# Table of contents

1 Foreword ......................................................................................................................... 5
   1.1 Notes on the documentation ......................................................................................... 5
   1.2 Safety instructions ........................................................................................................ 6

2 Overview .......................................................................................................................... 7

3 Function blocks .................................................................................................................. 8
   3.1 Bistable ......................................................................................................................... 8
       3.1.1 RS .......................................................................................................................... 8
       3.1.2 SR .......................................................................................................................... 8
   3.2 Counter .......................................................................................................................... 9
       3.2.1 CTD ....................................................................................................................... 9
       3.2.2 CTU ....................................................................................................................... 9
       3.2.3 CTUD .................................................................................................................... 10
   3.3 Timer .............................................................................................................................. 11
       3.3.1 TOF ...................................................................................................................... 11
       3.3.2 TON ...................................................................................................................... 11
       3.3.3 TP ......................................................................................................................... 12
   3.4 Timer (LTIME). ............................................................................................................. 13
       3.4.1 LTOF .................................................................................................................... 13
       3.4.2 LTON ................................................................................................................... 14
       3.4.3 LTP ....................................................................................................................... 15
   3.5 Trigger .......................................................................................................................... 16
       3.5.1 F_TRIG .................................................................................................................. 16
       3.5.2 R_TRIG .................................................................................................................. 16

4 String functions .................................................................................................................. 17
   4.1 CONCAT ......................................................................................................................... 17
   4.2 DELETE ......................................................................................................................... 17
   4.3 FIND .............................................................................................................................. 18
   4.4 INSERT ........................................................................................................................... 18
   4.5 LEFT ............................................................................................................................ 19
   4.6 LEN ............................................................................................................................... 19
   4.7 MID ............................................................................................................................... 20
   4.8 REPLACE ...................................................................................................................... 20
   4.9 RIGHT .......................................................................................................................... 21

5 String functions (WSTRING) ............................................................................................ 22
   5.1 WCONCAT ................................................................................................................... 22
   5.2 WDELETE ..................................................................................................................... 22
   5.3 WFIND .......................................................................................................................... 23
   5.4 WINSERT ..................................................................................................................... 23
   5.5 WLEFT ......................................................................................................................... 24
   5.6 WLEN .......................................................................................................................... 24
   5.7 WMID ........................................................................................................................... 25
   5.8 WREPLACE ................................................................................................................. 25
   5.9 WRIGHT ...................................................................................................................... 26
6 Global constants ................................................................................................................................................................................................. 27
   6.1 Library version .................................................................................................................................................................................. 27
1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:


with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.
1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="danger.png" alt="DANGER" /></td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="warning.png" alt="WARNING" /></td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td><img src="caution.png" alt="CAUTION" /></td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td><img src="note.png" alt="NOTE" /></td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

Tip or pointer

This symbol indicates information that contributes to better understanding.
2 Overview

The Standard Library includes all IEC61131-3 POUs. The POUs can be classified in:

- Bistable Function Blocks
- Trigger Function Blocks
- Counter
- Timer
- Timer (LTIME)
- String Functions
- String Functions (WSTRING)
3 Function blocks

3.1 Bistable

3.1.1 RS

Resetting Bistable Function Blocks

Q1 = RS (SET, RESET1)

means: Q1 = NOT RESET1 AND (Q1 OR SET)

VAR_INPUT

VAR_INPUT

SET : BOOL;
RESET1 : BOOL;
END_VAR

VAR_OUTPUT

VAR_OUTPUT

Q1 : BOOL;
END_VAR

Internal implementation of fb:

Q1: = NOT RESET1 AND (Q1 OR SET);

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

3.1.2 SR

Making Bistable Function Blocks Dominant

Q1 = SR (SET1, RESET)

means: Q1 = (NOT RESET AND Q1) OR SET1Q1, SET1 and RESET are BOOL variables.

VAR_INPUT

VAR_INPUT

SET1 : BOOL;
RESET : BOOL;
END_VAR

VAR_OUTPUT

VAR_OUTPUT

Q1 : BOOL;
END_VAR
Internal implementation of fb:

\[ Q1 := (\text{NOT} \ \text{RESET} \ \text{AND} \ Q1) \ \text{OR} \ \text{SET1}; \]

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

### 3.2 Counter

#### 3.2.1 CTD

Decrementer

**VAR_INPUT**

```
VAR_INPUT
    CD : BOOL; (* Count Down on rising edge *)
    LOAD : BOOL; (* Load Start Value *)
    PV : WORD; (* Start Value *)
END_VAR
```

**VAR_OUTPUT**

```
VAR_OUTPUT
    Q : BOOL; (* Counter reached 0 *)
    CV : WORD; (* Current Counter Value *)
END_VAR
```

When LOAD is TRUE, the counter variable CV will be initialized with the upper limit PV. If CD has a rising edge from FALSE to TRUE, CV will be lowered by 1 provided CV is greater than 0 (i.e., it doesn't cause the value to fall below 0). Q returns TRUE when CV is less than or equal to 0.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

#### 3.2.2 CTU

Incrementer

**VAR_INPUT**

```
VAR_INPUT
    CU : BOOL; (* Count Up *)
    RESET : BOOL; (* Reset Counter to 0 *)
    PV : WORD; (* Counter Limit *)
END_VAR
```
VAR_OUTPUT

VAR_OUTPUT
Q : BOOL; (* Counter reached the Limit *)
CV : WORD; (* Current Counter Value *)
END_VAR

The counter variable CV will be initialized with 0 if RESET is TRUE. If CU has a rising edge from FALSE to TRUE, the function block CV will be raised by 1 provided CV is smaller than PV (i.e., it doesn’t cause an overflow). Q will return TRUE when CV is greater than or equal to the upper limit PV.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

### 3.2.3 CTUD

![Incrementer and Decrementer](image)

If RESET is valid, the counter variable CV will be initialized with 0. If LOAD is valid, CV will be initialized with PV. If CU has a rising edge from FALSE to TRUE, CV will be raised by 1 provided CV does not cause an overflow. If CD has a rising edge from FALSE to TRUE, CV will be lowered by 1 provided this does not cause the value to fall below 0. QU returns TRUE when CV has become greater than or equal to PV. QD returns TRUE when CV has become less than or equal to 0.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>
3.3 Timer

3.3.1 TOF

Timer off-delay

**VAR_INPUT**

```plaintext
VAR_INPUT
  IN : BOOL; (* starts timer with falling edge, resets timer with rising edge *)
  PT : TIME; (* time to pass, before Q is set *)
END_VAR
```

**VAR_OUTPUT**

```plaintext
VAR_OUTPUT
  Q  : BOOL; (* is FALSE, PT seconds after IN had a falling edge *)
  ET : TIME; (* elapsed time *)
END_VAR
```

When IN is TRUE, Q is TRUE and ET is 0. As soon as IN becomes FALSE, the time will begin to be counted in milliseconds in ET until its value is equal to that of PT. It will then remain constant. Q is FALSE when IN is FALSE and ET is equal to PT. Otherwise it is TRUE. Thus, Q has a falling edge when the time indicated in PT in milliseconds has run out.

Graphic display of TOF behavior over time:

![Graphical display of TOF behavior over time](image)

The function TOF requires 15 byte data.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

3.3.2 TON

Timer on-delay
VAR_INPUT

IN : BOOL; (* starts timer with rising edge, resets timer with falling edge *)
PT : TIME; (* time to pass, before Q is set *)

END_VAR

VAR_OUTPUT

Q : BOOL; (* is TRUE, PT seconds after IN had a rising edge *)
ET : TIME; (* elapsed time *)

END_VAR

If IN is FALSE, Q is FALSE and ET is 0. As soon as IN becomes TRUE, the time will begin to be counted in milliseconds in ET until its value is equal to PT. It will then remain constant. Q is TRUE when IN is TRUE and ET is equal to PT. Otherwise it is FALSE. Thus, Q has a rising edge when the time indicated in PT in milliseconds has run out.

Graphic display of TON behavior over time:

IN

Q

ET

The function TON requires 15 byte data.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

3.3.3 TP

Pulse generator. This function block can be used to generate pulses with a defined pulse duration.

VAR_INPUT

IN : BOOL; (* Trigger for Start of the Signal *)
PT : TIME; (* The length of the High-Signal in ms *)

END_VAR

VAR_OUTPUT

Q : BOOL; (* The pulse *)
ET : TIME; (* The current phase of the High-Signal *)

END_VAR
If IN is FALSE, the outputs are FALSE or 0. As soon as IN becomes TRUE, Q also becomes TRUE and remains TRUE for the pulse duration PT. As long as Q is TRUE, the time is incremented in milliseconds in ET, until the value reaches PT. The value then remains constant. The output Q remains TRUE until the pulse duration has elapsed, irrespective of the state of the input IN. Q therefore supplies a signal over the interval specified in PT.

Graphic display of TP behavior over time:

The TP function requires 14 bytes of data.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

3.4 Timer (LTIME)

3.4.1 LTOF

Timer-off delay with 64-bit time data type (LTIME).

VAR_INPUT

```plaintext
VAR_INPUT
IN  :  BOOL; (*starts timer with falling edge, resets timer with rising edge*)
PT  :  LTIME; (*time to pass before Q is reset*)
END_VAR
```

VAR_OUTPUT

```plaintext
VAR_OUTPUT
Q   :  BOOL; (*is FALSE, PT seconds after IN had a falling edge*)
ET  :  LTIME; (*elapsed time since falling edge at IN*)
END_VAR
```

If IN is TRUE, the outputs are TRUE or 0. As soon as IN becomes FALSE, the time is incremented in nanoseconds in ET, until the value equals the value of PT. The value then remains constant. Q is FALSE, if IN = FALSE and ET = PT. Otherwise Q = TRUE. Q thus has a falling edge, once the time specified in nanoseconds in PT has elapsed.

Graph showing LTOF over time:
Timer-on delay with 64-bit time data type (LTIME).

**VAR_INPUT**

<table>
<thead>
<tr>
<th>VAR_INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN : BOOL; (<em>starts timer with rising edge, resets timer with falling edge</em>)</td>
</tr>
<tr>
<td>PT : LTIME; (<em>time to pass before Q is set.</em>)</td>
</tr>
<tr>
<td>END_VAR</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>VAR_OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q : BOOL; (<em>is TRUE, PT seconds after IN had a rising edge</em>)</td>
</tr>
<tr>
<td>ET : LTIME (<em>elapsed time since rising edge at IN</em>)</td>
</tr>
<tr>
<td>END_VAR</td>
</tr>
</tbody>
</table>

If IN is FALSE, the outputs are FALSE or 0. As soon as IN becomes TRUE, the time is incremented in nanoseconds in ET, until the value reaches PT. The value then remains constant. Q is TRUE, if IN = TRUE and ET = PT. Otherwise Q = FALSE. Q thus has a rising edge, once the time specified in nanoseconds in PT has elapsed.

Graph showing LTON over time:
### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

### 3.4.3 LTP

Pulse generator with 64-bit time data type (LTIME). This function block can be used to generate pulses with a defined pulse duration.

#### VAR_INPUT

```
VAR_INPUT
  IN : BOOL; (*Trigger for Start of the Signal*)
  PT : LTIME; (*The length of the High- Signal*)
END_VAR
```

#### VAR_OUTPUT

```
VAR_OUTPUT
  Q : BOOL; (*The pulse*)
  ET : LTIME (*elapsed time since pulse start*)
END_VAR
```

If IN is FALSE, the outputs are FALSE or 0. As soon as IN becomes TRUE, Q also becomes TRUE and remains TRUE for the pulse duration PT. As long as Q is TRUE, the time is incremented in nanoseconds in ET, until the value reaches PT. The value then remains constant. The output Q remains TRUE until the pulse duration has elapsed, irrespective of the state of the input IN. Q therefore supplies a signal over the interval specified in PT.

Graph showing LTP over time:

![Graph showing LTP over time]

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>
3.5 Trigger

3.5.1 F_TRIG

Detector for a Falling Edge

VAR_INPUT

VAR_INPUT
CLK : BOOL; (* Signal to detect *)
END_VAR

VAR_OUTPUT

VAR_OUTPUT
Q : BOOL; (* Edge detected *)
END_VAR

The output Q will remain FALSE as long as the input variable CLK returns TRUE. As soon as CLK returns FALSE, Q will return TRUE. This means each time the function is called up, Q will return FALSE until CLK has a rising followed by a falling edge.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

3.5.2 R_TRIG

Detector for a Rising Edge

VAR_INPUT

VAR_INPUT
CLK : BOOL; (* Signal to detect *)
END_VAR

VAR_OUTPUT

VAR_OUTPUT
Q : BOOL; (* Edge detected *)
END_VAR

The output Q will remain FALSE as long as the input variable CLK is FALSE. As soon as CLK returns TRUE, Q will return TRUE. This means each time the function is called up, Q will return FALSE until CLK has falling edge followed by a rising edge.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>
4 String functions

On the PLC samples page you can download a sample project about string functions.

4.1 CONCAT

Concatenation (combination) of two strings.

**FUNCTION CONCAT: STRING (255)**

```
VAR_INPUT
  STR1 : STRING(255);
  STR2 : STRING(255);
END_VAR

Example in IL:
LD 'SUSI'
CONCAT 'WILLI'
ST Var1 (* Result is 'SUSIWILLI' *)
```

Example in ST:
```
Var1 := CONCAT ('SUSI','WILLI');
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

4.2 DELETE

The function DELETE removes a partial string from a larger string at a defined position. The input variable STR is type STRING, LEN and POS are type INT, the return value of the function is type STRING.

DELETE (STR, LEN, POS) means: Delete LEN characters from STR beginning with the character in the POS.

**FUNCTION DELETE: STRING (255)**

```
VAR_INPUT
  STR  : STRING(255);
  LEN  : INT;
  POS  : INT;
END_VAR

Example in IL:
LD 'SUXYSI'
DELETE 2,3
ST Var1 (* Result is 'SUSI' *)
```

Example in ST:
```
Var1 := DELETE ('SUXYSI',2,3);
```
String functions

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

4.3 FIND

The function FIND searches for a partial string within a string. FIND (STR1, STR2) means: Find the position of the first character where STR2 appears in STR1 for the first time. If STR2 is not found in STR1, then OUT:=0.

FUNCTION FIND: INT

```
VAR_INPUT
  STR1 : STRING(255);
  STR2 : STRING(255);
END_VAR

Example in IL:
LD 'SUXYSI'
FIND 'XY'
ST Var1 (* Result is 3 *)

Example in ST:
Var1 := FIND('SUXYSI','XY');
```

4.4 INSERT

The function INSERT inserts a string into another string at a defined point. INSERT (STR1, STR2, POS) means: Insert STR2 into STR1 after position POS.

FUNCTION INSERT: STRING (255)

```
VAR_INPUT
  STR1 : STRING(255);
  STR2 : STRING(255);
  POS  : INT;
END_VAR

Example in IL:
LD 'SUSI'
INSERT 'XY',2
ST Var1 (* Result is 'SUXYSI' *)

Example in ST:
Var1 := INSERT('SUSI','XY',2);
```
4.5 LEFT

The function LEFT returns the left, initial string for a given string. LEFT (STR, SIZE) means: Take the first SIZE character from the left in the string STR.

**FUNCTION LEFT: STRING (255)**

```
VAR_INPUT
  STR : STRING(255);
  SIZE : INT;
END_VAR

Example in IL:
LD 'SUSI'
LEFT 3
ST Var1 (* Result is 'SUS' *)
```

Example in ST:

```
Var1 := LEFT ('SUSI',3);
```

4.6 LEN

The function LEN returns the length of a string.

**FUNCTION LEN: INT**

```
VAR_INPUT
  STR : STRING(255);
END_VAR

Example in IL:
LD 'SUSI'
LEN
ST Var1 (* Result is 4 *)
```

Example in ST:

```
Var1 := LEN ('SUSI');
```
4.7 MID

The function MID returns a partial string from within a string.

*MID* (STR, LEN, POS) means: Retrieve LEN characters from the STR string beginning with the character at position POS.

**FUNCTION MID: STRING (255)**

```plaintext
VAR_INPUT
    STR : STRING(255);
    LEN : INT;
    POS : INT;
END_VAR

Example in IL:

LD 'SUSI'
MID 2,2
ST Var1 (* Result is 'US' *)

Example in ST:

Var1 := MID ('SUSI',2,2);
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

4.8 REPLACE

The function REPLACE replaces a partial string from a larger string with a third string.

*REPLACE* (STR1, STR2, L, P) means: Replace L characters from STR1 with STR2 beginning with the character in the P position.

**FUNCTION REPLACE: STRING (255)**

```plaintext
VAR_INPUT
    STR1 : STRING(255);
    STR2 : STRING(255);
    L : INT;
    P : INT;
END_VAR

Example in IL:

LD 'SUXXSYI'
REPLACE 'K',2,2
ST Var1 (* Result is 'SKYSI' *)

Example in ST:

Var1 := REPLACE('SUXXSYI','K',2,2);
```
4.9 RIGHT

The function RIGHT returns the right, initial string for a given string.
RIGHT (STR, SIZE) means: Take the first SIZE character from the right in the string STR.

FUNCTION RIGHT: STRING (255)

```
VAR_INPUT
  STR : STRING(255);
  SIZE : INT;
END_VAR

Example in IL:
LD 'SUSI'
RIGHT 3
ST Var1 (* Result is 'USI' *)
```

Example in ST:
```
Var1 := RIGHT ('SUSI', 3);
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>
5 String functions (WSTRING)

5.1 WCONCAT

Concatenation (combination) of two WSTRINGs.

FUNCTION WCONCAT: WSTRING (255)

VAR_INPUT
  STR1 : WSTRING(255) (*Head part of the concatenated result*)
  STR2 : WSTRING(255) (*Tail part of the concatenated result*)
END_VAR

Example in IL:
LD "SUSI"
WCONCAT "WILLI"
ST Var1 (*Result is "SUSIWILLI"*)

Example in ST:
Var1 := WCONCAT ("SUSI","WILLI");

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

5.2 WDELETE

The function WDELETE deletes part of a WSTRING from a certain point. The input STR is of type WSTRING. LEN and POS are of type INT. The return value of the function is of type WSTRING.

FUNCTION WDELETE: WSTRING (255)

VAR_INPUT
  STR1 : WSTRING(255);
  LEN : INT;
  POS : INT;
END_VAR

Example in IL:
LD "SUXYSI"
WDELETE 2,3
ST Var1 (*Result is "SUSI"*)

Example in ST:
Var1 := WDELETE ("SUXYSI",2,3);

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>
5.3 W_FIND

The function W_FIND searches for a part in W_STRING.
W_FIND (STR1, STR2) means: Find the position of the first character of first occurrence of STR2 in STR1. If STR2 does not occur in STR1, then OUT := 0 applies.

FUNCTION W_FIND: INT
VAR_INPUT
  STR1 : W_STRING(255);
  STR2 : W_STRING(255);
END_VAR

Example in IL:
LD "SUXYSI"
W_FIND "XY"
ST Var1 (*Result is 3*)

Example in ST:
Var1 := W_FIND ("SUXYSI","XY");

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

5.4 W_INSERT

The function W_INSERT adds a WString in another WString from a particular location.
W_INSERT (STR1, STR2, POS) means: Add STR2 in STR1 after the POSth position.

FUNCTION W_INSERT: W_STRING (255)
VAR_INPUT
  STR1 : W_STRING(255);
  STR2 : W_STRING(255);
  POS  : INT;
END_VAR

Example in IL:
LD "SUSI"
W_INSERT "XY",2
ST Var1 (*Result is "SUXYSI"*)

Example in ST:
Var1 := W_INSERT ("SUSI","XY",2);

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>
5.5  WLEFT

The function WLEFT supplies a left-hand start string for a WSTRING.
WLEFT (STR, SIZE) means: Take the first SIZE characters from the left in WString STR.

FUNCTION WLEFT: WSTRING (255)

```
VAR_INPUT
   STR : WSTRING(255);
   SIZE : INT;
END_VAR

Example in IL:
LD "SUSI"
WLEFT 3
ST Var1 (*Result is "SU")

Example in ST:
Var1 := WLEFT ("SUSI",3);
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

5.6  WLEN

The function WLEN outputs the length of a WSTRING.

FUNCTION WLEN: INT

```
VAR_INPUT
   STR : WSTRING(255);
END_VAR

Example in IL:
LD "SUSI"
WLEN
ST Var1 (*Result is 4*)

Example in ST:
Var1 := WLEN ("SUSI");
```

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>
5.7 WMID

The function WMID supplies a substring of a WSTRING. WMID(STR, LEN, POS) means: Fetch LEN characters from WSTRING STR beginning with the character at position POS.

FUNCTION WMID: WSTRING (255)

VAR_INPUT
  STR : WSTRING(255);
  LEN : INT;
  POS : INT;
END_VAR

Example in IL:
LD "SUSI"
WMID 2,2
ST Var1 (*Result is "US"*)

Example in ST:
Var1 := WMID ("SUSI",2,2);

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>

5.8 WREPLACE

The function WREPLACE replaces a substring of a WSTRING with another WSTRING. WREPLACE (STR1, STR2, L, P) means: Replace L characters from STR1 with STR2 beginning with the Pth character.

FUNCTION WREPLACE: WSTRING (255)

VAR_INPUT
  STR1 : WSTRING(255);
  STR2 : WSTRING(255);
  L : INT;
  P : INT;
END_VAR

Example in IL:
LD "SUYSI"
WREPLACE "XY",2
ST Var1 (*Result is "SKYSI"*)

Example in ST:
Var1 := WREPLACE ("SUYSI","K",2,2);
5.9 **WRIGHT**

WRIGHT supplies a right-hand start string for a WSTRING. WRIGHT (STR, SIZE) means: Take the first SIZE characters from the right in WString STR.

**FUNCTION WRIGHT: WSTRING (255)**

```plaintext
VAR_INPUT
    STR : WSTRING(255);
    SIZE : INT;
END_VAR

Example in IL:
LD "SUSI"
WRIGHT 3
ST Var1 (*Result is "USI"*)

Example in ST:
Var1 := WRIGHT ("SUSI",3);
```

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target system type</th>
<th>PLC libraries to be linked</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.0.0</td>
<td>PC or CX (x86)</td>
<td>Tc2_Standard</td>
</tr>
</tbody>
</table>
6 Global constants

6.1 Library version

All libraries have a specific version. This version is shown in the PLC library repository too. A global constant contains the library version information:

Global_Version
VAR_GLOBAL CONSTANT
  stLibVersion_Tc2_Standard : ST_LibVersion;
END_VAR


To compare the existing version to a required version use the function F_CmpLibVersion.

All other possibilities known from TwinCAT2 libraries to query a library version are obsolete!
More Information:
www.beckhoff.com/te1000