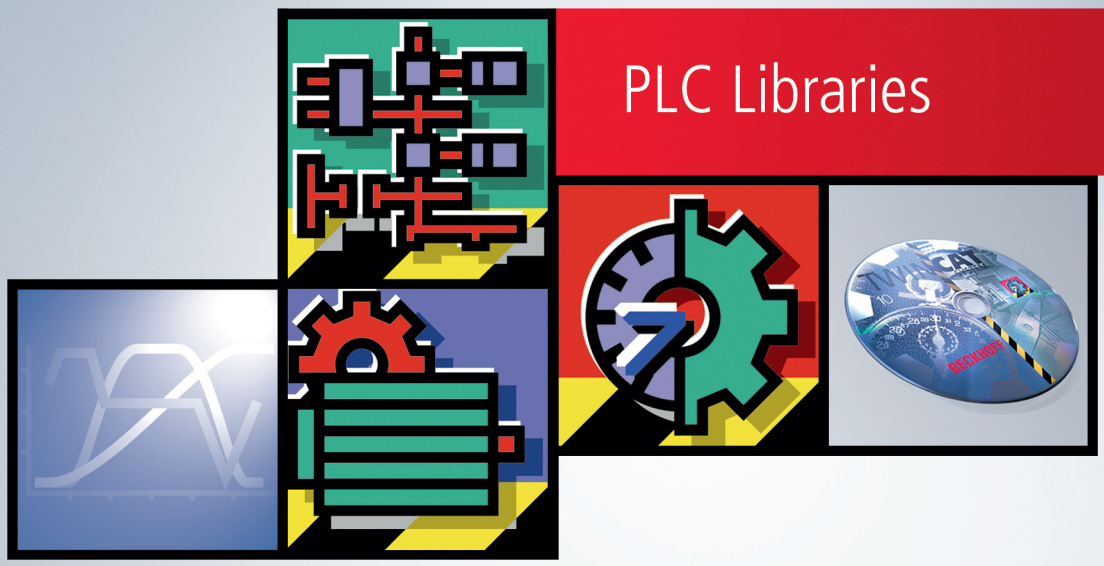
BECKHOFF New Automation Technology

Handbuch | DE

TX1200

TwinCAT 2 | PLC-Bibliothek: Standard



12.04.2022 | Version: 1.0

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, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche 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 weiter entwickelt.
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®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® 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 Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT®

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

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

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.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

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

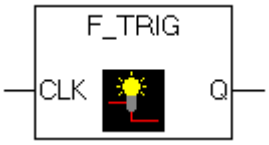
Die Standardbibliothek enthält folgenden in der Norm IEC61131-3 vorgesehenen Bausteine:

- Bistabile Funktionsblöcke [▶ 10]
- Zähler [▶ 12]
- Timer [▶ 14]
- Trigger [▶ 9]
- String Funktionen [▶ 18]

3 Funktionsbausteine

3.1 Trigger

3.1.1 F_TRIG



Detektor für fallende Flanke.

VAR_INPUT

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

VAR_OUTPUT

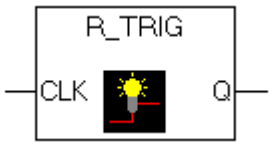
```
VAR_OUTPUT
    Q       : BOOL; (* Edge detected *)
END_VAR
VAR
    M       : BOOL;
END_VAR
```

Solange die Eingabevariable CLK TRUE liefert, solange werden die Ausgabe Q und die Hilfsvariable M FALSE sein. Sobald CLK FALSE liefert, wird zuerst Q TRUE liefern, und dann M auf TRUE gesetzt. 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-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.1.2 R_TRIG



Detektor für eine ansteigende Flanke.

VAR_INPUT

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

VAR_OUTPUT

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

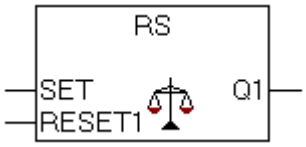
Solange die Eingabevariable CLK FALSE liefert, solange werden die Ausgabe Q und die Hilfsvariable M FALSE sein. Sobald CLK TRUE liefert, wird zuerst Q TRUE liefern, und dann M auf TRUE gesetzt. D.h.: bei jedem weiteren Aufruf der Funktion wird Q wieder FALSE liefern, bis CLK eine fallende und wieder eine steigende Flanke hat.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.2 Bistable

3.2.1 RS



Bistabiler Funktionsblock, reset ist dominant.

VAR_INPUT

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

VAR_OUTPUT

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

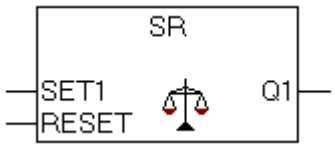
Interne implementierung des FB's:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.2.2 SR



Bistabiler Funktionsblock, setzen ist dominant.

VAR_INPUT

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

VAR_OUTPUT

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

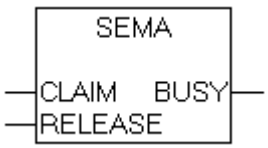
Interne implementierung des FB's:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.2.3 SEMA



Ein Software-Semaphor (unterbrechbar von einer anderen höherprioren Task!).

VAR_INPUT

```
VAR_INPUT
    CLAIM   : BOOL;
    RELEASE : BOOL;
END_VAR
```

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY    : BOOL;
END_VAR
```

Wenn SEMA aufgerufen wird und BUSY ist TRUE, dann bedeutet das, dass SEMA bereits vorher belegt wurde (SEMA wurde mit CLAIM = TRUE aufgerufen). Wenn BUSY FALSE ist, dann wurde SEMA noch nicht aufgerufen, oder es wurde freigegeben (Aufruf mit RELEASE = TRUE).

Interne Implementierung des FB's:

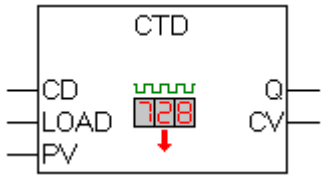
```
BUSY := X;

IF CLAIM THEN
    X:=TRUE;
ELSIF RELEASE
    THEN
        BUSY := FALSE;
        X:= FALSE;
END_IF
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.3 Counter

3.3.1 CTD



Abwärtszähler.

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
```

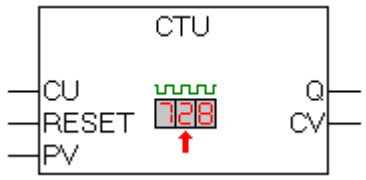
Wenn LOAD TRUE ist, wird die Zählvariable CV mit der Obergrenze PV initialisiert. Wenn CD eine steigende Flanke von FALSE auf TRUE hat, wird CV um 1 erniedrigt, solange CV größer als 0 ist (Wenn also kein Unterlauf verursacht wird). Q liefert TRUE, wenn CV kleiner oder gleich 0 ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.3.2 CTU



Aufwärtszähler.

VAR_INPUT

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

VAR_OUTPUT

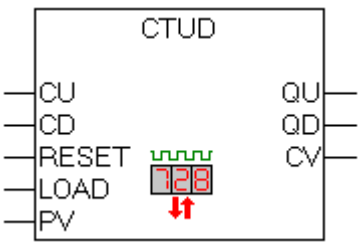
```
VAR_OUTPUT
    Q       : BOOL; (* Counter reached the Limit *)
    CV      : WORD; (* Current Counter Value *)
END_VAR
```

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). Q liefert TRUE, wenn CV größer oder gleich der Obergrenze PV ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.3.3 CTUD



Auf- und Abwärtszähler.

VAR_INPUT

```
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
```

VAR_OUTPUT

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

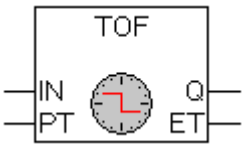
Wenn RESET gilt, dann wird die Zählvariable CV mit 0 initialisiert. Wenn LOAD gilt, dann wird CV mit PV initialisiert. Wenn CU eine steigende Flanke von FALSE auf TRUE hat, dann wird CV um 1 erhöht, solange CV keinen Überlauf verursacht. Wenn CD eine steigende Flanke von FALSE auf TRUE hat, dann wird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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.4 Timer

3.4.1 TOF



Timer off-delay.

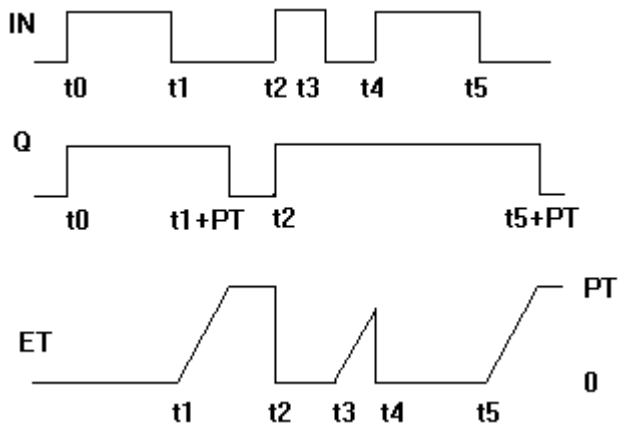
VAR_INPUT

```
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

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

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 gleich PT ist. Andernfalls ist es TRUE. Q hat somit eine fallende Flanke, wenn die in PT in Millisekunden angegebene Zeit abgelaufen ist. Graphische Darstellung des zeitlichen Verhaltens von TOF:

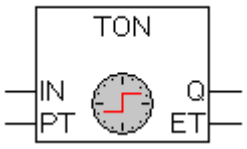


Die Funktion TOF benötigt 15 Byte Daten

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.4.2 TON



Timer on-delay

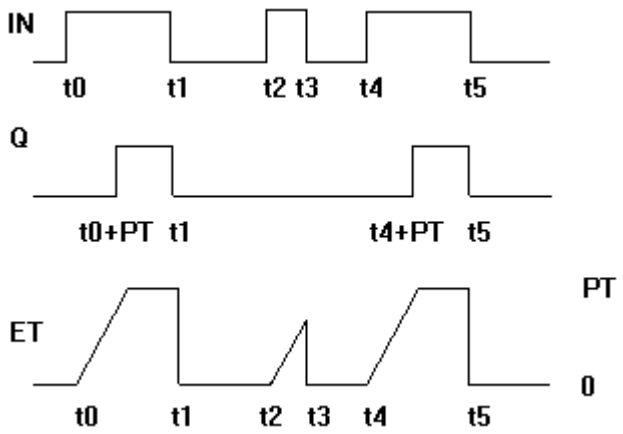
VAR_INPUT

```
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

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

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 gleich PT ist. Andernfalls ist es FALSE. Q hat somit eine steigende Flanke, wenn die in PT in Millisekunden angegebene Zeit abgelaufen ist. Graphische Darstellung des zeitlichen Verhaltens von TON:

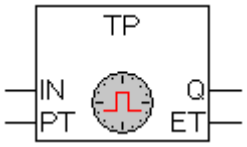


Die Funktion TON benötigt 15 Byte Daten

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

3.4.3 TP



Pulsgeber. Mit diesem Funktionsbaustein können Impulse mit einer definierten Impulsdauer generiert werden.

VAR_INPUT

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

VAR_OUTPUT

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

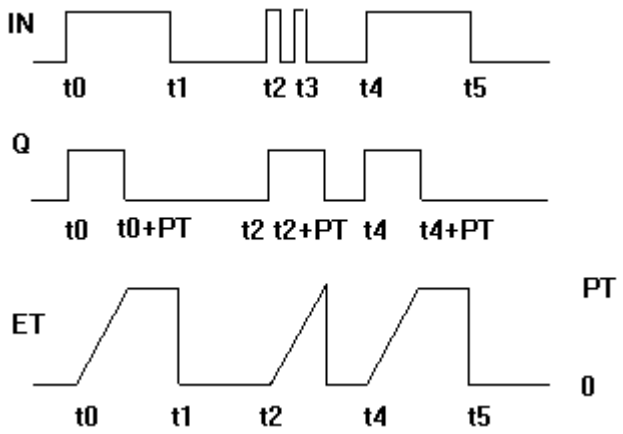
Wenn IN FALSE ist, sind die Ausgaben FALSE bzw. 0.

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.

Q liefert somit für den in PT angegebenen Zeitraum ein Signal. Graphische Darstellung des zeitlichen Ablaufs von TP:



Die Funktion TP benötigt 14 Byte Daten

Voraussetzungen

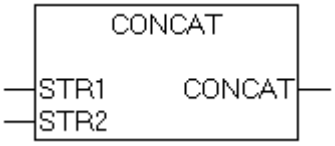
Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

4 Stringfunktionen

Bitte beachten:

String-Funktionen sind nicht sicher bei Taskwechsel: Bei der Verwendung von Tasks dürfen String-Funktionen nur in einer Task eingesetzt werden. Wird die gleiche Funktion in verschiedenen Tasks benützt, besteht die Gefahr des Überschreibens.

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' *)
```

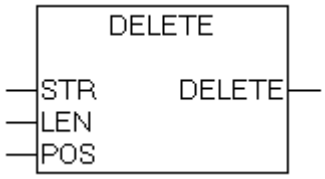
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

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 Funktionvom Typ STRING. DELETE(STR, LEN, POS) bedeutet: Lösche LEN Zeichen aus STR, beginnend 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-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

4.3 FIND

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:

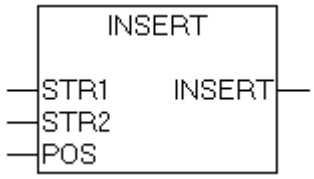
```
Var1 := FIND
('SUXYSI','XY');
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

4.4 INSERT



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-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

4.5 LEFT

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' *)
```

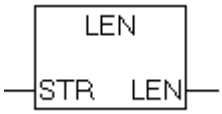
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

4.6 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 *)
```

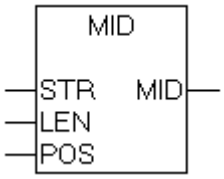
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

4.7 MID



Mid liefert einen Teilstring eines Strings. MID (STR, LEN, POS) bedeutet: Hole LEN Zeichen aus dem String STR, beginnend 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' *)
```

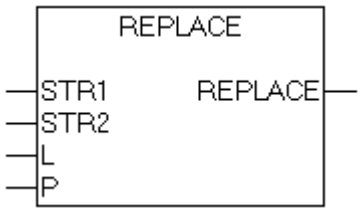
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

4.8 REPLACE



REPLACE ersetzt einen Teilstring eines Strings durch einen anderen String. REPLACE(STR1, STR2, L, P) bedeutet: Ersetze L Zeichen aus STR1 durch STR2 beginnend 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' *)
```

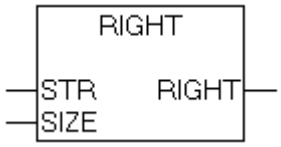
Beispiel in ST:

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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

4.9 RIGHT



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-Bibliotheken
TwinCAT Version >= 2.6.0	PC or CX (x86)	Standard.Lib
TwinCAT Version >= 2.6.0	BC (165)	Standard.Lb6
TwinCAT Version >= 2.9.0	BCxx50 or BX	Standard.lbx
TwinCAT Version >= 2.10.0 Build >= 1301	CX (ARM)	Standard.lib

Mehr Informationen:
www.beckhoff.de/tx1200

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