMS("#1,7-8,le#3") est une variante de la fonction de dialogue avec laquelle l'entrée est exécutée lors du démarrage du rapport et pas pendant l'exécution du rapport
En utilisant PARAMS dans les calculs sur un rapport, un nouveau bouton <Paramètre supplémentaire> sur l'écran de démarrage du rapport sera ajouté. En appuyant sur celui-ci, le dialogue contenant les champs sera affiché.
Valeur retour: Aucune.
Voir aussi: DIALOG
Exemple:

```
PARAMS("#1-3,11")              /* Construire un dialogue contenant les champs
indiqués
```

7.13. RETURN - Retourner à partir des calculs

nombre RETURN(nombre *par1*)
Paramètres: *par1* : la valeur à retourner
Description:
Cette fonction vous permettra de terminer les calculs pour l'enregistrement actuel et lu à partir du fichier principal. Si vous n'avez pas indiqué un paramètre ou *par1* est égal à 0, le rapport imprimera les lignes d'impression définies pour cet enregistrement. Si 1 est retourné, enregistrement actuel ne sera pas traité et imprimé.
Valeur retour: Aucune.
Voir aussi: GOSUB
Exemple: IF LE#6 < 1000 RETURN(1) /* *Aucune copie si le solde < 1000*

7.14. SORTKEY - Insertion de clé de tri supplémentaire(RAP)

nombre SORTKEY(fileid *par1*)
Paramètres: *par1* : 0, -1 ou fichier id
Description: Si vous souhaitez trier une liste de telle façon que chaque enregistrement apparaisse plusieurs fois, par exemple une liste de marchandise dans laquelle la marchandises doit être trouvée tant sous le fournisseur normal que sous un fournisseur éventuel
Si c'est le cas, vous devez faire le tri d'après un champ libre qui est calculé et une clé de tri insérée chaque fois que vous faites appel à la fonction SORTKEY
Vous pouvez également mélanger plusieurs fichiers à l'aide de cette fonction. Le fichier de tri contient un numéro, qui normalement point à un enregistrement dans le fichier principal du rapport. Avec SORTKEY(le), une clé sera insérée qui point au fichier le, et avec #20=SORTKEY(-1), vous pouvez lire le fichier qui fonctionne en tant que fichier principal pour le moment. Ainsi vous pouvez commander les calculs.
Valeur retour: Numéro de fichier principal, normalement 1.
Voir aussi: MERGE
Exemple:

```
#11=#9                  /* Sortworkfield = Alternative supplier
IF #11<>0 SORTKEY(0)    /* Extra sortkey with this
#11=#6                  /* Normal sortkey with normal supplier
```

7.15. SORTWORK - Utilisation d'un fichier de tri déterminé (RAP)

SORTWORK(nombre *par1*)
Paramètres: *par1* : Numéro de fichier de tri
Description: Lors de l'exécution de tri, RAPGEN créera des fichiers d'aide avec les noms c:/tmp/SIN00000.000 et c:/tmp/SUD00000.000 où c:/tmp/ est le chemin normal deTMP. Ces fichiers de tri ne seront pas supprimés après l'exécution, car en indiquant lors du démarrage du rapport ce qui suit :

START AT: SORT or SORTD

vous pouvez éviter le temps de tri en utilisant le même tri encore une fois,
En utilisant ceci plusieurs fois, vous pouvez vous protégez contre la suppression des fichiers d'aide en indiquant un numéro par exemple à l'aide de SORTWORK(47). Par ceci, les noms seront c:/tmp/SIN00000.047 et c:/tmp/SUD00000.047.
Valeur retour: Aucune.
Voir aussi:
Exemple: SORTWORK(47)

7.16. **WHEN** - Moment d'effectuer les calculs (RAP)

WHEN(nombre *par1*, nombre *par2*)

Description: Vous pouvez utiliser la commande WHEN s'il existe de différentes types de calculs

8. Contrôle de l'imprimante

Dans cette section nous allons voir les différents fonctions d'impression.

8.1. COPIES- Nombre des copies (RAP)

COPIES(nombre *par1*, Printer *par2*)

par2 : Numéro de l'imprimante éventuel
Description: COPIES(1) vous donnera une copie supplémentaire lors de l'impression. Vous pouvez indiquer 30 copies au maximum, mais votre PC doit posséder suffisamment de l'espace pour ces fichiers d'impression.
COPIES(1,7) produira une copie supplémentaire sur l'imprimante définie en tant que numéro dans la mise en place de l'imprimante. Veuillez noter qu'un saut de page inattendu peut apparaître si l'imprimante de copie a un format inférieur à l'orignal.
Valeur retour: Aucune.
Voir aussi: PRINT
Exemple: COPIES(1) */* *Imprimer deux fois*

8.2. PAGE - Changer la page de présentation du rapport (RAP)

nombre PAGE(nombre *par1*)
Paramètres: *par1* : la page du rapport voulu
Description: Normalement, quand un rapport imprime, il utilise la page de rapport noté 0 correspondante à sa présentation définie dans PAPGEN. Cependant, si le rapport, par exemple, doit imprimer une lettre de fournisseur dans une autre langue, le rapport pourra contenir 9 présentations de types différents. Ces présentations se trouvevent entre les pages de 0 à 9 . Vous pouvez les posséder à partir du menu 'Fichier', 'Présentation de page' quand vous éditez la forme.
Valeur retour: Retournera la page actuel sélectionnée en tant que page active d'impression.
Voir aussi: PRINT
Exemple:

```
PAGE(le#5)   /* select print page according to the suppliers language
PRINT(1-10)  /* print texte
```

8.3. PRINT - Impression des lignes à partir de la présentation du rapport (RAP)

PRINT(texte *par1*)
Paramètres: *par1* : les lignes destiné à être imprimées
Description: Cette fonction est utilisée pour faire imprimer les lignes à partir de la présentation du rapport, ou pour instaurer les commandes d'impression qui seront exécutées lors du saut de page et lors de l'impression des lignes de total. Vous pouvez utiliser la syntaxe suivante :

Fonction	**Description**
PRINT(1-10)	Imprimer les lignes 1 à 10
PRINT(1,+2,2)	Imprimer d'abord la ligne 1, puis 2 lignes blanches et enfin la ligne 2
PRINT(1,:60,2)	Imprimer la ligne 1, sauter vers la ligne 60 et imprimer la ligne 2
PRINT(:1003,1,3)	Sauter vers la ligne 3 avant le bas de page et imprimer les lignes 1 et 3
PRINT(1-10,:1,20)	Imprimer les lignes 1 et 10,Sauter une page et imprimer la ligne 20
PRINT(*H)	Les lignes définies en utilisant H= seront imprimées

Cette fonction sera également utilisée pour instaurer les commandes d'impression qui indiquent par exemple les lignes qui seront imprimées lors du saut de page :

Fonction	**Description**
PRINT(H=1-4)	When new page, print lines 1 to 4
PRINT(L=8)	The lines printed for each record in the main file is print line 8
PRINT(T=10)	The total line is line 10 (Applies also for grand total)
PRINT(D=9)	As heading for the Detaillines(READH) line 9 is printed
PRINT(B=:1002,17)	As Bottom on every page line 17 is printed
PRINT(N=3,:1,1-4)	Newpage 3 lines befor pageend, heading line 1-4
PRINT(A=10)	Line 10 is printed before a totalblock
PRINT(C=11)	Line 11 if printed after a totalblock

Il est à noter qu'un champ de texte peut être utilisé en tant que paramètre pour la commande d'impression comme par exemple :

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

PRINT(>2) active l'imprimante 2, Voir PRINT.
Valeur retour: None.
Voir aussi: PAGE , PRINT
Exemple: PRINT(:60,1-10) */* sauter vers la ligne 60 et imprimer les lignes 1 à 10*

8.3.1. **PRINT** - Contrôle de l'impression (RAP.)

PRINT(texte *par1*)
Paramètres: *par1* : option=valeur
Description: La commande PRINT est élargie pour inclure la syntaxe PRINT(xx=valeur yy), où les valeurs xx, et yy peuvent être :

	Fonction	**Description**
xx=	ml	Left margin
	mr	Right margin
	mt	Top margin
	mb	Bottom margin
	eh	Empty line height
	ce	Close report windows on exit
	fh	Standard font height for all lines
	cd	Close printer document and start new
yy=	cm	Centimetre
	in	Inches
	pt	Points
	<none>	Device pixels

8.3.2. **PRINT(?=** Lecture du montage d'imprimante (RAP.)

PRINT(?=texte *par1*)

Description: L'élargissement de la commande PRINT d'une fonction qui permet de lire les informations sur l'imprimante.
La valeur retournée yy sera notée en tant que points à moins que xx soit égal à 5, 8, 9, 15 ou 16.

8.4. PRINT(LAB= - Fonction d'étiquette (RAP)

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

par6 : Copies

Description: La hauteur et la largeur d'une étiquette sur la feuille peuvent être données en centimètres ou en pouce à l'aide de la syntaxe suivante :

```
7cm equals 7 centimetres
2in equals 2 inches
```

Dans l'exemple montré ci-dessous, les étiquettes sont imprimées de gauche à droite sur une feuille d'étiquette contenant 21 étiquettes, c'est à dire 3 sur chaque ligne et 7 lignes sur lesquels chaque étiquette mesure 7*7 centimètres. Chaque étiquette sera imprimée en deux copies.

Valeur retour: Aucune.

Voir aussi: PRINT

Exemple:

```
FIRST
PRINT(LAB=1,3,7,7cm,7cm,2) /* Définit l'impression d'étiquette
NORMAL
```

8.5. PRINTER- Sélection d'imprimante (RAP.)

PRINTER(Printer *par1*)
Paramètres: *par1* : Numéro d'imprimante
Description: Cette fonction est utilisée en liaison avec la sélection d'imprimante lors du démarrage. Pour instaurer l'imprimante standard pour un rapport vous pouvez insérer la ligne de calcul suivante :
Valeur retour: None.
Voir aussi: COPIES, PRINT
Exemple: PRINT(7) */* Imprimante standard pour ce rapport est l'imprimante 7*

8.5.1. PRINTER - Sortie sur des imprimantes multiples (RAP)

PRINTER(nombre *par1*, Imprimante *par2*)
Paramètres: *par1* : Numéro d'imprimante *par2* : Imprimante ID
Description: L'imprimante(2,7) ouvra l'autre imprimante définie en tant que numéro d'imprimante 7 dans la mise en place d'imprimante. Aucune sortie imprimante sera imprimée avant que :

PRINT(>2)

soit trouvé dans les calculs, et toutes les sorties imprimante. PRINT(>1) active l'imprimante standard.
Chaque imprimante a ses propres numéros de page avec de différentes tailles de papier. Vous pouvez utiliser 30 imprimantes au maximum (ou copies) simultanément.
Valeur retour: Aucune.
Voir aussi: COPIES, PRINT
Exemple: PRINTER(2,7) */* Ouvrer l'imprimante secondaire 7*

8.6. PRTTOTAL - Contrôle manuelle de l'impression de total (RAP)

PRTTOTAL(Niveau *par1*)
Paramètres: *par1* : Numéro de niveau de total
Description: Normalement, RAPGEN produit un sous-total quand une partie de la clé de tri change valeur. A l'aide de PRTTOTAL, vous pouvez commander de manière manuelle toutes l'impression des sous-totaux et imprimer ces derniers lorsque un champ change valeur.
Valeur retour: Aucune.
Voir aussi: ENDSUM
Exemple:

```
IF #7=1 PRTTOTAL(1)             /* Print subtotal if field 7 is 1
LAST
PRTTOTAL(2)                     /* Print grandetotal last
```

8.7. SCRPRT - Appel à la sortie imprimante à écran sauvegardée (IQ)

SCRPRT(Filename *par1*)
Paramètres: *Par1*: Nom de fichier contenant la sortie à l'écran sauvegardée.
Description: SCRPRT("nom de fichier") appellera à l'imprimante à écran et affichera la sortie sur l'imprimante sauvegardée dans le fichier indiqué. Ceci peut être intégré à un programme IQ en cliquant sur un champ.
Valeur retour: Aucune.
Voir aussi: PRINT
Exemple:

```
SCRPRT("c:/w/ab.cde")     /* Afficher le contenu de ce fichier avec l'imprimante
écran.
```

9. Lecture des fichiers

Dans ce chapitre nous allons décrire la fonction READ pour la lecture à partir d'un fichier secondaire et les fonctions START/NEXT/REPEAT pour la lecture des plusieurs enregistrements. Les principes fondamentaux pour les connexions de fichier sont décrits dans le manuel d'utilisation de RAPGEN dans la section consacrée à l'utilisation des fichiers multiples.

9.1. READ - Lecture d'un enregistrement à partir d'un fichier

nombre READ(file *par1*, index *par2*) ,connexion *par3*

par3 : Connexion optionnelle si la connexion standard n'est pas présente
Description: Cette fonction peut lire un enregistrement à partir d'un fichier.
READ(le) lira le fichier à l'aide de la connexion standard définie dans le dictionnaire de données.
READ(le),#9 lira le fichier le avec le champ 9 en tant que clé pour l'index 1, même s'il n'existe pas une connexion standard.
READ(va.02),#6 lira le fichier va avec le champ 6 en tant que clé pour l'index 2, même s'il n'existe pas une connexion standard.
READ(le),"1",#9(3,4),#7 créera une clé sous la forme d'une combinaison du constant "1" le champ 9 caractères 3-4 et le champ 7.
READ(le.00),#6 lira le fichier le avec le numéro d'enregistrement (index 0) comme indiqué dans le champ 6.
Valeur retour: 0 si l'enregistrement est lu.
Voir aussi: START, NEXT, REPEAT, END , PRIOR, READR, READX
Exemple: READ(le) */* Lire le fournisseur*

9.2. READH - Lecture d'un enregistrement et l'impression optionnelle de l'en-tête

nombre READH(file *par1*, index *par2*) ,connexion *par3*

par3 : Connexion optionnelle qui n'est pas présente
Description:
Cette fonction lira un enregistrement à partir d'un fichier selon le même procédé que READ. Si un autre enregistrement est lu après que READH a été utilisé la dernière fois, par exemple la modification du numéro de fournisseur, l'en-tête indiqué pour READH sera imprimée
Valeur retour: 0 si l'enregistrement est lu.
Voir aussi: READ
Exemple: READH(le) */* Lire le fournisseur avec l'en-tête optionnelle*

9.3. READR - Lecture d'un enregistrement en utilisant un numéro d'enregistrement déterminé

par2 : connexion optionnelle si la connexion standard n'est pas présente
Description: Cette fonction lira un enregistrement à partir du fichier avec l'indication du numéro d'enregistrement. READR peut être utilisée sur les bases de données travaillant avec des numéros d'enregistrement et elle est seulement incorporée en raison de la comptabilité des anciennes versions.
READ(le.00),#6 est le même que READR(le),#6
Voir aussi: READ , READX

9.4. READX - Lecture d'un enregistrement en indiquant le numéro d'enregistrement relatif.

nombre READX(file *par1*) ,connexion *par2*

par2 : connexion optionnelle si la connexion standard n'est pas présente
Description: Cette fonction lira un enregistrement à partir d'un fichier en indiquant le numéro d'enregistrement relatif en tant que clé. READX peut seulement être utilisée sur une base de données qui travail avec des numéros d'enregistrement. Elle est seulement apportée à ce système en raison de la comptabilité des anciennes versions.
READ(le.00),#6+N est le même que READX(le),#6
Voir aussi: READ , READR

9.5. START - Instaurer l'index et l'intervalle pour un fichier

nombre START(fichier *par1*, index *par2*) ,connexion *par3*

par3 : connexion optionnelle si la connexion standard n'est pas présente.
Description: Cette fonction prépare la lecture avec la fonction END en définissant l'intervalle des clés à utiliser.
La connexion standard entre les fichiers peut être utilisée, ou vous pouvez indiquer une clé individuelle comme décrit sous READ.
Lors du démarrage, vous devez seulement spécifier une partie de ces clés. Les lectures suivantes avec END trouvent tous les enregistrements dans lesquels la première partie de la clé corresponde à la partie indiquée au DEMARRAGE.
Valeur retour: retournera 0 si l'intervalle OK.
Voir aussi: READ, NEXT, REPEAT, END , PRIOR
Exemple:

```
#47=0                    /* Zero total field
START(va)                /* Start reading of articles
NEXT(va)                 /* Read next article
#47=#47+va#3             /* Totalize all articles
REPEAT(va)               /* Continue until end of range
```

9.6. NEXT - Rechercher l'enregistrement dans l'intervalle

nombre NEXT(fichier *par1*)
Paramètres: *par1* : abréviation du fichier
Description: Cette fonction est utilisé en liaison avec les boucles START/NEXT/REPEAT Les fonctions START() et END() instaurent l'intervalle du boucle voulu. START lit un enregistrement à partir du fichier. Une fois la ligne de calcul REPEAT() effectuée, la fonction START() sera exécutée encore une fois jusqu'à ce que qu'il ne se trouve plus d'enregistrements dans l'intervalle.
Valeur retour: Retournera 0 aussi longtemps qu'il se trouve des enregistrements dans l'intervalle.
Voir aussi: READ, START, REPEAT, END , PRIOR
Exemple:

```
PRINT                   /* Take over complete print control
PRINT(4,6,5)            /* Print supplier heading
START(va)               /* Start reading of articles
NEXT(va)                /* Read next article
PRINT(7)                /* Print all articles
REPEAT(va)              /* Continue until end of range
```

9.7. REPEAT - Répéter la lecture START

nombre REPEAT(file *par1*)
Paramètres: *par1* : abréviation du fichier
Description: Cette fonction est utilisée en liaison avec les boucles START/NEXT/REPEAT. A l'aide des fonctions START() et TERMINER() l'intervalle voulu du boucle sera inséré. Une fois l'intervalle inséré, la fonction START() lira un enregistrement dans le fichier. Quand la ligne de calcul REPEAT() est effectuée, la fonction START() est exécutée encore une fois jusqu'à ce qu'il n'aie plus d'enregistrements dans l'intervalle. **Valeur retour:** None
Voir aussi: START, NEXT , PRIOR
Exemple:

```
#47=0                    /* Zero total field
START(va)                /* Start reading of articles
NEXT(va)                 /* Read next article
#47=#47+va#3             /* Totalize all articles
REPEAT(va)               /* Continue until end of range
```

9.8. GETKEY - Rechercher la clé actuelle

texte GETKEY(fileid *par1*)
Paramètres: *par1* : Fichierid
Description: #20=GETKEY(va) recherchera la clé d'index de l'enregistrement lu dernièrement dans le fichier va. Cette fonction est utilisée pour les systèmes de base de données dans lesquels la clé ne doit pas enregistrée en tant que champ.
Valeur retour: Valeur de clé sous la forme de texte.
Voir aussi:
Exemple: #20 = GETKEY(va)

9.9. END - Insérer l'intervalle de fin après START

nombre END(file *par1*) ,connexion *par2*

par2 : Finir la spécification de l'intervalle
Description: La fonction START définit la clé au début et à la fin d'un intervalle de telle façon que les enregistrements avec le même numéro de débiteur soient lus. Normalement, vous ne devez pas utiliser END, seulement si vous souhaitez un intervalle spécial.
Valeur retour: Retournera 0 si l'intervalle OK.
Voir aussi: READ, START, REPEAT, NEXT , PRIOR
Exemple:

```
UPDATE(1)                  /* Empty workfile before use
START(xx),"0000"           /* Start reading from the very first
END(xx),"9999"             /* And go until the last one
NEXT(xx)                   /* Read next record
DELETE(xx)                 /* Delete all records
REPEAT(xx)                 /* Continue until file is empty
```

9.10. PRIOR - Rechercher enregistrement précédent d'un l'intervalle

nombre PRIOR(file *par1*)
Paramètres: *par1* : abréviation du fichier
Description: PRIOR fonctionne selon le même procédé que NEXT. Cependant, elle ne recherchera pas l'enregistrement suivant mais l'enregistrement précédent. Note : Ce ne sont pas toutes les bases de données qui supporteront la lecture effectuée dans l'ordre 'inverse'.
Valeur retour: retournera 0 aussi longtemps qu'il se trouve des enregistrements dans l'intervalle.
Voir aussi: READ, START, REPEAT, NEXT , END
Exemple:

```
PRINT                       /* Take over complete print control
#47=0                       /* Zero counter
START(va)                   /* Start reading of articles
PRIOR(va)                   /* Read prior article
#47=#47+1                   /* Count the articles
IF #47=1 PRINT(4,6,5)       /* Print supplier heading first time
PRINT(7)                    /* Print all articles in reverse order
REPEAT(va)                  /* Continue until end of range
IF #47>0 PRINT(7)           /* Print trailer if any articles
```

9.11. SPEED- Optimisation de la stratégie de lecture

SPEED()
Paramètres: Aucunes
Description: La fonction SPEED() peut optimiser la stratégie de lecture d'un rapport de telle façon qu'un enregistrement avec la même clé déjà utilisé ne soit pas lue encore une fois. Elle sera pris à partir du tampon interne de la dernière lecture. Cependant il faut être prudent s'il s'agit d'un rapport avec la mise à jour des fichiers.
Valeur retour: Aucune.
Voir aussi: READ
Exemple: SPEED() */* Optimiser READ sur un rapport*

10. Mode d'écriture dans les fichiers

Dans le chapitre suivant nous allons décrire les différents modes d'écrire dans les fichiers. Afin de pouvoir utiliser ces fonctions, le système installé doit posséder la mise à jour des fichier, la base de données utilisée doit posséder des fonctions pour la mise à jour des fichiers et l'utilisateur doit avoir une permission pour pouvoir utilisé le serveur.
Il faut toujours tester un programme qui met à jour des fichiers. L'utilisateur est rendu responsable :

the total responcibility of the user

de ce que la mise à jour soit testée et exécutée correctement.

10.1. UPDATE - Permission de la mise à jour des fichiers

nombre UPDATE(nombre *par1*, fields *par2*)

par2 :Fichier optionnel et champs permis.
Description: Il faut que UPDATE(1) soit placé dans le rapport de mise à jour avant d'utiliser les fonctions d'écriture pour l'activation de celles-ci.
L'élargissement de la fonction Update comprend la spécification des champs à mettre à jours.

```
UPDATE(1,"va#6")    /* causes the program to update field 6 in va only.
UPDATE(1,"le#3-4") /* when more files are involved each file must be separate
UPDATE(0)           /* can now be used in DATAMASTER to switch all update off
```

Valeur retour: None.
Voir aussi: DELETE, INSERT, REWRITE, WRITE, NOPAS
Exemple:

```
UPDATE(1)               /* the report updates
NOPAS()                 /* no password
#6=#6+10                /* Do the field modifications
REWRITE(le)             /* update the supplier in the file
```

10.2. REWRITE - Mettre á jour un enregistrement existant dans le ficher

nombre REWRITE(file *par1*)
Paramètres: *par1* : abréviation du fichier
Description: Cette fonction mettra à jour un enregistrement dans le fichier indiqué. Il faut que celui-ci a été lu. Les champs d'index peuvent seulement être modifiés si le système de base de données supporte cela. Il faut que le calculs UPDATE(1) soit exécuté afin de pouvoir activer cette fonction.
Valeur retour: 0 si l'enregistrement a été mis à jour.
Voir aussi: DELETE, INSERT, WRITE, NOPAS, UPDATE
Exemple:

```
UPDATE(1)               /* the report updates
NOPAS()                 /* no password
AFTER                   /* JUST AFTER SELECTIONS DONE
#6=#6+10                /* Do the field modifications
REWRITE(le)             /* update the supplier in the file
```

10.3. INSERT - Insérer un nouveau enregistrement dans le fichier

nombre INSERT(file *par1*)
Paramètres: *par1* : abréviation du fichier
Description: Cette fonction insère un nouveau enregistrement dans un fichier TOUS les champs dans le fichier doit posséder une valeur avant de pouvoir exécuter INSERT. Il faut que le calcul UPDATE(1) soit exécuté afin de pouvoir activer cette fonction.
Valeur retour: 0 si l'enregistrement a été inséré
Voir aussi: DELETE, REWRITE, WRITE, NOPAS, UPDATE, CLEAR, LET
Exemple:

```
UPDATE(1)               /* the report updates
NOPAS()                 /* no password
CLEAR(le)               /* zero all fields in supplier record
LET("le#1,3=#7,17")     /* fill the fields
INSERT(le)              /* insert new supplier in supplier file
```

10.4. DELETE - Supprimer un enregistrement dans un fichier

nombre DELETE(file *par1*)
Paramètres: *par1* : abréviation du fichier
Description: Cette fonction supprimera un enregistrement de manière physique dans le fichier voulu. Il faut que celui-ci soit lu avant l'exécution de la suppression. Il faut que la ligne de calcul UPDATE(1) soit exécutée afin de pouvoir activer cette fonction.
Valeur retour: 0 si l'enregistrement a été supprimé
Voir aussi: INSERT, REWRITE, WRITE, NOPAS, UPDATE
Exemple:

```
UPDATE(1)               /* the report updates
NOPAS()                 /* no password
AFTER                   /* JUST AFTER SELECTIONS DONE
DELETE(va)              /* the selected articles are removed
```

10.5. WRITE - Mettre à jour ou insérer un enregistrement dans le fichier

nombre WRITE(file *par1*)
Paramètres: *par1* : abréviation du fichier
Description: Cette fonction met à jour ou insère un enregistrement dans un fichier indiqué. Si le dernier READ du fichier a trouvé un enregistrement, cette fonction fonctionne selon le même procédé que REWRITE. Si aucun enregistrement n'est trouvé à l'aide de READ, celle-ci fonctionnera en tant que INSERT. Il faut que la ligne de calcul UPDATE(1) soit exécutée afin de pouvoir activer cette fonction.
Valeur retour: 0 si l'enregistrement est mis à jour ou inséré
Voir aussi: INSERT, REWRITE, DELETE, NOPAS, UPDATE
Exemple:

```
UPDATE(1)               /* the report updates
NOPAS()                 /* no password
READ(le),#6             /* Read supplier for this article
IF #OK THEN BEGIN       /* If supplier not present
le#1=#6                 /* Set supplier nombre
le#2="I made this"      /* and name
END
le#6=le#6+#3            /* Update supplier fields
WRITE(le)               /* insert or update supplier record
```

11. Exportation et Importation des fichiers externes

Dans le chapitre suivant nous allons voir les fonctions pour entrer et sortir de données dans les fichiers texte transmises aux autres sytèmes. .

11.1. **EXPORT** - Exportation des données vers un fichier texte

nombre EXPORT(champs *par1*, nom de fichier *par2*, texte *par3*, texte *par4**6, texte *par5*, texte *par6**6)

Description: La fonction EXPORT exportera les données vers un fichier texte. Cette fonction peut être utilisée pour transmettre les données entres les systèmes, les feuilles de calcul électronique et les systèmes du traitement de texte.
Il faut que les champs indiqués dans *par1* soient entrés en tant que texte, par exemple. "#1-99" (entre guillemets).
Le nom de fichier dans *par2* si un chemin n'est pas indiqué, le fichier sera placé dans TMP Si aucune extension n'est indiquée il sera .OUT. Si aucun nom de fichier n'est indiqué , le nom du rapport sera utilisé, par exemple c:/tmp/DM1007.OUT pour le numéro de rapport 7.
Avec *par3* and *par5*vous pouvez contrôler la longueur enregistrement et le séparateur de ligne du fichier
par4 est seulement utilisé pour les fichiers avec une taille d'enregistrement fixe et pour les transferts entre les systèmes de macroordinateur.
par6 consiste en 6 caractères utilisés pour contrôler la présentation d'un fichier séparé par une virgule. Note : " Ce bande doit être inscrit en tant que deux caractères \". Normalement, tous les champs alphanumériques sont inscrits en tant que "xxxx", où les caractères qui apparaissent en tant que " (guillemets) seront converti en ' (un guillemet). Les champs numériques seront inscrits en tant que 99.99, où . (point) sera le point décimal. Tous les champs seront séparés par une, (virgule).
Vous pouvez fermer le fichier d'exportation en utilisant EXPORT("CLOSE"). Ceci sera nécessaire si vous souhaitez voir le fichier créé en exécutant CHAIN au notepad.

Valeur retour: Aucune.
Voir aussi: IMPORT
Exemple:

```
AFTER                          /* AFTER selections
EXPORT("LE#1-99","le.csv")     /* all fields are exported (CSV)
EXPORT("#1-6","le.csv","","","","--,\"'.")  /* Same as above
```

Dans l'exemple montré ci-dessus, le fichier le.csv contiendra les lignes suivantes:

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

Exemple:

```
EXPORT("#1-6","le.ssv","000001","","","--;. ,")  /* all fields exported (SSV)
```

Dans l'exemple montré ci-dessus, le fichier le.ssv contiendra les lignes suivantes :

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

Exemple:

```
EXPORT("#1-2,5-6","a","-80","1")  /* fixed fieldlength and no crlf
```

11.2. IMPORT - Importation des données à partir d'un fichier texte (RAP)

IMPORT(champs *par1*, nom de fichier *par2*, texte *par3*, texte *par4**6, texte *par5*, texte *par6**6)

Description: Cette fonction entre toutes les données à partir d'un fichier texte
Il faut que les champs indiqués dans *par1* soient entrés dans "" (entre guillemet). Il sera possible d'indiquer des calculs simples en liaison avec les champs, par exemple IMPORT("#1-5,+6")

Operator	Fonction
+	Add up these fields
-	Subtract from these fields
&	Skip these fields
=	Set fields equal
:xx	Skip to position xx in the record

Le nom de fichier physique indiqué dans *par2* peut contenir un chemin, par exemple "c:\\export\\le.csv".
Valeur retour: Aucune.
Voir aussi: EXPORT
Exemple:

```
                              /* the reports mainfile is le (supplier file)
   UPDATE(1)                  /* the report updates
   NOPAS()                    /* no password
   IMPORT("#1-6","le.csv")    /* read one record from textefile
   READ(LE),#1                /* Check if supplier LE present
   LE#6=LE#6+#6               /* Add amount read to old balance
   IF #OK LET("LE#1-6=#1-6")  /* If new supplier move all fields
   WRITE(LE)                  /* Modify/Create LE supplier
```

Exemple:

```
   IMPORT("#1-6","le.ssv","","","","--;- -")  /* import fra (SSV) tekstfil
```

11.2.1. **IMPOCONT** - Importations en continu (RAP)

IMPOCONT(champ *par1*)
Paramètres: *par1* : indiquera les champs à insérer lors de la lecture des données à partir du fichier texte.
Description: IMPOCONT continue l'importation des champs supplémentaires du même fichier, comme dernière IMPORT et à partir de l'endroit où celui-ci a été terminé. Elle peut être utilisée quand un type enregistrement placé à l'endroit précédent de l'enregistrement entraîne l'importation des différents champs.
Voir aussi: IMPORT, IMPONEXT, IMPOTHIS

11.2.2. **IMPONEXT** - Importation de l'enregistrement suivant (RAP)

IMPONEXT(fields *par1*)
Paramètres: *par1* : ces champs insérés lors de la lecture des données à partir du fichier texte.
Description: IMPONEXT lit l'enregistrement suivant et entre des champs à partir de cet endroit. Elle est utilisée , par exemple, quand un champ indique qu'un ou plusieurs enregistrements suit avec des informations relatives pour cet enregistrement principal.
Voir aussi: IMPORT, IMPOCONT, IMPOTHIS

11.2.3. **IMPOTHIS** - Importation encore une fois de cet enregistrement (RAP)

IMPOTHIS(fields *par1*)
Paramètres: *par1* : les champs qui doivent être inscrits lors de la lecture des données à partir du fichier texte.
Description: La fonction IMPOTHIS importera l'enregistrement actuel encore une fois. Elle est utilisée quand un type d'enregistrement placé à l'endroit précédent de l'enregistrement indique un contenu de champ différent. **Voir aussi:** IMPORT, IMPOCONT, IMPONEXT

11.3. FTP - File Transfer Processor Transfert de fichier

nombre FTP(Nombre *par1*, Texte *par2*)

Par2: FTP command
Description: La fonction FTP a été créée pour que les utilisateurs avec des connaissance de programmation puissent transférer des fichiers, par exemple, un rapport basé sur un fichier de SSV contenant les noms de fichier. Pour acquérir les commandes de FTP, vous devez vous reporter au manuel de FTP. Il est à noter que les champs libres peuvent être utilisés en tant que paramètres pour la commande.La version de 32 bits de TRIO supportera des noms de fichiers longs.
L'exemple présenté ci-dessus montre les transferts d'un fichier entre un système Quattro à l'aide de la commande spécifique de Quattro pour le transfert du fichier complet avec le bloc d'en-tête y compris XQUAT qui supprimera les informations supplémentaires transférées à partir du serveur SSQ au Quattro de chaque bloc.
Valeur retour: Pour OPEN: FTP ident, tout autre: FTP code d'erreur, 0=OK
Exemple:

```
#10=FTP(0,"open 200.0.0.9")          /* Connect to Server
#11=FTP(#10,"user cms mypas")        /* Log in as user cms password mypas
#11=FTP(#10,"binary")                /* Switch on binary transfer
#11=FTP(#10,"quattro")               /* Switch on Quattro backup mode
#11=FTP(#10,"get /X.BASIC/0/AFIL c:/mydir/myfil") /* Get the file
if #11<>0 FTP(#10,"error")           /* Display error message
#11=FTP(#10,"xquat c:/mydir/myfil")  /* Convert from Quattro
#11=FTP(#10,"quit")                  /* Thats it
```

12. Plusieurs entreprises et une mélange des fichiers

Les fonctions présentées ci-dessus sont utilisées pour l'exécution des plusieurs entreprises où pour que les données des entreprises soient placées dans sa propre base de données ou sa propre table. De plus, elles sont utilisées pour fusionner des différents fichiers contenant la même définition.

12.1. **ACCESS**- Contrôler l'existence d'un fichier (IQ)

nombre ACCESS(Nom de fichier *par1*)
Paramètres: *par1* : nom de fichier
Description: Contrôler l'existence d'un fichier, retourner 0 si le fichier est trouvé.
Valeur retour: 0 si le fichier est trouvé.
Voir aussi: OPEN
Exemple: IF ACCESS("monfichier.ssv")=0 MESS("Ok ? ")

12.2. **COMNO** - Entreprise id

texte COMNO(Fichierid *par1*)
Paramètres: *par1* :blanc ou fichierid
Description: Cette fonction retournera le id d'entreprise actuel pour un fichier indiqué, et pour le fichier principal si rien n'est indiqué.
Valeur retour: Entreprise id.
Voir aussi: OPCOM
Exemple: #1 = OPCOM() */* Rechercher id d'entreprise actuelle, par exemple. "001"*

12.3. **ENDSUM** - Grand total supplémentaire lors de l'exécution des plusieurs fichiers principaux

ENDSUM()
Paramètres: None
Description: Dans un rapport avec plusieurs listes indépendantes en raison de l'utilisation des fonctions MERGE ou de OPCOM contenant des totaux pour chaque liste, vous pouvez obtenir un total supplémentaire des tous les enregistrements imprimés en inscrivant une ligne de calcul ENDSUM().
Les champs de système #CO and #CN seront imprimés en tant que ***à ENDSUM .
Valeur retour: Aucune
Voir aussi: KEYS, MERGE, OPCOM
Exemple: ENDSUM() */* Imprimer un total supplémentaire à la fin*

12.4. **FILENAME** - Nom de fichier actuel d'un fichier

texte FILENAME(fichier *par1*)
Paramètres: *par1* : abréviation de fichier
Description: Cette fonction retournera le nom du fichier ouvert actuellement avec le id indiqué.
Valeur retour: Nom de fichier physique.
Voir aussi: OPEN
Exemple: #1 = FILENAME(va) /* *donnera "c:/rapfil/ssv/isa/va.ssv"*

12.5. OPEN - Ouverture d'un fichier avec un nom spécifique

nombre OPEN(fichier *par1*, nom de fichier *par2*, driver *par3*)

par3 : 0 ou numéro de base de données de l'interface
Description: Avec cette fonction, vous pouvez ouvrir un fichier spécifique au lieu du fichier déjà ouvert pour le fichier id donné. L'ancien fichier sera fermé.
Une message d'erreur apparaîtra à l'écran si le fichier n'existe pas ou ne peut pas être ouvert.
En mettant *par3*, le fichier sera ouvert en tant que type de base de données défini dans BASIS.SSV lors de l'installation de driver de la base de données.
Valeur retour: 0=ok, <>0=erreur.
Voir aussi: ACCESS, FILENAME, MERGE, OPCOM
Exemple:

```
FIRST
OPEN(va,"c:/swtools/demo/va.ssv")  /* Use this specific file

OPEN(va,#50)                       /* Enter filename by start report
```

12.5.1. OPEN - Fermeture temporaire des fichiers

OPEN(fichierid *par1*, Constant *par2*)

par2 : "-"
Description: Vous pouvez fermer un fichier de telle façon que tous les programmes ENCHAINES puissent avoir accès aux fichiers NOTE : Il ne faut pas fermer le fichier PRINCIPAL selon ce procédé.
Valeur retour: 0=ok, <>0=erreur.
Voir aussi: FILENAME, MERGE, OPCOM
Exemple:

```
OPEN("ku","-")          /* will temporary close file to allow:
CHAIN("command.com /c edit c:\\windows\\system\\ku.ssv")

OPEN("ku","+")          /* will reopen file again
```

12.6. MERGE - Mélange des plusieurs fichiers principaux dans un rapport (RAP)

nombre MERGE(fichierid *par1*, Nom de fichier *par2*, Driver *par3*)

par3 : 0 or database interface nombre

Description: A l'aide de cette fonction vous pouvez mélanger plusieurs fichier dans un rapport. Pour la routine MERGE, vous pouvez indiquer un fichierid dans *par1* si le fichier est définit en tant qu'un fichier séparé ou en tant qu'un nom de fichier direct dans *par2* comme dans OPEN. Il faut que les fichiers utilisés aient la même structure.
Une message d'erreur apparaîtra si le fichier n'existe pas ou ne peut pas être ouvert.
Si *par3* est indiqué, le fichier sera ouvert comme ce type de base de données défini dans BASIS.SSV lors de l'installation de driver de la base de données.
Il faut qu'un rapport qui utilise MERGE soit trié afin d'obtenir une mélange des fichiers, par exemple, en suivant un ordre du numéro d'article. Si vous utilisez MERGE sans faire le tri, vous obtiendrez d'abord une liste à partir du fichier principal et normal et puis une liste à partir de chaque fichier mélangé. La fonction ENDSUM peut être utilisée pour obtenir un total des tous les enregistrements imprimés.
Si vous faisiez appel à MERGE sans des paramètres, un MERGENUMERO sera retourné qui sera 1 pour le fichier principal, 2 pour la première mélange, et 3 pour la suivante, etc.

Sans des paramètres: MERGENUMERO à partir de 1 et les suivants.

Voir aussi: ENDSUM, OPCOM, OPEN

Exemple:

```
MERGE(0,"c:/swtools/demo/va.ssv")   /* Merge this file
MERGE(le)                           /* And the le file
#12=MERGE()                         /* Get mergenombre 1,2 or 3
```

12.7. OPCOM - Ouvrir des fichiers dans les différentes entreprises

nombre OPCOM()

par3 : 0 ou numéro d'interface de la base de données
Description:
La fonction OPCOM vous permettra d'avoir l'accès aux différentes entreprises dans un rapport.
Vous pouvez exécuter un rapport une fois pour chaque entreprise, en mettant OPCOM("111,777-888") ou OPCOM(#50) où le champ 50 sera le champ d'entrée de début. Ensuite, ce rapport peut être construit avec un total pour toutes les entreprises avec ENDSUM ou vous pouvez le trier par exemple pour collectionner des informations sur un article dans toutes les entreprises.
Les champs de système #CO and #CN peuvent être utilisés pour imprimer le id d'entreprise et le nom d'entreprise dans l'en-tête. .
Une liste d'article définie sur le fichier va peut rechercher des informations à partir du fichier d'article d'une autre entreprise en mettant d'abord OPCOM(VA,"555") suivi par READ(VA) de telle façon que va#8 contienne le stock de l'entreprise actuelle. C'est à dire que, par exemple VA#8 sera le stock de l'entreprise 555.
Un rapport statistique ou chaque enregistrement dans le fichier statistique contient un numéro d'entreprise, peut ouvrir les fichiers d'entreprise à l'aide de OPCOM(0,#47)
Si *par3*est indiqué, le fichier sera ouvert avec le type d'interface de la base de données définie dans BASIS.SSV lors de l'installation de driver de la base de données.
Si OPCOM est utilisé sans paramètres, le numéro d'entreprise actuel sera retourné.
A partir du fichier COMPANY.SSV, vous pouvez rechercher les noms d'entreprise ainsi que les numéros permis, si vous avez indiqué l'intervalle dans le id d'entreprise*par2*

Sans paramètres : entrepriseid.
Voir aussi: COMNO, ENDSUM, MERGE, OPEN
Exemple:

```
OPCOM("001,777-888")          /* Run the report with these companies
OPCOM("*")                    /* Run the report with all companies
OPCOM(va,"123")               /* Use the article file company 123
OPCOM(0,"777")                /* Company 777 for all other but mainfile
OPCOM(-1,"888")               /* Use company 888 for all files

OPCOM(#50)                    /* Enter companies by start
```

13. Fonctions d'IQ et de DATAMASTER

Ces fonctions sont uniquement construites pour DATAMASTER de telle façon que vous ne puissiez pas les utiliser dans des rapports. Cependant, vous pouvez utiliser une partie de ses fonctions dans IQ. Ceci sera indiqué pour chaque fonction.

13.1. **DISABLE**- Désactiver l'entrée d'un programme (IQ)

DISABLE(programno *par1*)
Paramètres: *par1* : numéro de programme qui doit être désactivé.
Description: Désactiver toutes les entrées d'un numéro de programme voulu.
Valeur retour: None.
Voir aussi: ENABLE , FOCUS
Exemple: DISABLE(20)

13.2. **DISP** - Affichage des champs modifiés (IQ)

DISP(fields *par1*)
Paramètres: *par1* : "" ou champs
Description: Vous utiliserez DISP si vous sautez pendant un calcul d'un champ vers d'autres champ qui sont affichés à l'écran. Si DISP n'est pas présent, il ne sera pas certain que la valeur calculée soit réellement affichée.
La commande DISP() qui affichera à l'écran tous les champs encore une fois vous permettra également d'indiquer l'affichage à l'écran uniquement des champs sélectionnés comme DISP("#1,4")
Valeur retour: Aucune.
Voir aussi:
Exemple: DISP()

13.3. DOFONCTION - Exécuter la fonction externe (IQ)

DOFONCTION(Fonction *par1*, texte *par2*,numéro de programme *par3*)

par3 : numéro de programme optionnel
Description: DOFONCTION communique la message <no.fonction> au programme d'IQ en cours d'exécution ou au <programme> ouvert. Une clé peut être communiquée aux fonctions READ.
Une liste des numéros de fonctions peut être trouvée dans la boîte de liste de calcul pour 'Calculs en sélectionnant une fonction'.
Valeur retour: Aucune.
Voir aussi: CHAIN, PLSNEXT, TRANSMIT
Exemple:

```
DOFONCTION(505,#1,20) /* ask program 20 to read a record using key #1
DOFONCTION(550)       /* Zooms the current screen
```

13.4. **ENABLE**- Activer l'entrée d'un programme (IQ)

ENABLE(numéro de programme *par1*)
Paramètres: *par1* : Numéro de programme
Description: Active l'entrée pour un programme selon le numéro de programme donné.
Valeur retour: Aucune.
Voir aussi: DISABLE , FOCUS
Exemple: ENABLE(20) */* Activer le programme 20*

13.5. **FOCUS** - Activer un programme (IQ)

FOCUS(Numéro de programme *par1*)
Paramètres: *par1* : Numéro de programme à activer.
Description: Active l'entrée et focalise sur le numéro de programme indiqué.
Valeur retour: Aucune.
Voir aussi: DISABLE, ENABLE
Exemple: FOCUS(20) */* Le programme 20 deviendra actif*

13.6. FUNC - Mode de mise à jour actuel pour un enregistrement (IQ)

nombre FUNC(fichierid *par1*)
Paramètres: *par1* : Fichierid
Description: Selon l'entrée de données utilisateur, DATAMASTER décidera si une mise à jour d'un enregistrement sera nécessaire et la manière selon laquelle celle-ci sera exécutée. FUNC est utilisé dans la routine d'écriture pour la lecture de cela.
Valeur retour:

Mode	Fonction
0	No update nessesary
1	An existing record should be modified
2	A new record should be inserted
3	An existing record should be deleted

Voir aussi: SETUPD, ON
Exemple:

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

13.7. GETINFO Chercher les informations supplémentaires sur un programme (IQ/DM)

nombre GETINFO(nombre *par1*, texte *par2*)

par2 : Field reference

Description: Cette fonction permet de chercher des informations spécifiques à partir d'un programme IQ/DM. Le type 0 et 1 retournera le id unique de la fenêtre qui peut être utilisé par les autres fonctions lors de la mise à jours. Voir l'exemple dans le manuel OLE.

Il faut faire une référence de champ dans *par2*. pour les type 2 et 5. Pour obtenir la colonne de départ pour le champ 7 du fichier d'article, *par2* doit être égal à "va#7". Les coordinateur pour ce champ seront retournés selon la taille actuelle du champ défini dans IQ/DM.

Si vous souhaitez avoir les coordinateurs actuels affichés à l'écran d'après par exemple zoom in/out, vous devrez utiliser les types 6 et 9.

Type 2-9 retournera un champ de coordinateur. La valeur peut être conservée dans le format de champ 9,T2.

Exemple:

```
GETINFO(0)            /* Get the IQ program window id
GETINFO(2,"va#7");    /* Get the start x coordinate of va field 7
```

13.8. HELP - Afficher la boîte de message contenant l'aide en ligne d'un champ (IQ)

HELP(field *par1*)
Paramètres: *par1* : Référence de champ
Description: HELP("#31") Affichera une boîte de message contenant l'aide en ligne d'un champ actuel
Voir aussi: MESS
Exemple: HELP("#31")

13.9. ISACTIVE - Contrôler si les programmes sont actifs (IQ)

nombre ISACTIVE(Numéro de programme *par1*)
Paramètres: *par1* : Numéro de programme
Description: Tester si <programme> est actif.
Valeur retour: Retournera 1 si <programme>est actif, sinon 0.
Voir aussi: CHAIN, EXIT, WAIT
Exemple: IF ISACTIVE(20)=0 CHAIN(20) */* Démarrer programme 20 dans le cas où celui-ci ne serait pas exécuté*

13.10. KEYON - Sauvegarder ou afficher les champs d'entrée de clé (IQ)

KEYON(nombre *par1*)

Description: KEYON(0) supprimera le champ d'entrée de clé, (1) réactivera ceci.
Valeur retour: Aucune.
Voir aussi:
Exemple: KEYON(0) */* Supprimer le champ d'entrée de clé*

13.11. **LINE** - Rechercher ou insérer le numéro de ligne actuel (IQ/DM)

nombre LINE(nombre *par1*)
Paramètres: *par1* : ou le numéro de ligne
Description: Cette fonction recherchera ou insérera le numéro de ligne actuel dans un programme IQ/DM Ce numéro de ligne est le compteur de ligne dans un programme définit en tant que par exemple **va#1-6l** ou **le#1-6/va#1-6**.
En mettant *par1* est 0, la valeur sera recherchée par le compteur de ligne.
En mettant *par1* égal à -1, cette fonction recherchera le nombre de lignes définies. Si le programme est défini en tant que **va#1-6l,t5** il faut que la valeur return soit **5**
En mettant *par1* supérieur à 0, la fonction insérera le numéro de ligne actuel à *par1*.
Valeur retour: Un numéro de ligne/compteur pour par 1= 0/-1, ou 0.
Exemple:

```
#20=LINE()  /* Rechercher le numéro de ligne actuel
```

13.12. LOOP - Faire appel à une routine pour tous les enregistrements dans le tampon de ligne (IQ)

LOOP(étiquette *par1*)
Paramètres: *par1* : Etiquette(le nom de la routine) qui doit être appelé
Description: Pour chaque enregistrement lu dans un programme de liste et pour chaque transaction dans un programme de transaction, le tampon de ligne interne sera rempli avec les valeurs lues ainsi que le résultat des calculs.(non-global workfields).
Lors de l'écriture dans un tel programme, vous devez utiliser LOOP pour faire appel à une routine d'écriture pour chaque ligne. De plus, vous pouvez utiliser LOOP pour recalculer le SUM.
Valeur retour: Aucune.
Voir aussi: GOSUB, ON
Exemple:

```
LOOP(MAIN)          /* Writing the lines in a LIST-program
LOOP(TRANS)         /* Writing transaction lines
LOOP(SUMIT)         /* Recalculation of transaction SUM
LOOP(TRANSDEF)      /* Change of keyvalue for all transactions
```

13.13. **MENUCH** - Menu Flip et checked flags (IQ)

MENUCH(Numéro de menu *par1*)
Paramètres: *par1* : Numéros de menu
Description: Elle flip des numéros de menu actuels (Voir MENUS) et mettra à jour les drapeaux internes relatifs pour la commande de programme.
Valeur retour: Aucune.
Voir aussi: MENUUPD, MENUS
Exemple: MENUCH("31-32") */* Menu Flip de mise à jour IQ*

13.14. MENUS - Modification des menus (IQ)

MENUS(Numéro de menu *par1*)
Paramètres: *par1* : -xxx= Désactiver , +xxx= Activer les numéros de menu xxx

Menunombre	**Fonction**
1/11	Insert new record in mainfile/transactions
2/12	Amend a record in mainfile/transactions
3/13	Delete a record in mainfile/transactions
4/14	Superindex search on mainfile/transactions
5/15	Selections on mainfile/transactions
6/16	Superindex fielddefinition on mainfile/transactions
20	Search, list must match input
21/22/23/24/25	Transactions, Next/Previus/First/Last/Direction
26	Display key during search
27	Case sensitive search
31/32	Talk/Listen to other programs
41/42/43/44	Mainfile, Next/Previus/First/Last
51/52/53	Calculations/Amend form/Save program
54/55	Parameter menus
61/62/63/64	New program, Delete program, Print program, Start program
100-149	Index locked and index nombre
999	Activate everything

Description: La fonctions MENUS permet désactiver certains points de menu dans les programmes DATAMASTER et IQ
De plus, vous pouvez activer MENUS au début en faisant appel à IQ à partir de Windows et puis en mettant les paramètres -m+xxx or -m-xxx.. Afin de pouvoir modifier les calculs d'un programme quand cette fonction est désactivé, vous devez sélectionner, par exemple, : C:\SWTOOLS\IQWIN -m999
Valeur retour: Aucune.
Voir aussi: MENUCH, MENUUPD
Exemple: MENUS("-51-55") */* Désactiver les modifications exécutées dans ce programme*

13.15. MENUUPD - Additionner et Contrôler menu (IQ)

MENUUPD(Numéro de menu *par1*, nombre *par2*, nombre *par3*)

par3 : Texte
Description: Add / Contrôler menu manuellement.
MENUUPD(1,2000,"My &Own menu") additionnera la fonction 2000 au numéro de menu 1.
En sélectionnant ce nouveau point de menu, les calculs utilisateur avec l'étiquette FU2000: dans les calculs sélectionnant la fonction seront effectués.
Valeur retour: Aucune.
Voir aussi: MENUCH, MENUS
Exemple: MENUUPD(1,2000,"Mon &Own menu") */* Additionner la fonction 2000 au numéro de menu 1.*

13.16. NEXTFLD - Sauter vers le champ d'entrée (IQ)

NEXTFLD(champ *par1*)
Paramètres: *par1* : Numéro de champ pour l'entrée suivant
Description: Vous pouvez utiliser NEXTFLD pour survirer la séquence d'entrée fixe en fonction des calculs.
De plus vous pouvez indiquer le numéro de programme ou le numéro de ligne avec la spécifications de champ.
Valeur retour: Aucune.
Voir aussi: NEXTFLDSEQ, SEQ
Exemple:

```
IF #4<#3 NEXTFLD(#3)
NEXTFLD("#10")       /* sets next input field to field 10.
NEXTFLD("#10.2")     /* jumps to field 10 on line 2
NEXTFLD("5.#10")     /* jumps to program 5 field 10
```

13.17. NEXTFLDSEQ - Sauter vers un champ d'entrée dans la séquence (IQ)

NEXTFLDSEQ(nombre *par1*, nombre *par2*)

par2 : Numéro de champ dans cette séquence
Description: Sauter vers un champ voulu dans la séquence de champ.
Valeur retour: Aucune.
Voir aussi: SEQ , NEXTFLD
Exemple: NEXTFLDSEQ(2,1) */* Sauter vers le premier champ dans la séquence d'entrée 2*

13.18. OBJECTADDSTRING - Additionner une chaîne au objet (IQ)

OBJECTADDSTRING(Champ *par1*, texte *par2*, texte *par3*)

par3 : Texte à utiliser en tant qu'index
Description: Cette fonction insérera un texte dans un objet. Le résultat dépend du type d'objet selon la table présenté ci-dessus.

Object	**Meaning**
BUTTON	The fonction sets the texte displayed for the button
COMBOBOX	The fonction adds a new element to the list
EDITBOX	The fonction set the texte in the editbox. If the flag for multiple edit lines has been set the texte will be added to the previous texte
LISTBOX	The fonction adds a new element to the list

Il faut seulement utiliser le paramètre *par3* si le type d'objet est COMBOBOX ou LISTBOX Le paramètre doit contenir la valeur standard pour le champ.
Valeur retour: Aucune.
Voir aussi: OBJECTCLEAR
Exemple: OBJECTADDSTRING("va#7",gr#2,gr#1) */* Afficher le nom de groupe et utiliser le numéro en tant qu'index;*

13.19. OBJECTCLEAR - Garnir à zéro le contenu d'un objet (IQ)

OBJECTCLEAR(Champs *par1*)
Paramètres: *par1* : Champ à l'écran, par exemple va#7
Description: Cette fonction garnit à zéro le contenu d'un objet
Valeur retour: Aucune.
Voir aussi: OBJECTADDSTRING
Exemple:

```
OBJECTCLEAR("va#7")     /* clear all previous values
START(gr),""            /* read all values from Article group table
NEXT(gr)
   OBJECTADDSTRING("va#7",gr#2,gr#1) /* Display name and use no. as index
REPEAT(gr)
```

13.20. OBJECTGETSTRING- Rechercher le numéro de l'objet sélectionné (IQ/DM)

texte OBJECTGETSTRING(champ *par1*)
Paramètres: *par1* : numéro de champ sous la forme par exemple va#7
Description: Cette fonction recherchera le numéro de la ligne sélectionné dans un champ de boîte de list/combobox. C'est à dire la valeur qui corresponde à *par3* indiqué lors de la création de la boîte avec OBJECTADDSTRING.
Pour utiliser cette fonction, cliquer sur le champ de la liste de boîte/ combobox field.
Valeur retour: La valeur normale d'item sélectionné actuellement.
Voir aussi: OBJECTADDSTRING
Exemple:

```
   #20=OBJECTGETSTRING("va#6")  /* Rechercher le numéro de fournisseur
sélectionné
```

13.21. **PLSNEXT** - Préparer et lire la fichier principal (IQ)

PLSNEXT(nombre *par1*, texte *par2*, nombre *par3*,)

Description: Préparera et exécutera la lecture du fichier principal d'après le mode indiqué. Utilisée dans les menus et pour les fonctions en bas et en haut de la page, etc. En mettant le drapeau d'entrée, vous devez utiliser la clé sinon la lecture est suivante/précédente/direct.
Valeur retour: Aucune.
Voir aussi: DOFONCTION, TRANSMIT
Exemple: PLSNEXT(0,#1,1) */* Lira l'enregistrement suivant avec #1 comme clé*

13.22. SEQ - Changement de la séquence d'entrée (IQ)

SEQ(nombre *par1*, champs *par2*)

par2 : Fieldnumbers in the new sequence
Description: Vous changez le paramètre pour la séquence de champ à l'aide de cette fonction.
Valeur retour: Aucune.
Voir aussi: NEXTFLD, NEXTFLDSEQ
Exemple:

```
SEQ(2,"va#2-3,5")          /* Set the normal amendment sequence
IF #7=1 SEQ(2,"va#4,3")    /* Special for this article group
```

13.23. SETUPD - Repérer un fichier sur une ligne pour la mise à jour(IQ)

SETUPD(fileid *par1*)
Paramètres: *par1* : Fichierid qui doit être mis à jour
Description: Lorsque des modifications sont effectuées dans les champs critiques, il est nécessaire de repérer toutes les transactions pour la mise à jour sinon les enregistrements inscrits lors de l'entrée sera seulement écrits.
Valeur retour: Aucune.
Voir aussi: LOOP
Exemple: SETUPD(va)

13.24. **SHOW** Activer, Désactiver, Visualiser et Cacher un champ (IQ/DM)

Nombre SHOW(champ *par1*, nombre *par2*)

3 = Hide field
Description: Cette fonction permet d'activer ou désactiver l'entrée d'un champ ainsi que d'afficher ou de cacher un champ.
Valeur retour: Aucune.
Exemple:

```
SHOW("va#7",1)        /* Désactiver un champ va#7
```

13.25. SUPER - Préparer la recherche de superindex (IQ)

SUPER(fileid *par1*) , texte *par2*

par2 : Clé
Description: Cette fonction SUPER initialisera la lecture NEXT à utiliser le superindex
Valeur retour: Aucune.
Voir aussi: NEXT, START
Exemple:

```
SUPER(va),#21     /* NEXT uses superindex search for the texte in #21
NEXT(va)          /* Must be follow to actually read the record
SUPER(va)         /* Superindex is switched off
SUPER(va),"#1-3"  /* Superindex fields is set to field 1-3
```

13.26. TRANSMIT- Mettre à jour les autres programmes d'IQ (IQ)

TRANSMIT(nombre *par1*, texte *par2*, texte *par3*)

par3 : Optional Connection
Description: Transmettra les enregistrement actuels à un ou plusieurs programmes en utilisant les connexions définies et fixes ou, la connexion indiquée.

```
    Progid=""   Send to all other
           "20" Just update program 20 if active
           "le" Send to all programs using the file le as mainfile

    Connection = ""     Use automatic connections between files
                 "1,2P" Use field 1 and 2 packed as connection
                 "va.01.6" Use va as transmitting file to the other program
                           Read the other mainfile index 1 using field 6.
```

Valeur retour: Aucune.
Voir aussi: PLSNEXT, DOFONCTION
Exemple: TRANSMIT(0,"","") */* Mettra à jour tous les autres programmes à l'aide de la connexion standard .*

13.27. **TRANSSEL**- Définir les sélections de transactions d' IQ (IQ)

TRANSEL(texte *par1*, nombre *par2*)

Description: explorera une entrée indiqué et définira les sélections de transaction si l'entrée contient le formule en tant que #15>0. Utilisé par IQ en liaison avec les flèches en bas et en haut dans le champ de clé.
Valeur retour: Aucune.
Voir aussi:
Exemple: TRANSSEL("#15>20",1) */* Définir sélection*

14. Fonctions de système

Ces fonctions sont utilisées dans des programmes spéciaux. Par exemple si vous souhaitez avoir l'accès direct aux fichiers ou aux chemins.

14.1. DEBUG- Activer la fenêtre debug (IQ)

DEBUG(nombre *par1*)

Description: La fonction DEBUG(1) ouvra une fenêtre qui affiche tous les calculs et leurs numéros de programme ou d'étiquette quand ils sont exécutés
La fenêtre DEBUG est fermée quand IQ est fermé.
Valeur retour: Aucune
Voir aussi: WIF, WIFS
Exemple: DEBUG(1) */* Activer la fenêtre debug*

14.2. **EXEC**- Exécuter une chaîne de texte en tant que calcul

EXEC(texte *par1*, Numéro de programme *par2*)

par2 : **(IQ/DM)**numéro de programme

Description:

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

exécutera une chaîne de texte stockée dans le champ 20 en tant que calcul.
Lorsque vous utilisez des champs libres dans la fonction EXEC, vous devez utiliser la référence WW#nn pour ce champ. Vous pouvez l'obtenir à partir de l'impression des définitions de programme.
Normalement, la chaîne intégrée à la fonction EXEC n'est pas explorée et contrôlée comme une ligne de calcul normal. Cela est important quand elle est utilisée dans RAPGEN car la syntaxe C pour les calculs doit être associé. Nous conseillons les utilisateur sans aucune expérience de programmation d'utiliser la fonction EXEC dans RAPGEN d'une manière simple sans faire d'autres appels aux fonctions. Des paramètres de fonction non validés peuvent provoquer une protection d'erreur générale.
Nous vous conseillons spécialement de faire attention à ce que RAPGEN : #15=2 met le champ 15 égal à 2, qui est EGALEMENT utiliser lorsque vous utilisez l'expression IF #15=2 LET #16=3. Il faut que vous indiquez le signe d'égalité deux fois selon la syntaxe C dans une telle instruction: IF (#15==2) LET #16=3
IQ: EXEC(#20,15) change au programme actif 15 et exécute le calcul indiqué.
Valeur retour: Aucune
Voir aussi:
Exemple: EXEC(#20) */* Exécute un calcul entré lors du démarrage du rapport*

14.3. GETFLD- Indication de la structure des pointeurs SY (IQ)

GETFLD(texte *par1*)
Paramètres: *par1* : spécification de champ
Description: Cette fonction indiquera que le système variable (SY#..) doit pointer sur la définition du champ indiqué. Cette définition de champ peut être changée ou lue lors de l'exécution. Destinée aux programmeurs seulement.
Valeur retour: Aucune

14.4. INSTALL- Fonctions externes

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

par4 : propre nom de fonction, éventuellement
Description: Les programmeurs qui connaissent les définitions de fonction à partir d'autres DLL' peuvent incorporer celles-ci en tant que fonctions IQ.

NOTE: Improper use of this fonction may cause system breakdown.

Valeur retour: Aucune
Voir aussi:
Exemple:

```
INSTALL("a.dll","b","3,[ss]")
   activates #20=B(#21) from a.dll, #20 and #21 being short variables

INSTALL("some.dll","aname","3,[sCl]","FUNNY")
   activates #30=FUNNY(#31,#32) as fonction aname from some.dll
   return value #30 short, paramètres #31 as char pointer, #32 as long.
```

14.5. SYSPAR - Lecture du paramètre de système

texte SYSPAR(nombre *par1*)

Description: SYSPAR lira le paramètre indiqué. Seulement les valeurs présentées ci-dessus sont importantes.
Valeur retour: Paramètre système.
Voir aussi: SYSPARSET
Exemple: #1 = SYSPAR(4) */* Chercher le chemin TMP*

14.6. SYSPARSET - Instaurer la valeur d'un paramètre de système

SYSPARSET(nombre *par1*, texte *par2*)

par2 : New value of this system parameter
Description: SYSPARSET changera la valeur du paramètre indiqué.
Valeur retour: None.
Voir aussi: SYSPAR
Exemple: SYSPARSET(4,"c:/mytmp/") */* Instaurer une nouvelle valeur pour le chemin TMP*

14.7. USERINFO - Rechercher les informations sur l'utilisateur à partir du contrôle utilisateur

texte USERINFO(nombre *par1*)

17=User defined
Description: Cette fonction recherchera les informations sur l'utilisateur demandées.
Le numéro noté dans *Par1*fait référence au numéro de champ dans le fichier système US dans lequel vous pouvez définir les champs 11 à 17 individuellement pour chaque installation. (Soyez prudent si vous met à niveau la version de TRIO).
Valeur retour: Chaîne de texte contenant les informations sur l'utilisateur voulue.
Exemple:

```
#11=USERINFO(6)  /* Get the user first remark
```

14.8. **WIF** - Copie test (IQ)

WIF(texte *par1*)
Paramètres: *par1* : Texte qui doit être imprimé
Description: WIF donnera une copie d'impression au fichier c:/wif sans changer l'écran.
Valeur retour: Aucune
Voir aussi: WIFS, DEBUG
Exemple: WIF("Here I am") */* test de sortie*

14.9. **WIF**- Copie test (RAP)

WIF(texte *par1* , texte *par2*)

. **Description:** WIF donnera une copie test au fichier c:/wif
Valeur retour: Aucune
Voir aussi: WIFS, DEBUG
Exemple: WIF("Field equals %s.",#2) /* *Copie test*

14.10. **WIFS**- Copie test des champs (IQ)

WIFS(fields *par1*)
Paramètres: *par1* : Champs qui doivent être imprimés
Description: WIFS donnera une copie test au fichier c:/wif des valeurs notées
Valeur retour: Aucune
Voir aussi: WIF, DEBUG
Exemple: WIFS("va#1-3,le#2") */* Copie test des valeurs de champ*

Index