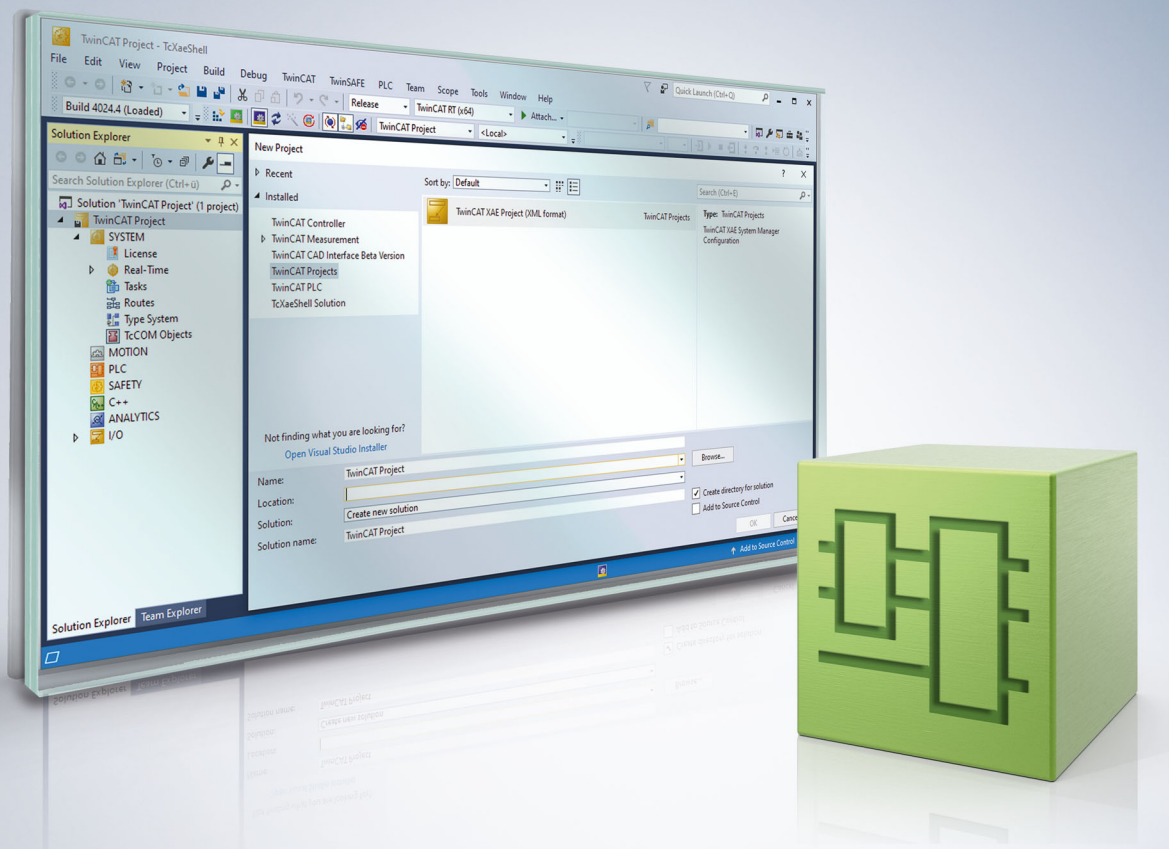
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | PLC-Bibliothek: Tc2_Standard



02.06.2025 | Version: 1.3.3

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® und XTS® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Kennzeichnungen führen.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie, lizenziert durch die Beckhoff Automation GmbH, Deutschland.

Copyright

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: https://www.beckhoff.com/trademarks.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Übersicht

Die Standard-Bibliothek beinhaltet alle IEC61131-3 POUs. Die POUs können eingeteilt werden in:

- Bistabile Funktionsblöcke
- Zähler
- Timer
- Timer (LTIME)
- Trigger
- Stringfunktionen
- Stringfunktionen (WSTRING)

3 Funktionsbausteine

3.1 Bistable

3.1.1 RS

Der Funktionsbaustein RS ist bistabil und RESET ist dominant.

Q1 = RS (SET, RESET1)

D. h.: Q1 = NOT RESET1 AND (Q1 OR SET)

Eingänge

```
VAR_INPUT
    SET    : BOOL;
    RESET1 : BOOL;
END_VAR
```

Name	Typ	Beschreibung
SET	BOOL	Setzeingang
RESET1	BOOL	Rücksetzeingang

Ausgänge

```
VAR_OUTPUT
    Q1 : BOOL;
END_VAR
```

Interne Implementierung des Funktionsbausteins:

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

Name	Typ	Beschreibung
Q1	BOOL	Der Ausgang wird zurückgesetzt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.1.2 SR

Der Funktionsbaustein SR ist bistabil und SET ist dominant.

Q1 = SR (SET1, RESET)

D. h.: Q1 = (NOT RESET AND Q1) OR SET1Q1, SET1 und RESET sind BOOL- Variablen.

Eingänge

```
VAR_INPUT
    SET1  : BOOL;
    RESET : BOOL;
END_VAR
```

Name	Typ	Beschreibung
SET1	BOOL	Setzeingang
RESET	BOOL	Rücksetzeingang

Ausgänge

```
VAR_OUTPUT
    Q1 : BOOL;
END_VAR
```

Interne Implementierung des Funktionsbausteine:

```
Q1 := (NOT RESET AND Q1) OR SET1;
```

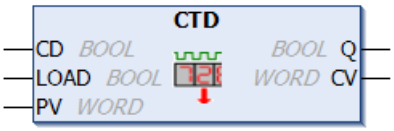
Name	Typ	Beschreibung
Q1	BOOL	Der Ausgang wird gesetzt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.2 Counter

3.2.1 CTD



Der Funktionsbaustein CTD ist ein Abwärtszähler.

Eingänge

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

Name	Typ	Beschreibung
CD	BOOL	Abwärtszählen bei steigender Flanke
LOAD	BOOL	Lade Startwert.
PV	Word	Startwert

Ausgänge

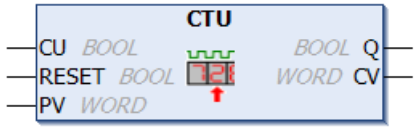
```
VAR_OUTPUT
    Q  : BOOL; (* Counter reached 0 *)
    CV : WORD; (* Current Counter Value *)
END_VAR
```

Name	Typ	Beschreibung
Q	BOOL	Wenn LOAD = TRUE ist, wird die Zählvariable CV mit der Obergrenze PV initialisiert. Wenn CD eine steigende Flanke von FALSE auf TRUE hat, wird CV um 1 erniedrigt, solange CV größer als 0 ist. (Wenn also kein Unterlauf verursacht wird.)
CV	WORD	Q liefert TRUE, wenn CV kleiner oder gleich 0 ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.2.2 CTU



Der Funktionsbaustein CTU ist ein Aufwärtszähler.

Eingänge

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

Name	Typ	Beschreibung
CU	BOOL	Aufwärtszählen
RESET	BOOL	Setze Zähler auf den Wert 0 zurück.
PV	WORD	Zählergrenze

Ausgänge

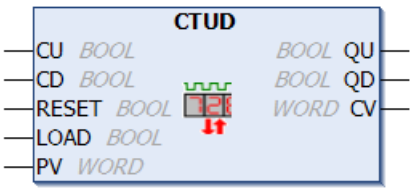
```
VAR_OUTPUT
    Q  : BOOL; (* Counter reached the Limit *)
    CV : WORD; (* Current Counter Value *)
END_VAR
```

Name	Typ	Beschreibung
Q	BOOL	Q liefert TRUE, wenn CV größer oder gleich der Obergrenze PV ist.
CV	WORD	Wenn RESET = TRUE ist, wird die Zählvariable CV mit 0 initialisiert. Wenn CU eine steigende Flanke von FALSE auf TRUE hat, dann wird der Funktionsblocks CV um 1 erhöht, solange CV kleiner als PV ist. (Wenn also kein Überlauf verursacht wird.)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.2.3 CTUD



Der Funktionsbaustein CTUD ist ein Auf- und Abwärtszähler.

Eingänge

```
VAR_INPUT
    CU    : BOOL; (* Count Up *)
    CD    : BOOL; (* Count Down *)
    RESET : BOOL; (* Reset Counter to Null *)
    LOAD  : BOOL; (* Load Start Value *)
    PV    : WORD; (* Start Value / Counter Limit *)
END_VAR
```

Name	Typ	Beschreibung
CU	BOOL	Aufwärtszählen
CD	BOOL	Abwärtszählen
RESET	BOOL	Setze Zähler auf den Wert 0 zurück.
LOAD	BOOL	Lade Startwert.
PV	WORD	Startwert/Zählerlimit

Ausgänge

```
VAR_OUTPUT
    QU : BOOL; (* Counter reached Limit *)
    QD : BOOL; (* Counter reached Null *)
    CV : WORD; (* Current Counter Value *)
END_VAR
```

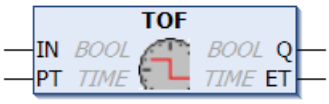
Name	Typ	Beschreibung
QU	BOOL	QU liefert TRUE, wenn CV größer oder gleich PV geworden ist.
QD	BOOL	QD liefert TRUE, wenn CV kleiner oder gleich 0 geworden ist.
CV	WORD	Wenn RESET gilt, dann wird die Zählvariable CV mit 0 initialisiert. Wenn LOAD gilt, dann wird CV mit PV initialisiert. Wenn CU eine steigende Flanke von FALSE auf TRUE hat, dann wird CV um 1 erhöht, solange CV keinen Überlauf verursacht. Wenn CD eine steigende Flanke von FALSE auf TRUE hat, dann wird CV jeweils um 1 erniedrigt, solange CV keinen Unterlauf verursacht.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.3 Timer

3.3.1 TOF



Der Funktionsbaustein TOF ist ein Timer off-delay.

Eingänge

```
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
```

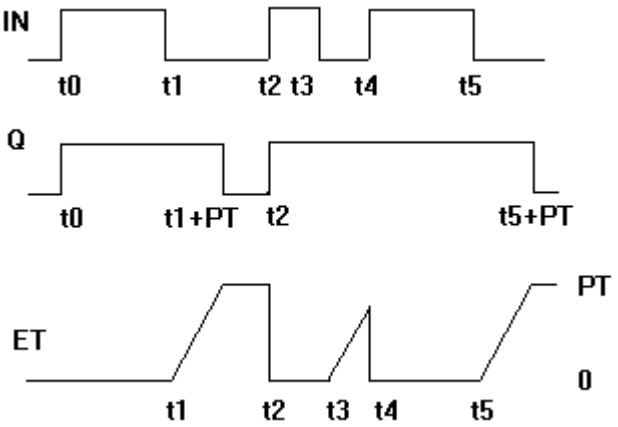
Name	Typ	Beschreibung
IN	BOOL	Startet den Timer mit fallender Flanke und setzt den Timer mit steigender Flanke zurück.
PT	TIME	Zeit, die vergeht, bevor Q gesetzt wird.

Ausgänge

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

Name	Typ	Beschreibung
Q	BOOL	Q hat eine fallende Flanke, wenn die in PT in Millisekunden angegebene Zeit abgelaufen ist.
ET	BOOL	Wenn IN = TRUE ist, sind die Ausgaben TRUE bzw. 0. Sobald IN = FALSE ist, wird in ET die Zeit in Millisekunden hochgezählt bis der Wert gleich dem in PT ist, dann bleibt er gleich. Q ist FALSE wenn IN = FALSE und ET = PT ist. Sonst ist Q = TRUE.

Graphische Darstellung des zeitlichen Verhaltens von TOF:

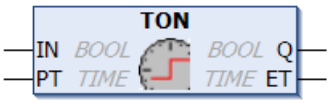


Die Funktion TOF benötigt 15 Byte Daten

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.3.2 TON



Der Funktionsbaustein TON ist ein Timer on-delay.

Eingänge

```
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
```

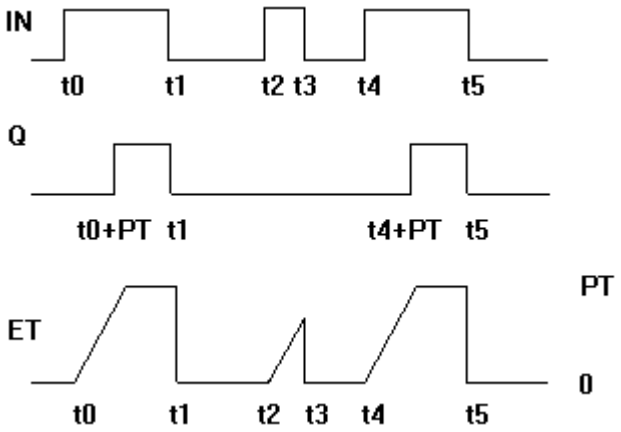
Name	Typ	Beschreibung
IN	BOOL	Startet den Timer mit steigender Flanke und setzt den Timer mit fallender Flanke zurück.
PT	TIME	Zeit, die vergeht, bevor Q gesetzt wird.

Ausgänge

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

Name	Typ	Beschreibung
Q	BOOL	Q hat eine steigende Flanke, wenn die in PT in Millisekunden angegebene Zeit abgelaufen ist.
ET	TIME	Wenn IN = FALSE ist, sind die Ausgaben FALSE bzw. 0. Sobald IN = TRUE ist, wird in ET die Zeit in Millisekunden hochgezählt, bis der Wert gleich dem in PT ist, dann bleibt er gleich. Q ist TRUE, wenn IN = TRUE und ET = PT ist. Sonst ist Q = FALSE.

Graphische Darstellung des zeitlichen Verhaltens von TON:

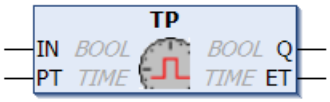


Die Funktion TON benötigt 15 Byte Daten

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.3.3 TP



Mit dem Funktionsbaustein TP, einem Pulsgeber, können Impulse mit einer definierten Impulsdauer generiert werden.

Eingänge

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

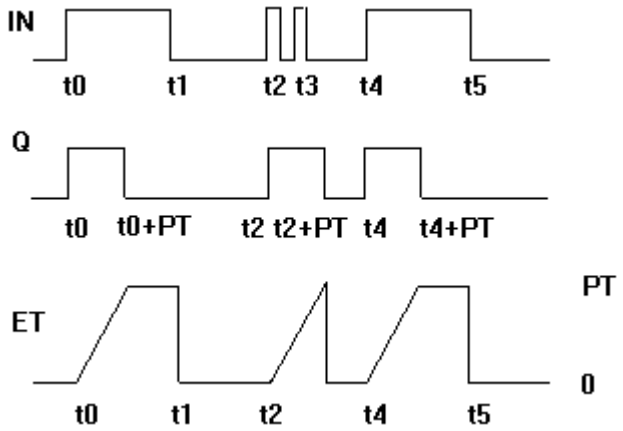
Name	Typ	Beschreibung
IN	BOOL	Auslöser für den Beginn des Signals
PT	TIME	Die Länge des High-Signals in ms

Ausgänge

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

Name	Typ	Beschreibung
Q	BOOL	Q liefert für den in PT angegebenen Zeitraum ein Signal.
ET	TIME	Wenn IN = FALSE ist, sind die Ausgaben FALSE bzw. 0. Sobald IN = TRUE ist, wird auch Q = TRUE und bleibt TRUE für die Impulsdauer PT. Solange Q = TRUE ist, wird in ET die Zeit in Millisekunden hochgezählt, bis der Wert gleich dem in PT ist, dann bleibt er gleich. Der Ausgang Q bleibt TRUE bis die Impulszeit verstrichen ist unabhängig von dem Zustand des Eingangs IN.

Grafische Darstellung des zeitlichen Ablaufs von TP:



Die Funktion TP benötigt 14 Byte Daten.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.4 Timer (LTIME)

3.4.1 LTOF

Der Funktionsbaustein LTOF ist ein Timer off delay mit 64- Bit- Zeitdatentyp (LTIME).

Eingänge

```
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
```

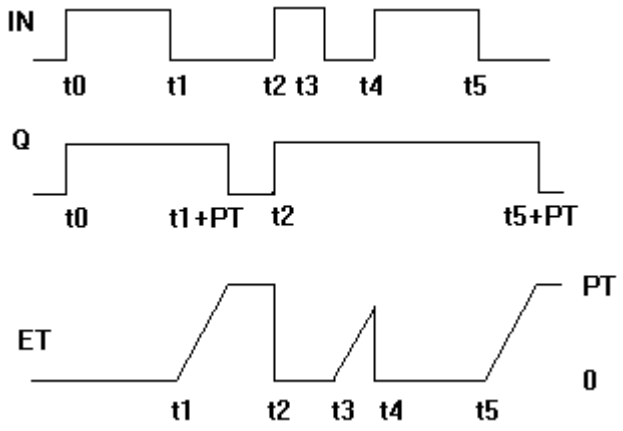
Name	Typ	Beschreibung
IN	BOOL	Startet den Timer mit fallender Flanke und setzt den Timer mit steigender Flanke zurück.
PT	TIME	Zeit, die vergeht, bevor Q gesetzt wird.

Ausgänge

```
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
```

Name	Typ	Beschreibung
Q	BOOL	Q hat eine fallende Flanke, wenn die in PT in Nanosekunden angegebene Zeit abgelaufen ist.
ET	LTIME	Wenn IN = TRUE ist, sind die Ausgaben TRUE bzw. 0. Sobald IN = FALSE ist, wird in ET die Zeit in Nanosekunden hochgezählt, bis der Wert gleich dem in PT ist. Dann bleibt er gleich. Q ist FALSE, wenn IN = FALSE und ET = PT ist.

Grafische Darstellung des zeitlichen Verhaltens von LTOF:



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.4.2 LTON

Der Funktionsbaustein LTON ist ein Timer on delay mit 64- Bit- Zeitdatentyp (LTIME).

Eingänge

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

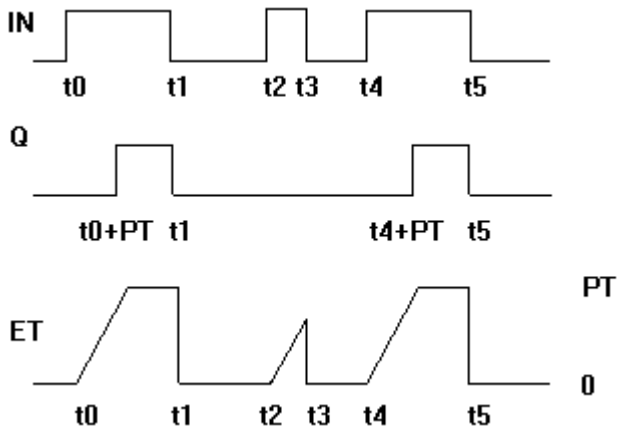
Name	Typ	Beschreibung
IN	BOOL	Startet den Timer mit steigender Flanke und setzt den Timer mit fallender Flanke zurück.
PT	TIME	Zeit, die vergeht, bevor Q gesetzt wird.

Ausgänge

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

Name	Typ	Beschreibung
Q	BOOL	Q hat eine steigende Flanke, wenn die in PT in Nanosekunden angegebene Zeit abgelaufen ist.
ET	LTIME	Wenn IN = FALSE ist, sind die Ausgaben FALSE bzw. 0. Sobald IN = TRUE ist, wird in ET die Zeit in Nanosekunden hochgezählt bis der Wert gleich dem in PT ist, dann bleibt er gleich. Q ist TRUE, wenn IN = TRUE und ET = PT ist. Sonst ist Q = FALSE.

Grafische Darstellung des zeitlichen Verhaltens von LTON:



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.4.3 LTP

Der Funktionsbaustein LTP ist ein Pulsgeber mit 64- Bit- Zeitdatentyp (LTIME). Mit diesem Funktionsbaustein können Impulse mit einer definierten Impulsdauer generiert werden.

Eingänge

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

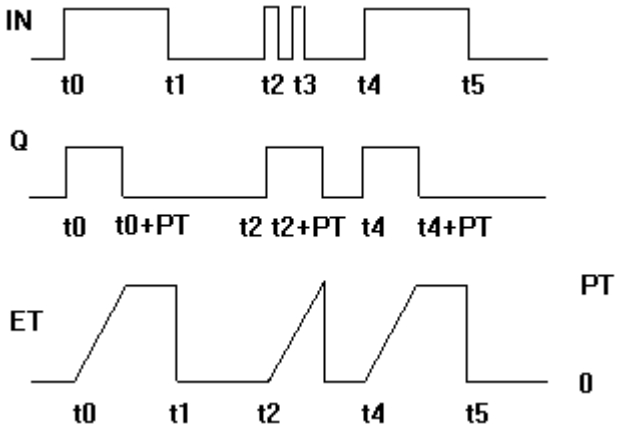
Name	Typ	Beschreibung
IN	BOOL	Auslöser für den Beginn des Signals
PT	LTIME	Die Länge des High-Signals in ms

Ausgänge

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

Name	Typ	Beschreibung
Q	BOOL	Q liefert für den in PT angegebenen Zeitraum ein Signal.
ET	LTIME	Wenn IN = FALSE ist, sind die Ausgaben FALSE bzw. 0. Sobald IN = TRUE ist, wird auch Q = TRUE und bleibt TRUE für die Impulsdauer PT. Solange Q = TRUE ist, wird in ET die Zeit in Nanosekunden hochgezählt bis der Wert gleich dem in PT ist, dann bleibt er gleich. Der Ausgang Q bleibt TRUE bis die Impulszeit verstrichen ist unabhängig vom dem Zustand des Eingangs IN.

Grafische Darstellung des zeitlichen Ablaufs von LTP:



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.5 Trigger

3.5.1 F_TRIG

Der Funktionsbaustein F_TRIG ist ein Detektor für eine fallende Flanke.

Eingänge

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

Name	Typ	Beschreibung
CLK	BOOL	Zu erkennendes Signal

Ausgänge

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

Name	Typ	Beschreibung
Q	BOOL	Solange die Eingabevariable CLK = TRUE liefert, wird die Ausgabe Q = FALSE sein. Sobald CLK = FALSE liefert, wird Q = TRUE liefern. D.h.: Bei jedem weiteren Aufruf der Funktion wird Q wieder FALSE liefern, bis CLK eine steigende und wieder eine fallende Flanke hat.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

3.5.2 R_TRIG

Der Funktionsbaustein R_TRIG ist ein Detektor für eine ansteigende Flanke.

Eingänge

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

Name	Typ	Beschreibung
CLK	BOOL	Zu erkennendes Signal

Ausgänge

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

Name	Typ	Beschreibung
Q	BOOL	Solange die Eingabevariable CLK = FALSE liefert, wird der Ausgang Q = FALSE sein. Sobald CLK = TRUE liefert, wird Q auch TRUE liefern. D.h.: Bei jedem weiteren Aufruf der Funktion wird Q wieder FALSE liefern, bis CLK eine fallende und wieder eine steigende Flanke hat.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4 STRING-Funktionen

Auf der SPS-Samples-Seite können Sie ein Beispielprojekt zum Thema Stringfunktionen herunterladen.

4.1 CONCAT

Konkatenation (Aneinanderhängen) von zwei Strings.

FUNCTION CONCAT: STRING (255)

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

Beispiel in AWL:

```
LD 'SUSI'
CONCAT 'WILLI'
ST Var1 (* Ergebnis ist 'SUSIWILLI' *)
```

Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4.2 DELETE

Die Funktion DELETE löscht ab einer bestimmten Stelle einen Teilstring aus einem String. Der Eingang STR ist vom Typ STRING, LEN und POS vom Typ INT, der Rückgabewert der Funktion vom Typ STRING. DELETE(STR, LEN, POS) bedeutet: Lösche LEN Zeichen aus STR und beginne mit dem POS-ten.

FUNCTION DELETE: STRING (255)

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

Beispiel in AWL:

```
LD 'SUXYSI'
DELETE 2,3
ST Var1 (* Ergebnis ist 'SUSI' *)
```

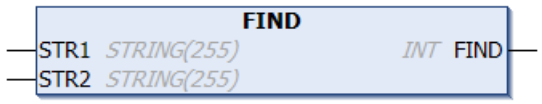
Beispiel in ST:

```
Var1 := DELETE ('SUXYSI',2,3);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4.3 FIND



Die Funktion FIND sucht einen Teilstring in einem String.
FIND(STR1, STR2) bedeutet: Finde die Position des ersten Zeichens des ersten Vorkommens von STR2 in STR1. Wenn STR2 in STR1 nicht vorkommt, dann gilt OUT := 0.

FUNCTION FIND: INT

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

Beispiel in AWL:

```
LD 'SUXYSI'
FIND 'XY'
ST Var1 (* Ergebnis ist 3 *)
```

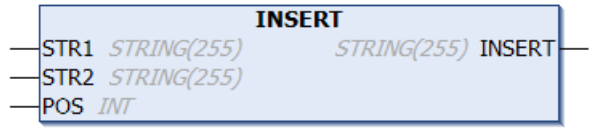
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4.4 INSERT



Die Funktion INSERT fügt einen String ab einer bestimmten Stelle in einen anderen ein.
INSERT(STR1, STR2, POS) bedeutet: Füge STR2 in STR1 nach der POS-ten Stelle ein.

FUNCTION INSERT: STRING (255)

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

Beispiel in AWL:

```
LD 'SUSI'
INSERT 'XY',2
ST Var1 (* Ergebnis ist 'SUXYSI' *)
```

Beispiel in ST:

```
Var1 := INSERT('SUSI','XY',2);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4.5 LEFT

Die Funktion LEFT liefert einen linken Anfangsstring eines Strings.
LEFT (STR, SIZE) bedeutet: Nehme die ersten SIZE Zeichen von links im String STR.

FUNCTION LEFT: STRING (255)

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

Beispiel in AWL:

```
LD 'SUSI'
LEFT 3
ST Var1 (* Ergebnis ist 'SUS' *)
```

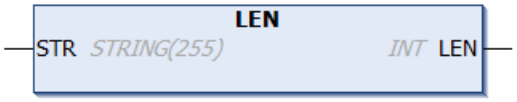
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4.6 LEN



Die Funktion LEN gibt die Länge eines Strings aus.

FUNCTION LEN: INT

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

Beispiel in AWL:

```
LD 'SUSI'
LEN
ST Var1 (* Ergebnis ist 4 *)
```

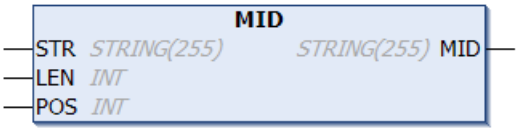
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4.7 MID



Die Funktion MID liefert einen Teilstring eines Strings.
MID (STR, LEN, POS) bedeutet: Hole LEN Zeichen aus dem String STR und beginne mit dem Zeichen an der Stelle POS.

FUNCTION MID: STRING (255)

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

Beispiel in AWL:

```
LD 'SUSI'
MID 2,2
ST Var1 (* Ergebnis ist 'US' *)
```

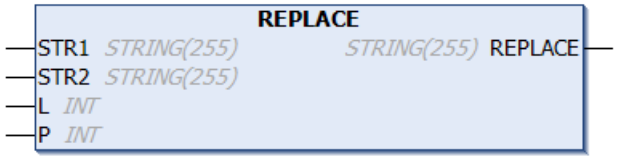
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4.8 REPLACE



Die Funktion REPLACE ersetzt einen Teilstring eines Strings durch einen anderen String.
REPLACE(STR1, STR2, L, P) bedeutet: Ersetze L Zeichen aus STR1 durch STR2 und beginne mit dem P-ten Zeichen.

FUNCTION REPLACE: STRING (255)

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

Beispiel in AWL:

```
LD 'SUXYSI'
REPLACE 'K',2,2
ST Var1 (* Ergebnis ist 'SKYSI' *)
```

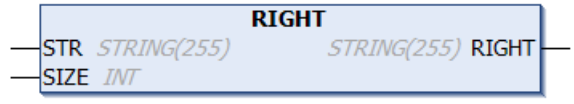
Beispiel in ST:

```
Var1 := REPLACE('SUXYSI','K',2,2);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

4.9 RIGHT



Die Funktion RIGHT liefert einen rechten Anfangsstring eines Strings.
RIGHT (STR, SIZE) bedeutet: Nehme die ersten SIZE Zeichen von rechts im String STR.

FUNCTION RIGHT: STRING (255)

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

Beispiel in AWL:

```
LD 'SUSI'
RIGHT 3
ST Var1 (* Ergebnis ist 'USI' *)
```

Beispiel in ST:

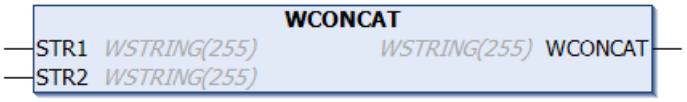
```
Var1 := RIGHT ('SUSI',3);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5 Stringfunktionen (WSTRING)

5.1 WCONCAT



Konkatenation (Aneinanderhängen) von zwei 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
```

Beispiel in AWL:

```
LD "SUSI"
WCONCAT "WILLI"
ST Var1 (*Ergebnis ist "SUSIWILLI"*)
```

Beispiel in ST:

```
Var1 := WCONCAT ("SUS","WILLI");
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5.2 WDELETE

Die Funktion WDELETE löscht ab einer bestimmten Stelle einen Teil aus einem WSTRING. Der Eingang STR ist vom Typ WSTRING. LEN und POS sind vom Typ INT. Der Rückgabewert der Funktion ist vom Typ WSTRING.
WDELETE (STR, LEN, POS) bedeutet: Lösche LEN Zeichen aus STR und beginne mit dem POS-ten.

FUNCTION WDELETE: WSTRING (255)

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

Beispiel in AWL:

```
LD "SUXYSI"
WDELETE 2,3
ST Var1 (*Ergebnis ist "SUSI"*)
```

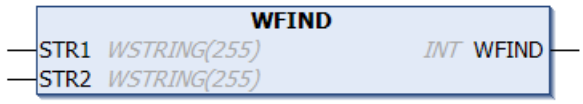
Beispiel in ST:

```
Var1 := WDELETE ("SUXYSI",2,3);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5.3 WFIND



Die Funktion WFIND sucht einen Teil in einem WSTRING.
WFIND(STR1, STR2) bedeutet: Finde die Position des ersten Zeichens des ersten Vorkommens von STR2 in STR1. Wenn STR2 in STR1 nicht vorkommt, dann gilt OUT := 0.

FUNCTION WFIND: INT

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

Beispiel in AWL:

```
LD "SUXYSI"
WFIND "XY"
ST Var1 (*Ergebnis ist 3*)
```

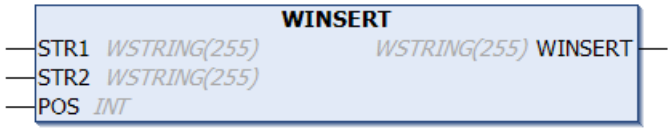
Beispiel in ST:

```
Var1 := WFIND ("SUXYSI","XY");
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5.4 WINSERT



Die Funktion WINSERT fügt einen WString ab einer bestimmten Stelle in einen anderen ein.
WINSERT (STR1, STR2, POS) bedeutet: Füge STR2 in STR1 nach der POS- ten Stelle ein.

FUNCTION WINSERT: WSTRING (255)

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

Beispiel in AWL:

```
LD "SUSI"
WINSERT "XY",2
ST Var1 (*Ergebnis ist "UXYSI"*)
```

Beispiel in ST:

```
Var1 := WINSERT ("SUSI","XY",2);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5.5 WLEFT

Die Funktion WLEFT liefert einen linken Anfangsstring eines WSTRINGs.
WLEFT (STR, SIZE) bedeutet: Nehme die ersten SIZE Zeichen von links im WString STR.

FUNCTION WLEFT: WSTRING (255)

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

Beispiel in AWL:

```
LD "SUSI"
WLEFT 3
ST Var1 (*Ergebnis ist "SUS"*)
```

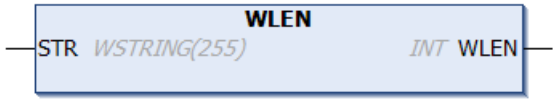
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5.6 WLEN



Die Funktion WLEN gibt die Länge eines WSTRINGs aus.

FUNCTION WLEN: INT

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

Beispiel in AWL:

```
LD "SUSI"
WLEN
ST Var1 (*Ergebnis ist 4*)
```

Beispiel in ST:

```
Var1 := WLEN ("SUSI");
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5.7 WMID

Die Funktion WMID liefert einen Teilstring eines WSTRINGs.
WMID(STR, LEN, POS) bedeutet: Hole LEN Zeichen aus dem WSTTRING STR und beginne mit dem Zeichen an der Stelle POS.

FUNCTION WMID: WSTRING (255)

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

Beispiel in AWL:

```
LD "SUSI"
WMID 2,2
ST Var1 (*Ergebnis ist "US"*)
```

Beispiel in ST:

```
Var1 := WMID ("SUSI",2,2);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5.8 WREPLACE

Die Funktion WREPLACE ersetzt einen Teilstring eines WSTRINGs durch einen anderen WSTRING.
WREPLACE (STR1, STR2, L, P) bedeutet: Ersetze L Zeichen aus STR1 durch STR2 und beginne mit dem P-ten Zeichen.

FUNCTION WREPLACE: WSTRING (255)

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

Beispiel in AWL:

```
LD "SUXYSI"
WREPLACE "XY",2
ST Var1 (*Ergebnis ist "SKYSI"*)
```

Beispiel in ST:

```
Var1 := WREPLACE ("SUXYSI","K",2,2);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

5.9 WRIGHT

Die Funktion WRIGHT liefert einen rechten Anfangswstring eines WSTRINGs.
WRIGHT (STR, SIZE) bedeutet: Nehme die ersten SIZE Zeichen von rechts im WString STR.

FUNCTION WRIGHT: WSTRING (255)

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

Beispiel in AWL:

```
LD "SUSI"
WRIGHT 3
ST Var1 (*Ergebnis ist "USI"*)
```

Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

6 Globale Konstanten

6.1 Bibliotheksversion

Alle Bibliotheken haben eine bestimmte Version. Diese Version wird im Repository der SPS-Bibliothek angezeigt.
Die Versionsnummer der Bibliothek ist in einer globalen Konstante gespeichert.

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_Standard : ST_LibVersion;
END_VAR
```

Name	Typ	Beschreibung
stLibVersion_Tc2_Standard	ST_LibVersion	Versionsnummer der Tc2_Standard-Bibliothek

Zum Vergleich zwischen vorhandener und erforderlicher Version dient die Funktion F_CmpLibVersion.

Kompatibilität zu TwinCAT 2

Abfragemöglichkeiten von TwinCAT 2 Bibliotheken sind nicht mehr verfügbar.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliothek
TwinCAT v3.0.0	PC oder CX (x86)	Tc2_Standard

Trademark statements

Mehr Informationen:
www.beckhoff.com/te1000

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com