Bearbeiten von Zeichenketten in S7-SCL

1 Aufbau eine Zeichenkette

Der Datentyp STRING definiert eine Zeichenkette von maximal 254 Einzelzeichen.
Der Standardbereich, der für eine Zeichenkette reserviert ist, besteht aus 256 Bytes. Dieser Platz wird benötigt, um 254 Zeichen und einen Kopf von 2 Byte zu speichern.
Im Kopf wird die maximale und die aktuelle Länge des Strings gespeichert.
In einer Zeichenkette sind alle Zeichen des ASCII-Codes zugelassen. Ein String kann auch Sonderzeichen, wie Steuerzeichen und nichtdruckbare Zeichen, enthalten.

2 Deklaration

Bei der Deklaration von Speicherplatz für Zeichenketten kann die maximale Anzahl der Zeichen, die in der Zeichenkette gespeichert werden sollen, definiert werden.
Wird hier keine Maximallänge angegeben, wird ein String der Länge 254 angelegt.

Beispiel :

```
VAR
        Text1   : STRING[123];
        Text2   : STRING;
END_VAR
```

Die Konstante „123“ bei der Deklaration der Variablen „Text1“ steht für die maximale Anzahl der Zeichen in der Zeichenkette.
Bei der Variablen „Text2“ wird eine Länge von 254 Zeichen reserviert.

Hinweis
Bei Rückgabewerten, sowie bei Ausgangs- und Durchgangsparametern, vom Typ STRING einer Funktion wird vom SCL Compiler angenommen, daß ein STRING der Länge 254 zurückgegeben wird. Deshalb legt der SCL Compiler eine temporäre Variable von 254 Zeichen Länge für den Funktionswert auf dem Stack an. Dies kann zu einem Laufzeitfehler durch Stacküberlauf führen.
Um dies zu vermeiden und um den Speicher Ihrer CPU besser zu nutzen, kann dieser Wert auf die im Programm erforderliche Länge reduziert werden, indem Sie vor der Übersetzung im SCL Editor den Menübefehl **Extras / Einstellungen** wählen und im folgenden Dialogfeld im Register “Compiler” die im Programm erforderliche “Maximale Stringlänge” eintragen.

Achtung!! Die Richtigkeit der Einstellung kann von SCL nicht überprüft werden!!

3 Initialisierung von Zeichenketten

Stringvariable lassen sich, wie andere Variablen auch, bei der Deklaration der Parameter von Funktionsbausteinen (FBs) mit konstanten Zeichenfolgen initialisieren.
Bei den Parametern von Funktionen (FCs) ist keine Initialisierung möglich.

Werden temporäre Variable (in VAR bei FCs und in VAR_TEMP bei FCs und FBs) vom Typ STRING z. B. zur Zwischenspeicherung von Ergebnissen benötigt, dann müssen diese in jedem Fall vor ihrer erstmaligen Verwendung mit einer Stringkonstanten beschrieben werden. Hierdurch wird der korrekte Stringaufbau sichergestellt.

Beispiel:

```
FUNCTION Test : STRING[45]

VAR
  x : STRING[45];
END_VAR

x := 'a';
x := concat (in1 := x, in2 := x);
Test := x;
END_FUNCTION
```

Ohne die Initialisierung **x := 'a';** würde die Funktion ein falsches Ergebnis liefern.

4 Sonderzeichen in Zeichenketten

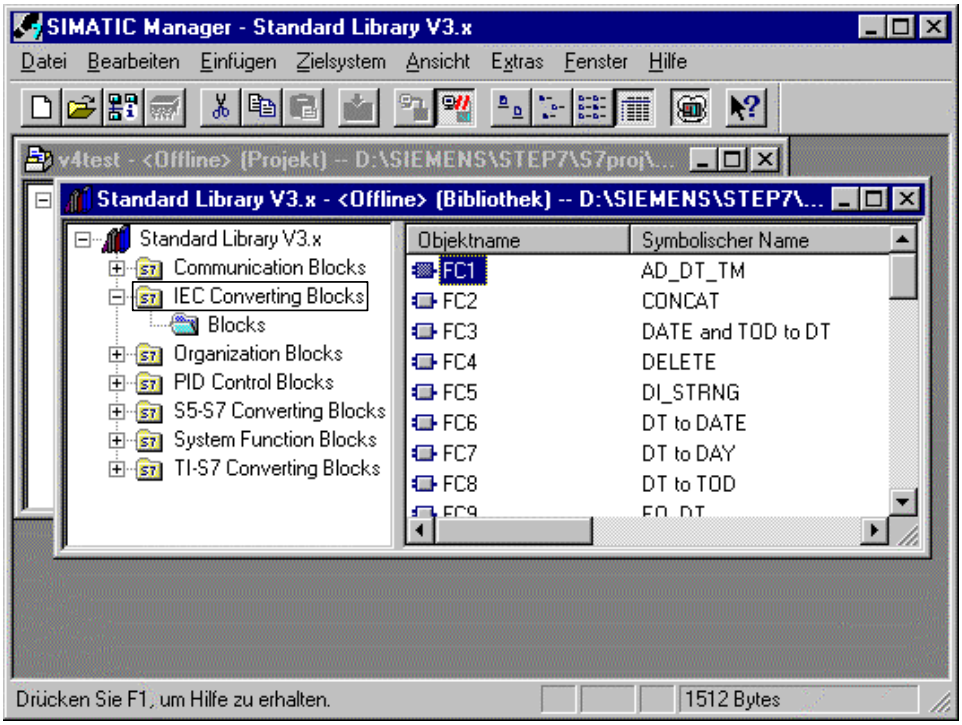
Sonderzeichen in Zeichenketten wie z.B. Druckersteuerzeichen wie Zeilenvorschub, Wagenrücklauf,...können über die Syntax $hh eingeben, wobei hh stellvertretend für den hexadezimal ausgedrückten Wert des ASCII-Zeichens steht.

Beispiel:

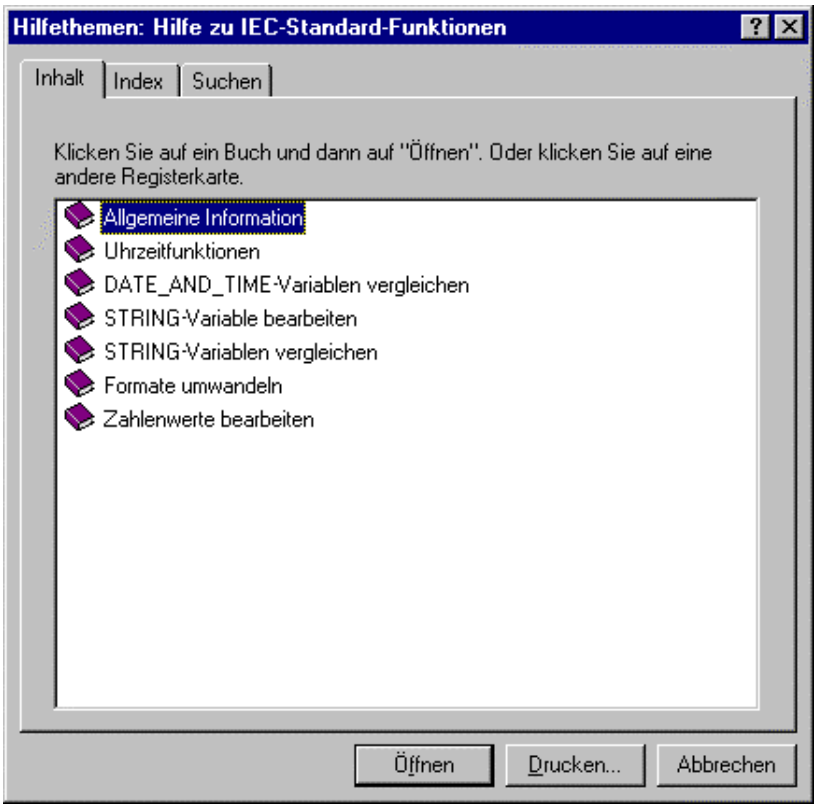
```
'Text1'$0D$0AText2'
```

5 Vorhandene Standardfunktionen

Die mit STEP7 gelieferte Standardlibrary enthält auch Standardfunktionen zum Manipulation und Vergleich von Zeichenketten, sowie Konvertierungsfunktionen.



Die Funktionen sind im folgenden kurz beschrieben, ein Aufrufschablone ist beigefügt. Die ausführliche Beschreibung der Funktionen ist im Hilfesystem von STEP 7 vorhanden. Das richtige Kapitel der Hilfe erreichen sie durch Anwahl einer Funktion der Library und Drücken der Taste F1. Über die Registerlasche „Inhalt“ kann schnell zu allen verfügbaren Funktionen gesprungen werden.



5.1 Manipulation an Zeichenketten

5.1.1 LEN

```
LEN (S :=  // IN: STRING
    ); // INT
```

Die Funktion LEN (FC21) gibt die aktuelle Länge einer Zeichenkette (Anzahl der gültigen Zeichen) als Rückgabewert aus.
Ein Leerstring ('') hat die Länge Null.
Die Funktion meldet keine Fehler.

5.1.2 CONCAT

```
CONCAT (IN1 :=  // IN: STRING
       ,IN2 :=  // IN: STRING
       ); // STRING
```

Die Funktion CONCAT (FC2) faßt zwei STRING-Variable zu einer Zeichenkette zusammen.
Ist die Ergebniszeichenkette länger als die am Ausgangsparameter angelegt Variable, wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt.
Fehler können über das OK Flag abgefragt werden.

5.1.3 LEFT bzw. Right

```
LEFT (IN :=  // IN: STRING
     ,L :=  // IN: INT
     ); // STRING


RIGHT (IN :=  // IN: STRING
      ,L :=  // IN: INT
      ); // STRING
```

Die Funktion LEFT bzw. RIGHT (FC20 bzw. FC32) liefert die ersten bzw. letzten L Zeichen einer Zeichenkette.
Ist L größer als die aktuelle Länge der STRING-Variablen, wird der komplette String zurückgeliefert.
Bei L = 0 wird ein Leerstring zurückgeliefert.
Ist L negativ wird ein Leerstring ausgegeben und das OK Flag auf „0“gesetzt.

5.1.4 MID

```
MID (IN :=  // IN: STRING
    ,L :=  // IN: INT
    ,P :=  // IN: INT
    ); // STRING
```

Die Funktion MID (FC26) liefert den mittleren Teil einer Zeichenkette (L Zeichen ab dem P. Zeichen einschließlich).
Geht die Summe aus L und (P-1) über die aktuelle Länge der STRING-Variablen hinaus, wird eine Zeichenkette ab dem P. Zeichen bis zum Ende des Eingangswerts geliefert. In allen anderen Fällen (P liegt außerhalb der aktuellen Länge, P und/oder L gleich Null oder negativ) wird ein Leerstring ausgegeben und das OK Flag auf "0“ gesetzt.

5.1.5 INSERT

```
INSERT (IN1 :=  // IN: STRING
       ,IN2 :=  // IN: STRING
       ,P :=  // IN: INT
       ); // STRING
```

Die Funktion INSERT (FC17) fügt die Zeichenkette am Parameter IN2 in die Zeichenkette am Parameter IN1 nach dem P. Zeichen ein.
Ist P gleich Null, wird die zweite Zeichenkette vor der ersten Zeichenkette eingefügt. Ist P größer als die aktuelle Länge der ersten Zeichenkette, wird die zweite Zeichenkette an die erste angehängt. Ist P negativ wird ein Leerstring ausgegeben und das Ok Flag auf ”0" gesetzt. Das OK Flag wird auch auf ”0" gesetzt, wenn die Ergebniszeichenkette länger ist als die am Ausgangsparameter angegebene Variable; in diesem Fall wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt.

5.1.6 DELETE

```
DELETE (IN :=  // IN: STRING
       ,L :=  // IN: INT
       ,P :=  // IN: INT
       ); // STRING
```

Die Funktion DELETE (FC4) löscht in einer Zeichenkette L Zeichen ab dem P. Zeichen (einschließlich).
Ist L und/oder P gleich Null oder ist P größer als die aktuelle Länge der Eingangszeichenkette, wird die Eingangszeichenkette zurückgeliefert. Ist die Summe aus L und P größer als die Eingangszeichenkette, wird bis zum Ende der Zeichenkette gelöscht. Ist L und/oder P negativ wird ein Leerstring ausgegeben und das OK Flag auf ”0" gesetzt.

5.1.7 REPLACE

```
REPLACE (IN1 :=  // IN: STRING
        ,IN2 :=  // IN: STRING
        ,L :=  // IN: INT
        ,P :=  // IN: INT
        ); // STRING
```

Die Funktion REPLACE (FC31) ersetzt L Zeichen der ersten Zeichenkette (IN1) ab dem P. Zeichen (einschließlich) durch die zweite Zeichenkette (IN2).
Ist L gleich Null wird die erste Zeichenkette zurückgeliefert. Ist P gleich Null oder Eins wird ab dem 1. Zeichen (einschließlich) ersetzt. Liegt P außerhalb der ersten Zeichenkette, wird die zweite Zeichenkette an die erste Zeichenkette angehängt. Ist L und/oder P negativ wird ein Leerstring ausgegeben und das OK Flag auf ”0" gesetzt. Das OK Flag wird auch auf ”0" gesetzt, wenn die Ergebniszeichenkette länger ist als die am Ausgangsparameter angegebene Variable; in diesem Fall wird die Ergebniszeichenkette auf die maximal eingerichtete Länge begrenzt.

5.1.8 FIND

```
FIND (IN1 :=  // IN: STRING
     ,IN2 :=  // IN: STRING
     ); // INT
```

Die Funktion FIND (FC11) liefert die Position der zweiten Zeichenkette (IN2) innerhalb der ersten Zeichenkette (IN1).
Die Suche beginnt links; es wird das erste Auftreten der Zeichenkette gemeldet. Ist die zweite Zeichenkette in der ersten nicht vorhanden, wird Null zurückgemeldet. Die Funktion meldet keine Fehler.

5.2 Zeichenkettenvergleiche

Vergleiche mit Zeichenketten als Opranden sind in SCL mit den normalen Vergleichsoperatoren, also ==, <>, <, <=,> und >=, möglich. Der Compiler baut den ensprechenden Funktionsaufruf automatisch ein. Die Funktionen sind hier nur verständnishalber aufgeführt, um die Art des Vergleiches bei Zeichenketten zu dokumentieren.

5.2.1 EQ_STRNG und NE_STRNG

Die Funktion EQ_STRNG (FC10) bzw. NE_STRNG (FC29) vergleicht die Inhalte zweier Variablen im Format STRING auf gleich (ungleich) und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand ”1", wenn die Zeichenkette am Parameter S1 gleich (bzw. ungleich) der Zeichenkette am Parameter S2 ist. Die Funktion meldet keine Fehler

5.2.2 GE_STRNG und LE_STRNG

Die Funktion GE_STRNG (FC13) bzw. LE_STRNG (FC19)vergleicht die Inhalte zweier Variablen im Format STRING auf größer (kleiner) oder gleich und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand ”1", wenn die Zeichenkette am Parameter S1 größer (kleiner) oder gleich der Zeichenkette am Parameter S2 ist. Die Zeichen werden beginnend von links über ihre ASCII-Codierung verglichen (z.B. ist `a' größer als `A'). Das erste unterschiedliche Zeichen entscheidet über das Vergleichsergebnis. Ist der linke Teil der längeren Zeichenkette identisch mit der kürzeren Zeichenkette, gilt die längere Zeichenkette als größer. Die Funktion meldet keine Fehler.

5.2.3 GT_STRNG und LT_STRNG

Die Funktion GT_STRNG (FC15) bzw. LT_STRNG (FC24) vergleicht die Inhalte zweier Variablen im STRING-Format auf größer (kleiner) und gibt das Vergleichsergebnis als Rückgabewert aus. Der Rückgabewert führt Signalzustand ”1", wenn die Zeichenkette am Parameter S1 größer (kleiner) als die Zeichenkette am Parameter S2 ist. Die Zeichen werden beginnend von links über ihre ASCII-Codierung verglichen (z.B. ist `a' größer als `A'). Das erste unterschiedliche Zeichen entscheidet über das Vergleichsergebnis. Ist der linke Teil der längeren Zeichenkette identisch mit der kürzeren Zeichenkette, gilt die längere Zeichenkette als größer. Die Funktion meldet keine Fehler.

5.3 Datentypwandlungen

5.3.1 I_STRNG und STRNG_I

```
I_STRNG (I :=  // IN: INT
        ); // STRING

STRNG_I (S :=  // IN: STRING
        ); // INT
```

Die Funktion I_STRNG (FC16) wandelt eine Variable im INT-Format in eine Zeichenkette, bzw. STRNG_I (FC38) eine Zeichenkette in das INT-Format.
Die Zeichenkette wird mit einem führenden Vorzeichen darstellt. Ist die am Rückgabeparameter angegebene Variable zu kurz, findet keine Wandlung statt und das OK Flag wird auf ”0" gesetzt.

5.3.2 DI_STRNG und STRNG_DI

```
DI_STRNG (I :=  // IN: DINT
```

```
         ); // STRING

STRNG_DI (S :=  // IN: STRING
         ); // DINT
```

Die Funktion DI_STRNG (FC5) wandelt eine Variable im DINT-Format in eine Zeichenkette, bzw. STRNG_DI (FC37) eine Zeichenkette in das DINT-Format.
Die Zeichenkette wird mit einem führenden Vorzeichen darstellt. Ist die am Rückgabeparameter angegebene Variable zu kurz, findet keine Wandlung statt und das OK Flag wird auf ”0" gesetzt.

5.3.3 R_STRNG und STRNG_R

```
R_STRNG (IN :=  // IN: REAL
        ); // STRING

STRNG_R (S :=  // IN: STRING
        ); // REAL
```

Die Funktion R_STRNG (FC30) wandelt eine Variable im REAL-Format in eine Zeichenkette, bzw. STRNG_R(FC39) eine Zeichenkette in das REAL-Format.
Die Zeichenkette wird mit 14 Stellen darstellt:

±v.nnnnnnnE±xx

Ist die am Rückgabeparameter angegebene Variable zu kurz oder liegt am Parameter IN keine gültige Gleitpunktzahl an, findet keine Wandlung statt und das OK Flag wird auf ”0" gesetzt.

6 Beispiel

6.1 Meldetexte zusammensetzen

```
(*****  Meldetexte prozeßgesteuert zusammensetzen und abspeichern  ****)

(*********************************************************************
** Der Baustein enthält die benötigten Meldetexte und die letzten 20
**    generierten Meldungen
*********************************************************************)

DATA_BLOCK Meldetexte
  STRUCT
    Index  : INT;
    textpuffer : ARRAY [0..19] OF
                    STRING[30];
    HW    : ARRAY [1..5] of STRING[15];  (* 5 verschiedene Geräte *)
    stati : ARRAY [1..5] of STRING[11];  (* 5 verschiedene Zustände *)
  END_STRUCT
BEGIN
  Index:=0;
  HW[1] := 'Motor ';
  HW[2] := 'Ventil ';
  HW[3] := 'Presse ';
  HW[4] := 'Schweisstation ';
  HW[5] := 'Brenner ';
  Stati[1] := ' gestört';
  Stati[2] := ' gestartet';
  Stati[3] := ' Temperatur';
  Stati[4] := ' repariert';
  Stati[5] := ' gewartet';
END_DATA_BLOCK

(*********************************************************************
Die Function setzt Meldetexte zusammen und trägt sie in den DB Meldetexte
ein. Die Meldetexte werden in einem Umlaufpuffer abgelegt. Der nächste
freie Index des Textpuffers steht ebenfalls im DB Meldetexte und wird
von der Funktion aktualisiert.
*********************************************************************)
FUNCTION Textgenerator : BOOL;

VAR_INPUT
  unit   : INT;       // Index des Gerätetextes
  nr     : INT;       // IDNr. des Gerätes
  status : INT;
  wert   : INT;
END_VAR
VAR
   text : string[31];
   i : int;
END_VAR
  // initialisierung der temporären Variablen
  text  := '';
  Textgenerator := TRUE;
  CASE unit OF
    1..5 :
      CASE status OF
        1..5 : text := CONCAT(in1 := Meldetexte.HW[unit],
                              in2 := RIGHT(l:=2,in:=I_STRNG(nr)));
               text := CONCAT(in1 := text,
                              in2 := Meldetexte.stati[status]);
               IF wert <> 0 THEN
                  text := CONCAT(in1 := text,
                                 in2 := I_STRNG(wert));
               END_IF;
        ELSE Textgenerator := FALSE;
      END_CASE;
    ELSE Textgenerator := FALSE;
  END_CASE;
  i := Meldetexte.index;
  Meldetexte.textpuffer[i] := text;
  Meldetexte.index := (i+1) mod 20;
END_FUNCTION

(*********************************************************************
 Die Function wird im zyklischen Programm bei Flankenwechsel
 im %M10.0 aufgerufen  und bewirkt den einmaligen Eintrag einer
 Meldung,  falls sich ein Parameter ändert.
*********************************************************************)
ORGANIZATION_BLOCK zyklus

VAR_TEMP
       Besy_ifx : ARRAY [0..20] OF BYTE;
       fehler : BOOL;
```

```
END_VAR
(*************************************************************************
 Der folgende Aufruf bewirkt den Eintrag "Motor 12 gestartet " im
 Textpuffer des DB Meldetexte, wobei über %MW0 eine 1, über %EW2 eine 12
 und über %MW2 eine 2 zugeführt wird.
*************************************************************************)

IF %M10.0 <> %M10.1 THEN
  fehler := Textgenerator (unit   := WORD_TO_INT(%MW0),
                           nr     := WORD_TO_INT(%EW2),
                           status := WORD_TO_INT(%MW2),
                           wert   := 0);
  %M10.1 := %M10.0;
END_IF;

END_ORGANIZATION_BLOCK
```