Syntax von Formeln und Ausdrücken des Parsers SParser (V2)

Inhaltsverzeichnis

1 Allgemeine Erläuterungen

Der Parser dient zur Berechung bzw. Auswertung und syntaktischen Überprüfung von Formeln und Ausdrücken.
Er ist typempfindlich, d.h. er unterscheidet zwischen folgenden (Daten-) Typen:

- Zahlen
- Zeichenketten (Texte)

Darüber hinaus unterstützt er Variablen, die Werte beider Typen annehmen können und lediglich als Platzhalter fungieren.
In einem auszuwertenden Ausdruck dürfen sowohl Zahlen als auch Zeichenketten enthalten sein.

Zahlen

- Dezimaltrennzeichen ist ein Punkt.
- führende Nullen (z.B. bei 0.5) dürfen weggelassen werden.
- Tausendertrennzeichen dürfen keine angegeben werden
- wissenschaftliche Darstellung ist zulässig, z.B.:
  1.5e4 **à** 15000
  3e-5 **à** 0.00003
  5E2 **à** 500

Zeichenketten

- Zeichenketten werden durch das Einschließen beliebiger Zeichen in doppelte Anführungszeichen (" ") gekennzeichnet, z. B.:
  "Zeichenkette"
- Zeichenketten mit Unterstützung für „Escape-Sequenzen“ (\\ \n \t \r \’ \“) müssen in Hochkommata eingeschlossen werden, z.B.:
  ’Zeile1\nZeile2’
- zwischen Groß- und Kleinschreibung wird unterschieden (case-sensitive)

Variablen

Variablen sind Platzhalter und können Zahlen oder auch Zeichenketten enthalten.

Variablennamen beginnen immer mit einem Groß- oder Kleinbuchstaben oder einem Unterstrich (_). Darauf dürfen beliebig viele Groß- oder Kleinbuchstaben, Ziffern und Unterstriche folgen. Als Variablennamen unzulässig sind die folgenden Funktionsnamen bzw. Schlüsselwörter:

AND	OR	NOT	XOR		
IF	THEN	ELSE	SWITCH	CASE	DEFAULT

Funktionen

Funktionsnamen unterliegen prinzipiell den gleichen Regeln wie Variablen. Ein Funktionsaufruf ist an den darauffolgenden Klammern () erkennbar. Funktionsparameter können sowohl durch Semikolon ; als auch durch Komma , getrennt werden.

Weitere Merkmale des Parsers

- Bei **allen** Funktionen werden die angegebenen Werte / Parameter **nicht** verändert.

- Alle Funktionen können unter Einhaltung der jeweiligen Syntax beliebig kombiniert und geschachtelt werden.

- Leerzeichen und Zeilenumbrüche werden außer in Zeichenketten ignoriert.

Zur besseren Lesbarkeit dieses Dokuments werden die folgenden Symbole und Schriftarten verwendet:

Symbolik

Hinweis		Weiterführende Informationen
Tipp		Nützliche Tipps & Tricks
Anmerkung		Anmerkungen zur Vertiefung und Verdeutlichung
Beispiel	**Beispiel**	
Syntax	**Syntax**	

Schriftstile und Schriftfarben

`Text in Serifenschrift`	Ausdrücke wie sie direkt eingegeben werden können („Quellcode“)

kursiver Text in blauer Farbe	Parameter bzw. Ausdrücke vom Typ **ZAHL**
kursiver Text in grüner Farbe	Parameter bzw. Ausdrücke vom Typ **Zeichenkette**
kursiver Text in grauer Farbe	Parameter bzw. Ausdrücke von **beliebigem** Typ

Funktionsparameter in eckigen Klammern [ ] sind optional.

2 Mathematische Funktionen

Es werden die gängigsten mathematischen Grundfunktionen unter Einhaltung der algebraischen Regeln (Klammerpriorität, Punkt vor Strich usw.) unterstützt.

Grundfunktionen	+	Addition
	-	Subtraktion
	*	Multiplikation
	/	Division
	^	Potenz
	SQRT(*Zahl*)	Quadratwurzel
	+ -	Vorzeichen
Vergleichs-funktionen	<	kleiner als
	>	größer als
	<=	kleiner oder gleich
	>=	größer oder gleich
	=	ist gleich
	<>	ungleich
Trigonom. Funktionen	SIN(*Zahl*)	Sinus (Argument in Grad)
	ARCSIN(*Zahl*)	Arkussinus
	COS(*Zahl*)	Kosinus (Argument in Grad)
	ARCCOS(*Zahl*)	Arkuskosinus
	TAN(*Zahl*)	Tangens (Argument in Grad)
	ARCTAN(*Zahl*)	Arkustangens
Weitere mathematische Funktionen	EXP(*Zahl*)	Exponentialfunktion (e-Funktion)
	LN(*Zahl*)	natürlicher Logarithmus
	MOD(*Zahl*)	Modulofunktion, Nachkommastellen fallen weg
	PREC(*Zahl*)	Vorkommastellen fallen weg
	ABS(*Zahl*)	Betrag

Die Vergleichsfunktionen liefern den Wert 1 zurück, falls die Aussage zutrifft und andernfalls den Wert 0.

Beispiele

Ausdruck	Ergebnis
`2+3`	5
`2-3`	-1
`2*3`	6
`2/3`	0.666667
`2^3`	8
`SQRT(2)`	1.414214
`--2`	2
`2<3`	1
`2>3`	0
`2<=3`	1

2>=3	0
2=3	0
2<>3	1
SIN(30)	0.5
ARCSIN(.5)	30
COS(60)	0.5
ARCCOS(.5)	60
TAN(45)	1
ARCTAN(1)	45
EXP(1)	2.718282
LN(2.718282)	1
MOD(2.3)	2
PREC(2.3)	0.3
ABS(-2)	2

2.1 Logische Funktionen

AND	Und	Beide Bedingungen müssen erfüllt sein
OR	Oder	Mindestens eine der beiden Bedingungen muss erfüllt sein
XOR	Exklusiv Oder	Genau eine der beiden Bedingungen muss erfüllt sein
NOT	Nicht	Negiert das Argument

Eine Bedingung ist erfüllt, wenn der Wert des auszuwertenden Ausdrucks von 0 verschieden ist.

Beispiele

Ausdruck	Ergebnis
0 AND 1	0
2 AND 3	1
0 OR 1	1
2 OR 3	1
0 XOR 1	1
2 XOR 3	0
NOT 0	1
NOT 2	0

3 Funktionen zur Verarbeitung von Zeichenketten (Stringfunktionen)

Neben Zahlenwerten können auch Texte bzw. Zeichenketten verwendet werden. Hierzu stehen folgende Funktionen zur Verfügung:

Verkettung	&
Groß-Kleinschreibung	UCASE
	LCASE
Teilstring / Extraktion	LEFT
	RIGHT
	MID
Länge	LEN
	ISEMPTY
Suche	FIND
	RFIND
Manipulation	REPLACE
	INSERT
	LTRIM
	RTRIM

Neben diesen Funktionen können für Zeichenketten dieselben Vergleichsoperatoren wie bei den Zahlen verwendet werden.

< > <= >= = <>

Auch bei den Vergleichsfunktionen wird die Groß- bzw. Kleinschreibung der Zeichenketten berücksichtigt, d.h. "a" ist nicht gleich "A", sondern "a" ist größer "A".

Beispiele

Ausdruck	Ergebnis
`"Hans" < "Hugo"`	1
`"hans" < "Hugo"`	0
`"Hans" > "Hugo"`	0
`"Hans" <= "Hugo"`	1
`"Hans" >= "Hugo"`	0
`"Hans" = "Hugo"`	0
`"Hans" <> "Hugo"`	1

Für die folgenden Beispiele in diesem Kapitel seinen folgende Variablen definiert:

Name	"Homag"
PrgName1	"Platte01.mpr"

&

Syntax *Text1* & *Text2*

Verbindet zwei Zeichenketten miteinander derart, dass *Text2* direkt an *Text1* angehängt wird.

Die Funktion kann beliebig viele Texte miteinander verketten.

Beispiele

Ausdruck	Ergebnis
`Name & " Holzbearbeitungssysteme AG"`	Homag Holzbearbeitungssysteme AG
`"c:\machine1\" & "A1\"`	c:\machine1\A1\
`"c:\machine1\A1\" & "mp4\" & PrgName1`	c:\machine1\A1\mp4\Platte01.mpr

UCASE

(**u**pper **case**)

Syntax UCASE(*Text1*)

Konvertiert alle Kleinbuchstaben in der Zeichenkette *Text1* zu Großbuchstaben. Andere Zeichen bleiben unverändert.

Beispiele

Ausdruck	Ergebnis
`UCASE("TEST")`	TEST
`UCASE(Name)`	HOMAG
`UCASE("Teil-0815-A")`	TEIL-0815-A

LCASE

(**l**ower **case**)

Syntax LCASE(*Text1*)

Konvertiert alle Großbuchstaben in der Zeichenkette *Text1* zu Kleinbuchstaben. Andere Zeichen bleiben unverändert.

Beispiele

Ausdruck	Ergebnis
`LCASE("TEST")`	test
`LCASE(Name)`	homag
`LCASE("Teil-0815-A")`	teil-0815-a

Mit UCASE bzw. LCASE lässt sich der Parser bei Vergleichen sozusagen unempfindlich bezüglich der Groß- und Kleinschreibung machen:
`LCASE("Homag") = LCASE("HOMAG")`

LEFT

Syntax LEFT(*Text1*; *Länge*)

Liefert von der Zeichenkette *Text1* die durch *Länge* angegebene Anzahl von Zeichen von links.

Beispiele

Ausdruck	Ergebnis
`LEFT("Holzbearbeitungssysteme"; 4)`	Holz
`LEFT(PrgName1; 8)`	Platte01

RIGHT

Syntax RIGHT(*Text1*; *Länge*)

Liefert die durch *Länge* angegebene Anzahl von Zeichen von rechts in der Zeichenkette *Text1*.

Beispiele

Ausdruck	Ergebnis
`RIGHT("Holzbearbeitungssysteme AG"; 2)`	AG
`RIGHT(PrgName1; 3)`	mpr

MID
(**mid**dle)

Syntax MID(*Text1*; *Startposition*[; *Länge*])

Liefert einen Teil der Zeichenkette *Text1* zurück. Mit dem Parameter *Startposition* wird die Anzahl der zu überspringenden Zeichen angegeben. Falls der Parameter *Länge* angegeben wird, bestimmt er die Länge der Teilzeichenkette. Wird er nicht angegeben, so wird der Rest der Zeichenkette zurückgegeben.

Beispiele

Ausdruck	Ergebnis
`MID("Homag Holzbearbeitungssysteme AG"; 6)`	Holzbearbeitungssysteme AG
`MID("Homag Holzbearbeitungssysteme AG"; 6; 4)`	Holz
`MID(PrgName1; 6)`	01.mpr
`MID(PrgName1; 6; 2)`	01

LEN
(**len**gth)

Syntax LEN(*Text1*)

Liefert die Anzahl der Zeichen in der Zeichenkette *Text1* zurück.

Beispiele

Ausdruck	Ergebnis
`LEN("")`	0
`LEN("Homag Holzbearbeitungssysteme AG")`	32
`LEN(PrgName1)`	12

ISEMPTY

Syntax ISEMPTY(*Text1*)

Überprüft, ob die Zeichenkette *Text1* leer ist. Wenn sie leer ("") ist, so wird 1 ansonsten 0 zurückgeliefert.

Die Funktion
ISEMPTY(*Text1*)
ist gleichbedeutend mit:
LEN(*Text1*) = 0

Beispiele

Ausdruck	Ergebnis
`ISEMPTY("")`	1
`ISEMPTY("Homag Holzbearbeitungssysteme AG")`	0
`ISEMPTY(PrgName1)`	0

FIND

Syntax FIND(*Heuhaufen*; *Nadel* [;*Startposition*])

Durchsucht die Zeichenkette *Heuhaufen* auf Vorkommen der Zeichenkette *Nadel* und gibt dessen Position zurück. Mit dem Parameter *Startposition* kann eine Anzahl zu überspringender Zeichen angegeben. Wird er nicht angegeben, so wird beim ersten Zeichen begonnen. Kommt die zu suchende Zeichenkette *Nadel* nicht in *Heuhaufen* vor, so wird -1 zurückgegeben.

Beispiele

Ausdruck	Ergebnis
`FIND("Homag Holzbearbeitungssysteme AG"; "H")`	0
`FIND("Homag Holzbearbeitungssysteme AG"; "H"; 1)`	6
`FIND("Homag Holzbearbeitungssysteme AG"; "Holz")`	6
`FIND("Homag Holzbearbeitungssysteme AG"; "HOLZ")`	-1

RFIND

(**r**everse **find**)

Syntax RFIND(*Heuhaufen*; *Nadel* [;*Startposition*])

Diese Funktion verhält sich analog zu FIND, außer dass das Durchsuchen der Zeichenkette *Heuhaufen* von hinten nach vorne durchgeführt wird.

Beispiele

Ausdruck	Ergebnis
`RFIND("Homag Holzbearbeitungssysteme AG"; "H")`	6
`RFIND("Homag Holzbearbeitungssysteme AG"; "H" ; 5)`	0
`RFIND("Homag Holzbearbeitungssysteme AG"; "HOLZ")`	-1
`RFIND("c:\machine1\a1\mp4"; "\")`	14

Soll die Anzahl zu überspringender Zeichen vom letzten Zeichen aus gerechnet werden, so kann dies mit dem Ausdruck
LEN(*Heuhaufen*) - *Zeichenanzahl*
für den Parameter *Startposition* bewerkstelligt werden.

REPLACE

Syntax REPLACE(*Text1*; *Alt*; *Neu*)

Ersetzt in der Zeichenkette *Text1* alle Teilzeichenketten *Alt* durch *Neu*. Dabei können die Zeichenketten *Alt* und *Neu* jeweils beliebig lang sein.

Beispiele

Ausdruck	Ergebnis
`REPLACE("c:\machine1\a1\mp4"; "\"; "/")`	c:/machine1/a1/mp4
`REPLACE(PrgName1; "01"; "02")`	Platte02.mpr
`REPLACE(PrgName1; "0"; "0000")`	Platte00001.mpr
`REPLACE(PrgName1; "01"; "")`	Platte.mpr

Das Löschen von Teilzeichenketten kann mit einer leeren Zeichenkette ("") für den Parameter *Neu* erzielt werden.

INSERT

Syntax INSERT(*Text1*; *Position*; *Neu*)

Fügt in die Zeichenkette *Text1* an der Stelle *Position* die Zeichenkette *Neu* ein.

Beispiele

Ausdruck	Ergebnis
`INSERT("123456"; 3; "---")`	123---456
`INSERT (PrgName1; 6; "_")`	Platte_01.mpr

LTRIM

(**l**eft **trim**)

Syntax LTRIM(*Text1*)

Entfernt alle Leerzeichen am Anfang der Zeichenkette *Text1*.

Beispiele

Ausdruck	Ergebnis
`LTRIM (Name)`	Homag
`LTRIM ("   Text")`	Text

RTRIM

(**r**ight **trim**)

Syntax RTRIM(*Text1*)

Entfernt alle Leerzeichen am Ende der Zeichenkette *Text1*.

Beispiele

Ausdruck	Ergebnis
`RTRIM (Name)`	Homag
`RTRIM ("c:\machine1\a1\mp4 ")`	c:\machine1\a1\mp4

4 Bedingte Ausdrücke

Der Parser unterstützt zwei Konstrukte für bedingte Ausdrücke.
Das IF-THEN-ELSE Konstrukt und das SWITCH-CASE-DEFAULT Konstrukt. Sie sind in den beiden folgenden Kapiteln näher beschrieben.

4.1 IF-THEN-ELSE Konstrukt

Syntax IF *Bedingung* THEN *WennErfüllt* ELSE *Sonst*

Falls die Bedingung *Bedingung* erfüllt ist, d.h. ihr Wert ungleich 0 ist, so wird der Wert des Ausdrucks *WennErfüllt* zurückgeliefert, ansonsten der Wert des Ausdrucks *Sonst*. Die Ausdrücke *WennErfüllt* und *Sonst* dürfen von beliebigem Typ sein. Der Typ des zurückgelieferten Werts ist identisch mit dem des jeweiligen Ausdrucks.
Das IF-THEN-ELSE Konstrukt darf beliebig oft geschachtelt werden, d.h. die Ausdrücke *WennErfüllt* oder *Sonst* dürfen selbst wiederum ein IF-THEN-ELSE Konstrukt sein.

Für die folgenden Beispiele in diesem Kapitel seinen folgende Variablen definiert:

L	1200
B	800
Z	25
AlsZeichenkette	1
k	3

Beispiele

Ausdruck		Ergebnis
`IF 0 THEN 12 ELSE 34`		34
`IF 1 THEN 12 ELSE 34`		12
`IF L>800 THEN 100 ELSE 200`		100
`IF B>=850 THEN B/3 ELSE B/2`		400
`IF L<B THEN "hochformat" ELSE "querformat"`		querformat
`IF AlsZeichenkette THEN "123" ELSE 123`		123
`IF Z<0 THEN -1 ELSE IF Z=0 THEN 0 ELSE 1`	*	1
`IF Z<=0 THEN IF Z=0 THEN 0 ELSE -1 ELSE 1`	*	1
`IF k=1 THEN "A" ELSE IF k=2 THEN "B" ELSE IF k=3 THEN "C" ELSE "D"`		C

Die beiden mit dem Sternchen gekennzeichneten Beispiele führen zum selben Ergebnis.

4.2 SWITCH-CASE-DEFAULT Konstrukt

Mit diesem Konstrukt kann ein Ausdruck mehrfach verglichen werden, ohne ihn mehrfach aufführen zu müssen, d.h. im Unterschied zum IF-THEN-ELSE Konstrukt wird hier immer der gleiche Ausdruck (Switch-Argument) für die Auswertung hinzugezogen.

Syntax SWITCH *Referenz* CASE *VergleichX* THEN *WertX* DEFAULT *Sonst*

Nach dem Schlüsselwort SWITCH folgt der mit allen Zweigen zu Vergleichende Ausdruck *Referenz*. Nun können beliebig viele CASE-Zweige, jedoch mindestens einer, folgen. Entspricht der Wert des Ausdrucks *Referenz* dem des Ausdrucks *VergleichX*, so wird der Wert des auf das Schlüsselwort THEN folgenden Ausdrucks zurückgegeben.
Falls keiner der Zweige zutrifft wird der Wert nach dem abschließenden Schlüsselwort DEFAULT zurückgeliefert.

Darüber hinaus darf jeder CASE-Zweig mehrere CASE-Vergleiche enthalten. Somit lassen sich komplexe Verzweigungen wie im folgenden Beispiel realisieren:

SWITCH *Referenz*
    CASE *VergleichA1* CASE *VergleichA2* CASE *VergleichA3*
    THEN *WertA*
    CASE *VergleichB1* CASE *VergleichB2* CASE *VergleichB3* CASE *VergleichB4*
    THEN *WertB*
    CASE *VergleichC1* CASE *VergleichC2*
    THEN *WertC*
DEFAULT *Sonst*

Bei der Auswertung der Zweige wird verglichen, ob der Wert des Ausdrucks *Referenz* **gleich** dem Wert des jeweiligen Ausdrucks *VergleichX* ist.

Die Typen der Vergleichswerte müssen identisch zu dem des Referenzwerts sein.
Die Typen der zurückzugebenden Ausdruckswerte können beliebig gewählt werden.

Beispiele

Ausdruck	Ergebnis
`SWITCH 20 CASE 10 THEN "A" DEFAULT "XYZ"`	XYZ
`SWITCH 20 CASE 10 THEN "A" CASE 20 THEN "B" DEFAULT "XYZ"`	B
`SWITCH 20 CASE 10 CASE 20 THEN "A" DEFAULT "XYZ"`	A
`SWITCH "R" CASE "A" THEN 1 CASE "B" THEN 2 DEFAULT 9`	9

Zur Verdeutlichung der Verzweigungen weitere Beispiele mit tabellarischer Darstellung der Resultate:

Beispiel

SWITCH k CASE 1 THEN A CASE 2 THEN B CASE 3 THEN C DEFAULT D

Abhängig vom Wert von k wird folgender Wert ermittelt:

k	Ergebnis
1	A
2	B
3	C
sonst	D

Beispiel

SWITCH k CASE 1 CASE 3 CASE 5 THEN U CASE 2 CASE 4 THEN G DEFAULT X

Abhängig vom Wert von k wird folgender Wert ermittelt:

k	Ergebnis
1, 3 oder 5	U
2 oder 4	G
sonst	X

Beispiel

SWITCH k CASE 1 CASE 3 CASE 5 THEN U CASE 2 CASE 4 THEN G DEFAULT X

Abhängig vom Wert von k wird folgender Wert ermittelt:

k	Ergebnis
1, 3 oder 5	U
2 oder 4	G
sonst	X

Intervalle

Soll der Ausdruck *Referenz* mit einem ganzen Bereich von Werten verglichen werden, so können auf das Schlüsselwort CASE **Intervalle** folgen. Ein Intervall wird mit folgender Syntax angegeben (zwei direkt aufeinanderfolgende Punkte):

Anfang .. *Ende*

Die Auswertung eines Intervalls ist positiv, wenn der Ausdruck *Referenz* gleich oder zwischen den Werten der Ausdrücke *Anfang* und *Ende* ist.

Falls der Ausdruck *Anfang* mit einer konstanten Ganzzahl endet, so muß mindestens ein Leerzeichen folgen, da ansonsten der erste Punkt als Dezimaltrennzeichen interpretiert wird.

Der Ausdruck
```
SWITCH Breite CASE 100 .. 500 THEN 2 DEFAULT 3
```
ist gleichbedeutend mit:
```
IF 100 <=  Breite AND Breite <= 500 THEN 2 ELSE 3
```

Beispiel

SWITCH LEFT(Name, 1) CASE "A" .. "F" THEN N1 CASE "G" .. "L" THEN N2 DEFAULT N3

Abhängig von Name wird folgender Wert ermittelt:

Anfangsbuchstabe von Name	Ergebnis
A bis F	N1
G bis L	N2
sonst	N3

Das IF-THEN-ELSE Konstrukt und das SWITCH-CASE-DEFAULT Konstrukt dürfen beliebig oft verschachtelt und miteinander gemischt werden.

5 Konvertierungs- und Sonderfunktionen

Konvertierungsfunktionen

STR

Syntax STR(*Ausdruck*)

Konvertiert den Ausdruck *Ausdruck* in eine Zeichenkette.

Beispiele

Ausdruck	Ergebnis
`STR(123)`	123
`STR(4 + 5.6)`	9.6
`STR("Text")`	Text

VAL

Syntax VAL(*Ausdruck*)

Konvertiert den Ausdruck *Ausdruck* in eine Zahl. Handelt es sich um eine Zeichenkette, die nicht mit Ziffern, Vorzeichen oder Dezimaltrennzeichen beginnt, so wird 0 zurückgegeben (die Auswertung der Zeichenkette wird abgebrochen, sobald ein ungültiges Zeichen auftritt).

Beispiele

Ausdruck	Ergebnis
`VAL("234")`	234
`VAL("-.5")`	-0.5
`VAL("Text")`	0
`VAL("15 Platten")`	15

STR() und VAL() akzeptieren als Parameter sowohl Zeichenketten als auch Zahlen. Damit kann man den Wert einer Variable, die ggf. Werte verschiedener Typen annimmt, mit Typsicherheit weiterverwenden.

Sonderfunktionen

VARDEF

Syntax VARDEF (*Variablenname*)

Überprüft, ob die Variable *Variablenname* definiert ist. Ist sie definiert so wird 1 zurückgegeben, andernfalls 0.

Für die folgenden Beispiele sei folgende Variable definiert:

X	100

Beispiele

Ausdruck	Ergebnis
`VARDEF("X")`	1
`VARDEF("Y")`	0
`IF VARDEF("X") THEN X ELSE 123`	100
`IF VARDEF("Y") THEN Y ELSE 123`	123

6 Bindungen

Zur Einhaltung der algebraischen Regeln (z.B. „Punkt vor Strich“) verwendet der Parser die in der folgenden Tabelle aufgelisteten Bindungsebenen und Bindungsrichtungen. Die stärkste Bindungsebene ist oben in der Tabelle.

Operatoren	Bindungsrichtung
Funktionen und Klammern	
(Vorzeichen) **+ - NOT**	**à**
^	**ß**
*** /**	**à**
+ - &	**à**
< <= >= >	**à**
= <>	**à**
AND	**à**
XOR	**à**
OR	**à**
*Bedingte Ausdrücke (*IF *und* SWITCH*)*	

starke Bindung

schwache Bindung

Bindungsebenen

In Ausdrücken, die Kombinationen von Operatoren enthalten, werden die stärksten Bindungen zuerst berechnet. Um die Reihenfolge explizit festzulegen, dürfen die Ausdrücke beliebig oft geklammert werden.

Beispiele

Ausdruck	Ergebnis	Bindung
1+2*3	7	1+ 2*3
(1+2)*3	9	1+2 *3
1+2*3^4	163	1+ 2* 3^4
NOT a AND b	$\bar{a} \wedge b$	NOT a AND b
NOT(a AND b)	$\overline{a \wedge b}$	NOT a AND b
a OR b AND c	$a \vee b \wedge c$	a OR b AND c
a OR b AND c XOR d	$a \vee b \wedge c \oplus d$	a OR b AND c XOR d

`a < b = c < d`	*(die beiden Ausdrücke sind äquivalent)*	a<b = c<d
`a<b AND c<d OR a>=b AND c>=d`		a<b AND c<d OR a>=b AND c>=d
`L - p1 / 2 > p_min OR p2 * -2 < p3`		L- p1/2 >p_min OR p2* -2 <p3
`"c:\" & Ordner1 & Datei5`		"c:\" & Ordner1 & Datei5
`ARCSIN(SQRT(2)/2)`	45	ARCSIN(SQRT(2) /2)
`IF L/2 <= min_X THEN "T123" ELSE "T" & STR(TNr)`	IF L/2 <= min_X THEN "T123" ELSE "T" & STR(TNr)	

Bindungsrichtung

Folgen mehrere Operatoren der selben Bindungsebene aufeinander, so entscheidet die Bindungsrichtung über die Reihenfolge der Auswertung. Bei allen Operatoren außer der Potenz ^ ist die Bindungsrichtung links nach rechts. Die Bindungsrichtung der Potenz ist im woodWOP6-Modus (PM_WW6) von rechts nach links und im woodWOP5-Modus (PM_WW5) von links nach rechts.

Beispiele

Ausdruck	Ergebnis	Bindung
`1+2+3`	6	1+2 +3
`4-3-2-1`	-2	4-3 -3 -1
`5+4-3+2-1`	7	5+4 -3 +2 -1
`4^3^2`	262144	4^ 3^2
`1.1^1.2^1.3^1.4`	1.13203	1.1^ 1.2^ 1.3^1.4

7 Irrelevante Zweige

Bei der Auswertung mancher Ausdrücke steht das Ergebnis frühzeitig fest, was zur Folge hat, dass der Rest des Ausdrucks bzw. bestimmte Teile davon nicht mehr ausgewertet werden müssen.

Beispiel `IF Bed1 THEN (L-2*RX)/2 ELSE (B-2*RY)/2`

Wenn die Bedingung Bed1 erfüllt ist, so ist nur der Teil (L-2*RX)/2 von Bedeutung und der Teil (B-2*RY)/2 muss nicht berechnet werden. Falls die Bedingung nicht erfüllt ist entsprechend umgekehrt.
Der Parser erkennt dies und überprüft die irrelevanten Zweige lediglich auf syntaktische Korrektheit, führt aber keine Auswertung durch.
Der Vorteil davon ist in folgendem Beispiel zu erkennen.

Beispiel `IF X<>0 THEN 1/X ELSE 0`

Würde der Parser im Zweig 1/X die Auswertung durchführen, auch wenn die Bedingung X<>0 nicht zutreffend war, würde der Fehler „Division durch Null“ gemeldet, obwohl der Zweig für genau diese Bedingung irrelevant ist und eben dies die Intension für diesen speziellen Ausdruck ist.

Irrelevante Zweige können in folgenden Konstrukten und Operatoren auftreten:

IF-THEN-ELSE	Je nachdem, ob die Bedingung erfüllt ist oder nicht, wird nur der jeweils zutreffende Ausdruck ausgewertet.
SWITCH-CASE-DEFAULT	Sobald der erste Vergleich (CASE-Argument) positiv ist, werden alle folgenden Vergleiche, sowohl im selben CASE-Zweig wie auch in allen folgenden Zweigen, nicht mehr ausgewertet.
AND	Wenn der 1.Parameter 0 ist, so kann der Ausdruck nicht mehr erfüllt werden, weshalb dann der 2. Parameter nicht mehr ausgewertet wird.
OR	Wenn der 1. Parameter ungleich 0 ist, dann ist der Ausdruck bereits erfüllt, unabhängig vom 2. Parameter, der dann nicht mehr ausgewertet wird.

Beispiele

```
IF VARDEF("X") THEN X ELSE 0
IF X <= 0 THEN 5 ELSE LN(X)
SWITCH X CASE 0 THEN 0 DEFAULT 1/X
SWITCH X CASE -1.0 .. 1.0 THEN ARCSIN(X) DEFAULT 180
SWITCH k CASE 1 CASE 5 THEN 1.2*L CASE 6 ..10 THEN 1.5*L2 DEFAULT 1.8*L3
VARDEF("X") AND X<>0
X >= L OR X >= B
```

8 Fehlermeldungen

Mathematische Fehler

Mathematische Fehler werden erst bei der Berechnung / Auswertung des Ausdrucks erkannt, also nach der syntaktischen Überprüfung.

Fehlertext	**Division durch Null**
Fehlernummer	101
Fehlercode	SPSC_M_DIVISION_BY_ZERO
Beschreibung	Der Divisor ist Null. (a^{-b} kann auch als $\frac{1}{a^b}$ geschrieben werden.)
Beispiele	`10 / 0`
	`5 / (4-2^2)`
	`0^-2`

Fehlertext	**Negatives Argument der Wurzelfunktion**
Fehlernummer	102
Fehlercode	SPSC_M_NONREAL_RESULT
Beschreibung	Die Wurzel einer negativen Zahl liefert keine reelle Zahl. ($a^{\frac{p}{q}}$ kann auch als $\sqrt[q]{a^p}$ geschrieben werden.)
Beispiele	`SQRT(-2)`
	`SQRT(5-8)`
	`-10^2.5`

Fehlertext	**Undefiniertes Ergebnis**
Fehlernummer	103
Fehlercode	SPSC_M_UNDEFINED_RESULT
Beschreibung	Die Funktion ist für diesen Wert nicht definiert.
Beispiele	`TAN(90)`
	`TAN(-10*27)`

Fehlertext	**Wert außerhalb des Definitionsbereichs**
Fehlernummer	104
Fehlercode	SPSC_M_VALUE_OUT_OF_DOMAIN
Beschreibung	Die Funktion ist für diesen Wert nicht definiert. Die Arkusfunktionen sind nur von -1.0 bis +1.0 definiert.
Beispiele	`ARCSIN(2)`
	`ARCCOS(-2)`

Fehler bei Variablen

Fehlertext	**Unbekannte Variable** □
Fehlernummer	201
Fehlercode	SPSC_V_VAR_UNKNOWN
Beschreibung	Die angegebene Variable existiert nicht.
Beispiele	`L/2`
	`0.8 * B + Offset`

Fehlertext	Keine Variablenliste verfügbar
Fehlernummer	202
Fehlercode	SPSC_V_NO_VARLIST_SPECIFIED
Beschreibung	Der Ausdruck enthält eine oder mehrere Variablen, dem Parser steht aber keine Variablenliste zur Verfügung. Dieser Fehler ist im Allgemeinen kein Eingabefehler des Benutzers, sondern deutet auf ein Problem des Programms hin.
Beispiele	`L/2`
	`0.8 * B + Offset`

Fehlertext	Fehler beim Ermitteln der Variable □
Fehlernummer	203
Fehlercode	SPSC_V_INQUIRY_FAILURE
Beschreibung	Der Ausdruck enthält eine oder mehrere Variablen. Die Ermittlung des Werts dieser Variable konnte nicht erfolgreich abgeschlossen werden. Dieser Fehler ist im Allgemeinen kein Eingabefehler des Benutzers, sondern deutet auf ein Problem des Programms hin.
Beispiele	`L/2`
	`0.8 * B + Offset`

Typfehler

Typfehler treten auf, wenn eine Funktion oder ein Operator auf Parameter falschen Datentyps stößt. Sie werden ebenfalls erst bei der Berechnung / Auswertung, d.h. nach der syntaktischen Überprüfung festgestellt.

Fehlertext	Unverträgliche Datentypen
Fehlernummer	301
Fehlercode	SPSC_T_INCOMPATIBLE_TYPES
Beschreibung	Zwei oder mehr miteinander korrelierende Parameter sind von verschiedenem Typ.
Beispiele	`2 < "A"`
	`SWITCH 12 CASE "A" THEN 0.1 DEFAULT 0.9`
	`SWITCH "Text" CASE 10 .. 20 THEN 0.1 DEFAULT 0.9`

Fehlertext	Falscher Datentyp
Fehlernummer	302
Fehlercode	SPSC_T_WRONG_TYPE
Beschreibung	Der Typ des Parameters entspricht nicht dem erwarteten Typ.
Beispiele	`1 + "x"`
	`0 OR "Text"`
	`IF "Text" THEN 2 ELSE 3`
	`SIN("x")`
	`VARDEF(0)`
	`LEFT(2, "Text")`
	`REPLACE("TextA", "A", 1)`
	`ISEMPTY (0)`

Fehler bei Funktionen

Fehlertext	Funktion □ unbekannt
Fehlernummer	401
Fehlercode	SPSC_F_FUNCTION_UNKNOWN
Beschreibung	Die angegebene Funktion ist nicht bekannt.
Beispiele	`MYFUNCTION()`
	`ANOTHERFUN(1;2)`
	`1 + Sin(30)`

Fehlertext	Zu viele Argumente
Fehlernummer	402
Fehlercode	SPSC_F_TOO_MANY_ARGUMENTS
Beschreibung	Es wurde versucht einer Funktion mehr Parameter zu übergeben als zulässig.
Beispiele	`SIN(30; 45)`
	`UCASE("Test"; 1)`
	`SQRT(8; 3)`

Fehlertext	Zu wenige Argumente
Fehlernummer	403
Fehlercode	SPSC_F_TOO_FEW_ARGUMENTS
Beschreibung	Es wurde versucht einer Funktion weniger Parameter zu übergeben als zulässig.
Beispiele	`SIN()`
	`LEFT("Text")`

Fehlertext	Falsche Anzahl an Argumenten
Fehlernummer	404
Fehlercode	SPSC_F_WRONG_ARGUMENT_COUNT
Beschreibung	Die Anzahl der übergebenen Parameter passt zu keiner Funktion. Dieser Fehler kann nur auftreten wenn zu einem Funktionsnamen mehrere Überladungen registriert sind.
Beispiele	`MYFUNC(1;2)`
	`(wenn die Funktion MYFUNC zwei Überladungen hätte, beispielsweise mit einem und mit drei Parametern)`

Fehlertext	Die Typen der Argumente passen zu keiner Funktion
Fehlernummer	405
Fehlercode	SPSC_F_PARAMS_MATCH_NO_OVERLOAD
Beschreibung	Die Typen der übergebenen Parameter passen zu keiner Funktion. Dieser Fehler kann nur auftreten wenn zu einem Funktionsnamen mehrere Überladungen registriert sind, die sich lediglich in den Parametertypen, nicht aber in der Parameteranzahl unterscheiden.
Beispiele	`MYFUNC(1;"ABC")`
	`(wenn die Funktion MYFUNC zwei Überladungen hätte, beispielsweise mit den Signaturen (double; double) und (string; string)`

Anwendungsspezifische Fehler

Fehlertext	Anwendungsspezifischer Fehler
Fehlernummer	500
Fehlercode	SPSC_CUSTOM_FAILURE
Beschreibung	Fehler in applikationsspezifischen Implementierungen der Variablentabelle und/oder Funktionen. Weitergehende Informationen können in diesem Fall über die Methode `getCustomErrorNumber()`, `getCustomErrorText` und `getCustomErrorObject()` ermittelt werden.
Beispiele	**`MYFUNCTION(1;2;3)`**

Syntaktische Fehler

Syntaktische Fehler treten bei fehlenden oder falschen Schlüsselwörtern, unzulässigen Zeichen, mehrdeutigen Eingaben usw. auf.

Fehlertext	Symbol *1* anstatt *2* gefunden
Fehlernummer	1120
Fehlercode	SPSC_MISMATCHED_TOKEN
Beschreibung	Das angegebene Symbol *1* ist an dieser Stelle unzulässig. Stattdessen wird das Symbol *2* erwartet. Dieser Fehler deutet meistens auf falsche Schlüsselwörter oder eine fehlerhafte Klammersetzung hin.
Beispiele	**`IF 0 ELSE 10`**
	`3*((1+2)`
	`SWITCH L CASE 1 THEN 10 ELSE 20`

Fehlertext	Unerwartetes Symbol *1*
Fehlernummer	1140
Fehlercode	SPSC_NO_VIABLE_ALT
Beschreibung	Das angegebene Symbol *1* ist an dieser Stelle unzulässig. Dieser Fehler deutet häufig auf einen fehlenden Parameter oder Operator hin.
Beispiele	**`2 ** 3`**
	`0.5.0`
	`L 2`
	`SWITCH CASE 1 THEN 10 DEFAULT 0`

Fehlertext	Unvollständiger Ausdruck
Fehlernummer	1141
Fehlercode	SPSC_NO_VIABLE_ALT_EOF
Beschreibung	Der Ausdruck endet, ohne einem zulässigen Ausdruck zu genügen. Dieser Fehler deutet auf einen abgeschnittenen Ausdruck hin.
Beispiele	**`1 >`**
	`B -`
	`IF`
	`IF x THEN`
	`SWITCH x CASE 1`

Fehlertext	Unerwartetes Zeichen
Fehlernummer	1220
Fehlercode	SPSC_TOKEN_STREAM_RECOGNITION_FAILURE
Beschreibung	Der Ausdruck beinhaltet unzulässige oder falsch platzierte Zeichen.
Beispiele	`#12`
	`100 + Überstand`
	`0.5.`

Sonstige Fehler

Fehlertext	Unbekannter Fehler
Fehlernummer	10
Fehlercode	SPSC_UNKNOWN_ERROR
Beschreibung	Bei der Analyse des Ausdrucks auftretende schwerwiegende Probleme, z. B. Speichermangel (Stack Overflow), können diesen Fehler verursachen.

Raum für Notizen