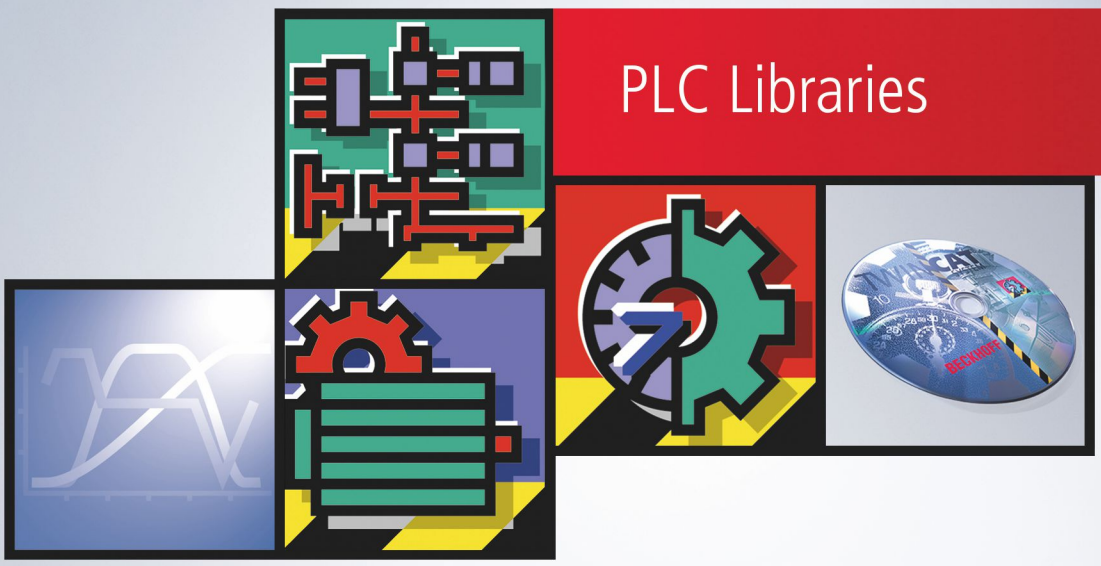
BECKHOFF New Automation Technology

Handbuch | DE

TX1200

TwinCAT 2 | PLC-Bibliothek: TcPlcUtilitiesBC



10.11.2022 | Version: 1.1

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 Bibliothek beinhaltet nützliche Funktionsbausteine für die SPS Controller (BCxxxx Controller). Neben den RTC-Bausteinen beinhaltet die Bibliothek einen Funktionsbaustein für die Dekodierung des DCF-77 Zeitsignals und einige Konvertierungsfunktionen. Intern werden die Systemfunktionen der SPS Controller aufgerufen.

Anforderungen

Einige der Funktionen werden nur von SPS Controllern mit einer neueren Firmware-Version unterstützt. Die erforderlichen Firmware-Versionen sind in der Dokumentation zu der PlcSystemBC Bibliothek aufgeführt.

Inhalt der Bibliothek

Funktionsbausteine

Name	Beschreibung
RTC [▶ 10]	Real Time Clock
RTC_EX [▶ 11]	Real Time Clock mit einem zusätzlichen Millisekunden-Ausgang
DCF77_TIME [▶ 13]	Die DCF77 Funkzeit lesen
ReadWriteTerminalReg [▶ 17]	Zugriff auf die Register der Klemmen
DRAND [▶ 19]	Randomzahlen-Generator
FB_BasicPID [▶ 20]	Einfacher PID Controller

Funktionen

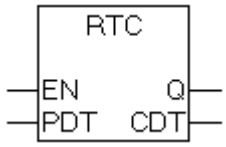
Name	Beschreibung
DT_TO_SYSTEMTIME [▶ 23]	DATE_AND_TIME in Windows Systemzeit-Struktur konvertieren
SYSTEMTIME_TO_DT [▶ 23]	Windows Systemzeit-Struktur in DATE_AND_TIME konvertieren
TIME_TO_OTSTRUCT [▶ 24]	TIME in eine Struktur mit Millisekunden, Sekunden, Minuten, Stunden, Tagen und Wochen konvertieren
OTSTRUCT_TO_TIME [▶ 25]	Struktur mit Millisekunden, Sekunden, Minuten, Stunden, Tagen und Wochen in eine TIME-Variable konvertieren
SETBIT32 [▶ 26]	Setzt ein Bit in einer 32-Bit Variablen auf 1
CSETBIT32 [▶ 27]	Setzt / rücksetzt ein Bit in einer 32-Bit Variablen
GETBIT32 [▶ 27]	Ermittelt den Wert eines Bits einer 32-Bit Variable
CLEARBIT32 [▶ 28]	Setzt ein Bit in einer 32-Bit Variablen auf Null
F_SwapReal [▶ 29]	Konvertiert Buscontroller REAL-Zahlen in den 4Byte REAL Intel-Format.
IsFinite [▶ 30]	Überprüft die Formatierung einer Gleitkommazahl nach der IEEE

Datenstrukturen

Name	Beschreibung
TIMESTRUCT [▶ 33]	Aufbau der Windows Systemzeit-Struktur
OTSTRUCT [▶ 33]	Aufbau der Struktur: Operating time

3 Funktionsbausteine

3.1 RTC



Mit dem Funktionsbaustein "RTC" (Real Time Clock) kann eine interne Uhr in einem SPS Controller (BCxxxx) realisiert werden. Die Uhr muss mit einem Anfangsdatum und einer Uhrzeit initialisiert werden. Nach der Initialisierung wird die Uhrzeit und das Datum nach jedem Aufruf des Funktionsbausteins zyklisch aktualisiert. Um die aktuelle Uhrzeit und das Datum zu berechnen, wird ein Systemtakt des Controllers benutzt. Der Systemtakt hat eine Auflösung von einer Millisekunde. Damit die aktuelle Zeit berechnet werden kann, sollte der Funktionsbaustein in jedem Zyklus der SPS einmal aufgerufen werden. Der RTC hat eine Abweichung von ca. 1 Minute pro 24 Stunden. Um größere Abweichungen zu vermeiden, kann der RTC zyklisch (z.B. mit einer Funkuhr oder über den Feldbus mit einem TwinCAT-PC) synchronisiert werden. Am Ausgang des Funktionsbausteines steht das aktuelle Datum und Uhrzeit in dem gängigen DATE_AND_TIME (DT) Format zur Verfügung. In einem SPS-Programm können mehrere Instanzen von dem RTC-Funktionsbaustein erzeugt werden.

VAR_INPUT

```
VAR_INPUT

    EN  :  BOOL;

    PDT : DATE_AND_TIME;

END_VAR
```

EN: Bei einer steigenden Flanke an diesem Eingang wird der Funktionsbaustein mit einer vorgegebenen Uhrzeit und Datum neu initialisiert.
PDT: (Preset Date and Time) Die Initialisierungswerte für das Datum und Uhrzeit des Funktionsbausteins. Bei einer steigenden Flanke an dem EN-Eingang wird dieser Wert von dem Funktionsbaustein übernommen.

VAR_OUTPUT

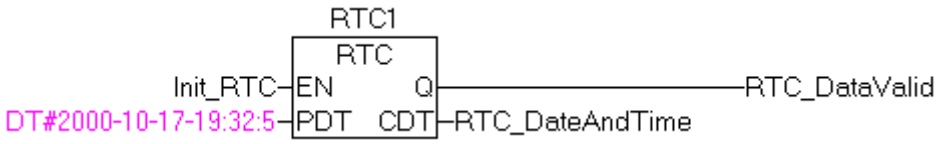
```
VAR_OUTPUT

    Q  :   BOOL;

    CDT: DATE_AND_TIME;

END_VAR
```

Q: Wurde der Funktionsbaustein mindestens einmal initialisiert, wird dieser Ausgang gesetzt. Ist dieser Ausgang gesetzt, dann sind die Werte für das Datum und Uhrzeit am PDT-Ausgang gültig.
CDT: (Current Date and Time) Aktuelles Datum und Uhrzeit vom RTC. Der CDT-Ausgang wird nur dann aktualisiert, wenn der Funktionsbaustein aufgerufen wurde. Daher sollten die Instanzen des Funktionsbausteines einmal in jedem Zyklus der SPS aufgerufen werden.

Beispiel für einen Aufruf in FUP:

```
PROGRAM MAIN
VAR
    RTC1           : RTC;
    Init_RTC       : BOOL;
```

```
    RTC_DataValid  : BOOL;
    RTC_DateAndTime: DT;
END_VAR
```

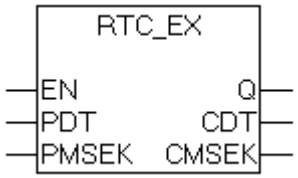


Voraussetzungen

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

3.2 RTC_EX



Mit dem Funktionsbaustein "RTC_EX" kann eine interne Uhr in einem SPS Controller (BCxxxx) realisiert werden. Die Uhr muss mit einem Anfangsdatum und einer Uhrzeit initialisiert werden. Nach der Initialisierung wird die Uhrzeit und das Datum nach jedem Aufruf des Funktionsbausteins zyklisch aktualisiert. Um die aktuelle Uhrzeit und das Datum zu berechnen, wird ein Systemtakt des Controllers benutzt. Der Systemtakt hat eine Auflösung von einer Millisekunde. Damit die aktuelle Zeit berechnet werden kann, sollte der Funktionsbaustein in jedem Zyklus der SPS einmal aufgerufen werden.
Der RTC_EX-Funktionsbaustein hat eine Abweichung von ca. 1 Minute pro 24 Stunden. Um größere Abweichungen zu vermeiden, kann der RTC_EX-Funktionsbaustein zyklisch (z.B. mit einer Funkuhr oder über den Feldbus mit einem TwinCAT-PC) synchronisiert werden. Am Ausgang des Funktionsbausteins steht das aktuelle Datum und Uhrzeit in dem gängigen DATE_AND_TIME (DT) Format zur Verfügung. Im Gegensatz zu dem RTC [▶ 10]-Funktionsbaustein hat RTC_EX eine Millisekunden-Genauigkeit. Da die DATE_AND_TIME-Variablen nur eine Sekunden-Genauigkeit besitzen, werden Millisekunden über die CMSEK-Ausgangsvariable ausgegeben. In einem SPS-Programm können mehrere Instanzen von dem RTC_EX-Funktionsbaustein erzeugt werden.

VAR_INPUT

```
VAR_INPUT

EN   :    BOOL;

PDT  :   DATE_AND_TIME;
```

```
PMSEK: DWORD;

END_VAR
```

EN: Bei einer steigenden Flanke an diesem Eingang wird der RTC_EX-Funktionsbaustein mit einer vorgegebenen Uhrzeit, Datum und Millisekunden neu initialisiert.
PDT: (Preset Date and Time) Die Initialisierungswerte für das Datum und Uhrzeit des Funktionsbausteins. Bei einer steigenden Flanke an dem EN-Eingang wird dieser Wert von dem Funktionsbaustein übernommen.
PMSEK: (Preset Milliseconds) Der Initialisierungswert für die Millisekunden. Bei einer steigenden Flanke an dem EN-Eingang wird dieser Wert von dem Funktionsbaustein übernommen.

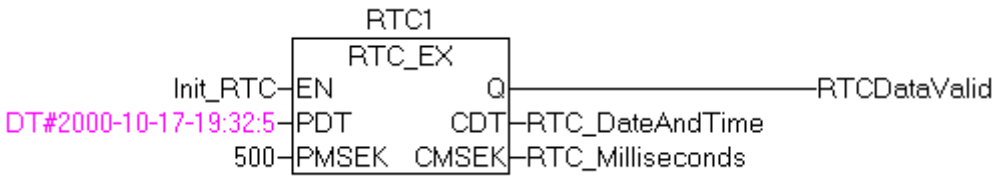
VAR_OUTPUT

```
VAR_OUTPUT


Q    : BOOL;


CDT  : DATE_AND_TIME;


CMSEK: DWORD;

END_VAR
```

Q: Wurde der Funktionsbaustein mindestens ein Mal initialisiert, wird dieser Ausgang gesetzt. Ist dieser Ausgang gesetzt, dann sind die Werte für das Datum, Uhrzeit und Millisekunden am PDT-Ausgang und CMSEK-Ausgang gültig.
CDT: (Current Date and Time) Aktuelles Datum und Uhrzeit vom RTC. Der CDT-Ausgang wird nur dann aktualisiert, wenn der Funktionsbaustein aufgerufen wurde. Daher sollten die Instanzen des Funktionsbausteines ein Mal in jedem Zyklus der SPS aufgerufen werden.
CMSEK: (Current Milliseconds) Der Millisekunden-Ausgang.

Beispiel für einen Aufruf in FUP:

```
PROGRAM MAIN
VAR
    RTC1                : RTC_EX;
    Init_RTC            : BOOL;
    RTCDataValid        : BOOL;
    RTC_DateAndTime     : DT;
    RTC_Milliseconds    : DWORD;
END_VAR
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

3.3 DCF77_TIME

Mit dem Funktionsbaustein "DCF77_TIME" kann das DCF-77 Funkuhr-Signal dekodiert werden. Über eine steigende Flanke am RUN-Eingang wird die Dekodierung gestartet und wird fortgesetzt, solange der RUN-Eingang gesetzt ist. Der Funktionsbaustein braucht im schlechtesten Fall maximal eine Minute um sich zu synchronisieren und eine weitere Minute um Daten für die nächste Minute zu dekodieren. Während dieser Zeit wird auf die fehlende 59. Sekundenmarke gewartet. Intern findet bei dem Funktionsbaustein eine Abtastung des DCF-77 Signals statt. Um die Flanken fehlerfrei abtasten zu können, sollte der Funktionsbaustein in jedem Zyklus der SPS ein Mal aufgerufen werden. Bei einer Zykluszeit <= 25ms. können zufriedenstellende Ergebnisse erzielt werden. Bei einem fehlenden oder fehlerhaften DCF-77 Signal wird der ERR-Ausgang auf TRUE gesetzt und entsprechender Fehlercode am ERRID-Ausgang gesetzt. Bei dem nächsten fehlerfreien Empfang werden die ERR- und ERRID-Ausgänge zurückgesetzt. Manche der Empfänger liefern ein invertiertes DCF-77 Signal. In so einem Fall muss das Signal zuerst invertiert werden und dann an den DCF_PULSE-Eingang geleitet werden. Bei einem fehlerfreien Betrieb wird die aktuelle Zeit im Minutentakt am CDT-Ausgang aktualisiert. Dabei wird für einen Zyklus der SPS bei der Nullten Sekunde der READY-Ausgang auf TRUE gesetzt. In dieser Zeit ist die DCF-77 Zeit am CDT-Ausgang gültig und kann im SPS-Programm ausgewertet werden. Der READY-Ausgang wird nur dann gesetzt, wenn die Daten für die kommende Minute erkannt wurde. Die Fehlererkennung wird mit Hilfe der übertragenen Paritätsbits durchgeführt. Bei schlechten Empfangsverhältnissen kann eine zu 100% fehlerfreie Erkennung nicht gewährleistet werden. D.h. bei zwei fehlerhaften (invertierten) Bits kann der Funktionsbaustein keinen Fehler erkennen und setzt den READY-Ausgang ebenfalls auf TRUE. Um eine zuverlässige Zeitinformation zu erhalten müssen zusätzliche Sicherungsmechanismen implementiert werden z.B. die Auswertung der Redundanz der Zeitinformation in aufeinanderfolgenden Minuten.

Ab TwinCAT v2.10 Build > 1340 und TwinCAT v2.11 Build > 1543 wurde in dem DCF77_TIME-Funktionsbaustein eine einfache Plausibilitätsprüfung von zwei aufeinanderfolgenden Telegrammen implementiert. Diese Funktionalität kann über eine globale boolesche Variable für alle Instanzen des DCF77_TIME-Bausteins aktiviert werden. Bei der Aktivierung der Plausibilitätsprüfung verlängert sich die erste Synchronisierung um eine weitere Minute auf maximal 3 Minuten.

```
GLOBAL_DCF77_SEQUENCE_CHECK : BOOL := FALSE;
```

TRUE = Plausibilitätsprüfung ist aktiviert (zwei aufeinanderfolgende Telegramme werden geprüft)

FALSE = Plausibilitätsprüfung ist deaktiviert

Die auftretenden Fehler während des Empfangs werden von dem Funktionsbaustein registriert. Der ERRCNT-Ausgang ist ein Fehlerzähler. Der Zähler gibt Auskunft über die Anzahl der aufgetretenen Fehler seit dem letzten fehlerfreien Empfang. Bei dem nächsten fehlerfreien Empfang wird dieser Zähler zurückgesetzt.

Zeitcode

Während jeder Minute werden die Nummern von Jahr, Monat, Tag, Wochentag, Stunde, Minute BCD-codiert durch Impulsmodulation der Sekundenmarken übertragen. Die übertragenen Informationen gelten jeweils für die nachfolgende Minute. Jede Sekunde wird eine der Sekundenmarken übertragen. Dabei entspricht eine Sekundenmarke mit der Dauer von 0.1s einer binären Null und eine mit der Dauer von 0.2s einer binären Eins. Die Informationen werden mit 3 Prüfbits ergänzt. Bei der 59. Sekunde fehlt die Sekundenmarke und diese "Lücke" kann zur Synchronisation des Empfängers benutzt werden.

Ab TwinCAT v2.10 Build > 1340 und TwinCAT v2.11 Build > 1543 kann auch über eine globale Variable die Länge des kurzen und langen Impulssignals konfiguriert werden. Bei einem gestörten Signal sind die Impulsbreiten immer kleiner. In der Empfängerspezifikationen finden sich meistens Angaben über minimale und maximale Impulsbreiten der beiden logischen Signale, wobei breitere Impulse bei höheren Feldstärken zu erwarten sind und schmale Impulse bei niedrigen Feldstärken oder entsprechenden Störverhältnissen. In der Nähe des Senders (wo die Feldstärke sehr groß ist) können ebenfalls Probleme entstehen wenn die Impulsbreite der logischen Null zu groß wird. Deshalb wird abhängig von der Spezifikation des Empfängers eine feste Grenze zur Unterscheidung zwischen Null und Eins festgelegt. **Prüfen Sie deshalb die Spezifikation des eingesetzten Empfängers und konfigurieren entsprechend die Impulslänge.**

```
GLOBAL_DCF77_PULSE_SPLIT : TIME := T#140ms;
```

0 == pulse < 140ms, 1 == pulse > 140

Z.B.: in der Spezifikation von Atmel T4227 (Time Code Receiver) wird folgende Pulslänge angegeben:
100ms Pulse (Null): Min: 70ms, Typical: 95ms, Max: **130ms**
200ms Pulse (Eins): Min **170ms**, Typical 195ms, Max 235ms
Für diesen IC wäre ein Grenzwert von 150ms optimal = 130 + ((170ms - 130ms) / 2).

Tipp:

Wenn der konfigurierte Grenzwert für die Impulslänge einen zu kleinen Wert hat dann werden kurze Impulse als lange erkannt. Umgekehrt gilt, wenn der konfigurierte Grenzwert zu groß ist dann werden lange Impulse als kurze erkannt. Der Empfänger kann bei einer passenden Checksumme diese Fehler nicht erkennen. Im ersten Fall liefert der Empfänger möglicherweise Zeiten die weit in der Zukunft liegen und im zweiten Fall Zeiten aus der Vergangenheit.

VAR_INPUT

```
VAR_INPUT
    DCF_PULSE   :BOOL;
    RUN         :BOOL;
END_VAR
```

DCF_PULSE: Das DCF-77 Signal.
RUN: Eine steigende Flanke an diesem Eingang initialisiert den Funktionsbaustein und startet die Dekodierung des DCF-77 Signals. Wird dieser Eingang zurückgesetzt, dann wird die Dekodierung gestoppt

VAR_OUTPUT

```
VAR_OUTPUT
    BUSY        :BOOL;
    ERR         :BOOL;
    ERRID       :UDINT;
    ERRCNT      :UDINT;
    READY       :BOOL;
    CDT         :DATE_AND_TIME;
END_VAR
```

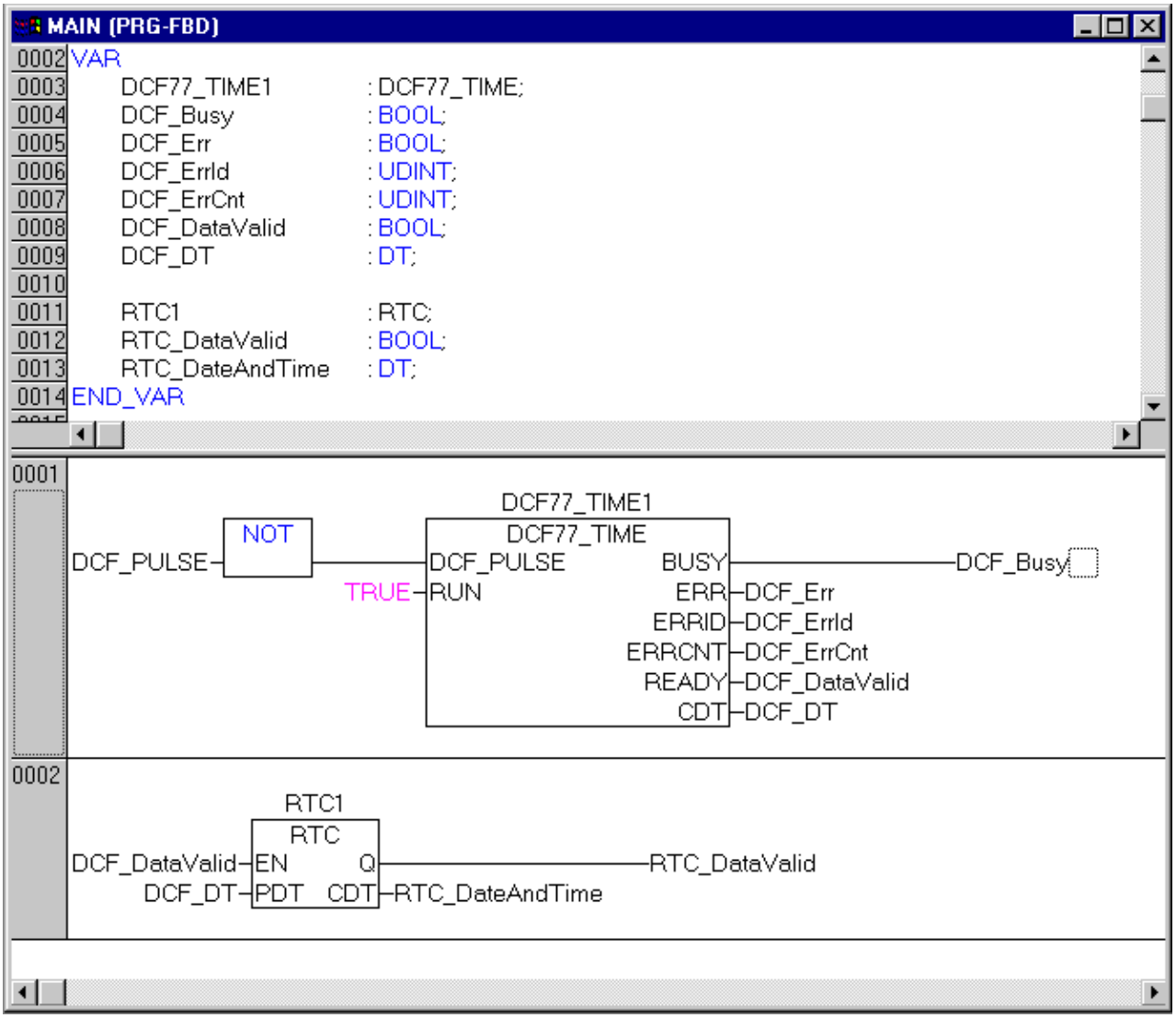
BUSY: Bei der Aktivierung des Funktionsbausteins wird dieser Ausgang gesetzt.
ERR: Ist ein Fehler bei der Dekodierung aufgetreten, dann wird dieser Ausgang gesetzt.
ERRID: Liefert bei einem gesetzten ERR-Ausgang die Fehlernummer.
ERRCNT: Anzahl der aufgetretenen Fehler seit dem letzten fehlerfreien Empfang.
READY: Ist dieser Ausgang gesetzt, dann sind die Daten am CDT-Ausgang gültig.
CDT: Die DCF-77 Zeit im DATE_AND_TIME-Format.

Fehlerbeschreibung:

Fehlercodes	Fehlerbeschreibung
0	kein Fehler
0x100	Timeout-Fehler. Möglicherweise kein DCF-77 Signal erkannt.
0x200	Parity-Fehler. In den empfangenen Daten wurden fehlerhafte Bits erkannt.
0x300	Fehlerhafte Daten wurden empfangen. Da bei der Parity Prüfung nur ein falsches Bit erkannt werden kann, werden die empfangenen Daten noch auf Gültigkeit überprüft (bei Monat = 13 wird z.B. dieser Fehlercode gesetzt).
0x400	Der letzte Dekodierungszyklus war zu lang. Dieser Fehler kann bei einem schlechten Empfang auftreten (zu wenig Sekundenmarken wurden empfangen).
0x500	Der letzte Dekodierungszyklus war zu kurz. Dieser Fehler kann bei einem schlechten Empfang auftreten (zusätzliche Flanken wurden empfangen).

Beispiel für einen Aufruf in FUP

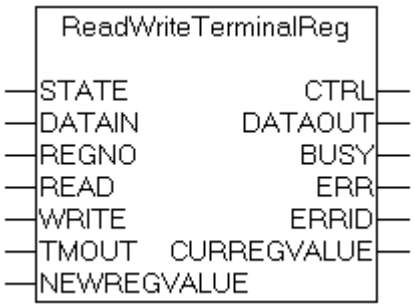
In der Beispielanwendung wird bei einem fehlerfreien Empfang der Real Time Clock mit der Funkzeit synchronisiert.



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

3.4 ReadWriteTerminalReg



Der Funktionsbaustein "ReadWriteTerminalReg" ermöglicht einen komfortableren Zugriff auf die Register der Klemme über das Status-/Controlbyte des Klemmenkanals (Registerkommunikation). In der Standardbetriebsart werden die Daten Ein-/Ausgänge der intelligenten Klemmen (z.B. einer analogen Ausgangsklemme) für den Austausch der analogen Ausgangsdaten benutzt. Ein Handshake über das Status-/Controlbyte ermöglicht einen Registerzugriff, dabei werden die Daten Ein-/Ausgangsvariablen zum Übertragen der Registerwerte benutzt. Die Klemme muss als komplexe SPS-Klemme gemappt sein, damit das Status-/Controlbyte im Prozessabbild sichtbar wird. Es kann nicht auf die Register der Klemme zugegriffen werden, wenn die Klemme als kompakte SPS-Klemme oder als Feldbusklemme (Prozessabbild der Klemme nicht für den Bus-Controller sichtbar) gemappt wurde.
Über eine positive Flanke an dem READ- oder WRITE-Eingang wird das Register mit der Nummer REGNO gelesen bzw. in das Register geschrieben. Der Registerschreibschutz wird von dem Funktionsbaustein bei einem Schreibzugriff aufgehoben und anschließend neu gesetzt. Bei einem Schreibzugriff auf ein Register wird der neue Registerwert gelesen und steht an dem Ausgang CURRREGVALUE zur Verfügung. Um Änderungen der Registerwerte permanent zu speichern, muss die Spannungsversorgung des Kopplers unterbrochen oder ein Software-Reset des Kopplers durchgeführt werden.

VAR_INPUT

```
VAR_INPUT
    STATE           : BYTE;
    DATAIN          : WORD;
    REGNO           : BYTE;
    READ            : BOOL;
    WRITE           : BOOL;
    TMOUT           : TIME;
    NEWREGVALUE     : WORD;
END_VAR
```

STATE: Statusbyte des Klemmenkanals.
DATAIN: Dateneingangswort des Klemmenkanals.
REGNO: Nummer des Registers auf den ein Schreib- bzw. Lesezugriff erfolgen soll.
READ: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der aktuelle Registerwert gelesen. Bei Erfolg steht der Registerwert in der Ausgangsvariablen CURRREGVALUE zur Verfügung.
WRITE: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und in den Register REGNO der Wert der Eingangsvariablen NEWREGVALUE geschrieben. Anschließend wird der aktuelle Wert des Registers gelesen und steht bei Erfolg in der Ausgangsvariablen CURREGVALUE zur Verfügung.
TMOUT: Gibt die Timeout-Zeit an, die bei der Ausführung der Funktion nicht überschritten werden darf.
NEWREGVALUE: Datenwort, das bei einem Schreibzugriff in dem Register mit der Nummer REGNO geschrieben werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    CTRL          :BYTE;
    DATAOUT       :WORD;
    BUSY          :BOOL;
    ERR           :BOOL;
```

```
    ERRID         :UDINT;
    CURREGVALUE   :WORD;
END_VAR
```

CONTROL: Controlbyte des Klemmenkanals.
DATAOUT: Datenausgangswort des Klemmenkanals.
BUSY: Bei der Aktivierung des Bausteines wird dieser Ausgang gesetzt und bleibt gesetzt, bis die Ausführung der Funktion abgeschlossen wurde.
ERR: Sollte ein Fehler bei der Ausführung der Funktion auftreten, dann wird dieser Ausgang gesetzt, nachdem der BUSY-Ausgang zurückgesetzt wurde.
ERRID: Liefert bei einem gesetzten ERR-Ausgang die Fehlernummer.
CURREGVALUE: Bei einem erfolgreichen Lese- oder Schreibzugriff wird über die Variable der aktuelle Registerwert ausgegeben.

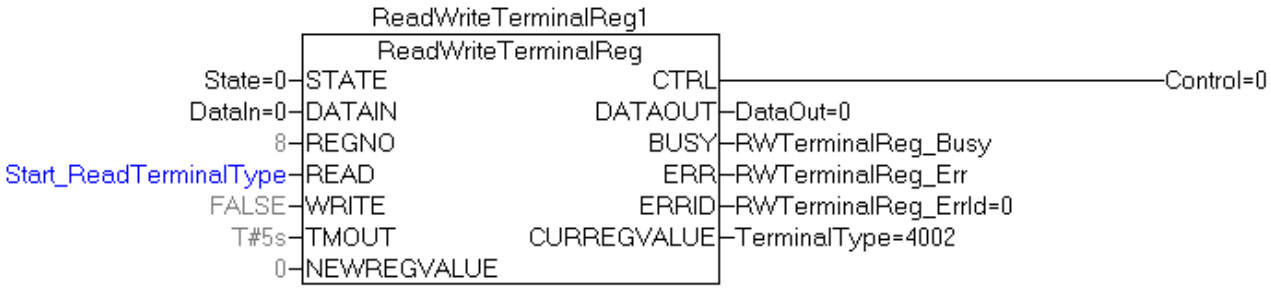
Fehlerbeschreibung:

Fehlernummer	Fehlerbeschreibung
0	kein Fehler
0x100	Timeout-Fehler. Die zulässige Ausführungszeit wurde überschritten.
0x200	Parameter Fehler (z.B. bei einer unzulässigen Registernummer).
0x300	Der gelesene Wert unterscheidet sich von dem geschriebenen Wert (Schreibzugriff auf diesen Register möglicherweise nicht erlaubt oder fehlgeschlagen)

Beispiele für einen Aufruf in FUP:

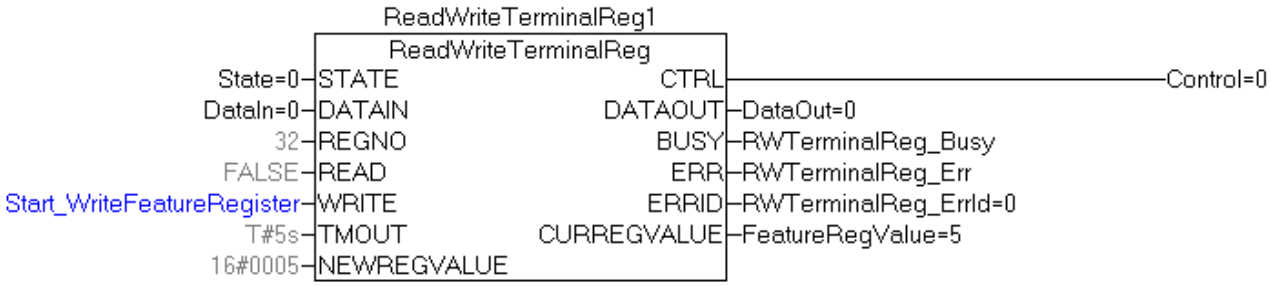
```
VAR
ReadWriteTerminalReg1     : ReadWriteTerminalReg;
State AT%IB0              : BYTE;
Control AT%QB0            : BYTE;
DataIn AT%IW2             : WORD;
DataOut AT%QW2            : WORD;
Start_ReadTerminalType    : BOOL;
Start_WriteFeatureRegister: BOOL;
RWTerminalReg_Busy        : BOOL;
RWTerminalReg_Err         : BOOL;
RWTerminalReg_ErrId       : UDINT;
TerminalType              : WORD;
FeatureRegValue           : WORD;
END_VAR
```

Beispiel 1



Im Beispiel 1 wird aus dem Register 8 einer analogen Ausgangsklemme die Klemmenbezeichnung ausgelesen. Die Variablen *State*, *Control*, *DataIn* und *DataOut* werden mit den entsprechenden IO-Variablen der Klemme im TwinCAT System Manager verknüpft. Die Klemmenbezeichnung lautet: KL**4002**.

Beispiel 2

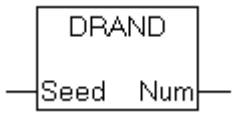


Im Beispiel 2 wird in dem Feature-Register (Register 32) einer analogen Ausgangsklemme KL4002 die Anwenderskalierung aktiviert. Der neue Wert des Feature-Registers wird dann von dem Funktionsbaustein gelesen und kann über die Ausgangsvariable **CURREGVALUE** überprüft werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

3.5 DRAND



Funktionsbausteine werden gemäß IEC61131-3 instanziiert und dann innerhalb des SPS-Programms über diesen Instanznamen aufgerufen beziehungsweise angesprochen. Der Funktionsbaustein DRAND erlaubt die Erzeugung einer (Pseudo-)Zufallszahl des Typs REAL.

VAR_INPUT

```
VAR_INPUT
    Seed    : INT;
END_VAR
```

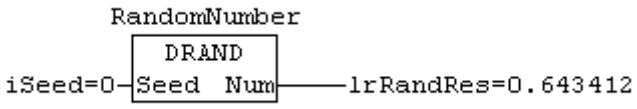
Seed : Initialwert zur Festlegung der Zufallszahlenreihe.

VAR_OUTPUT

```
VAR_OUTPUT
    Num     : REAL;
END_VAR
```

Num : Dieser Ausgang gibt eine Pseudo-Zufallszahl im Bereich 0.0 .. 1.0 mit einfacher Genauigkeit zurück. Der Generator erzeugt dabei eine Zahlenfolge mit 1075 stochastischen Werten je Periode.

Beispiel für den Aufruf des Bausteins in FBD:

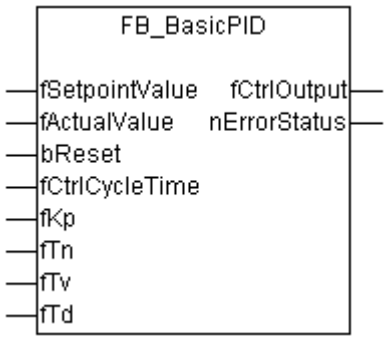


Im Beispiel wird der REAL-Wert 0.643412 erzeugt und zurückgegeben. Mit dem Eingangsparameter "Seed" kann der Initialwert der Reihe beeinflusst werden. Um beispielsweise eine deterministisch reproduzierbare Zufallszahlenfolge in verschiedenen Sitzungen zu erreichen, muss ein identischer "Seed"-Wert verwendet werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

3.6 FB_BasicPID

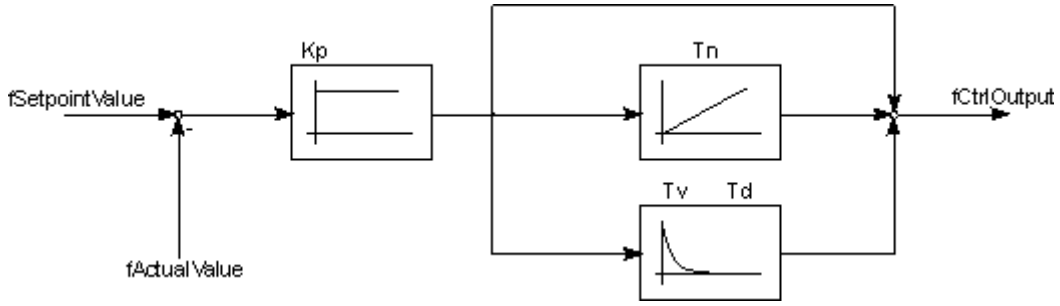


Der Funktionsbaustein stellt ein einfaches diskretisiertes PID-Glied dar.

Übertragungsfunktion:

$$G(s) = K_p(1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s})$$

Wirkungsplan:



VAR_INPUT

```
VAR_INPUT
    fSetpointValue  : REAL;
    fActualValue    : REAL;
    bReset          : BOOL;
    fCtrlCycleTime  : REAL;
    fKp             : REAL;
    fTn             : REAL;
    fTv             : REAL;
    fTd             : REAL;
END_VAR
```

fSetpointValue: Sollwert der Regelgröße.

fActualValue: Istwert der Regelgröße.

bReset: Ein TRUE an diesem Eingang setzt die internen Zustandsgrößen und den Ausgang des Reglers zurück.

fCtrlCycleTime: Zykluszeit, mit der der Funktionsbaustein aufgerufen wird und der Regelkreis bearbeitet wird [s].

fKp : Reglerverstärkung / Reglerbeiwert

fTn : Nachstellzeit [s]

fTv: Vorhaltzeit [s]

fTd : Dämpfungszeit [s]

VAR_OUTPUT

```
VAR_OUTPUT
    fCtrlOutput     : REAL;
    nErrorStatus    : UINT
END_VAR
```

fCtrlOutput : Ausgang des PID-Gliedes.

nErrorStatus : Liefert die Fehlernummer, wenn ein Fehler vorliegt (nErrorStatus <> 0).

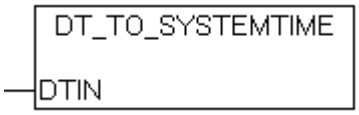
0 = nERR_NOERROR : Kein Fehler.
1 = nERR_INVALIDPARAM : Ungültige Parameter
2 = nERR_INVALIDCYCLETIME : Ungültige Zykluszeit.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.8.0 Build > 747 TwinCAT v2.9.0 Build > 947	PC (i386)	TcUtilities.Lib (Standard.Lib; TcBase.Lib; TcSystem.Lib werden automatisch eingebunden)
TwinCAT v2.7.0 Build > 522 TwinCAT v2.8.0 Build > 747 TwinCAT v2.9.0 Build > 947	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4 Funktionen

4.1 DT_TO_SYSTEMTIME



Mit der Funktion "DT_TO_SYSTEMTIME" kann eine im DATE_AND_TIME - Format (DT) SPS-Variable in eine Windows Systemzeit-Struktur konvertiert werden. Die Systemzeit hat eine Auflösung von 1ms, und das DATE_AND_TIME eine Auflösung von 1s. Die Variable "wMilliseconds" in der Systemzeit-Struktur liefert daher immer den Wert Null zurück.

FUNCTION DT_TO_SYSTEMTIME : TIMESTRUCT

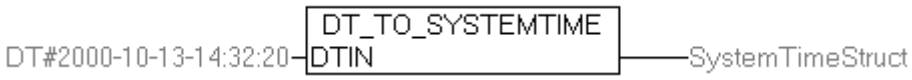
TIMESTRUCT [▶ 33]

```
VAR_INPUT
    DTIN        : DT;
END_VAR
```

DTIN:: Das zu konvertierende Datum und Uhrzeit in DATE_AND_TIME - Format.

Beispiel für einen Aufruf in FUP:

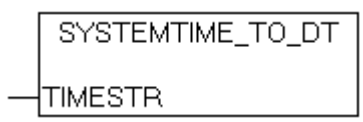
```
PROGRAM SystemTimeTest
VAR
    SystemTimeStruct  :TIMESTRUCT;
END_VAR
```



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4.2 SYSTEMTIME_TO_DT



Mit der Funktion "SYSTEMTIME_TO_DT" kann die Windows Systemzeit-Struktur in das in der SPS gängige DATE_AND_TIME - Format (DT) konvertiert werden. Die Systemzeit hat eine Auflösung von 1ms, und das DATE_AND_TIME eine Auflösung von 1s. Die Millisekunden aus der Systemzeit werden bei der Konvertierung berücksichtigt und auf den DATE_AND_TIME - Rückgabewert entsprechend aufgerundet.

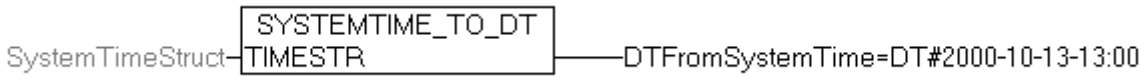
FUNCTION SYSTEMTIME_TO_DT : DT

```
VAR_INPUT
    TIMESTR         :TIMESTRUCT;
END_VAR
```

TIMESTR: Struktur mit der zu konvertierenden Windows Systemzeit..

Beispiel für einen Aufruf in FUP:

```
PROGRAM SystemTimeTest
VAR
    SystemTimeStruct  :TIMESTRUCT;
    DTFromSystemTime  :DT;
END_VAR
```



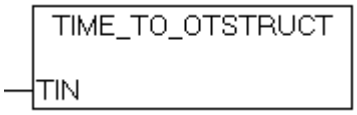
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

Sehen Sie dazu auch

TYPE TIMESTRUCT [▶ 33]

4.3 TIME_TO_OTSTRUCT



Mit der Funktion "TIME_TO_OTSTRUCT" kann eine TIME-Konstante oder Variable in eine Struktur mit den aufgelösten Millisekunden, Sekunden, Minuten, Stunden, Tagen und Wochen konvertiert werden.

FUNCTION TIME_TO_OTSTRUCT : OTSTRUCT

OTSTRUCT [▶ 33]

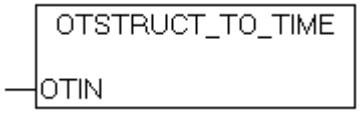
```
VAR_INPUT
    TIN             :TIME;
END_VAR
```

TIN: Die zu konvertierende TIME-Variable.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4.4 OTSTRUCT_TO_TIME



Mit der Funktion "OTSTRUCT_TO_TIME" kann eine Struktur mit den aufgelösten Millisekunden, Sekunden, Minuten, Stunden, Tagen und Wochen in eine TIME-Variable konvertiert werden.

FUNCTION OTSTRUCT_TO_TIME : TIME

```
VAR_INPUT
    OTIN        : OTSTRUCT;
END_VAR
```

OTIN: Die zu konvertierende Struktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

Sehen Sie dazu auch

TYPE OTSTRUCT [▶ 33]

4.5 SETBIT32

Die Funktion setzt das über eine Bitnummer angegebene Bit in dem ihr übergebenen 32-bit Wert und gibt den resultierenden Wert als Ergebnis zurück.

FUNCTION SETBIT32 : DWORD

VAR_INPUT

```
VAR_INPUT
    inVal32         :DWORD;
    bitNo           :SINT;
END_VAR
```

inVal32: Der zu verändernde 32-bit Wert.

bitNo: Die Nummer des zu setzenden Bits (0-31). Diese Zahl wird vor der Ausführung intern modulo 32 verrechnet.

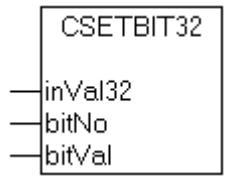
Beispiel für den Aufruf der Funktion in FBD:

Hierbei wird das Bit 31 in dem Eingangswert ‚0' gesetzt. Es ergibt sich der Wert (hex) ‚80000000'.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4.6 CSETBIT32



Die Funktion setzt/rücksetzt das, über eine Bitnummer angegebene Bit, in dem ihr übergebenen 32-bit Wert und gibt den resultierenden Wert als Ergebnis zurück.

FUNCTION CSETBIT32 : DWORD

VAR_INPUT

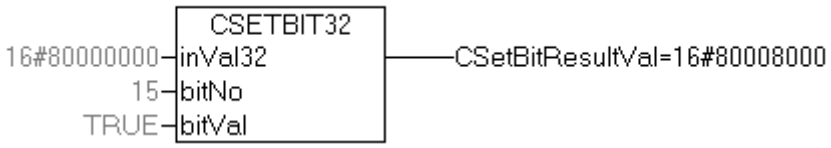
```
VAR_INPUT
    inVal32         :DWORD;
    bitNo           :SINT;
    bitVal          :BOOL;
END_VAR
```

inVal32: Ein 32-bit Wert;

bitNo: Die Nummer des Bits, der gesetzt bzw. zurückgesetzt werden soll (0-31). Diese Zahl wird vor der Ausführung intern modulo 32 verrechnet;

bitVal: Der Wert auf den das Bit gesetzt bzw. zurückgesetzt werden soll (TRUE = 1, FALSE = 0);

Beispiel für den Aufruf der Funktion in FBD:



Hierbei wird das Bit 15 in dem Eingangswert ‚16#80000000' auf 1 gesetzt. Das Ergebnis (16#80008000) wird der Variablen *CSetBitResultVal* zugewiesen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4.7 GETBIT32

Die Funktion gibt den Status des, über eine Bitnummer angegebenen Bits, in dem ihr übergebenen 32-bit Wert als boolsches Ergebnis zurück. Der Eingangswert wird nicht verändert.

FUNCTION GETBIT32 : BOOL

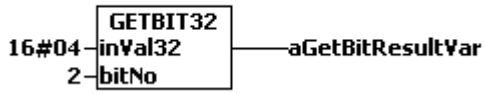
VAR_INPUT

```
VAR_INPUT
    inVal32         :DWORD;
    bitNo           :SINT;
END_VAR
```

inVal32: Der 32-bit Wert;

bitNo: Die Nummer des Bits, der gelesen werden soll (0-31). Diese Zahl wird vor der Ausführung intern modulo 32 verrechnet;

Beispiel für den Aufruf der Funktion in FBD:

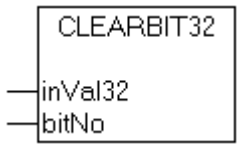


Hierbei wird das Bit 2 in dem Eingangswert ‚16#04' abgefragt und der boolschen Variablen *aGetBitResultVar* zugewiesen. Die Prüfung ergibt in diesem Beispiel TRUE.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4.8 CLEARBIT32



Die Funktion setzt das, über eine Bitnummer angegebene, Bit in dem ihr übergebenen 32-bit Wert auf Null und gibt den resultierenden Wert als Ergebnis zurück.

FUNCTION CLEARBIT32 : DWORD

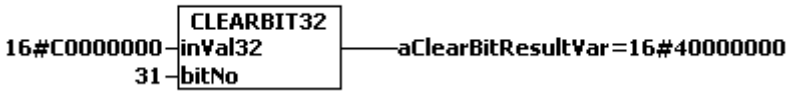
VAR_INPUT

```
VAR_INPUT
    inVal32         :DWORD;
    bitNo           :SINT;
END_VAR
```

inVal32: Der zu verändernde 32-bit Wert;

bitNo: Die Nummer des zu setzenden Bits (0-31). Diese Zahl wird vor der Ausführung intern modulo 32 verrechnet;

Beispiel für den Aufruf der Funktion in FBD:

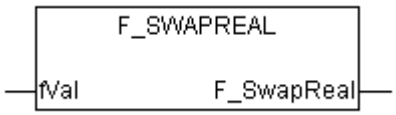


Hierbei wird das Bit 31 in dem Eingangswert ‚C0000000' zurückgesetzt. Es ergibt sich der Wert (hex) ‚40000000'.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4.9 F_SwapReal



Die Speicherdarstellung einer REAL-Zahl auf einem Buscontroller (165) unterscheidet sich von der Speicherdarstellung einer REAL-Zahl auf einem Intel-System (PC). Um eine REAL-Zahl eines Buscontrollers auf einem PC richtig darstellen zu können müssen die Hi- und Lo-Words der REAL-Zahl vertauscht werden. Die Programmierumgebung macht dies bereits im Online- oder Simulations-Mode. Um die REAL-Daten eines Buscontrollers über das Netzwerk (ADS-Protokoll, ADSDLL, AdsOcx usw.) anzufordern und auf einem Intel-PC richtig darzustellen, müssen die REAL-Daten in das richtige Format konvertiert werden. Dieses kann auf der Buscontroller- oder PC-Seite gemacht werden. Mit der Funktion F_SwapReal können die REAL-Variablen (die z.B. von einer VB-Applikation eingelesen werden oder mit TwinCAT Scope View aufgezeichnet werden sollen) auf der PC-Seite in das passende Format konvertiert werden. Der übergebene *fVal*-Parameter wird dabei von der Konvertierung nicht beeinflusst. Die Funktion liefert eine neue REAL-Zahl mit veränderten Speicherdarstellung.

FUNCTION F_SwapReal : REAL

VAR_INPUT

```
VAR_INPUT
    fVal    :REAL;
END_VAR
```

fVal: Der zu konvertierende REAL Wert.

Beispiel in ST:

```
PROGRAM MAIN
VAR
    Real_165        : REAL;
    Real_i386 AT%MB0: REAL;
END_VAR
```

```
(* ADSREAD(.... ADR(Real_165), SIZEOF(Real_165)... );
...
...
...
*)
Real_i386 := F_SwapReal( Real_165 );
Real_i368 := Real_i368 + 0.001;
```

Im Beispiel wird eine 4-Byte REAL-Variable des Buscontrollers in den 4-Byte REAL Intel-Format konvertiert. Die Variable Real_i386 kann dann z.B. auf einer VB-Form richtig dargestellt werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.7.0 Build > 518 TwinCAT v2.8.0 Build > 735	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4.10 IsFinite

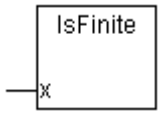


Abb. 1: IsFinite

Die Funktion IsFinite() liefert TRUE zurück, wenn deren Argument einen endlichen Wert besitzt (INF < x < +INF). D.h. die Funktion liefert FALSE zurück, wenn das Argument unendlich oder NaN ist (NaN = Not a number). IsFinite() überprüft, ob die Formatierung einer LREAL oder REAL-Variablen der IEEE entspricht.

INF-Zahlen können auf einem Laufzeitsystem dann vorkommen, wenn das Ergebnis einer mathematischen Operation den darstellbaren Bereich überschreitet oder unterschreitet. Z.B.:

```
PROGRAM MAIN
VAR
    fSingle : REAL := 12.34;
END_VAR
(*Cyclic called program code*)
fSingle := fSingle*2;
```

NaN-Zahlen können im Laufzeitsystem dann vorkommen, wenn deren eigentliche Formatierung (Speicherinhalte) durch unerlaubten Zugriff (z.B. durch Benutzung der MEMCPY, MEMSET Funktionen) überschrieben wurden. Z.B.:

```
PROGRAM MAIN
VAR
    fSingle : REAL := 12.34;
END_VAR
(*Cyclic called program code*)
MEMSET( ADR( fSingle ), 16#FF, SIZEOF( fSingle ) ); (* Invalid initialization of REAL variable *)
```

Beim Aufruf einer Konvertierungsfunktion mit einer NaN oder INF-Zahl als Parameter wird auf einem PC-System (i368) eine FPU-Exception ausgelöst. Diese Exception führt anschließend zum Stopp der SPS. Mit der Funktion IsFinite() kann der Wert der Variablen überprüft, die FPU-Exception vermieden und die Programmausführung fortgesetzt werden.

FUNCTION IsFinite : BOOL

```
VAR_INPUT
    x :T_Arg;
END_VAR
```

x: Eine Hilfsstruktur mit Informationen zu der zu überprüfenden REAL oder LREAL-Variablen. Die Strukturparameter müssen beim Aufruf von IsFinite() mit Hilfsfunktionen F_REAL [▶ 32] oder F_LREAL [▶ 32] erzeugt und als Parameter übergeben werden.

Beispiel für einen Aufruf in ST:

Im folgenden Beispiel wird die Formatierung einer REAL- und einer LREAL-Variablen überprüft und eine FPU-Exception vermieden.

```
PROGRAM MAIN
VAR
    fSingle         : REAL := 12.34;
    fDouble         : LREAL := 56.78;
    singleAsString  : STRING;
    doubleAsString  : STRING;
END_VAR
fSingle := fSingle*2;
IF IsFinite( F_REAL( fSingle ) ) THEN
    singleAsString := REAL_TO_STRING( fSingle );
ELSE
  (* report error !*)
 fSingle := 12.34;
END_IF

fDouble := fDouble*2;
IF IsFinite( F_LREAL( fDouble ) ) THEN
    doubleAsString := LREAL_TO_STRING( fDouble );
ELSE
  (* report error !*)
 fDouble := 56.78;
END_IF
```

Im folgenden Fall kann eine FPU-Exception durch Überprüfung mit IsFinite() nicht vermieden werden:

```
PROGRAM MAIN
VAR
    bigFloat : LREAL := 3.0E100;
    smallDigit: INT;
END_VAR
IF IsFinite( F_LREAL( bigFloat ) ) THEN
    smallDigit := LREAL_TO_INT( bigFloat );
END_IF
```

Die bigFloat-Variable besitzt zwar richtige Formatierung, der Variablenwert ist aber zu groß um diesen in einen INT-Typ konvertieren zu können. Auf einem PC-System (i368) wird eine Exception ausgelöst und das Laufzeitsystem gestoppt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 Build > 522 TwinCAT v2.8.0 Build > 747 TwinCAT v2.9.0 Build > 947	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

Sehen Sie dazu auch

TYPE T_Arg [▶ 33]

4.11 F_REAL

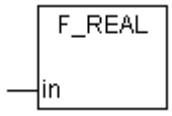


Abb. 2: F_REAL

Eine Hilfsfunktion die in einer Struktur Informationen zu einer REAL-Variablen zurückliefert.

FUNCTION F_REAL : T_Arg

T_Arg [▶ 33]

VAR_IN_OUT

```
VAR_IN_OUT
    in      :REAL;
END_VAR
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 Build > 522 TwinCAT v2.8.0 Build > 747 TwinCAT v2.9.0 Build > 947	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

4.12 F_LREAL

Nicht vorhanden auf dem BCxxxx (165)

5 Datenstrukturen

5.1 TYPE TIMESTRUCT

```
TYPE TIMESTRUCT
STRUCT
   wYear         : WORD;
   wMonth        : WORD;
   wDayOfWeek    : WORD;
   wDay          : WORD;
   wHour         : WORD;
   wMinute       : WORD;
   wSecond       : WORD;
   wMilliseconds: WORD;
END_STRUCT
END_TYPE
```

wYear : Das Jahr: 1970 ~ 2106;

wMonth : Der Monat: 1 ~ 12 (Januar = 1, Februar = 2 usw.);

wDayOfWeek : Der Wochentag: 0 ~ 6 (Sonntag = 0, Montag = 1 usw.);

wDay : Tag des Monats: 1 ~ 31;

wHour : Stunde: 0 ~ 23;

wMinute : Minute: 0 ~ 59;

wSecond : Sekunde: 0 ~ 59;

wMilliseconds : Millisekunde: 0 ~ 999;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

5.2 TYPE OTSTRUCT

```
TYPE OTSTRUCT
STRUCT
   wWeek         : WORD;
   wDay          : WORD;
   wHour         : WORD;
   wMinute       : WORD;
   wSecond       : WORD;
   wMilliseconds: WORD;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 und höher	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

5.3 TYPE T_Arg

```
TYPE T_Arg :
STRUCT
    eType    : E_ArgType      := ARGTYPE_UNKNOWN
```

```
    cbLen   : UDINT         := 0
    pData   : UDINT         := 0
END_STRUCT
END_TYPE
```

eType : Datentypkennung;

cbLen : Anzahl der Bytes, die im Speicher belegt werden;

pData : Adresspointer;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 Build > 522 TwinCAT v2.8.0 Build > 747 TwinCAT v2.9.0 Build > 947	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

Sehen Sie dazu auch

TYPE E_ArgType [▶ 34]

5.4 TYPE E_ArgType

```
TYPE E_ArgType :(

    ARGTYPE_UNKNOWN         := 0,
    ARGTYPE_BYTE,
    ARGTYPE_WORD,
    ARGTYPE_DWORD,
    ARGTYPE_REAL,
    ARGTYPE_LREAL,
    ARGTYPE_SINT,
    ARGTYPE_INT,
    ARGTYPE_DINT,
    ARGTYPE_USINT,
    ARGTYPE_UINT,
    ARGTYPE_UDINT,
    ARGTYPE_STRING,
    ARGTYPE_BOOL,
    ARGTYPE_BIGTYPE
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v2.7.0 Build > 522 TwinCAT v2.8.0 Build > 747 TwinCAT v2.9.0 Build > 947	BCxxxx (165)	Standard.Lb6, PlcSystemBC.Lb6, TcPlcUtilitiesBC.Lb6

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