

Working with Strings in S7-SCL

1 Structure of a string

The Datatype STRING defines a string of maximal 254 characters. The field reserved for a standard string is 256 Bytes. This space is required in order to save 254 characters and a string-head of 2 bytes.

The string-head contains the maximal and the actual length of the string.

In a string all characters of the ASCII-Code can be used. A string can even contain special characters like control keys and non printable characters.

2 Declaration

In the string declaration the maximal characters of the string can be indicated. If it is not indicated the string will have the default maximal value of 254 characters.

Example :

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

The constant „123“ at the declaration of the „Text1“ variable represents the maximal length of characters reserved for the string. The variable „Text2“ will have the default length of 254 characters.

Note

By Return-Value or OUTPUT- and IN_OUT-Parameter of the type STRING in a function the SCL-Compiler assumes that the String to be returned has the length 254. That's why the SCL Compiler puts a temporary variable of 254 characters for the returned value on the stack. This might cause a runtime error because of a stack overflow.

In order to avoid this and optimize memory space of the CPU the required space can be reduced to the really needed length. Therefore the compiler option „Maximal String Length“ in the menu **Extras / Customize** has to be set, before compiling the block.

Attention: The correctness of the user settings in S7-SCL cannot be checked by the system !!

3 Initialising Strings

Strings, like other variables, can be initialised with a constant amount of characters in the variable declaration section.

If temporary variables of type STRING are used (in the VAR section of FCs or in the VAR_TEMP section of FCs and FBs) they have to be initialised before their first use. This is necessary in order to guarantee the correct string structure.

Example:

```

FUNCTION Test : STRING[45]

VAR
  x : STRING[45];
END_VAR

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

```

Without the initialisation `x := 'a'`; the function would deliver a wrong result.

4 Special characters in strings

Special characters like for example carriage return or line feed can be entered with the synthax `$hh`, where `hh` indicated the hexadecimal value of the corresponding ASCII-character.

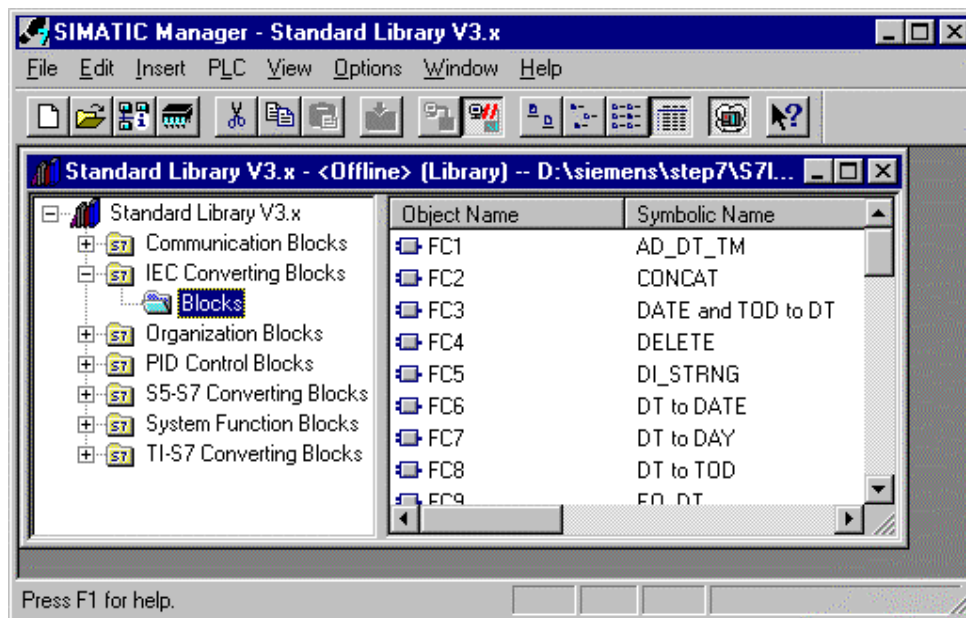
Example:

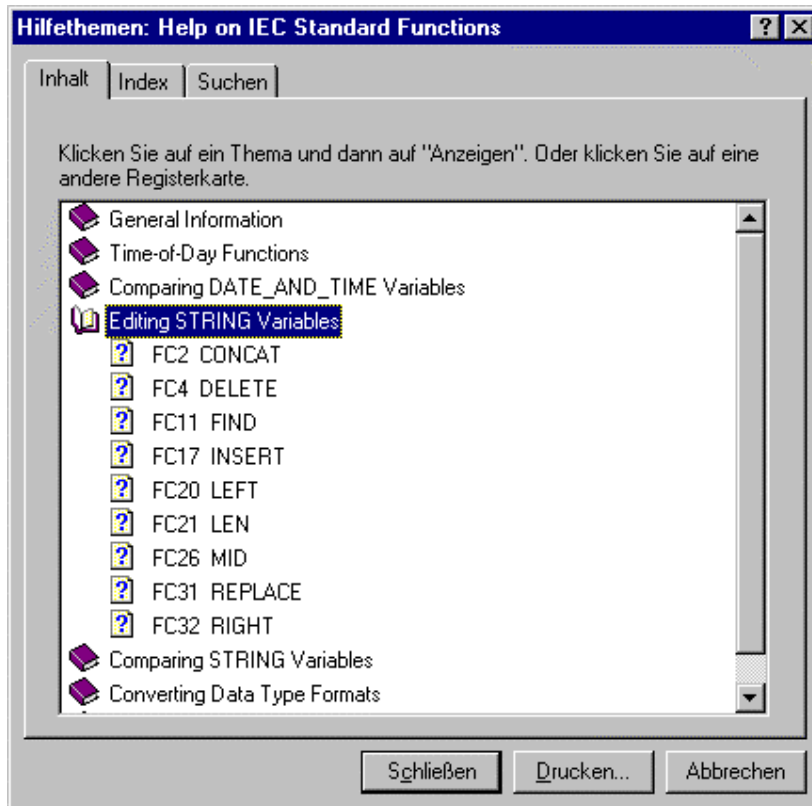
```
'Text1$0D$0AText2'
```

5 Existing standard functions

STEP7 contains a Standard-Library with standard functions for the handling and comparison of strings or for string conversions.

These functions will be shortly explained here, with a synthax description. A complete description can be find in the online-Help system of STEP7. The correspondant chapter can be find by selecting a function of the library and pressing F1. In „Contents“ each function is explained in details.





5.1 Editing String Variables

5.1.1 LEN (FC21)

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

A STRING variable contains two lengths: the maximum length (this is given in square brackets when the variables are being defined) and the current length (this is the number of currently valid characters). The current length must be smaller than or equal to the maximum length. The function FC21 outputs the current length of a string (number of valid characters) as a return value. A blank string (' ') has the length zero. The maximum length is 254. The function does not report any errors.

5.1.2 CONCAT (FC2)

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

The function FC2 concatenates two STRING variables together to form one string. If the resulting string is longer than the variable given at the output parameter, the result string is limited to the maximum set length.

Errors can be determined through the OK-Flag.

5.1.3 LEFT resp. Right (FC20 resp. FC32)

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

```

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

```

The function LEFT (FC20) resp. RIGHT (FC32) provides the first resp. The last L characters of a string (where L stands for a number). If L is greater than the current length of the STRING variables, the input value is returned. With L = 0 and with a blank string as the input value, a blank string is returned. If L is negative, a blank string is returned and the OK-Flag is set to "0".

5.1.4 MID (FC26)

```

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

```

The function MID (FC26) provides the middle part of a string (L characters from the character P inclusive). If the sum of L and P exceeds the current length of the STRING variables, a string is returned from the character P to the end of the input value. In all other cases (P is outside the current length, P and/or L are equal to zero or negative), a blank string is returned and the OK-Flag is set to "0".

5.1.5 INSERT (FC17)

```

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

```

The function INSERT (FC17) inserts a string at parameter IN2 into the string at parameter IN1 after the character at position P. If P equals zero, the second string is inserted before the first string. If P is greater than the current length of the first string, the second string is appended to the first, If P is negative, a blank string is output and the OK-Flag is set to "0". The OK-Flag is also set to "0" if the resulting string is longer than the variable given at the output parameter; in this case the result string is limited to the maximum set length.

5.1.6 DELETE (FC4)

```

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

```

The function DELETE (FC4) deletes a number of characters (L) from the character at position P (inclusive) in a string. If L and/or P are equal to zero or if P is greater than the current length of the input string, the input string is returned. If the sum of L and P is greater than the input string, the string is deleted up to the end. If L and/or P are negative, a blank string is returned and the OK-Flag is set to "0".

5.1.7 REPLACE (FC31)

```

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

```

The function REPLACE (FC31) replaces a number of characters (L) of the first string (IN1) from the character at position P (inclusive) with the second string (IN2). If L is equal to zero, the first string is returned. If P is equal to zero or one, the string is replaced from the first character (inclusive). If P is outside the first string, the second string is appended to the first string. If L and/or P is negative, a blank string is returned and the OK-Flag is set to "0". The OK-Flag is also set to "0" if the resulting string is longer than the variable given at the output parameter; in this case the result string is limited to the maximum set length.

5.1.8 FIND (FC11)

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

The function FIND (FC11) provides the position of the second string (IN2) within the first string (IN1). The search starts on the left; the first occurrence of the string is reported. If the second string is not found in the first, zero is returned. The function does not report any errors.

5.2 Comparing Strings

In SCL Strings can be compared with the normal comparison operators, which means ==, <>, <, <=, > und >= . The compiler will call automatically the needed function. For the comprehension these functions will be explained in the following.

5.2.1 EQ_STRNG (FC10) resp. NE_STRNG (FC29)

The function EQ_STRNG (FC10) resp. NE_STRNG (FC29) compares the contents of two variables in the data type format STRING to find out if they are equal resp. unequal and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter S1 is equal resp. not equal to the string at parameter S2.

The function does not report any errors.

5.2.2 GE_STRNG (FC13) resp. LE_STRNG (FC19)

The function GE_STRNG (FC13) resp. LE_STRNG (FC19) compares the contents of two variables in the data type format STRING to find out if the first is greater (resp. smaller) than or equal to the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter S1 is greater (resp. smaller) than or equal to the string at parameter S2.

The characters are compared by their ASCII code (for example, 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the first characters are the same, the longer string is greater.

The function does not report any errors.

5.2.3 GT_STRNG (FC15) resp. LT_STRNG (FC24)

The function GT_STRNG (FC15) resp. LT_STRNG (FC24) compares the contents of two variables in the data type format STRING to find out if the first is greater resp. smaller than the other and outputs the result of the comparison as a return value. The return value has the signal state "1" if the string at parameter S1 is greater resp. smaller than the string at parameter S2.

The characters are compared by their ASCII code (for example, 'a' is greater than 'A'), starting from the left. The first character to be different decides the result of the comparison. If the first characters are the same, the longer string is greater resp. the shorter string is smaller.

The function does not report any errors.

5.3 Conversion of datatypes

5.3.1 I_STRNG (FC16) and STRNG_I (FC38)

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

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

The function I_STRNG (FC16) converts a variable in INT data type format to a string. The string is shown preceded by a sign. If the variable given at the return parameter is too short, no conversion takes place and OK-Flag is set to "0".

The function STRNG_I (FC38) converts a string to a variable in INT data type format. The first character in the string may be a sign or a number, the characters which then follow must be numbers. If the length of the string is equal to zero or greater than 6, or if invalid characters are found in the string, no conversion takes place and OK-Flag is set to "0".

5.3.2 DI_STRNG (FC5) and und STRNG_DI (FC37)

```
DI_STRNG (I := // IN: DINT
         ); // STRING
```

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

The function DI_STRNG (FC5) converts a variable in DINT data type format to a string. The string is shown preceded by a sign. If the variable given at the return parameter is too short, no conversion takes place and OK-Flag is set to "0".

The function STRNG_DI (FC37) converts a string to a variable in DINT data type format. The first character in the string may be a sign or a number, the characters which then follow must be numbers. If the length of the string is equal to zero or greater than 11, or if invalid characters are found in the string, no conversion takes place and the OK-Flag is set to "0". If the result of the conversion is outside the DINT range, the result is limited to the corresponding value and the OK-Flag is set to "0".

5.3.3 R_STRNG (FC30) and STRNG_R (FC39)

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

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

The function R_STRNG (FC30) converts a variable in REAL data type format to a string. The string is shown with 14 digits:

$\pm v.nnnnnnnE\pm xx \quad \pm \quad \text{Signv} \quad 1 \text{ digit before the decimal point,} \quad 7 \text{ digits after the decimal point}$
 $x \quad 2 \text{ exponential digits}$

If the variable given at the return parameter is too short or if no valid floating-point number is given at parameter IN, no conversion takes place and the OK-Flag is set to "0".

The function STRN_R (FC39) converts a string to a variable in REAL data type format. The string must have the following format:

$\pm v.nnnnnnnE\pm xx \quad \pm \quad \text{Signv} \quad 1 \text{ digit before the decimal point,} \quad 7 \text{ digits after the decimal point}$
 $x \quad 2 \text{ exponential digits}$

If the length of the string is smaller than 14, or if it is not structured as shown above, no conversion takes place and OK-Flag is set to "0". If the result of the conversion is outside the REAL range, the result is limited to the corresponding value and the OK-Flag is set to "0".

6 Example

6.1 Preparing Message

```
(***** Preparing and saving process dependant messages *****)
(*****
** This block contains the required texts and the last 20 messages which have
** been generated.
*****)

DATA_BLOCK MessageText
STRUCT
  Index : INT;
  textbuffer : ARRAY [0..19] OF
    STRING[30];
  HW : ARRAY [1..5] of STRING[15]; (* 5 different verschiedene tools *)
  stati : ARRAY [1..5] of STRING[11]; (* 5 different states *)
END_STRUCT
BEGIN
  Index:=0;
  HW[1] := 'Motor ';
  HW[2] := 'Valve ';
  HW[3] := 'Press ';
  HW[4] := 'Welding station ';
  HW[5] := 'Torch ';
  Stati[1] := ' faulty';
  Stati[2] := ' started';
  Stati[3] := ' Temperatur';
  Stati[4] := ' repaired';
  Stati[5] := ' waited';
END_DATA_BLOCK

(*****
This function concatenates the text of the message and write it in the Data Block MessageText.
The generated messages are stored in a ringbuffer. The next free index is also stored in the
data Block MessageText and is actualized by this function.
*****)
FUNCTION Textgenerator : BOOL;

VAR_INPUT
  unit : INT; // Index of the Tool-Text
  nr : INT; // IDNr. Of the Tool
  status : INT;
  wert : INT;
END_VAR
VAR
  text : string[31];
  i : int;
END_VAR
// initialisation of the temporary variables
text := '';
Textgenerator := TRUE;
CASE unit OF
  1..5 :
    CASE status OF
      1..5 : text := CONCAT(in1 := MessageText.HW[unit],
        in2 := RIGHT(l:=2,in:=I_STRNG(nr)));
      text := CONCAT(in1 := text,
        in2 := MessageText.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 := MessageText.index;
  MessageText.textbuffer[i] := text;
  MessageText.index := (i+1) mod 20;
END_FUNCTION

(*****
The function is called cyclically in the programm bei edge changing at %M10.0.
It puts a message in the buffer if a parameter has changed.
*****)
ORGANIZATION_BLOCK zyklus

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

The following call puts the message "Motor 12 started " in the
Textbuffer of the data block MessageText, where %MW0 adds 1, %EW2 adds 12 and %MW2 adds 2.
*****)

```
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
```