

TopoL

GEO-INFORMATIONSSYSTEME

TopoL xT Workflow

ToPas-Makro:
„Text Suchen und Ersetzen“

© 2011 by Dirk Schönewolf, TopoL Support-Center

Alle Rechte vorbehalten. Ohne ausdrückliche, schriftliche Genehmigung ist es nicht gestattet, die Dokumentation oder Teile daraus in irgendeiner Form durch Fotokopie, Mikrofilm oder ein anderes Verfahren zu vervielfältigen oder zu verbreiten. Dasselbe gilt für das Recht der öffentlichen Wiedergabe.

Der Herausgeber macht darauf aufmerksam, daß die genannten Firmen- und Markenzeichen sowie Produktbezeichnungen in der Regel marken-, patent-, oder warenzeichenrechtlichem Schutz unterliegen.

Die Herausgeber übernehmen keine Gewähr für die Funktionsfähigkeit beschriebener Verfahren, Programme oder Algorithmen.



1. EINLEITUNG 4

2. TOPAS-MAKRO ZUM SUCHEN UND ERSETZEN VON TEXT 5

2.1 AUFGABE UND GRUNDBEDINGUNGEN 5
2.2 MAKRODATEI 6
2.3 QUELLTEXT 7
2.7 AUFRUF DER MAKRO-DATEI MIT DEM AUSDRUCKSEDITOR 8

3. ANHANG TOPAS BEFEHLE 11

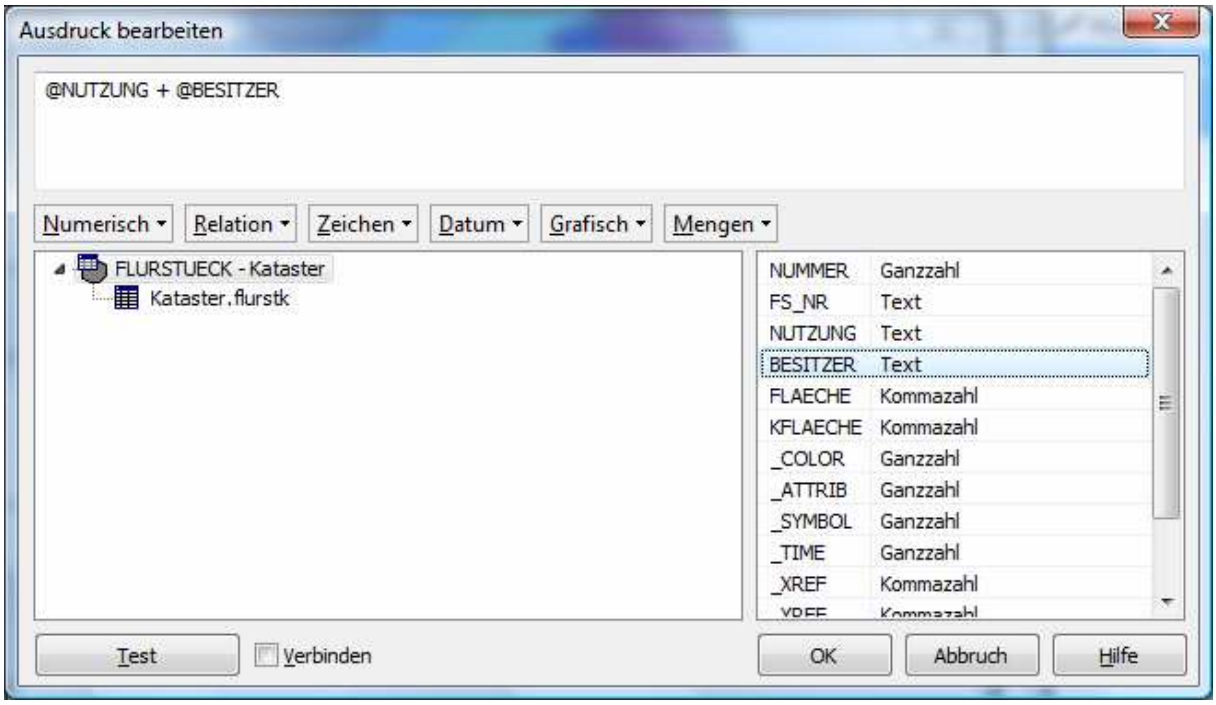
3.1 TEXT-FUNKTIONEN 12
3.2 MATHEMATISCHE FUNKTIONEN 13

1. Einleitung

Zum *„Suchen und Ersetzen“* von Text in Datenbankfeldern bietet sich die TopoL-Makrosprache **ToPas** an. Hierbei handelt es sich um einen *Pascal* bzw. *Delphi* – ähnliche Programmiersprache. ToPas-Makros können u.a. verwendet werden für:

- Beschreibungstext nach Ausdruck
- Farbtabelle nach Ausdruck
- Symboltabelle nach Ausdruck
- Auswahl nach Ausdruck
- Feld überschreiben nach Ausdruck
- Benutzerdefinierte Objektinformation

Im Gegensatz zu den normalen Bordmittlen des TopoL-Ausdruckseditors



können in einer ToPas-Makrodatei die typischen Mittel der Programmierung verwendet werden:

- Schleifen (for, do, while …)
- Variablen
- Bedingungen (if ..)

Hiermit besteht die Möglichkeit komplexe Abfragen mehrerer Variablen durchzuführen.

2. ToPas-Makro zum Suchen und Ersetzen von Text

2.1 Aufgabe und Grundbedingungen

- In einem TopoL-BLK soll in einem Datenbankfeld Text gesucht und ersetzt werden
- Typ des Datenbankfeld ist Text/String
- Es soll nur ein Teil des Textes gesucht und ersetzt werden
- Der zu ersetzende Text hat die gleiche Länge wie der gesuchte Text

Beispiel der BLK-Datenbank:

Projekt\FLURSTK.BLK\FLURSTUECK

NUMMER	FS_NR	NUTZUNG	FLAECHE	KFLAECHE	_COLOR	_ATTRIB
1	310	Sonderkultur	0,188	0,21	12	
2	311	Wald	0,308	0,37	-16640	
3	312	Gartenland	0,09	0,08	-512	
4	313	Gartenland	0,236	0,22	-512	
5	317	Wald	0,143	0,149	-16640	
6	315	Acker	0,236	0,221	14	
7	323	Acker	0,255	0,257	14	
8	325	Grünland	0,508	0,509	9	
9	288	Acker	0,747	0,757	14	
10	286	Acker	0,702	0,71	14	
11	285	Sonderkultur	1,014	1,012	12	
12	283	Acker	0,549	0,566	14	
13	11/01	Acker	2,109	2,09	14	
14	11/02	Grünland	1,858	1,9	9	
15	279	Acker	0,693	0,71	14	
16	319	Wald	0,544	0,567	-16640	
17	276	Grünland	2,38	2,365	9	
18	275	Acker	0,648	0,71	14	

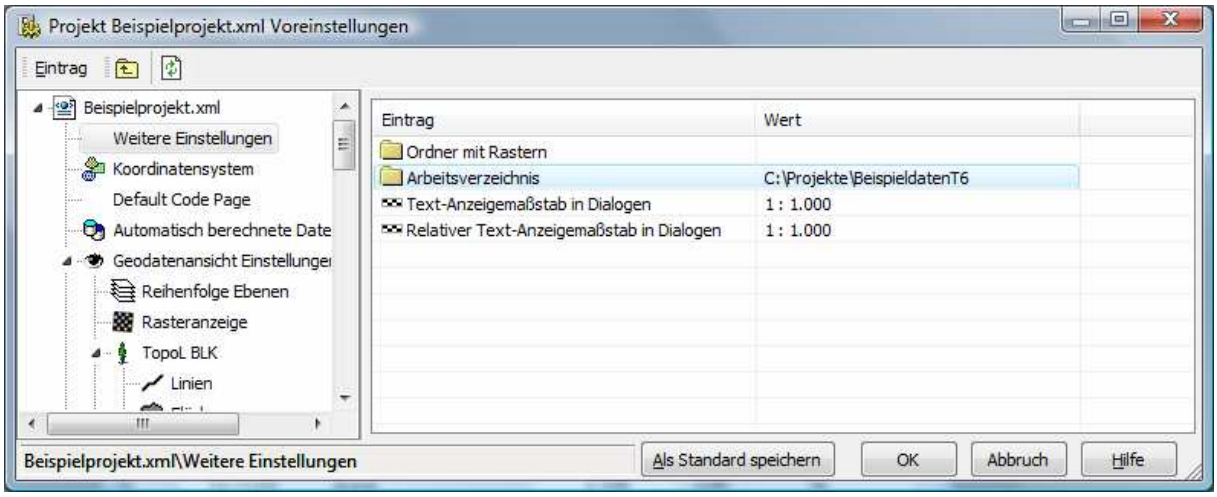
20/20

Datenbankfeld: **FS_NR** Typ: **Text**

Suche: ‚**11/**' Ersetze: ‚**06/**'

2.2 Makrodatei

- ToPas-Makros werden in ASCII-Textdateien erstellt und gespeichert.
- Für Makro-Dateien wird die Dateiendung ***.mac** empfohlen
- Die Makro-Datei sollte im **Arbeitsverzeichnis** gespeichert werden. Das Arbeitsverzeichnis wird im TopoL-Projekt über *Projekt* > *Eigenschaften* > *Weitere Einstellungen* definiert. Wird die Makro-Datei in einem anderen Ordner gespeichert, dann ist beim Zugriff der komplette Pfad mit anzugeben.



- Kommentare und Notizen im Makro werden mit { } gekennzeichnet.

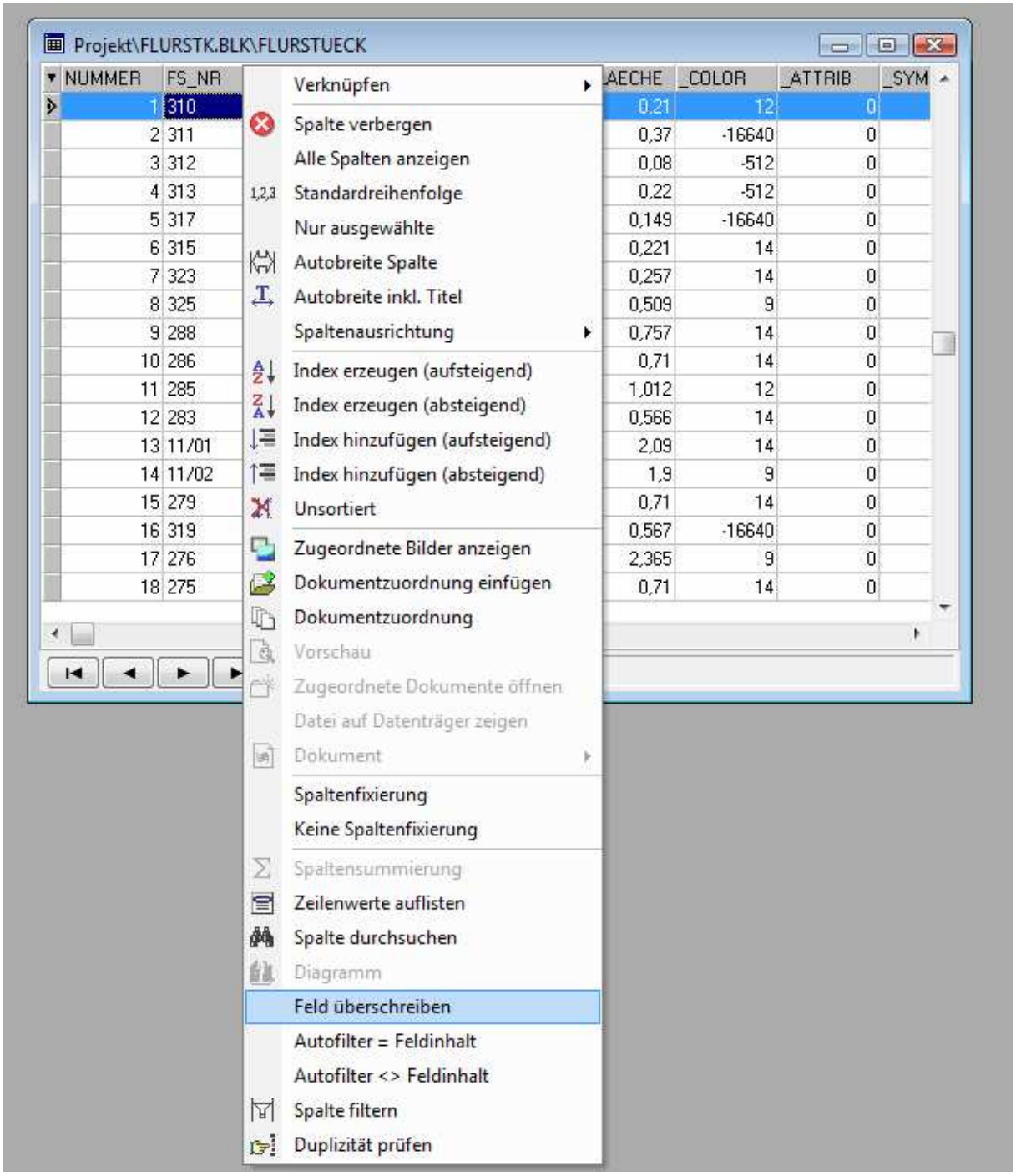
2.3 Quelltext

```
valuemacro TextReplace;

var

 ResultValue, SearchString, ReplaceString, SourceString : String;

 PosSearch, LengthSearch : longint;

begin

 SearchString := '11/';  { <--- Der gesuchte Text in ' '}

 ReplaceString := '06/'; { <--- Der zu ersetzende Text in ' '}

 SourceString := @FS_NR; { <--- Das betroffene Datenbankfeld/Spalte mit einem vorangestellten @ als Kennzeichnung für DB-Felder}

 LengthSearch := StrLength(SearchString);

{ WICHTIG: Dieses Beispiel funktioniert nur wenn der zu suchende Text die gleiche Länge hat wie der zu ersetzende Text }

 if SubStr(SearchString, SourceString) then

          begin

          PosSearch := StrPos(SearchString,SourceString);

       SourceString := StrDelete(SourceString,PosSearch,LengthSearch);

          ResultValue := StrInsert(ReplaceString,SourceString,PosSearch);

          end


          else

           ResultValue := SourceString;


end.
```

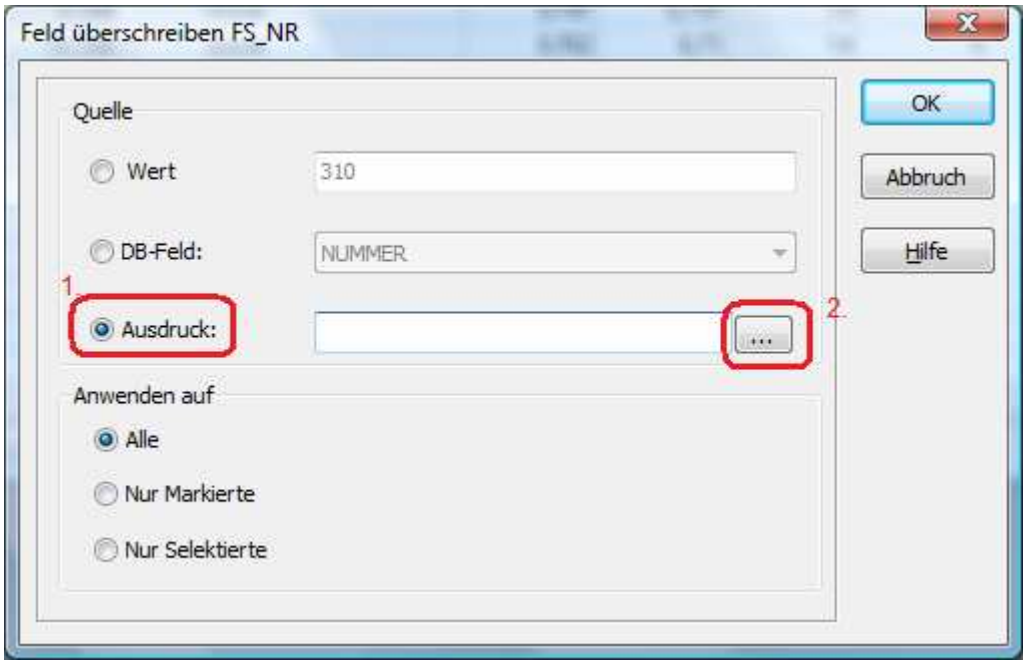
Wichtig: *Ein ToPas-Makro zur Ermittlung von Werten darf generell als Rückgabewert nur **STRINGS** (Text) zurückgeben. Der Rückgabewert wird dabei **immer** der TextVariable **ResultValue** übergeben. Möchte man Zahlenwerte zurückgeben, sind diese auch vorher in Text zu konvertieren (Siehe Anhang oder ToPas-Doku).*

2.7 Aufruf der Makro-Datei mit dem Ausdruckseditor

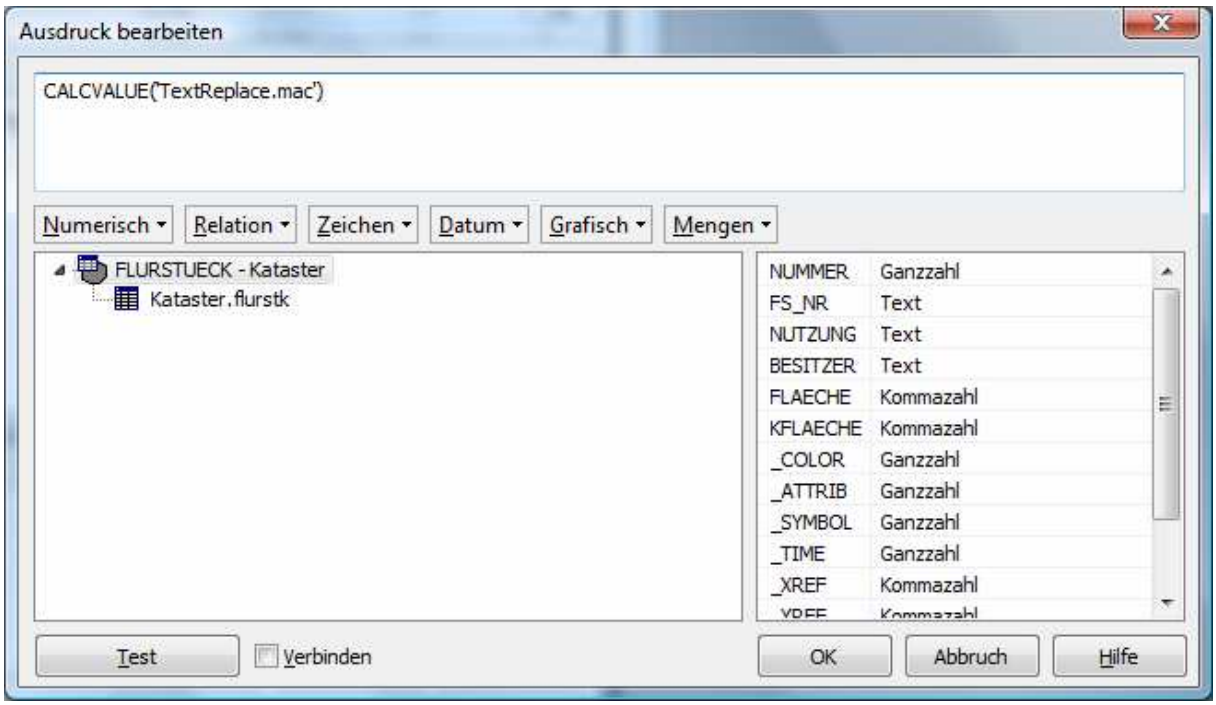
1. Öffnen des *Datenbankansicht* - Fensters
2. Anklicken der zu überschreibenden Spalte mit der rechten Maustaste
3. Die Funktion *Feld überschreiben* auswählen:



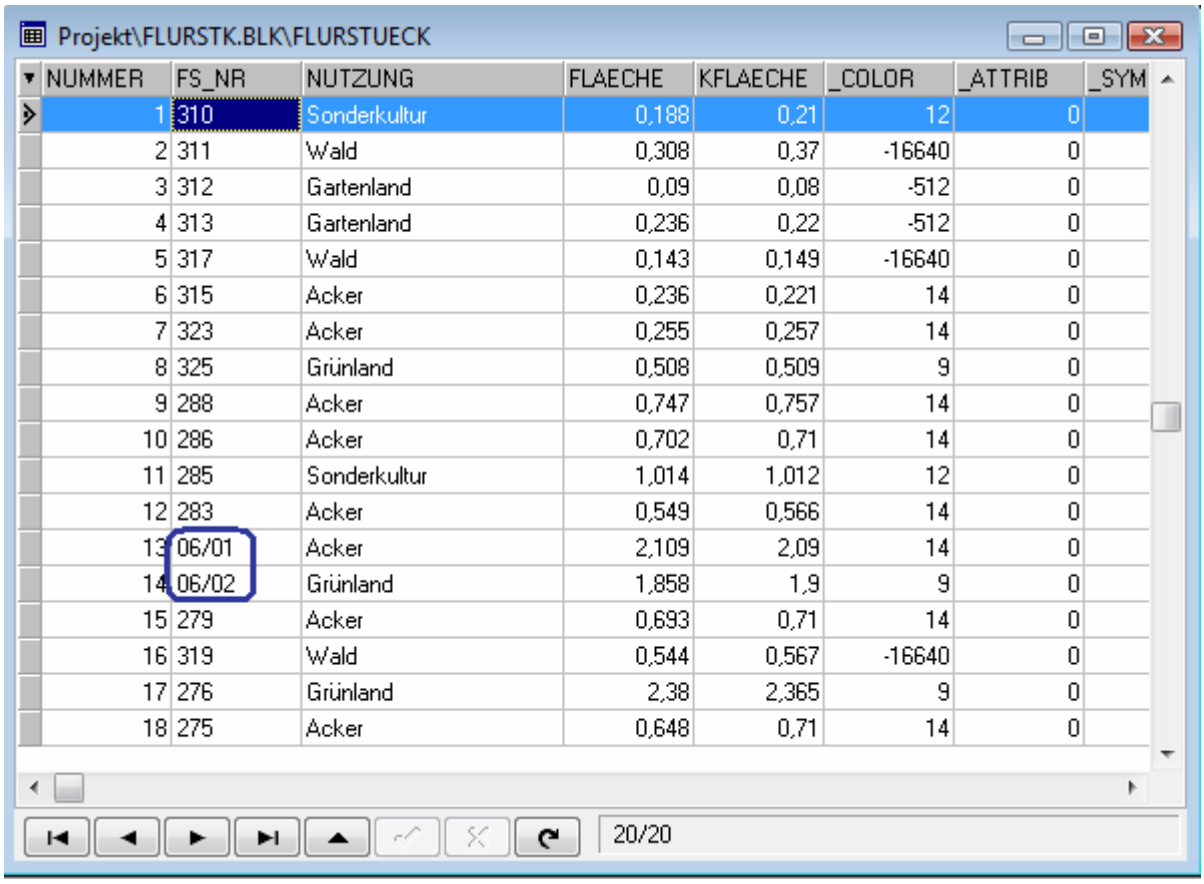
4. nach *Ausdruck* definieren:



5. Einen Ausdruck *Neu* erstellen und im Ausdruckseditor folgenden Befehl eintragen: **CALCVALUE('TextReplace.mac')**



6. Über *Test* die Syntax überprüfen und dann mit mehreren *OK* zurück und das Resultat schaut wie folgt aus:



Projekt\FLURSTK.BLK\FLURSTUECK

NUMMER	FS_NR	NUTZUNG	FLAECHE	KFLAECHE	_COLOR	_ATTRIB	_SYM
1	310	Sonderkultur	0,188	0,21	12	0	
2	311	Wald	0,308	0,37	-16640	0	
3	312	Gartenland	0,09	0,08	-512	0	
4	313	Gartenland	0,236	0,22	-512	0	
5	317	Wald	0,143	0,149	-16640	0	
6	315	Acker	0,236	0,221	14	0	
7	323	Acker	0,255	0,257	14	0	
8	325	Grünland	0,508	0,509	9	0	
9	288	Acker	0,747	0,757	14	0	
10	286	Acker	0,702	0,71	14	0	
11	285	Sonderkultur	1,014	1,012	12	0	
12	283	Acker	0,549	0,566	14	0	
13	06/01	Acker	2,109	2,09	14	0	
14	06/02	Grünland	1,858	1,9	9	0	
15	279	Acker	0,693	0,71	14	0	
16	319	Wald	0,544	0,567	-16640	0	
17	276	Grünland	2,38	2,365	9	0	
18	275	Acker	0,648	0,71	14	0	

20/20

Hinweis: *Die Funktion **CALCVALUE(,MACRODATEI')** ruft die Makrodatei auf und erwartet als Rückgabewert immer einen **STRING** (Textwert).*

3. Anhang ToPas Befehle

Nachfogend werden einige **ToPas**-Befehle aufgeführt. Die komplette Referenz finden Sie im Dokument „**TopoL for Windows - ToPas language**“, verfügbar zum Download in einer ZIP-Datei mit Beispiel-Scripten unter :

http://www.topol.de/SUPPORT/files/help/ToPas_Manuals_6x.zip



3.1 Text-Funktionen

`Char(expression, index)`
returns a string containing a single character located in the string `expression` on the position `index`
`Ord(expression, index)`
returns an integer code of a single character located in the string `expression` on the position `index`
`Chr(expression)`
returns a string containing a single character, to which is converted the integer value of the expression
`StrLength(expression)`
returns an integer length of the string `expression`
`StrUpCase(expression)`
returns a string `expression` converted to uppercase
`StrLowCase(expression)`
returns a string `expression` converted to lowercase
`StrTrim(expression)`
returns a string `expression` without leading and trailing spaces
`StrInTrim(expression)`
returns a string `expression` without leading and trailing spaces as well as without duplicate spaces inside the string
`Language`
returns an integer code of the current character code page (character encoding for the current block) - typical results: 1252 (ANSI Windows), 1251 (ANSI Windows Cyr), 1250 (ANSI Windows CE), 437 (ASCII), 850 (PC Latin 1), 852 (PC Latin2), 867 (Kamenických encoding)
`StrConvToAsc(expression)`
returns the string `expression` converted to ASCII code (diacritics removal)
`StrConvToKam(expression)`
returns the string `expression` converted from the current code page into Kamenických encoding
`StrConvToLat2(expression)`
returns the string `expression` converted from the current code page into PC Latin 2 encoding
`PrefStr(prefix, expression)`
returns TRUE, if the string `prefix` is found at the beginning of the string `expression`, otherwise FALSE
`SubStr(substring, expression)`
returns TRUE, if the string `substring` is found within the string `expression`, otherwise FALSE
`StrPos(substring, expression)`
returns an integer value that is the index of the first character of the first occurrence of the string `substring` within the string `expression`, if not found returns value of 0
`Like(v1, v2)`
returns TRUE, if the strings `v1, v2` are similar (are different only in letter case, redundant spaces and diacritics), otherwise FALSE
`StrCopy(expression, index, count)`
returns a string containing `count` characters copied from the string `expression` starting at the position `index`
`StrDelete(expression, index, count)`
returns a string after the removal of `count` characters starting at the position `index` from the string `expression`
`StrInsert(source, expression, index)`
returns a string after insert of `source` string into the string `expression` at the position `index`

3.2 Mathematische Funktionen

`Abs(expression)`
returns the absolute value of the expression
`Odd(expression)`
returns TRUE for an odd result of expression and FALSE for an even one
`Round(expression)`
returns value of the expression rounded to an integer value
`Trunc(expression)`
returns the value of the expression truncated (rounded toward zero) to an integer value
`Int(expression)`
returns the integer part of the value of the expression
`Frac(expression)`
returns the fractional part of the value of the expression
`Sqr(expression)`
returns the square of the value of the expression
`Sqrt(expression)`
returns the square root of the value of the expression
`Exp(expression)`
returns the exponential of the value of the expression, i.e. the base of the natural logarithms raised to the power of the expression
`Ln(expression)`
returns the natural logarithm of the value of the expression
`Pi`
returns the value of Pi (i.e. 3.1415926535897932385)
`Sin(expression)`
returns the sine of the value of the expression (expressed in radians)
`Cos(expression)`
returns the cosine of the value of the expression (expressed in radians)
`Tan(expression)`
returns the tangent of the value of the expression (expressed in radians)
`ArcTan(expression)`
returns the arctangent of the value of the expression (expressed in radians)