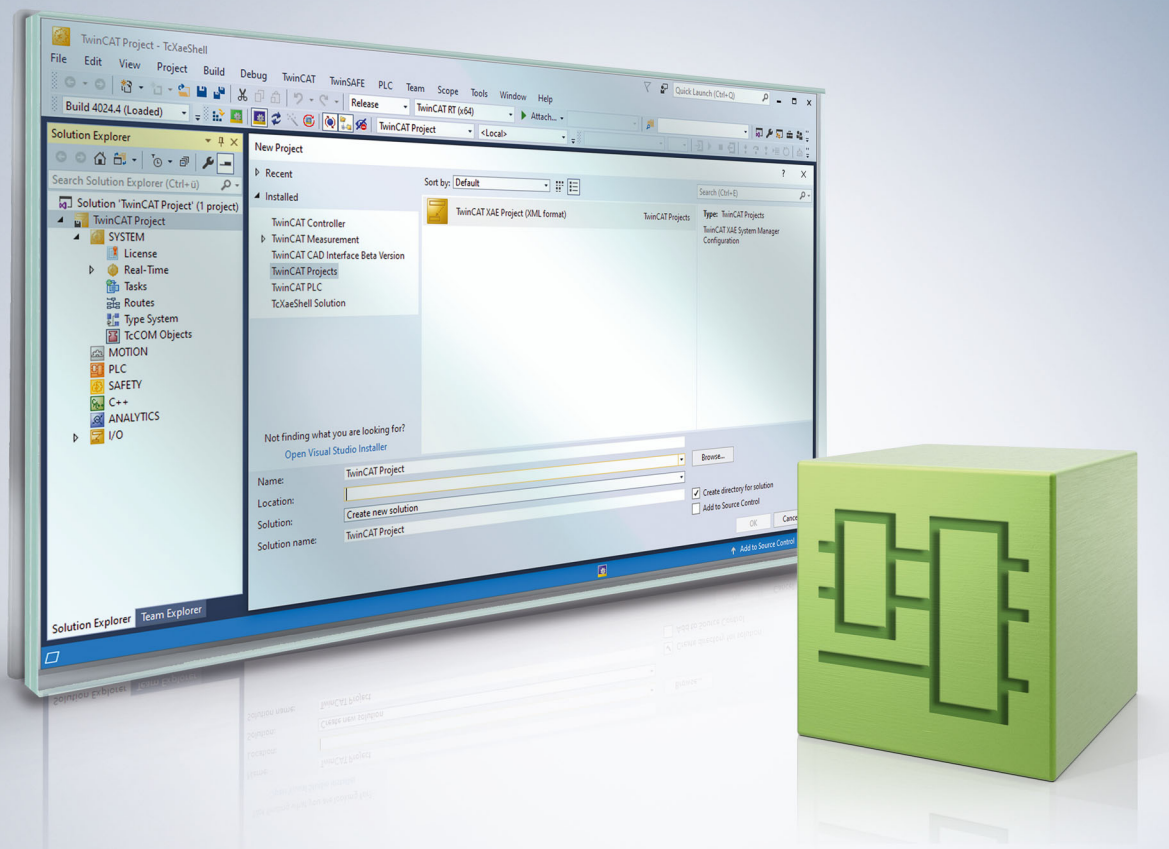
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | PLC Bibliothek: Tc3_BA2_Common



23.10.2023 | Version: 1.0.1

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

Copyright

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 Einleitung

Die TwinCAT3 Building Automation Bibliothek (Tc3_BA2_Common) beinhaltet Funktionsbausteine, die für das Arbeiten mit den TwinCAT Functions TF8020 als auch TF8040 erforderlich sind.

3 Allgemeine Informationen

Weitere erforderliche Bibliotheken

Für PC-Systeme und Embedded-PCs (CXxxxx):

- Tc2_IoFunctions
- Tc2_Standard
- Tc2_System
- Tc2_Utilities
- Tc2_DataExchange

4 Programmierung

4.1 POUs

4.1.1 Funktionen

4.1.1.1 Vergleichen

4.1.1.1.1 F_BA_CompareVersion

Die Funktion F_BA_CompareVersion vom Rückgabetyp BOOL vergleicht zwei Versionsnummern *stVersion1* und *stVersion2*, jeweils vom Typ ST_BA_Version [▶ 80]. Dabei gibt die Eingabevariable *nLimit* an, wie viele Stellen verglichen werden sollen, beginnend mit 1 = "Major". Die Enumeration *eCompare* gibt die Vergleichsoperation an. Der Funktionsrückgabewert wechselt auf TRUE, wenn der Vergleich erfüllt ist.

Ist beispielsweise stVersion1 kleiner als stVersion2, dann liefert der Vergleich *eCompare* = *E_BA_CompareMode.eLower* ein TRUE als Funktionsrückgabe.

Syntax

```
FUNCTION F_BA_CompareVersion : BOOL
VAR_INPUT
  stVersion1   : ST_BA_Version;
  stVersion2   : ST_BA_Version;

  eCompare     : E_BA_CompareMode  := E_BA_CompareMode.eEqual;
  nLimit       : UINT(1 .. 4)      := 4;
END_VAR
```

Eingänge

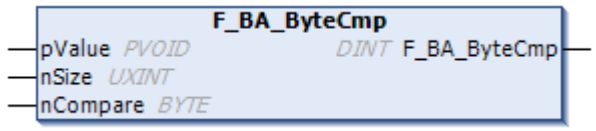
Name	Typ	Beschreibung
stVersion1, stVersion2	ST_BA_Version [▶ 80]	Zu vergleichende Versionsnummern.
eCompare	E_BA_CompareMode [▶ 51]	Auszuführende Vergleichsoperation.
nLimit	UINT(1 ... 4)	Anzahl der Stellen, die verglichen werden sollen, beginnend mit „Major“.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.2 Speicher

4.1.1.2.1 F_BA_ByteCmp



Die Funktion F_BA_ByteCmp vom Rückgabetyp DINT vergleicht den Inhalt eines Speicherbereiches in Byte-Schritten mit einem Vergleichs-Byte *nCompare*.

Sobald innerhalb des Speicherbereichs ein Byte gefunden wird, welches vom Wert her kleiner ist als das Vergleichbyte, bricht die Funktion ihren Vergleich ab und nimmt den Rückgabewert "-1" an. Wird ein Byte gefunden, welches größer ist als das Vergleichsbyte, so bricht die Funktion ebenfalls ab und nimmt den Rückgabewert "1" an. Wird hingegen kein Unterschied gefunden, d.h. alle Bytes des zu untersuchenden Speicherbereichs sind identisch mit dem Vergleichsbyte, so nimmt die Funktion bei Beendigung des Vergleiches den Wert "0" an.

Die Eingangsvariable *pValue* markiert den Beginn des Speicherbereiches, die Variable *nSize* die Länge.

Bei einer fehlerhaften Eingabe, d.h. *pValue* = 0 oder *nSize* = 0 wird die Funktion ebenfalls sofort abgebrochen und nimmt als Rückgabewert die "-1" an.

Syntax

```
FUNCTION F_BA_ByteCmp : DINT
VAR_INPUT
  pValue    : PVOID;
  nSize     : UXINT;
  nCompare  : BYTE;
END_VAR
```

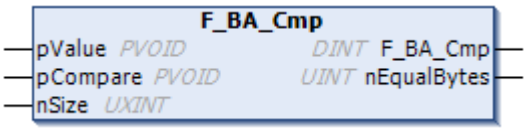
Eingänge

Name	Typ	Beschreibung
pValue	PVOID	Zeiger auf den Beginn des Speicherbereiches, der untersucht werden soll.
nSize	UXINT	Länge des Speicherbereichs.
nCompare	BYTE	Vergleichsbyte.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.2.2 F_BA_Cmp



Die Funktion F_BA_Cmp vom Rückgabetyp DINT vergleicht zwei Speicherbereiche gleicher Größe. Dabei definiert *pValue* den Beginn des Betrachtungsbereichs und *pCompare* den Beginn des zu vergleichenden Bereiches. Es wird Byte für Byte verglichen. Sobald ein Byte im Betrachtungsbereich gefunden wird, welches kleiner ist als das des vergleichenden Bereiches, so bricht die Funktion den Vergleich ab und nimmt den Rückgabewert "-1" an. Wird ein Byte im Betrachtungsbereich gefunden, welches größer ist als das des vergleichenden Bereiches, so bricht die Funktion ebenfalls ab und nimmt den Rückgabewert "1" an. Wird hingegen kein Unterschied gefunden, d.h. alle Bytes des Betrachtungsbereichs sind identisch mit denen des vergleichenden Bereiches, so nimmt die Funktion bei Beendigung des Vergleiches den Wert "0" an.

Der Ausgang *nEqualBytes* zeigt an, wie viele Bytes gleich waren, bevor die Vergleichsoperation beendet oder abgebrochen wurde.

Die Eingangsvariable *nSize* definiert die Größe der beiden Speicherbereiche.

Bei einer fehlerhaften Eingabe, d.h. *pValue* = 0 oder *nSize* = 0 wird die Funktion ebenfalls sofort abgebrochen und nimmt als Rückgabewert die "-1" an.

Syntax

```
FUNCTION F_BA_Cmp : DINT
VAR_INPUT
  pValue          : PVOID;
  pCompare        : PVOID;
  nSize           : UXINT;
END_VAR
VAR_OUTPUT
  nEqualBytes     : UINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pValue	PVOID	Zeiger auf den Beginn des Speicherbereiches, der untersucht werden soll.
pCompare	PVOID	Zeiger auf den Beginn des Vergleich-Speicherbereiches.
nSize	UXINT	Länge des Speicherbereiches.

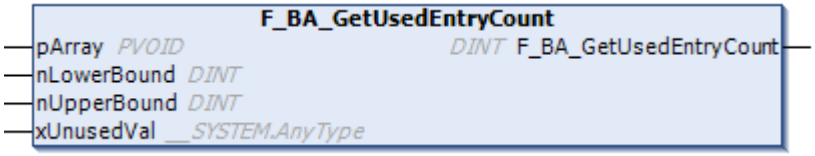
Ausgänge

Name	Typ	Beschreibung
nEqualBytes	UINT	Anzeige wie viele Bytes gleich waren, bevor die Vergleichsoperation beendet oder abgebrochen wurde.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.2.3 F_BA_GetUsedEntryCount



Die Funktion F_BA_GetUsedEntryCount vom Rückgabetyp DINT durchsucht den Speicherbereich eines ARRAYs nach dem ersten Eintrag, der mit xUnusedVal als "ungenutzt" gekennzeichnet ist.

Dabei wird der Speicherbereich des Arrays beginnend ab der Adresse pArray schrittweise untersucht: die Schrittweite ist durch die Größe von xUnusedVal vorgegeben, die Anzahl der Vergleichsschritte durch die Differenz von nLowerBound und nUpperBound. Dabei wird davon ausgegangen, dass die Einträge nLowerBound und nUpperBound richtig sind.

Der Rückgabewert der Funktion zeigt an, wie viele Elemente nicht dem Wert xUnusedVal entsprechen, bis dieser gefunden wird. Wird der Wert gefunden, wird die Suche abgebrochen.

Bei einer fehlerhaften Eingabe, d.h. pArray = 0 oder wenn die Größe von xUnusedVal = 0 ist, wird die Funktion sofort abgebrochen und nimmt als Rückgabewert "0" an.

Syntax

```
FUNCTION F_BA_GetUsedEntryCount : DINT
VAR_INPUT
  pArray         : PVOID;
  nLowerBound    : DINT;
  nUpperBound    : DINT;
  xUnusedVal     : ANY;
END_VAR
```

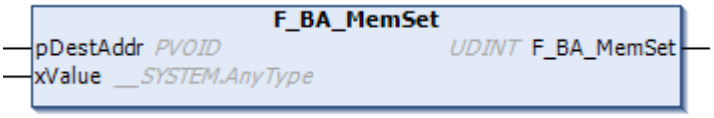
Eingänge

Name	Typ	Beschreibung
pArray	PVOID	Zeiger auf den Beginn des Speicherbereiches, in dem das zu untersuchende Array liegt.
nLowerBound	DINT	Untere Bereichsgrenze des Arrays.
nUpperBound	DINT	Obere Bereichsgrenze des Arrays.
xUnusedVal	ANY	Besitzt ein Element des Arrays den hier angegebenen Wert, so gilt es als "ungenutzt".

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.2.4 F_BA_MemSet



Die Funktion F_BA_MemSet vom Rückgabetyp UDINT beschreibt den Adressbereich beginnend mit *pDestAddr* mit dem Wert einer Variablen *xValue* beliebigen Typs.

Dem Funktionsrückgabewert selbst wird das Ergebnis der internen MEMCPY-Funktion zugewiesen, d.h. "0" für fehlerhaftes Kopieren und Werte größer "0" für die Anzahl kopierter Bytes.

Syntax

```
FUNCTION F_BA_MemSet : UDINT
VAR_INPUT
  pDestAddr    : PVOID;
  xValue       : ANY;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pDestAddr	PVOID	Zieladresse des Schreibvorgangs.
xValue	ANY	Wert, der geschrieben werden soll.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.2.5 F_BA_MemSetEx

Die Funktion F_BA_MetSetEx vom Rückgabetyp UDINT füllt den Adressbereich beginnend mit *pDestAddr* und der Länge *nDestSize* mit dem Wert einer Variablen *xValue* beliebigen Typs.

Voraussetzung dafür ist, dass der definierte Füllbereich um ein ganzzahliges Vielfaches größer als die Füllvariable *xValue* selbst ist: *nDestSize* = n * *xValue.diSize*.

Dem Funktionsrückgabewert selbst wird das Ergebnis der internen MEMCPY-Funktion zugewiesen, d.h. "0" für fehlerhaftes Kopieren und Werte größer "0" für die Anzahl kopierter Bytes.

Der Wert "0" wird auch zurückgegeben, wenn der definierte Füllbereich nicht um ein ganzzahliges Vielfaches größer als die Füllvariable *xValue* ist, und das Füllen nicht stattgefunden hat.

Syntax

```
FUNCTION F_BA_MemSetEx : UDINT
VAR_INPUT
  pDestAddr    : PVOID;
  nDestSize    : UDINT;
  xValue       : ANY;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pDestAddr	PVOID	Zieladresse des Füllbereiches.
nDestSize	UDINT	Größe des Füllbereiches.
xValue	ANY	Füllwert

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.2.6 F_BA_OffsetPtr

Die Funktion F_BA_OffestPtr vom Rückgabetyp PVOID addiert zu der eingegebenen Adresse *pAddr* ein Offset *nOffset* und gibt das Ergebnis als Rückgabewert der Funktion aus. Dieser Wert stellt dann seinerseits eine Adresse dar.

Die Funktion unterscheidet intern, ob das verwendete Laufzeitsystem vom Typ x64 oder x86 ist: beim x64-System wird der Offset zu einer Variable des Typs ULINT gewandelt und addiert, bei einem x86-System zu einer Variable des Typs UDINT.

Syntax

```
FUNCTION F_BA_OffsetPtr : PVOID
VAR_INPUT
  pAddr      : PVOID;
  nOffset    : DINT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pAddr	PVOID	Basisadresse
nOffset	DINT	Offset

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.2.7 F_BA_DiffPtr

Die Funktion F_BA_DiffPrt vom Rückgabetyp UDINT gibt die Differenz zweier Speicheradressen als Absolutwert an: intern wird die kleinere Adresse von der größeren abgezogen.

Syntax

```
FUNCTION F_BA_DiffPtr : UDINT
VAR_INPUT
  pAddr1    : PVOID;
  pAddr2    : PVOID;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
pAddr1	PVOID	Adresse 1
pAddr2	PVOID	Adresse 2

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3 Typen

4.1.1.3.1 AnyLEN

Die Funktion AnyLEN mit dem Rückgabetyp INT ermittelt die beschriebene Länge einer String-Variable. Diese wird an den Eingang STR vom Typ ANY gelegt und ist damit analysierbar.

Für String-Variablen werden je nach Deklaration feste Speicherbereiche reserviert, die jedoch meist nur zum Teil mit einem Text gefüllt werden.

Ist die angelegte Variable vom Typ STRING, so wird der Speicherbereich, in dem die Variable liegt, byteweise durchsucht, bis ein Byte mit dem Wert "0" gefunden wird, also das Ende des füllenden Textes erreicht ist. Der Rückgabewert der Funktion ist gleich der Anzahl der durchsuchten Bytes, exklusive der gefundenen "0".

Wird kein Byte mit dem Wert "0" gefunden, so füllt ein Text den gesamten Bereich des Strings aus, der Rückgabewert der Funktion entspricht dann der Speichergröße des angelegten Strings in Bytes.

Ist die angelegte Variable nicht vom Typ STRING, so wird dem Rückgabewert der Funktion die Speichergröße der Variablen in Bytes zugewiesen.

Syntax

```
FUNCTION AnyLEN : INT
VAR_INPUT
  STR    : ANY;
END_VAR
```

Eingänge

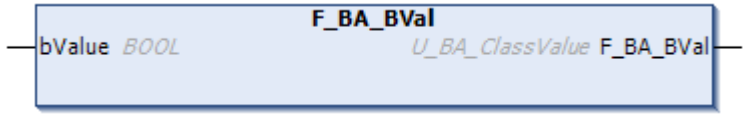
Name	Typ	Beschreibung
STR	ANY	Zu untersuchende Variable. Hier kann grundsätzlich ein beliebiger simpler Variablentyp angelegt werden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.2 ClassValue

4.1.1.3.2.1 F_BA_BVal



Der Rückgabewert der Funktion F_BA_BVal wird von der Struktur des Typs U_BA_ClassValue [▶ 82] erstellt. Der Eingangswert *bValue* wird auf die Variable *bVal* dieser Struktur geschrieben. Bedingt durch den Datentyp UNION, dessen Elemente alle ab derselben Speicheradresse beginnen, verändern sich auch alle anderen Werte dieser Struktur.

Syntax

```
FUNCTION F_BA_BVal : U_BA_ClassValue
VAR_INPUT
  bValue    : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bValue	BOOL	Dieser Wert wird *iVal* in der Ausgabestruktur zugewiesen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.2.2 F_BA_ByteVal

Der Rückgabewert der Funktion F_BA_ByteVal wird von der Struktur des Typs U_BA_ClassValue [▶ 82] erstellt. Die Eingangswerte *nByte1*, *nByte2*, *nByte3* und *nByte4* werden in dieser Struktur auf den jeweiligen Werten *nByteVal[1]*, *nByteVal[2]*, *nByteVal[3]* und *nByteVal[4]* zugewiesen. Bedingt durch den Datentyp UNION, dessen Elemente alle ab derselben Speicheradresse beginnen, verändern sich auch alle anderen Werte dieser Struktur.

Syntax

```
FUNCTION F_BA_ByteVal : U_BA_ClassValue
VAR_INPUT
  nByte1    : BYTE;
  nByte2    : BYTE;
  nByte3    : BYTE;
  nByte4    : BYTE;
END_VAR
```

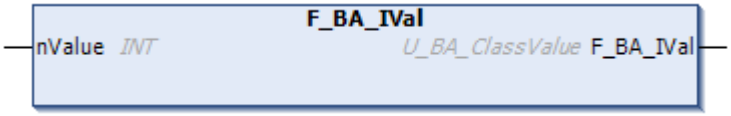
Eingänge

Name	Typ	Beschreibung
nByte1	BYTE	Dieser Wert wird _nByteVal[1]_der Ausgabestruktur zugewiesen.
nByte2	BYTE	Dieser Wert wird _nByteVal[2]_der Ausgabestruktur zugewiesen.
nByte3	BYTE	Dieser Wert wird _nByteVal[3]_der Ausgabestruktur zugewiesen.
nByte4	BYTE	Dieser Wert wird _nByteVal[4]_der Ausgabestruktur zugewiesen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.2.3 F_BA_IVal



Der Rückgabewert der Funktion F_BA_IVal wird von der Struktur des Typs U_BA_ClassValue [▶ 82] erstellt. Der Eingangswert *nValue* wird auf die Variable *iVal* dieser Struktur geschrieben. Bedingt durch den Datentyp UNION, dessen Elemente alle ab derselben Speicheradresse beginnen, verändern sich auch alle anderen Werte dieser Struktur.

Syntax

```
FUNCTION F_BA_IVal : U_BA_ClassValue
VAR_INPUT
  nValue    : INT;
END_VAR
```

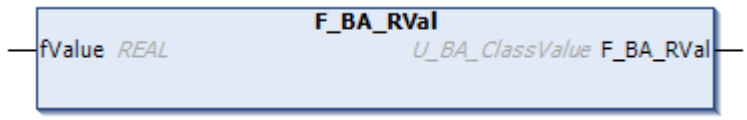
Eingänge

Name	Typ	Beschreibung
nValue	INT	Dieser Wert wird *iVal* in der Ausgabestruktur zugewiesen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.2.4 F_BA_RVal



Der Rückgabewert der Funktion F_BA_RVal wird von der Struktur des Typs U_BA_ClassValue [▶ 82] erstellt. Der Eingangswert *fValue* wird auf die Variable *rVal* dieser Struktur geschrieben. Bedingt durch den Datentyp UNION, dessen Elemente alle ab derselben Speicheradresse beginnen, verändern sich auch alle anderen Werte dieser Struktur.

Syntax

```
FUNCTION F_BA_RVal : U_BA_ClassValue
VAR_INPUT
  fValue    : REAL;
END_VAR
```

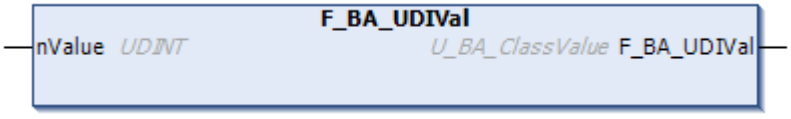
Eingänge

Name	Typ	Beschreibung
fValue	REAL	Dieser Wert wird *rVal* in der Ausgabestruktur zugewiesen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.2.5 F_BA_UDIVal



Der Rückgabewert der Funktion F_BA_UDIVal wird von der Struktur des Typs U_BA_ClassValue [▶ 82] erstellt. Der Eingangswert *nValue* wird auf die Variable *udiVal* dieser Struktur geschrieben. Bedingt durch den Datentyp UNION, dessen Elemente alle ab derselben Speicheradresse beginnen, verändern sich auch alle anderen Werte dieser Struktur.

Syntax

```
FUNCTION F_BA_UDIVal : U_BA_ClassValue
VAR_INPUT
  nValue    : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nValue	UINT	Dieser Wert wird *udiVal* in der Ausgabestruktur zugewiesen.

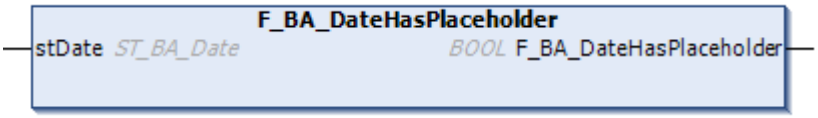
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3 Date and Time

4.1.1.3.3.1 Check

4.1.1.3.3.1.1 F_BA_DateHasPlaceholder



Die Funktion F_BA_DateHasPlaceholder vom Rückgabetyp BOOL liefert ein TRUE, wenn von der Eingabestruktur die Jahres-, Monats-, Tages- oder Wochentagskomponente den Platzhalterwert 16#FF innehat.

Syntax

```
FUNCTION F_BA_DateHasPlaceholder : BOOL
VAR_INPUT
  stDate    : ST_BA_Date;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stDate	ST_BA_Date [▶ 75]	Betrachteter Datums-Zeiteintrag.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.1.2 F_BA_DateUnspecified

Die Funktion F_BA_DataUnspecified vom Rückgabetyp BOOL liefert ein TRUE, wenn von der Eingabestruktur die Jahres-, Monats-, Tages- und Wochentagskomponenten den Platzhalterwert 16#FF innehaben.

Syntax

```
FUNCTION F_BA_DateUnspecified : BOOL
VAR_INPUT
  stDate    : ST_BA_Date;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stDate	ST_BA_Date [▶ 75]	Betrachteter Datums-Zeiteintrag.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.1.3 F_BA_IsLeapYear

Die Funktion F_BA_IsLeapYear vom Rückgabetyp BOOL liefert ein TRUE, wenn das eingegebene Jahr ein Schaltjahr ist.

Syntax

```
FUNCTION F_BA_IsLeapYear : BOOL
VAR_INPUT
  nYear  : UDINT;
END_VAR
```

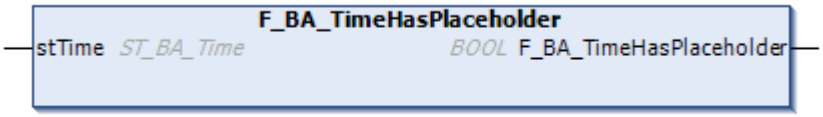
Eingänge

Name	Typ	Beschreibung
nYear	UDINT	Zu untersuchendes Jahr.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.1.4 F_BA_TimeHasPlaceholder



Die Funktion F_BA_TimeHasPlaceholder vom Rückgabetyp BOOL liefert ein TRUE, wenn von der Eingabestruktur die Stunden-, Minuten- oder Sekundenkomponente den Platzhalterwert 16#FF innehat.

Syntax

```
FUNCTION F_BA_TimeHasPlaceholder : BOOL
VAR_INPUT
  stTime    : ST_BA_Time;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stTime	ST_BA_Time [▶ 76]	Betrachteter Tages-Zeiteintrag.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.1.5 F_BA_TimeUnspecified

Die Funktion F_BA_TimeUnspecified vom Rückgabetyp BOOL liefert ein TRUE, wenn von der Eingabestruktur die Stunden-, Minuten- und Sekundenkomponenten den Platzhalterwert 16#FF innehaben.

Syntax

```
FUNCTION F_BA_TimeUnspecified : BOOL
VAR_INPUT
  stTime    : ST_BA_Time;
END_VAR
```

Eingänge

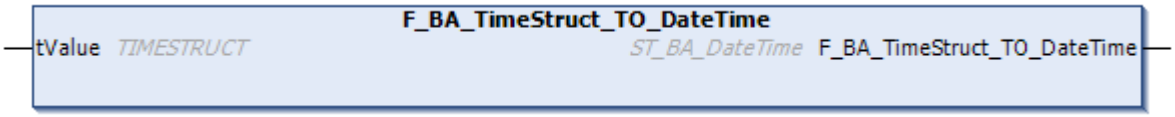
Name	Typ	Beschreibung
stTime	ST_BA_Time [▶ 76]	Betrachteter Tages-Zeiteintrag.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2 Convert

4.1.1.3.3.2.1 F_BA_TimeStruct_TO_DateTime



Die Funktion F_BA_TimeStruct_TO_DateTime vom Rückgabetyp ST_BA_DateTime [▶ 76] wandelt einen Zeitwert *tValue* vom Typ TIMESTRUCT in einen Zeitwert des Typs ST_BA_DateTime [▶ 76] um.

Syntax

```
FUNCTION F_BA_TIMESTRUCT_TO_DateTime : ST_BA_DateTime
VAR_INPUT
  tValue    : TIMESTRUCT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
tValue	TIMESTRUCT	Zu wandelnder Zeit-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.2 F_BA_TimeStruct_TO_Time

Die Funktion F_BA_TimeStruct_TO_Time vom Rückgabetyp ST_BA_Time [▶ 76] wandelt einen Zeitwert *tValue* vom Typ TIMESTRUCT in einen Zeitwert des Typs ST_BA_Time [▶ 76] um.

Syntax

```
FUNCTION F_BA_TIMESTRUCT_TO_Time : ST_BA_Time
VAR_INPUT
  tValue    : TIMESTRUCT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
tValue	TIMESTRUCT	Zu wandelnder Zeit-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.3 F_BA_To100msDate

Die Funktion F_BA_To100msDate vom Rückgabetyp UDINT wandelt ein Datum *stBACnetDate* vom Typ ST_BA_Date [▶ 75] in einen Zehntelsekunden-Wert bezogen auf den 01.01.1900 00:00:00 Uhr um.

Syntax

```
FUNCTION F_BA_To100msDate : UDINT
VAR_INPUT
  stBACnetDate    : ST_BA_Date;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stBACnetDate	ST_BA_Date [▶ 75]	Zu wandelnder Datums-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.4 F_BA_To100msTime

Die Funktion F_BA_To100msTime vom Rückgabetyp UDINT wandelt eine Tageszeit *stBACnetTime* vom Typ ST_BA_Time [▶ 76] in Zehntelsekunden beginnend ab 00:00:00 Uhr um.

Nicht spezifizierte, d.h. mit 16#FF vorgegebene Einträge des Zeit-Eingabewertes *stBACnetTime*, werden in der Berechnung mit 0 gewertet.

Syntax

```
FUNCTION F_BA_To100msTime : UDINT
VAR_INPUT
  stBACnetTime    : ST_BA_Time;
END_VAR
```

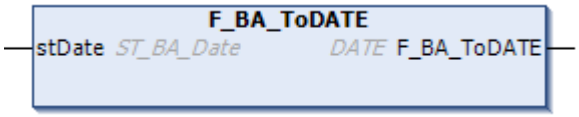
Eingänge

Name	Typ	Beschreibung
stBACnetTime	ST_BA_Time [▶ 76]	Zu wandelnder Zeit-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.5 F_BA_ToDate



Die Funktion F_BA_ToDATE vom Rückgabetyp DATE wandelt ein Datum *stDate* vom Typ ST_BA_Date [▶ 75] in ein Datum des Typs DATE um.

Syntax

```
FUNCTION F_BA_ToDATE : DATE
VAR_INPUT
  stDate    : ST_BA_Date;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stDate	ST_BA_Date [▶ 75]	Zu wandelnder Datums-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.6 F_BA_ToDT

Die Funktion F_BA_ToDT vom Rückgabetyp DATE wandelt ein Datum *stDate* vom Typ ST_BA_Date [▶ 75] in ein Datum des Typs DATE um.

Syntax

```
FUNCTION F_BA_ToDATE : DATE
VAR_INPUT
  stDate    : ST_BA_Date;
END_VAR
```

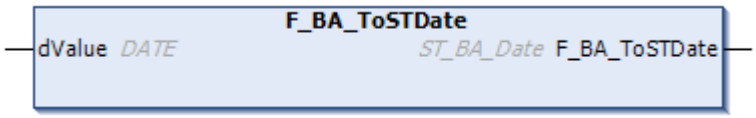
Eingänge

Name	Typ	Beschreibung
stDate	ST_BA_Date [▶ 75]	Zu wandelnder Datums-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.7 F_BA_ToSTDate



Die Funktion F_BA_ToSTDate vom Rückgabetyp ST_BA_Date [▶ 75] wandelt ein Datum *dValue* vom Typ DATE in ein Datum des Typs ST_BA_Date [▶ 75] um.

Syntax

```
FUNCTION F_BA_ToSTDate : ST_BA_Date
VAR_INPUT
  dValue    : DATE;
END_VAR
```

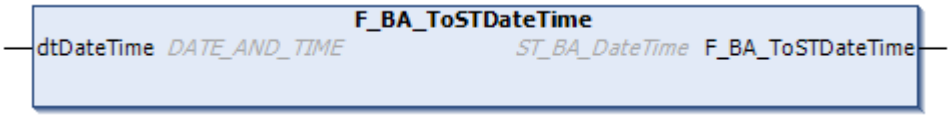
Eingänge

Name	Typ	Beschreibung
dValue	DATE	Zu wandelnder Datums-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.8 F_BA_ToSTDateTime



Die Funktion F_BA_ToSTDateTime vom Rückgabetyp ST_BA_DateTime [▶ 76] wandelt einen Zeitwert *dtDateTime* vom Typ DT (DATE_AND_TIME) in einen Zeitwert des Typs ST_BA_DateTime [▶ 76] um.

Syntax

```
FUNCTION  F_BA_ToSTDateTime  :  ST_BA_DateTime
VAR_INPUT
    dtDateTime    :  DT;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
dtDateTime	DT	Zu wandelnder Zeit-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.9 F_BA_ToSTTime

Die Funktion F_BA_ToSTTime vom Rückgabetyp ST_BA_Time [▶ 76] wandelt einen Zeitwert *tTime* vom Typ TIME in einen Zeitwert des Typs ST_BA_Time [▶ 76] um.

Syntax

```
FUNCTION F_BA_ToSTTime : ST_BA_Time
VAR_INPUT
  tTime    : TIME;
END_VAR
```

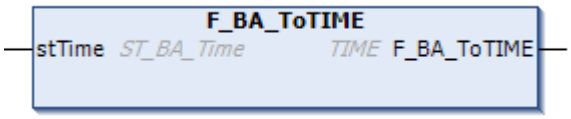
Eingänge

Name	Typ	Beschreibung
tTime	TIME	Zu wandelnder Zeit-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.2.10 F_BA_ToTime



Die Funktion F_BA_ToTime vom Rückgabetyp TIME wandelt einen Zeitwert *stTime* vom Typ ST_BA_Time in einen Zeitwert des Typs TIME um.

Nicht spezifizierte, d.h. mit 16#FF vorgegebene Einträge des Zeit-Eingabewertes *stTime*, werden in der Berechnung mit 0 gewertet.

Syntax

```
FUNCTION F_BA_ToTIME : TIME
VAR_INPUT
  stTime     : ST_BA_Time;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stTime	ST_BA_Time [▶ 76]	Zu wandelnder Zeit-Eingabewert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.3 F_BA_CountLeapYears

Die Funktion F_BA_CountLeapYears vom Rückgabetyp UDINT liefert errechnet die Anzahl der Schaltjahre vom eingetragenen Startjahr *nStartYear* bis zum Ende des Betrachtungszeitraumes *nYear* (inklusive).

Syntax

```
FUNCTION F_BA_CountLeapYears : UDINT
VAR_INPUT
  nYear         : UDINT;
  nStartYear    : UDINT    := 0;
END_VAR
```

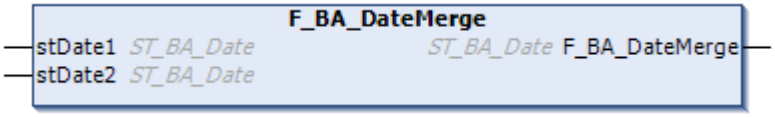
Eingänge

Name	Typ	Beschreibung
nYear	UDINT	Ende des Betrachtungszeitraums.
nStartYear	UDINT	Beginn des Betrachtungszeitraums.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.4 F_BA_DateMerge



Die Funktion F_BA_DateMerge vom Rückgabetyp ST_BA_Date [▶ 75] ersetzt vom Eingangsdatum *stDate1* diejenigen Teilkomponenten, welche als nicht spezifiziert (16#FF) gewertet sind, mit den entsprechenden Komponenten vom Eingangsdatum *stDate2*.

Syntax

```
FUNCTION F_BA_DateMerge : ST_BA_Date
VAR_INPUT
  stDate1       : ST_BA_Date;
  stDate2       : ST_BA_Date;
END_VAR
```

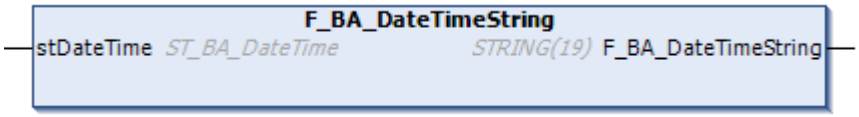
Eingänge

Name	Typ	Beschreibung
stDate1	ST_BA_Date [▶ 75]	Das Eingangsdatum, welches geprüft wird, ob Teile ersetzt werden müssen.
stDate2	ST_BA_Date [▶ 75]	Eingangsdatum, welches zum Ersetzen von *stDate1* verwendet wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.5 F_BA_DateTimeString



Die Funktion F_BA_DateTimeString vom Rückgabetyp STRING(19) wandelt ein Datum *stDateTime* in einen STRING vom Format TT.MM.JJJJ hh:mm:ss.

Dezimalwerte kleiner 10 werden mit "0" ergänzt, so wird z.B. der Monat September mit "09" dargestellt, statt einfach nur mit "9".

Nicht plausible Werte (z.B. Monat > 12) oder als nicht spezifiziert, d.h. mit 16#FF gewertete Einträge, werden mit "**" bzw. "****" für die Jahresangabe ausgegeben.

Syntax

```
FUNCTION F_BA_DateTimeString : STRING(19)
VAR_IN_OUT
  stDateTime    : ST_BA_DateTime;
END_VAR
```

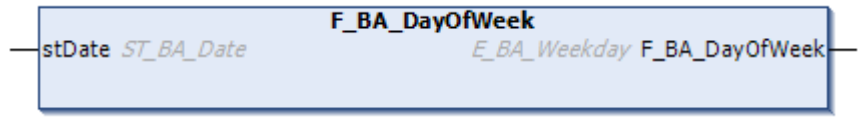
Ein- / Ausgänge

Name	Typ	Beschreibung
stDateTime	ST_BA_DateTime [▶ 76]	Datum und Uhrzeit, aus denen der Rückgabe-String gebildet wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.6 F_BA_DayOfWeek



Die Funktion F_BA_DayOfWeek vom Rückgabetyp E_BA_Weekday [▶ 57] ermittelt aus einem vorgegebenen Datum *stDate* den entsprechenden Wochentag.

Syntax

```
FUNCTION F_BA_DayOfWeek : E_BA_Weekday
VAR_INPUT
  stDate    : ST_BA_Date;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stDate	ST_BA_Date [▶ 75]	Datum, aus dem der Wochentag ermittelt wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.7 F_BA_DaysInMonth

Die Funktion F_BA_DaysInMonth vom Rückgabetyp UDINT ermittelt aus einem vorgegebenen Datum *stBACnetDate* die Anzahl der Tage des angegebenen Monats, sofern der Monat und das Jahr nicht als unspezifiziert, d.h. mit 16#FF gewertet sind. Im Falle der fehlenden Spezifizierung ist der Rückgabewert der Funktion 16#FFFFFFFF.

Syntax

```
FUNCTION F_BA_DaysInMonth : UDINT
VAR_INPUT
  stBACnetDate    : ST_BA_Date;
END_VAR
```

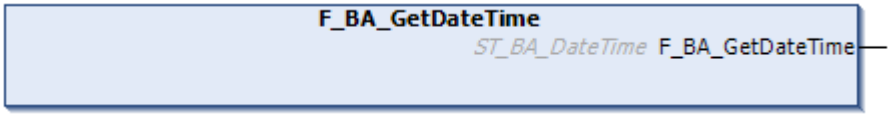
Eingänge

Name	Typ	Beschreibung
stBACnetDate	ST_BA_Date [▶ 75]	Datum, aus dessen Monat die Anzahl der Tage ermittelt werden soll.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.8 F_BA_GetDateTime



Die Funktion vom Rückgabetyp ST_BA_DateTime [▶ 76] das Datum und die Uhrzeit aus.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.9 F_BA_GetDT

Die Funktion vom Rückgabetyp DT (DATE_AND_TIME) das Datum und die Uhrzeit aus.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.10 F_BA_TimeMerge

Die Funktion F_BA_TimeString vom Rückgabetyp ST_BA_Time [▶ 76] ersetzt vom Eingangszeitstempel *stTime1* diejenigen Teilkomponenten, welche als nicht spezifiziert (16#FF) gewertet sind, mit den entsprechenden Komponenten vom Eingangszeitstempel *stTime2*.

Syntax

```
FUNCTION F_BA_TimeMerge : ST_BA_Time
VAR_INPUT
  stTime1    : ST_BA_Time;
  stTime2    : ST_BA_Time;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
stTime1	ST_BA_Time [▶ 76]	Eingangszeitstempel, welcher geprüft wird.
stTime2	ST_BA_Time [▶ 76]	Eingangszeitstempel, welcher zur Ersetzung nicht spezifizierter Teile von *stTime1* herangezogen wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.3.11 F_BA_TimeString

Die Funktion F_BA_TimeString vom Rückgabetyp STRING(12) wandelt einen Zeitstempel *stTime* in einen STRING im Format hh:mm:ss:**.
Die letzten beiden Stellen der Hundertstel-Sekunden werden dabei außer Acht gelassen und mit "*" gefüllt. Dezimalwerte kleiner 10 werden mit "0" ergänzt, so wird z.B. die neunte Stunde am Vormittag mit "09" dargestellt, statt einfach nur mit "9".

Nicht plausible Werte (z.B. Sekunde > 59) oder als nicht spezifiziert, d.h. mit 16#FF gewertete Einträge, werden mit "**" ausgegeben.

Syntax

```
FUNCTION F_BA_TimeString : STRING(12)
VAR_IN_OUT
  stTime    : ST_BA_Time;
END_VAR
```

Ein- / Ausgänge

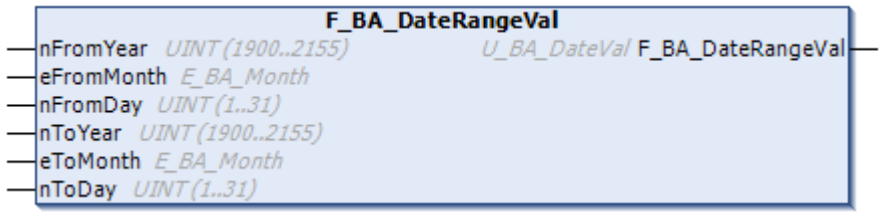
Name	Typ	Beschreibung
stTime	ST_BA_Time [▶ 76]	Zeitstempel, aus dem der Rückgabe-String gebildet wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.4 DateValue

4.1.1.3.4.1 F_BA_DateRangeVal



Die Funktion F_BA_DateRangeVal vom Rückgabetyp U_BA_DateVal [▶ 77] füllt anhand der Eingabevariablen *nFromYear*, *eFromMonth*, *nFromDay*, *nToYear*, *eToMonth* und *nToDay* die Teilkomponente ST_BA_DateRange [▶ 75] der Funktionsrückgabe U_BA_DateVal [▶ 77]. Damit wird ein Zeitbereich definiert.

Syntax

```
FUNCTION F_BA_DateRangeVal : U_BA_DateVal
VAR_INPUT
  nFromYear     : UINT(1900 .. 2155);
  eFromMonth    : E_BA_Month := E_BA_Month.Unspecified;
  nFromDay      : UINT(1 .. 31);

  nToYear       : UINT(1900 .. 2155);
  eToMonth      : E_BA_Month := E_BA_Month.Unspecified;
  nToDay        : UINT(1 .. 31);
END_VAR
```

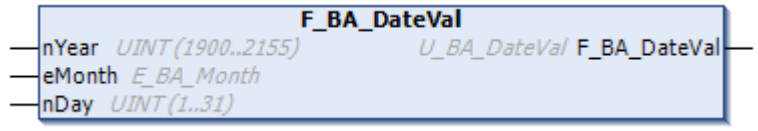
Eingänge

Name	Typ	Beschreibung
nFromYear	UINT	Anfangsjahr des Zeitbereiches.
eFromMonth	E_BA_Month [▶ 56]	Anfangsmonat des Zeitbereiches.
nFromDay	UINT	Anfangstag des Zeitbereiches.
nToYear	UINT	End-Jahr des Zeitbereiches.
eToMonth	E_BA_Month [▶ 56]	End-Monat des Zeitbereiches.
nToDay	UINT	End-Tag des Zeitbereiches.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.4.2 F_BA_DateVal



Die Funktion F_BA_DateVal vom Rückgabetyp U_BA_DateVal [▶ 77] füllt anhand der Eingabevariablen *nYear*, *eMonth* und *nDay* die Teilkomponente ST_BA_Date [▶ 75] der Funktionsrückgabe U_BA_DateVal [▶ 77]. Damit wird ein Datum definiert.

Syntax

```
FUNCTION F_BA_DateVal : U_BA_DateVal
VAR_INPUT
  nYear    : UINT(1900 .. 2155);
  eMonth    : E_BA_Month := E_BA_Month.Unspecified;
  nDay    : UINT(1 .. 31);
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nYear	UINT	Eintrag des Jahres.
eMonth	E_BA_Month [▶ 56]	Eintrag des Monats.
nDay	UINT	Eintrag des Tages.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.4.3 F_BA_WeekNDayVal

Die Funktion F_BA_WeekNDayVal vom Rückgabetyp U_BA_DateVal [▶ 77] füllt anhand der Eingabevariablen *eWeekday*, *eWeekOfMonth* und *eMonth* die Teilkomponente ST_BA_WeekNDay [▶ 76] der Funktionsrückgabe U_BA_DateVal [▶ 77]. Damit wird ein Wochentag innerhalb eines Monats definiert.

Syntax

```
FUNCTION F_BA_WeekNDayVal : U_BA_DateVal
VAR_INPUT
  eWeekday        : E_BA_Weekday  := E_BA_Weekday.Invalid;
  eWeekOfMonth    : E_BA_Week     := E_BA_Week.Invalid;
  eMonth          : E_BA_Month    := E_BA_Month.Invalid;
END_VAR
```

Eingänge

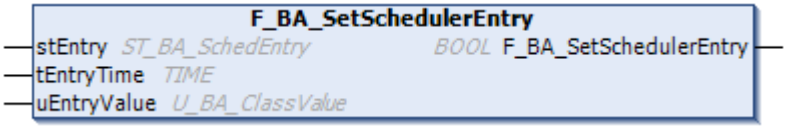
Name	Typ	Beschreibung
eWeekday	E_BA_Weekday [▶ 57]	Eintrag des Wochentages.
eWeekOfMonth	E_BA_Week [▶ 56]	Eintrag der Woche innerhalb des Monats.
eMonth	E_BA_Month [▶ 56]	Eintrag des Monats.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.5 Scheduler

4.1.1.3.5.1 F_BA_SetSchedulerEntry



Die Funktion F_BA_SetSchedulerEntry vom Rückgabetyp BOOL überprüft zunächst, ob die Eingabe-Struktur *uEntryValue* als undefiniert, d.h. mit 16#FF-Werten gefüllt ist.
Ist dies der Fall, so wird der Statusvariable *eState* der Ein- Ausgabestruktur *stEntry* der Wert E_BA_SchedEntryState.eNull zugewiesen, im gegenteiligen Fall der Wert E_BA_SchedEntryState.eValue.

Die Zeitkomponente *stTime* der der Ein- Ausgabestruktur *stEntry* wird der Eingang *tEntryTime*, angepasst über die Funktion F_BA_ToSTTime [▶ 23] zugewiesen und der Wertekomponente *uValue* der Eingangswert *uEntryValue*.
Dem Funktionsrückgabewert selbst wird nichts zugewiesen.

Syntax

```
FUNCTION F_BA_SetSchedulerEntry : BOOL
VAR_IN_OUT
  stEntry        : ST_BA_SchedEntry;
END_VAR
VAR_INPUT
  tEntryTime     : TIME;
  uEntryValue    : U_BA_ClassValue;
END_VAR
```

Ein- / Ausgänge

Name	Typ	Beschreibung
stEntry	ST_BA_SchedEntry [▶ 78]	Verweis auf die zu lesende und schreibende Zeitschaltplan-Eintrags-Struktur.

Eingänge

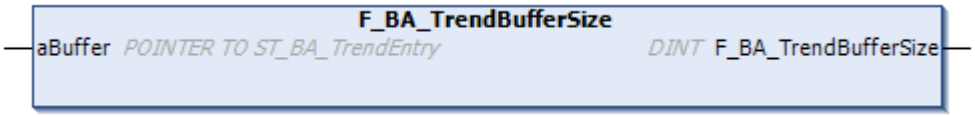
Name	Typ	Beschreibung
tEntryTime	TIME	Eingabe der Zeit.
uEntryValue	U_BA_ClassValue [▶ 82]	Eingabe des Wertes.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.6 Trend

4.1.1.3.6.1 F_BA_TrendBufferSize



Die Funktion F_BA_TrendBufferSize vom Rückgabetyp DINT ermittelt die Größe eines Trends, welcher als ARRAY vom Typ ST_BA_TrendEntry [▶ 80] definiert ist.

Die Ein-/ Ausgangsvariable *aBuffer* verweist dabei auf den zu untersuchenden Trend und erlaubt durch den Syntax ARRAY [*] das Verknüpfen von Feldern unbestimmter Größe.

Syntax

```
FUNCTION F_BA_TrendBufferSize : DINT
VAR_IN_OUT
  aBuffer    : ARRAY [*] OF ST_BA_TrendEntry;
END_VAR
```

Ein- / Ausgänge

Name	Typ	Beschreibung
aBuffer	ST_BA_TrendEntry [▶ 80]	Verweis auf den zu untersuchenden Trend.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.3.7 F_BA_IsDisturbed

Die Funktion F_BA_IsDisturbed vom Rückgabetyp BOOL überprüft die Teilkomponenten der Eingabestruktur *stStateFlags*.

Ist die Variable *bFault* oder *bInAlarm* gesetzt, so ist der Rückgabewert der Funktion TRUE.

Syntax

```
FUNCTION F_BA_IsDisturbed : BOOL
VAR_INPUT
  stStateFlags    : ST_BA_StatusFlags;
END_VAR
```

Eingänge

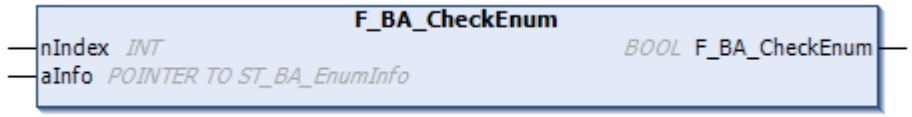
Name	Typ	Beschreibung
stStateFlags	ST_BA_StatusFlags [▶ 79]	Statusflags, die geprüft werden.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.4 Universal

4.1.1.4.1 F_BA_CheckEnum



Die Funktion F_BA_CheckEnum vom Rückgabetyp BOOL ermittelt, ob der Wert *nIndex* einer Enumeration innerhalb einer vordefinierten Liste *aInfo* vom Typ ARRAY [*] OF ST_BA_EnumInfo [▶ 79], vorhanden ist.

Durch den Syntax ARRAY [*] ist der Verweis auf Felder unbestimmter Größe möglich.

Beispiel:

```
aAction  : ARRAY[1 .. 2] OF ST_BA_EnumInfo := [
  (*eDirect*)  (sName := 'Direkt',    sDescription := 'Gleichläufiger Wirksinn',  sShortcut := ''),
  (*eReverse*) (sName := 'Indirekt',  sDescription := 'Gegenläufiger Wirksinn',   sShortcut := '')
  ];
```

Würde dieses Feld an die IN_OUT-Variable *aInfo* angelegt werden, würde die Funktion für die Werte 1 und 2 an *nIndex* den Rückgabewert "TRUE" ausgeben, anderenfalls "FALSE".

Syntax

```
FUNCTION F_BA_CheckEnum : BOOL
VAR_INPUT
  nIndex      : INT;
END_VAR
VAR_IN_OUT
  aInfo       : ARRAY [*] OF ST_BA_EnumInfo;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nIndex	INT	Zu untersuchender Index.

Ein- / Ausgänge

Name	Typ	Beschreibung
aInfo	ST_BA_EnumInfo	Liste, in der das Element mit dem Index *nIndex* zu finden sein soll.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.4.2 TcLog

4.1.1.4.2.1 FB_BA_LogMessage

Intern arbeitet die Funktion FB_BA_LogMessage mit dem Baustein FB_FormatString.
Im ersten Durchlauf wird ein Melde-String aus *sLogText* erzeugt. Der Text *sLogText* braucht neben dem Meldetext eine gültige Formatspezifikation.
Dem daraus entstandenen Text wird im zweiten Durchlauf von FB_FormatString ein Textargument in eckigen Klammern vorangestellt.

Beispiel

LogMessage

```
1  PROGRAM LogMessage
2  VAR
3      i               :   UINT;
4      sText1          :   T_MaxString := 'Message';
5  END_VAR
6
```

```
1  IF i < 10 THEN
2  F_BA_LogMessage(ADSLOG_MSGTYPE_ERROR,TO_STRING(i), sText1);
3  END_IF
4
5  i:=i+1;
6
7  IF i >= 10 THEN
8      i := 10;
9  END_IF
```

Die Funktion F_BA_LogMessage wird in zehn aufeinanderfolgenden Zyklen durchlaufen, danach nicht mehr. Der Logtyp ist ADSLOG_MSGTYPE_ERROR, das bedeutet, die Meldeausgabe ist vom Typ Fehlermeldung. Der Variable *sLogCode* wird die Laufvariable *i* zugeordnet, welche im Text in eckigen Klammern auftritt. Die Eingangsvariable *sLogText* der Funktion enthält den Text "Message".

Ausgabe

Error List

Entire Solution | 10 Errors | 0 of 1 Warning | 0 Messages | Clear | Build + IntelliSense

Code	Description
	30.08.2023 11:33:22 969 ms | 'PlcTask' (350): [0] Message
	30.08.2023 11:33:23 014 ms | 'PlcTask' (350): [1] Message
	30.08.2023 11:33:23 059 ms | 'PlcTask' (350): [2] Message
	30.08.2023 11:33:23 104 ms | 'PlcTask' (350): [3] Message
	30.08.2023 11:33:23 149 ms | 'PlcTask' (350): [4] Message
	30.08.2023 11:33:23 194 ms | 'PlcTask' (350): [5] Message
	30.08.2023 11:33:23 239 ms | 'PlcTask' (350): [6] Message
	30.08.2023 11:33:23 284 ms | 'PlcTask' (350): [7] Message
	30.08.2023 11:33:23 329 ms | 'PlcTask' (350): [8] Message
	30.08.2023 11:33:23 374 ms | 'PlcTask' (350): [9] Message

Der erzeugte Code dient zur Identifizierung einer bestimmten Stelle im Quellcode. Das oben gezeigte Beispiel dient zur Verdeutlichung der Funktionsweise dieser Funktion. Grundsätzlich sollen zur Erzeugung der Log Messages explizit ein markantes Kürzel (z.B. „IO50“ wenn innerhalb der Methode *InitObject* in Zeile 50 eine Meldung ausgegeben wird) angegeben sein.

Damit wird ein Fehler zur Laufzeit angezeigt und durch die Tastenkombination „Strg+F“ wird die fehlerhafte Stelle erreicht.

Syntax

```
FUNCTION F_BA_LogMessage
VAR_INPUT
  nLogType    : DWORD     := ADSLOG_MSGTYPE_ERROR;
  sLogCode    : STRING    := '';
  sLogText    : T_MaxString;
END_VAR
```

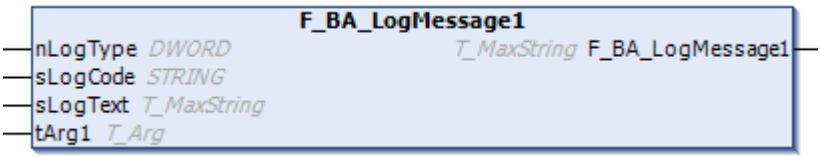
Eingänge

Name	Typ	Beschreibung
nLogType	DWORD	Logtyp, der als Maske gesetzt werden kann. Die Meldung wird dann je nach Setzen dieser Maske ausgegeben. Die Maske für ADSLOG_MSGTYPE_LOG wird intern mit gesetzt.
sLogCode	STRING	Ein Argument in textueller Form, welches der Meldung vorangestellt wird.
sLogText	T_MaxString	Meldung als Text

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.4.2.2 FB_BA_LogMessage1



Intern arbeitet die Funktion FB_BA_LogMessage1 mit dem Baustein FB_FormatString.
Im ersten Durchlauf wird ein Melde-String aus *sLogText* erzeugt, welche *tArg1* zugeordnet ist. Die zugeordnete Variable muss mit der richtigen Datentypkonvertierung auf *tArg1* verknüpft werden, der Text *sLogText* braucht neben dem Meldetext eine gültige Formatspezifikation.
Dem daraus entstandenen Text wird im zweiten Durchlauf von FB_FormatString ein Textargument in eckigen Klammern vorangestellt.

Beispiel

LogMessage1

```
1  PROGRAM LogMessage1
2  VAR
3      i               :   UINT;
4      sText1          :   T_MaxString := 'Appendix';
5  END_VAR
```

```
1  IF i < 10 THEN
2      F_BA_LogMessage1(ADSLOG_MSGTYPE_ERROR,TO_STRING(i), 'Message %s', F_STRING(sText1));
3  END_IF
4
5  i:=i+1;
6
7  IF i >= 10 THEN
8      i := 10;
9  END_IF
```

Die Funktion F_BA_LogMessage1 wird in zehn aufeinanderfolgenden Zyklen durchlaufen, danach nicht mehr. Der Logtyp ist ADSLOG_MSGTYPE_ERROR, das bedeutet, die Meldeausgabe ist vom Typ Fehlermeldung. Der Variable *sLogCode* wird die Laufvariable *i* zugeordnet, welche im Text in eckigen Klammern auftritt. Die Eingangsvariable *sLogText* der Funktion enthält neben dem Text "Meldung" auch noch eine Formatspezifikation "%s", welche den Text "Appendix" der Variable *sText1* mit anzeigen lässt. Ohne die Formatspezifikation würde " Appendix " nicht erscheinen.

Ausgabe

Error List

Entire Solution | 10 Errors | 0 of 1 Warning | 0 of 11 Messages | Clear | Build + IntelliSense

Code	Description
	30.08.2023 11:29:27 433 ms | 'PlcTask' (350): [0] Message Appendix
	30.08.2023 11:29:27 478 ms | 'PlcTask' (350): [1] Message Appendix
	30.08.2023 11:29:27 523 ms | 'PlcTask' (350): [2] Message Appendix
	30.08.2023 11:29:27 568 ms | 'PlcTask' (350): [3] Message Appendix
	30.08.2023 11:29:27 613 ms | 'PlcTask' (350): [4] Message Appendix
	30.08.2023 11:29:27 658 ms | 'PlcTask' (350): [5] Message Appendix
	30.08.2023 11:29:27 703 ms | 'PlcTask' (350): [6] Message Appendix
	30.08.2023 11:29:27 748 ms | 'PlcTask' (350): [7] Message Appendix
	30.08.2023 11:29:27 793 ms | 'PlcTask' (350): [8] Message Appendix
	30.08.2023 11:29:27 838 ms | 'PlcTask' (350): [9] Message Appendix

Der erzeugte Code dient zur Identifizierung einer bestimmten Stelle im Quellcode. Das oben gezeigte Beispiel dient zur Verdeutlichung der Funktionsweise dieser Funktion. Grundsätzlich sollen zur Erzeugung der Log Messages explizit ein markantes Kürzel (z.B. „IO50“ wenn innerhalb der Methode *InitObject* in Zeile 50 eine Meldung ausgegeben wird) angegeben sein.

Damit wird ein Fehler zur Laufzeit angezeigt und durch die Tastenkombination „Strg+F“ wird die fehlerhafte Stelle erreicht.

Syntax

```
FUNCTION F_BA_LogMessage
VAR_INPUT
  nLogType    : DWORD     := ADSLOG_MSGTYPE_ERROR;
  sLogCode    : STRING    := '';
  sLogText    : T_MaxString;
  tArg1       : T_Arg;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nLogType	DWORD	Logtyp, der als Maske gesetzt werden kann. Die Meldung wird dann je nach Setzen dieser Maske ausgegeben. Die Maske für ADSLOG_MSGTYPE_LOG wird intern mit gesetzt.
sLogCode	STRING	Ein Argument in textueller Form, welches der Meldung vorangestellt wird.
sLogText	T_MaxString	Meldung als Text.
tArg1	T_Arg	Weiterer Meldetext, welcher hinten angestellt wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.4.2.3 FB_BA_LogMessage2

Diese Funktion arbeitet wie F_BA_LogMessage1 [▶ 34], jedoch können zwei Meldetexte tArg1 und tArg2 hintenangestellt werden.

Dem Beispielprogramm aus F_BA_LogMessage1 [▶ 34] folgend, muss der Meldung ein zweites "%s" hintenangestellt werden, um die weitere Meldung anzuzeigen.

Beispiel

LogMessage2

```
PROGRAM LogMessage2
VAR
    i               :   UINT;
    sText1          :   T_MaxString := 'Appendix_1';
    sText2          :   T_MaxString := 'Appendix_2';
END_VAR
```

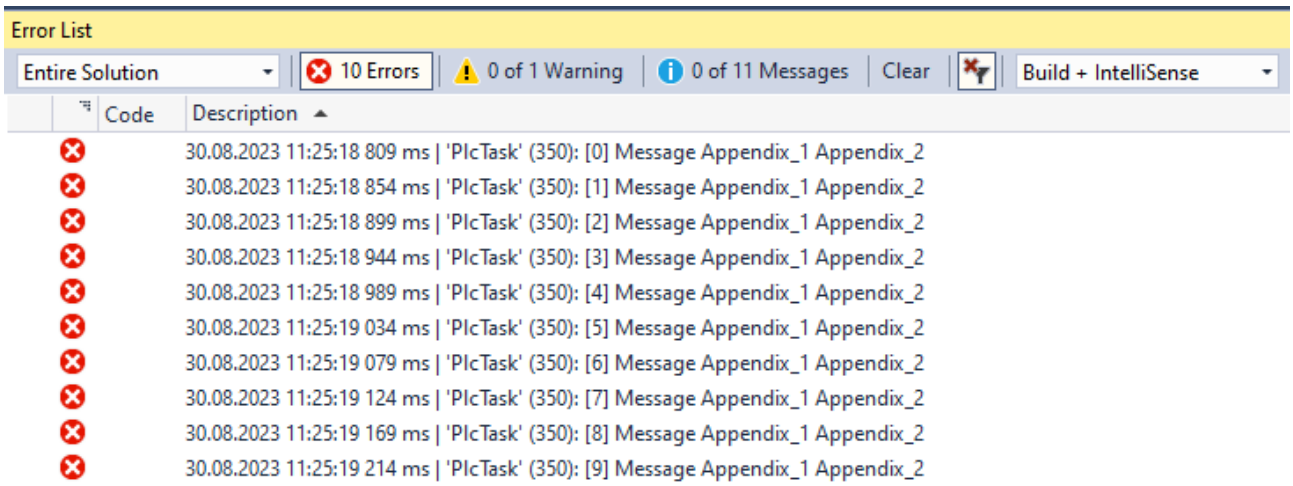
```
IF i < 10 THEN
    F_BA_LogMessage2(ADSLOG_MSGTYPE_ERROR,TO_STRING(i), 'Message %s %s', F_STRING(sText1), F_STRING(sText2));
END_IF

i:=i+1;

IF i >= 10 THEN
    i := 10;
END_IF
```

Es existieren noch acht weitere Bausteine des gleichen Typs: F_BA_LogMessage3 bis F_BA_LogMessage10. Mit diesen Funktionen ist es möglich, drei bis zehn zusätzliche Meldungen anzufügen. Für jede Meldung muss dann eine weitere Textvariable angelegt werden, *sText3* bis *sText10* und der Meldung in Zeile zwei des obigen Beispiels jeweils ein "%s" hinzugefügt werden.

Ausgabe



Der erzeugte Code dient zur Identifizierung einer bestimmten Stelle im Quellcode. Das oben gezeigte Beispiel dient zur Verdeutlichung der Funktionsweise dieser Funktion. Grundsätzlich sollen zur Erzeugung der Log Messages explizit ein markantes Kürzel (z.B. „IO50“ wenn innerhalb der Methode *InitObject* in Zeile 50 eine Meldung ausgegeben wird) angegeben sein.

Damit wird ein Fehler zur Laufzeit angezeigt und durch die Tastenkombination „Strg+F“ wird die fehlerhafte Stelle erreicht.

Syntax

```
FUNCTION F_BA_LogMessage
VAR_INPUT
  nLogType    : DWORD     := ADSLOG_MSGTYPE_ERROR;
  sLogCode    : STRING    := '';
  sLogText    : T_MaxString;
  tArg1       : T_Arg;
  tArg2       : T_Arg;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
nLogType	DWORD	Logtyp, der als Maske gesetzt werden kann. Die Meldung wird dann je nach Setzen dieser Maske ausgegeben. Die Maske für ADSLOG_MSGTYPE_LOG wird intern mit gesetzt.
sLogCode	STRING	Ein Argument in textueller Form, welches der Meldung vorangestellt wird.
sLogText	T_MaxString	Meldung als Text.
tArg1	T_Arg	Weiterer Meldetext, welcher hintenangestellt wird.
tArg2	T_Arg	Weiterer Meldetext, welcher hintenangestellt wird.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.4.2.4 FB_BA_LogMessage3

Anwendung siehe F_BA_LogMessage2 [▶ 36].

4.1.1.4.2.5 FB_BA_LogMessage4

Anwendung siehe F_BA_LogMessage2 [▶ 36].

4.1.1.4.2.6 FB_BA_LogMessage5

Anwendung siehe F_BA_LogMessage2 [▶ 36].

4.1.1.4.2.7 FB_BA_LogMessage6

Anwendung siehe F_BA_LogMessage2 [▶ 36].

4.1.1.4.2.8 FB_BA_LogMessage7

Anwendung siehe F_BA_LogMessage2 [▶ 36].

4.1.1.4.2.9 FB_BA_LogMessage8

Anwendung siehe F_BA_LogMessage2 [▶ 36].

4.1.1.4.2.10 FB_BA_LogMessage9

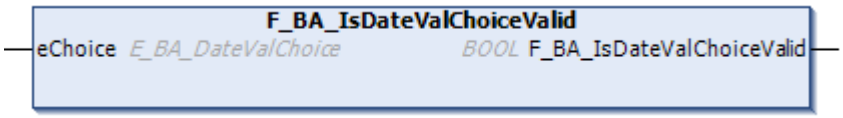
Anwendung siehe F_BA_LogMessage2 [▶ 36].

4.1.1.4.2.11 FB_BA_LogMessage10

Anwendung siehe F_BA_LogMessage2 [▶ 36].

4.1.1.5 Validierungsfunktionen

4.1.1.5.1 F_BA_IsDateValChoiceValid



Die Funktion F_BA_IsDateValChoiceValid vom Rückgabetyp BOOL überprüft, ob der Wert der Enumeration, die am Eingang *eChoice* anliegt, innerhalb der Grenzen First und Last liegt, wie sie unter E_BA_DateValChoice [▶ 54] definiert sind.

Syntax

```
FUNCTION F_BA_IsDateValChoiceValid : BOOL
VAR_INPUT
  eChoice    : E_BA_DateValChoice;
END_VAR
```

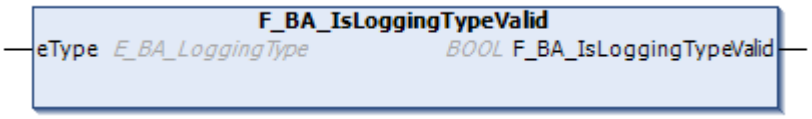
Eingänge

Name	Typ	Beschreibung
eChoice	E_BA_DateValChoice [▶ 54]	Zu überprüfende Enumeration.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.5.2 F_BA_IsLoggingTypeValid



Die Funktion F_BA_IsLoggingTypeValid vom Rückgabetyp BOOL überprüft, ob der Wert der Enumeration, die am Eingang *eType* anliegt, innerhalb der Grenzen First und Last liegt, wie sie unter E_BA_LoggingType definiert sind.

```
Syntax
FUNCTION F_BA_IsLoggingTypeValid : BOOL
VAR_INPUT
  eType    : E_BA_LoggingType;
END_VAR
```

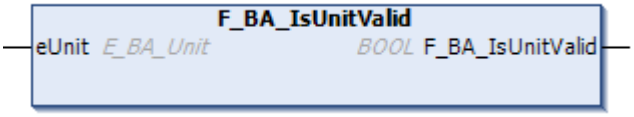
Eingänge

Name	Typ	Beschreibung
eType	E_BA_LoggingType [▶ 60]	Zu überprüfende Enumeration.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.5.3 F_BA_IsUnitValid



Die Funktion F_BA_IsUnitValid vom Rückgabetyp BOOL überprüft, ob der Wert der Enumeration, die am Eingang *eUnit* anliegt, innerhalb der Grenzen First und Last liegt, wie sie unter E_BA_Unit [▶ 61] definiert sind.

Syntax

```
FUNCTION F_BA_IsUnitValid : BOOL
VAR_INPUT
  eUnit    : E_BA_Unit;
END_VAR
```

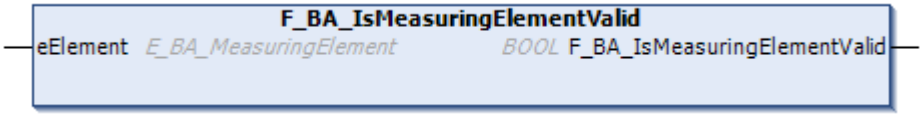
Eingänge

Name	Typ	Beschreibung
eUnit	E_BA_Unit [▶ 61]	Zu überprüfende Enumeration.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.5.4 F_BA_IsMeasuringElementValid



Die Funktion F_BA_IsMeasuringElementValid vom Rückgabetyp BOOL überprüft, ob der Wert der Enumeration, die am Eingang *eElement* anliegt, innerhalb der Grenzen First und Last liegt, wie sie unter E_BA_MeasuringElement [▶ 71] definiert sind.

Syntax

```
FUNCTION F_BA_IsMeasuringElementValid : BOOL
VAR_INPUT
  eElement    : E_BA_MeasuringElement;
END_VAR
```

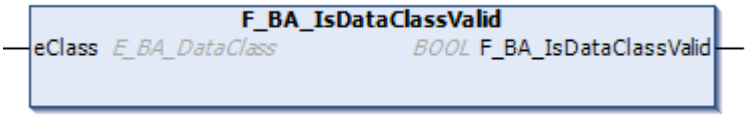
Eingänge

Name	Typ	Beschreibung
eElement	E_BA_MeasuringElement [▶ 71]	Zu überprüfende Enumeration.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.5.5 F_BA_IsDataClassValid



Die Funktion F_BA_IsDataClassValid vom Rückgabetyp BOOL überprüft, ob der Wert der Enumeration, die am Eingang *eClass* anliegt, innerhalb der Grenzen First und Last liegt, wie sie unter E_BA_DataClass [▶ 58] definiert sind.

Syntax

```
FUNCTION F_BA_IsDataClassValid : BOOL
VAR_INPUT
  eClass    : E_BA_DataClass;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eClass	E_BA_DataClass [▶ 58]	Zu überprüfende Enumeration.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.5.6 F_BA_IsDataTypeValid

Die Funktion F_BA_IsDataTypeValid vom Rückgabetyp BOOL überprüft, ob der Wert der Enumeration, die am Eingang *eType* anliegt, innerhalb der Grenzen First und Last liegt, wie sie unter E_BA_DataType [▶ 58] definiert sind.

Syntax

```
FUNCTION F_BA_IsDataTypeValid : BOOL
VAR_INPUT
  eType    : E_BA_DataType;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eType	E_BA_DataType [▶ 58]	Zu überprüfende Enumeration.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.1.5.7 F_BA_IsWeekdayValid

Die Funktion F_BA_IsWeekdayValid vom Rückgabetyp BOOL überprüft, ob der Wert der Enumeration, die am Eingang *eDay* anliegt, innerhalb der Grenzen First und Last liegt, wie sie unter E_BA_Weekday [▶ 57] definiert sind.

Syntax

```
FUNCTION F_BA_IsWeekdayValid : BOOL
VAR_INPUT
  eDay   : E_BA_Weekday;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
eDay	E_BA_Weekday [▶ 57]	Zu überprüfende Enumeration.

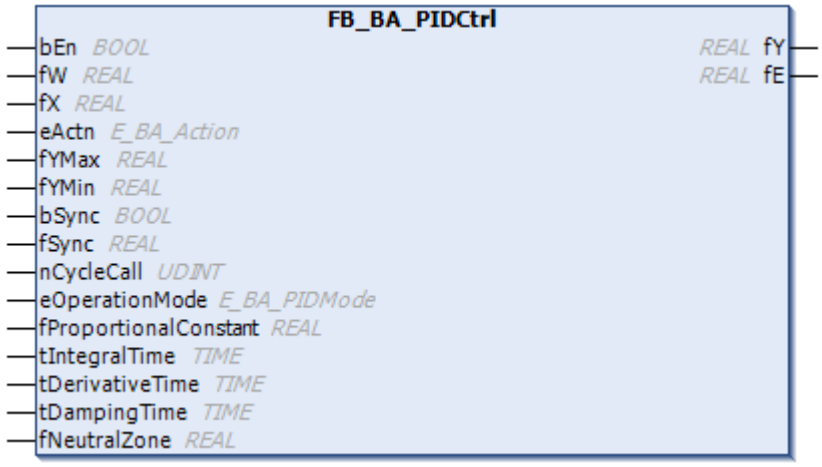
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.2 Funktionsbausteine

4.1.2.1 Regler

4.1.2.1.1 FB_BA_PIDCtrl



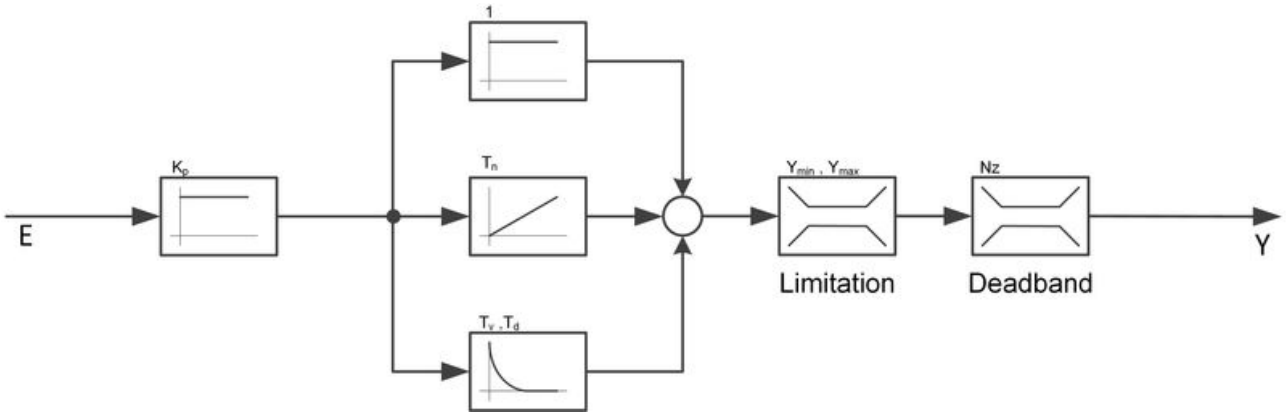
Der Funktionsbaustein FB_BA_PIDCtrl ist ein Universal-PID-Regler.

Der Regler ist intern in zwei aufeinander folgende Teile gegliedert:

- der Regler selbst, wie in den unten aufgeführten Wirkungsplänen als P-, I- und D-Anteil dargestellt mit einer Ausgangsbegrenzung (Limitation).
- einem Totbandglied (neutrale Zone), welches die Ausgangsänderungen des Reglers mit einer Hysterese beaufschlagt.

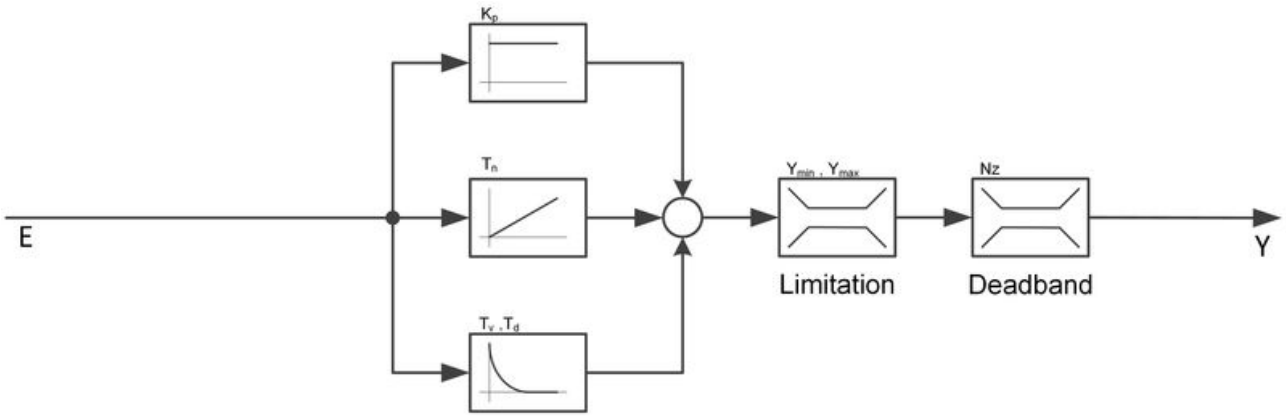
Betriebsmodus "P-Anteil vorgelagert":

(*eOperationMode* = E_BA_PIDMode.eP1ID)



Betriebsmodus "Parallelstruktur":

(*eOperationMode* = E_BA_PIDMode.ePID)



Wirksinn

Mit *bActn* = FALSE wird der Wirksinn des Reglers so umgekehrt, dass eine Regelabweichung kleiner als 0 eine Stellgrößenänderung ins Positive bewirkt. Dies wird dadurch erreicht, dass die Regelabweichung negativ berechnet wird:

bActn	*fE* (Regelabweichung)	Wirksinn
TRUE	*fX-fW* (Istwert-Sollwert)	direkt (Kühlen)
FALSE	*fW-fX* (Sollwert-Istwert)	indirekt (Heizen)

Passiv-Verhalten (bEn = FALSE)

Die Ausgänge werden wie folgt gesetzt:

fY	0.0
fE	aktuelle Regelabweichung, siehe Wirksinn.

Die internen Werte für den P-, I-, und D-Anteil werden auf 0 gesetzt, ebenso die Werte für den I- und D-Anteil vom vorhergehenden Zyklus. Damit wird die Stellgröße bei einem Neustart im ersten Zyklus ohne Vergangenheitswerte berechnet.

Aktiv-Verhalten (bEn = TRUE)

Im ersten Zyklus werden I- und D-Anteil wie bereits erwähnt ohne Vergangenheitswerte berechnet und frei von alten Werten aufgestartet.

Anti-Reset-Windup

Ist der I-Anteil aktiv, so sorgt der Regler dafür, dass dieser festgehalten wird, sollte der Regler Ausgang *rY* über die Grenzen *fYMin* oder *fYMax* hinausgehen wollen. Innerhalb des Reglers wird in jedem Zyklus eine Vorberechnung des Regler Ausgangs gemacht.

Anti-Reset-Windup an Min Grenze

Ist die Vorberechnung kleiner als die untere Ausgangsgrenze *fYMin*, so wird der I-Anteil daran gehindert weiter zu fallen und auf den Wert des letzten PLC-Zyklus begrenzt. Ein Ansteigen des I-Anteils bleibt jedoch möglich.

Anti-Reset-Windup an Max Grenze

Ist die Vorberechnung hingegen größer als die obere Grenze *fYMax*, so wird der I-Anteil daran gehindert weiter zu steigen und ebenfalls auf den Wert des letzten PLC-Zyklus begrenzt. In diesem Fall bleibt ein Abfallen des I-Anteiles weiter möglich.

Synchronisationen

Es gibt verschiedene Fälle in denen Regler Ausgang nicht nur begrenzt, sondern durch Manipulation des I-Anteils (wenn nicht aktiv, dann des D-Anteils bzw. des P-Anteils) auf einen neuen Wert synchronisiert werden muss. Diese Fälle sind wegen der möglichen Gleichzeitigkeit priorisiert:

Prio	Beschreibung	Bedingungen	Bemerkungen
1	Synchronisation über *bSync*/*fSync*	Regler freigegeben – *bEn* = TRUE	Ein positives Signal an *bSync* setzt den I-Anteil so, dass die Stellgröße den Wert *fSync* annimmt. Diese Methode kann, wenn *bEn* und *bSync* gleichzeitig gesetzt werden, zum Setzen eines Initialwertes genutzt werden, von dem aus die Regelung startet. Ist der I-Anteil nicht aktiv, so wird der D-Anteil entsprechend gesetzt. Zu beachten ist, dass intern nur die steigende Flanke von *bSync* ausgewertet wird, da es sich um eine Setz-Aktion handelt. Für ein erneutes Synchronisieren, etwa auf einen Übergabewert, muss am Eingang *bSync* ein erneutes TRUE-Signal angelegt werden.
2	Bereichssynchronisation *fYMin*	Regler freigegeben – *bEn* = TRUE *fYMin* <> fYMin_1 (letzterZyklus) fY_Test < *fYMin*	Hat sich die untere Bereichsgrenze geändert und die vorberechnete Regler Ausgabe ist nun kleiner als *fYMin* so wird auf *fYMin* synchronisiert.
3	Bereichssynchronisation *fYMax*	Regler freigegeben – *bEn* = TRUE *fYMax* <> fYMax_1 (letzterZyklus) fY_Test > *fYMax*	hat sich die obere Bereichsgrenze geändert und die vorberechnete Regler Ausgabe ist nun größer als *fYMax*, so wird auf *fYMax* synchronisiert
4	Synchronisation bei Wirksinnumkehr	Regler freigegeben – *bEn* = TRUE *eActn* <> eActn_1 (letzter Zyklus)	Es wird so synchronisiert, dass der Ausgang den Wert VOR der Umkehr innehat: *fY* = fY_1 (letzter Zyklus)
5	Anti-Reset-Windup	Regler freigegeben – *bEn* = TRUE	siehe Anti-Reset-Windup

Neutrale Zone

Ein Wert von *fNeutralZone* > 0.0 gibt die Funktion der neutralen Zone (Deadband) frei. Ein Wert gleich Null deaktiviert das Totbandglied und die Werte am Eingang werden direkt durchgereicht.

Ist beim aktiven Regler die Änderung am Eingang des Gliedes in einem SPS-Zyklus im Vergleich zum vorhergehenden SPS-Zyklus kleiner als *fNeutralZone* / 2, so wird der Ausgang auf dem Wert des vorhergehenden Zyklus gehalten, bis die Änderung größer oder gleich *fNeutralZone* / 2 ist.

Durch diese Funktion sollen unnötig viele Stellimpulse vermieden werden.

Syntax

```
VAR_INPUT
  bEn                    : BOOL;
  fW                     : REAL;
  fX                     : REAL;
  eActn                  : E_BA_Action := E_BA_Action.eReverse;
  fYMax                  : REAL := 100;
  fYMin                  : REAL := 0;
  bSync                  : BOOL;
  fSync                  : REAL;
END_VAR
VAR_INPUT CONSTANT PERSISTENT
  nCycleCall             : UDINT := 5;
  eOperationMode         : E_BA_PIDMode := E_BA_PIDMode.ePlID;
```

```
  fProportionalConstant    : REAL;
  tIntegralTime            : TIME;
  tDerivativeTime          : TIME;
  tDampingTime             : TIME;
  fNeutralZone             : REAL := 0.0;
END_VAR
VAR_OUTPUT
  fY                       : REAL;
  fE                       : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe des Reglers.
fW	REAL	Sollwert der Regelstrecke.
fX	REAL	Istwert der Regelstrecke.
eActn	E_BA_Action [▶ 70]	Wirksinn des Reglers.
fYMax	REAL	Obere Ausgangsbegrenzung des Reglers [%].
fYMin	REAL	Untere Ausgangsbegrenzung des Reglers [%]. Der Wert *fYMin* wird nach oben hin durch *fYMax* begrenzt.
bSync	BOOL	Synchronisiert den Regler auf den Wert von *fSync*.
fSync	REAL	Synchronisationswert. Der Wert *fSync* wird intern begrenzt auf Werte von *fYMin* bis *fYMax*.

Eingänge CONSTANT PERSISTENT

Name	Typ	Beschreibung
nCycleCall	UDINT	Aufrufzyklus des Bausteines als Vielfaches der Zykluszeit. Intern begrenzt auf einen Minimalwert von 1.
eOperationMode	E_BA_PIDMode [▶ 52]	Wirkungsweise des Reglers: PID-Modus oder P-ID-Modus
fProportionalConstant	REAL	Reglerverstärkung. Wirkt nur auf den P-Anteil. Intern begrenzt auf einen Minimalwert von 0.
TIntegralTime	TIME	Nachstellzeit des I-Anteiles [ms]. Ein Nullwert an diesem Parameter schaltet den I-Anteil ab.
tDerivativeTime	TIME	Vorhaltezeit des D-Anteiles [ms]. Ein Nullwert an diesem Parameter schaltet den D-Anteil ab.
tDampingTime	TIME	Dämpfungszeit des D-Anteiles [ms].
fNeutralZone	REAL	Totband

Ausgänge

Name	Typ	Beschreibung
fY	REAL	Stellgröße. Bereich durch *fYMin* und *fYMax* eingeschränkt.
fE	REAL	Regelabweichung (Die Berechnung ist abhängig vom Wirksinn).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.2.2 Universal

4.1.2.2.1 IO

4.1.2.2.1.1 FB_BA_KL32xx

Der Funktionsbaustein FB_BA_KL32xx dient der Konfiguration von Busklemmen der Typen KL3208_0010, KL3201, KL3202 und KL3204.

Syntax

```
FUNCTION_BLOCK FB_BA_KL32xx
VAR_INPUT
  bConfigurate            : BOOL;
  bReadConfig             : BOOL;
  eSensor                 : E_BA_MeasuringElement;
END_VAR
VAR_OUTPUT
  nState                  : USINT;
  nData                   : INT;
  fVal                    : REAL;
  bErr                    : BOOL;
  bWireBreak              : BOOL;
  bShortCircuit           : BOOL;/
END_VAR
VAR_IN_OUT
  nRawState               : USINT;
  nRawDataIn              : INT;
  nRawCtrl                : USINT;
  nRawDataOut             : INT;
END_VAR
VAR  // [Output-Properties]
  nTerminalType           : WORD;
  nSpecialType            : WORD;
  nFirmwareVersion        : WORD;
  sTerminalDescription    : STRING;
  sSensorName             : STRING;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bConfigurate	BOOL	Eine steigende Flanke startet die Konfiguration der Busklemme.
bReadConfig	BOOL	Eine steigende Flanke startet das Auslesen der Busklemme.
eSensor	E_BA_MeasuringElement	Auswahl des Sensortyps.

Ausgänge

Name	Typ	Beschreibung
nState	USINT	Ausgabe des aktuellen Klemmenstatus.
nData	INT	Ausgabe der aktuellen Prozessdaten.
fVal	REAL	Skalierter Ausgangswert.
bErr	BOOL	Fehler in der Klemmenkonfiguration.
bWireBreak	BOOL	Ein TRUE zeigt einen Drahtbruch am Sensor an.

Name	Typ	Beschreibung
bShortCircuit	BOOL	Ein TRUE zeigt einen Kurzschluss am Sensor an.

Ein- / Ausgänge

Name	Typ	Beschreibung
nRawState	USINT	Verknüpfung mit dem entsprechenden Status Byte der Busklemme im E/A Bereich des Programms.
nRawDataIn	INT	Verknüpfung mit den entsprechenden Rohdaten (Data In) der Busklemme im E/A Bereich des Programms (0...32767).
nRawCtrl	USINT	Verknüpfung mit dem entsprechenden Control Byte der Busklemme im E/A Bereich des Programms.
nRawDataOut	INT	Verknüpfung mit den entsprechenden Rohdaten (Data Out) der Busklemme im E/A Bereich des Programms.

Eigenschaften

Name	Typ	Zugriff	Beschreibung
FirmwareVersion	WORD	Get	Anzeige der Firmware der Klemme.
SensorName	STRING	Get	Anzeige des Sensortyps.
SpecialType	WORD	Get	Anzeige der speziellen Version der Klemme.
TerminalDescription	STRING	Get	Anzeige des Klemmentyps und der Firmware
TerminalType	WORD	Get	Anzeige des Klemmentyps.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.2.2.2 Trigger

4.1.2.2.2.1 FB_BA_ATrigCOV

Der Funktionsbaustein FB_BA_ATrigCOV dient zum Erkennen einer Wertänderung an einer Variablen *xValue* beliebigen Typs.

Die Größe des Variablentyps ist intern auf 4 Bytes festgelegt. Bei einer Werteänderung an *xValue* wird der Ausgang *bQ* für einen Zyklus auf TRUE gesetzt, ebenso bei einer steigenden Flanke an *bForce*. Der Bausteinausgang *bReady* wechselt auf TRUE, wenn die verknüpfte Variable an *xValue* die Grenze von 4 Bytes nicht überschreitet. Wird diese Grenze überschritten, so erscheint eine Fehlermeldung im TwinCAT-Ausgabefenster und in der Fehlerliste. Der Baustein wird die Variable *xValue* nicht weiter überprüfen und nur noch auf Änderungen am Eingang *bForce* reagieren. Der Ausgang *bReady* steht dann auf FALSE.

Syntax

```
FUNCTION_BLOCK FB_BA_ATrigCOV
VAR_INPUT
  xValue      : ANY;
  bForce      : BOOL;
END_VAR
VAR_OUTPUT
  bReady      : BOOL;
  bQ          : BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
xValue	ANY	Zu überwachender Wert. Er darf 4 Bytes nicht überschreiten.
bForce	BOOL	Eine steigende Flanke an diesem Eingang erzwingt einen Trigger-Impuls am Ausgang.

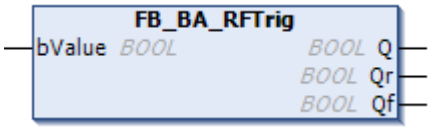
Ausgänge

Name	Typ	Beschreibung
bReady	BOOL	Wechselt auf TRUE, wenn die an *xValue* angelegte Variable 4 Byte nicht überschreitet.
bQ	BOOL	Bei einer Werteänderung an *xValue* oder bei einer steigenden Flanke an *bForce* wechselt dieser Ausgang für einen SPS-Zyklus auf TRUE.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.2.2.2.2 FB_BA_RFTrig



Der Funktionsbaustein FB_BA_RFTrig dient zum Erkennen einer steigenden oder fallenden Flanke an einer booleschen Variablen.

Syntax

```
FUNCTION_BLOCK FB_BA_RFTrig
VAR_INPUT
  bValue    :BOOL;
END_VAR
VAR_OUTPUT
  Q         :BOOL;
  Qr        :BOOL;
  Qf        :BOOL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bValue	BOOL	Zu überwachender Wert.

Ausgänge

Name	Typ	Beschreibung
Q	BOOL	Für einen Zyklus TRUE, wenn eine steigende oder fallende Flanke an *bValue* erkannt wurde.
Qr	BOOL	Für einen Zyklus TRUE, wenn eine steigende Flanke an *bValue* erkannt wurde (überwachter Wert von FALSE nach TRUE wechselt).
Qf	BOOL	Für einen Zyklus TRUE, wenn eine fallende Flanke an *bValue* erkannt wurde (überwachter Wert von TRUE nach FALSE wechselt).

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.2.2.3 Rampen Filter

4.1.2.2.3.1 FB_BA_FltrPT1

Der Funktionsbaustein FB_BA_FltrPT1 dient als Filter erster Ordnung.

Beim ersten Aufruf des Bausteines (Systemstart) wird der Ausgang *fOut* einmalig automatisch gleich dem Eingang *fIn* gesetzt.

Syntax

```
FUNCTION_BLOCK FB_BA_FltrPT1
VAR_INPUT
  fIn            : REAL;
  nDampConst     : UDINT;
  bSetActl       : BOOL;
END_VAR
VAR_OUTPUT
  fOut           : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
fIn	REAL	Eingangssignal
nDampConst	UDINT	Filterzeitkonstante [s]. Intern begrenzt auf Werte von 0 bis 86400.
bSetActl	BOOL	Eine steigende Flanke an diesem Eingang setzt den Ausgangswert *fOut* auf den Eingangswert *fIn*.

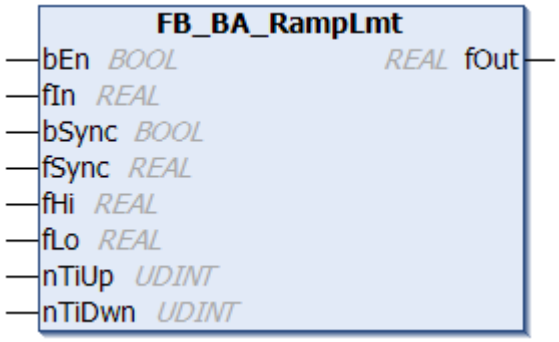
Ausgänge

Name	Typ	Beschreibung
fOut	REAL	Gefiltertes Ausgangssignal

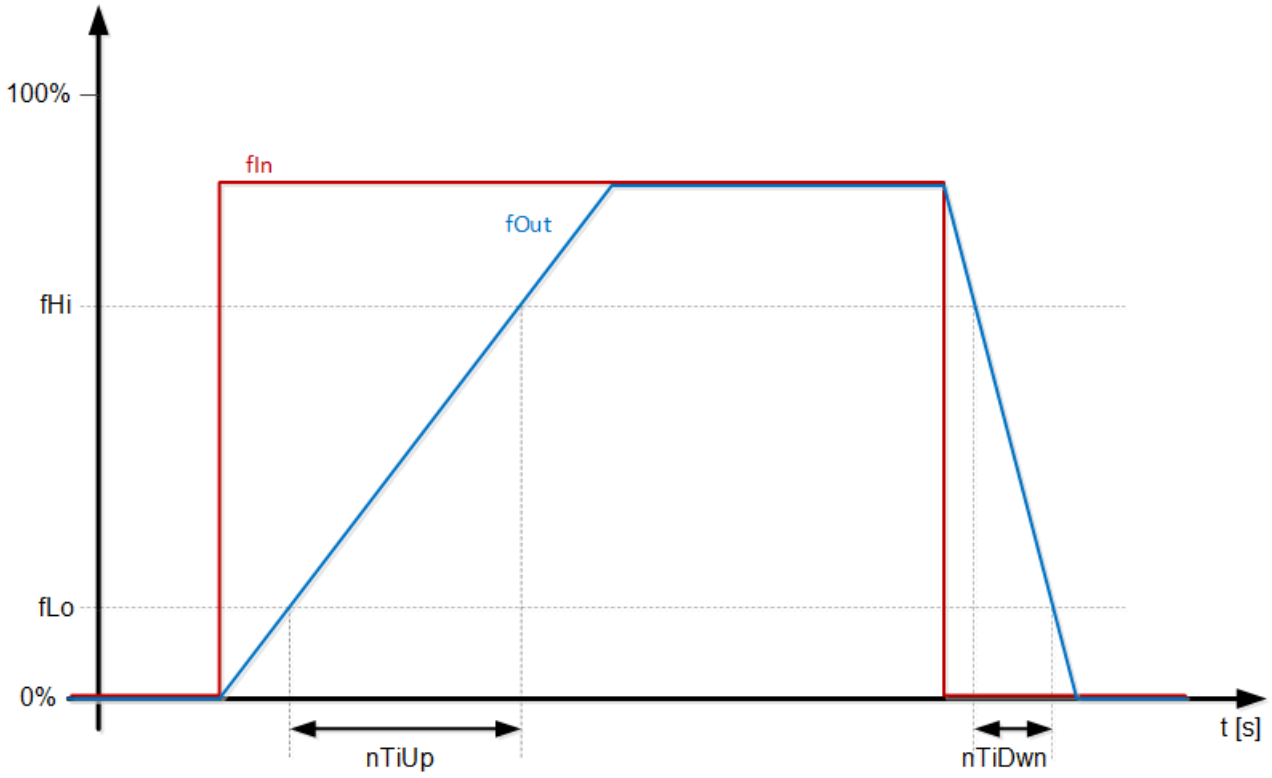
Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.1.2.2.3.2 FB_BA_RampLmt



Der Funktionsbaustein FB_BA_RampLmt limitiert die Anstiegs- bzw. Abfallgeschwindigkeit eines Eingangssignals.
Beim Anstieg von *fIn* wird der Ausgang *fOut* auf die Steigung von (*fHi-fLo*)/*nTiUp* begrenzt.
Beim Abfallen von *fIn* wird der Ausgang *fOut* auf die Steigung von (*fHi-fLo*)/*nTiDwn* begrenzt.



Syntax

```
FUNCTION_BLOCK FB_BA_RampLmt
VAR_INPUT
  bEn             : REAL;
  fIn             : REAL;
  bSync           : BOOL;
  fSync           : REAL;
  fHi             : REAL;
  fLo             : REAL;
  nTiUp           : UDINT;
  nTiDwn          : UDINT;
END_VAR
VAR_OUTPUT
  fOut            : REAL;
END_VAR
```

Eingänge

Name	Typ	Beschreibung
bEn	BOOL	Freigabe Baustein, wenn FALSE, dann ist *fOut* = 0.0.
bEnRamp	BOOL	Freigabe Rampenbegrenzung, wenn FALSE, dann ist *fOut* = *fIn*.
fIn	REAL	Eingangswert der Rampenfunktion
fHi	REAL	Obere Stützstelle zur Berechnung der Rampen
fLo	REAL	Untere Stützstelle zur Berechnung der Rampen. *fHi* muss größer als *fLo* sein, ansonsten wird ein Fehler ausgegeben!
nTiUp	UDINT	Anstiegszeit [s]
nTiDwn	UDINT	Abfallzeit [s]

Ausgänge

Name	Typ	Beschreibung
fOut	REAL	Durch die Rampen steigungs-begrenztes Ausgangssignal.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2 DUTs

4.2.1 Enumerationen

4.2.1.1 [Funktional]

4.2.1.1.1 E_BA_CompareMode

Die Variable vom Typ Aufzählung dient der Anzeige und Klassifizierung von Vergleichergebnissen.

Syntax

```
TYPE E_BA_CompareMode :
(
  Invalid           := 0,

  eLower            := 1,
  eLowerOrEqual     := 2,
  eEqual            := 3,
  eNotEqual         := 4,
  eHigherOrEqual    := 5,
  eHigher           := 6

) BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eLower	Der Vergleichswert ist kleiner.
eLowerOrEqual	Der Vergleichswert ist kleiner oder gleich.
eEqual	Der Vergleichswert ist gleich.
eNotEqual	Der Vergleichswert ist nicht gleich.
eHigherOrEqual	Der Vergleichswert ist größer oder gleich.

Name	Beschreibung
eHigher	Der Vergleichswert ist größer.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.2 Controller

4.2.1.2.1 E_BA_PIDMode

Auswahl der Reglerstruktur.

Syntax

```
TYPE E_BA_PIDMode:
(
  Invalid         := 0,
  eP1ID           := 1,
  ePID            := 2
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eP1ID	P-Glied ist vorgelagert.
ePID	P-, I- und D-Glied in Parallelstruktur.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.3 Conversion

4.2.1.3.1 E_BA_ByteMappingMode

```
TYPE E_BA_ByteMappingMode :
(
  Invalid             := 0,
  eIndex1N            := 1,
  eBinary1N           := 2,
  eIndexUpDown        := 3,
  eBinaryUpDown       := 4,
  eBinaryDecimal      := 5,
) BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eIndex1N	Setzt das indizierte Bit auf TRUE. Beispiel: 2#0000_0001 | 1 2#0000_0010 | 2 2#0000_0100 | 3 2#0000_1000 | 4
eBinary1N	Setzt nur das erste Bit eines binär gemappten Dezimalwertes auf TRUE. Beispiel:

Name	Beschreibung
	2#0000_0001 | 1 2#0000_0010 | 2, 3 2#0000_0100 | 4, 5, 6 2#0000_1000 | 8, 9, 10, 11, 12, 13, 14, 15
eIndexUpDown	Setzt eine spezifische Anzahl von Bits auf TRUE. Beispiel: 2#0000_0001 | 1 2#0000_0011 | 2 2#0000_0111 | 3 2#0000_1111 | 4
eBinaryUpDown	Setzt alle Bits eines binär gemappten Dezimalwertes auf TRUE. Beispiel: 2#0000_0001 | 1 2#0000_0011 | 2, 3 2#0000_0111 | 4, 5, 6 2#0000_1111 | 8, 9, 10, 11, 12, 13, 14, 15
eBinaryDecimal	Binär gemappter Dezimalwert. Beispiel: 2#0000_0001 | 1 2#0000_0010 | 2 2#0000_0011 | 3 2#0000_0100 | 4 2#0000_0101 | 5 2#1111_1111 | 255

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.4 Event

4.2.1.4.1 E_BA_EventState

Die Enumeration dient der Anzeige und Klassifizierung von Events.

Syntax

```
TYPE E_BA_EventState:
(
  Invalid         := 0,
  Unknown         := 1,

  eNormal         := 2,
  eFault          := 3,
  eOffNormal      := 4,
  eLowLimit       := 5;
  eHighLimit      := 6
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender
eUnknown	Status unbekannt

Name	Beschreibung
eNormal	Normalbetrieb
eFault	Unzulässiger Wert
eOffnormal	Alarmzustand
eLowLimit	Unterer Grenzwert unterschritten
eHighLimit	Oberer Grenzwert überschritten

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.4.2 E_BA_EventTransition

Beschreibt die möglichen Übergangszustände von Event-Objekten.

Syntax

```
TYPE E_BA_EventTransition:
(
  Invalid        := 0,

  eToOffnormal   := 1,
  eToFault       := 2,
  eToNormal      := 3
);
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eToOffnormal	Übergang in den Zustand *Alarm*.
eToFault	Übergang in den Zustand *Fehler*.
eToNormal	Übergang in den Zustand *Normal*.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5 Typen

4.2.1.5.1 Date and Time

4.2.1.5.1.1 E_BA_DateValChoice

Die Enumeration dient der Wahl von Zeiträumen in Zeitschaltplänen.

Syntax

```
TYPE E_BA_DateValChoice:
(
  Invalid        := 0,
  eDate          := 1,
  eDateRange     := 2,
  eWeekDay       := 3
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.

Name	Beschreibung
eDate	Auswahl eines einzelnen Datums.
eDateRange	Auswahl eines Datumsbereichs.
eWeekNDay	Auswahl eines Wochentags.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.1.2 E_BA_Day

Die Enumeration dient der Spezifizierung von Tagen innerhalb eines Monats.

Syntax

```
TYPE E_BA_Day:
(
  Invalid           := 0,
  Unspecified       := 16#FF,

  eDay01            := 1,
  eDay02            := 2,
  eDay03            := 3,
  eDay04            := 4,
  eDay05            := 5,
  eDay06            := 6,
  eDay07            := 7,
  eDay08            := 8,
  eDay09            := 9,
  eDay10            := 11,
  eDay11            := 11,
  eDay12            := 12,
  eDay13            := 13,
  eDay14            := 14,
  eDay15            := 15,
  eDay16            := 16,
  eDay17            := 17,
  eDay18            := 18,
  eDay19            := 19,
  eDay20            := 20,
  eDay21            := 21,
  eDay22            := 22,
  eDay23            := 23,
  eDay24            := 24,
  eDay25            := 25,
  eDay26            := 26,
  eDay27            := 27,
  eDay28            := 28,
  eDay29            := 29,
  eDay30            := 30,
  eDay31            := 31,

  eLastDayOfMonth   := 32,
  eOddDaysOfMonth   := 33,
  eEvenDaysOfMonth  := 34
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
Unspecified	Nicht spezifiziert.
eDayNN	Tag Nr. NN (1…31) des Monats ist spezifiziert.
eLastDayOfMonth	Der letzte Tag des Monats ist spezifiziert.
eOddDaysOfMonth	Die ungeraden Tage des Monats sind spezifiziert.
eEvenDaysOfMonth	Die geraden Tage des Monats sind spezifiziert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.1.3 E_BA_Month

Die Enumeration dient der Spezifizierung von Monaten.

Syntax

```
TYPE E_BA_Month:
(
  Invalid         := 0,
  Unspecified     := 16#FF,

  eJanuary        := 1,
  eFebruary       := 2,
  eMarch          := 3,
  eApril          := 4,
  eMay            := 5,
  eJune           := 6,
  eJuly           := 7,
  eAugust         := 8,
  eSeptember      := 9,
  eOctober        := 10,
  eNovember       := 11,
  eDecember       := 12,

  eOddMonths      := 13,
  eEvenMonths     := 14
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
Unspecified	Nicht spezifiziert.
eJanuary	Januar ist spezifiziert.
eFebruary	Februar ist spezifiziert.
eMarch	März ist spezifiziert.
eApril	April ist spezifiziert.
eMay	Mai ist spezifiziert.
eJune	Juni ist spezifiziert.
eJuly	Juli ist spezifiziert.
eAugust	August ist spezifiziert.
eSeptember	September ist spezifiziert.
eOctober	Oktober ist spezifiziert.
eNovember	November ist spezifiziert.
eDecember	Dezember ist spezifiziert.
eOddMonths	Die Monate mit ungerader Ordnungszahl sind spezifiziert.
eEvenMonths	Die Monate mit gerader Ordnungszahl sind spezifiziert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.1.4 E_BA_Week

Die Enumeration dient der Spezifizierung von Wochen innerhalb eines Monats.

Syntax

```
TYPE E_BA_Week:
(
  Invalid        := 0,
  Unspecified    := 16#FF,

  eWeek1         := 1,
  eWeek2         := 2,
  eWeek3         := 3,
  eWeek4         := 4,
  eWeek5         := 5
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
Unspecified	Nicht spezifiziert.
eWeek1	Die erste Woche innerhalb des Monats ist spezifiziert.
eWeek2	Die zweite Woche innerhalb des Monats ist spezifiziert.
eWeek3	Die dritte Woche innerhalb des Monats ist spezifiziert.
eWeek4	Die vierte Woche innerhalb des Monats ist spezifiziert.
eWeek5	Die fünfte Woche innerhalb des Monats ist spezifiziert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.1.5 E_BA_Weekday

Die Enumeration dient der Spezifizierung von Wochentagen.

Syntax

```
TYPE E_BA_Weekday:
(
  Invalid        := 0,
  Unspecified    := 16#FF,

  eMonday        := 1,
  eTuesday       := 2,
  eWednesday     := 3,
  eThursday      := 4,
  eFriday        := 5,
  eSaturday      := 6,
  eSunday        := 7
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
Unspecified	Nicht spezifiziert.
eMonday	Montag ist spezifiziert.
eTuesday	Dienstag ist spezifiziert.
eWednesday	Mittwoch ist spezifiziert.
eThursday	Donnerstag ist spezifiziert.
eFriday	Freitag ist spezifiziert.
eSaturday	Samstag ist spezifiziert.
eSunday	Sonntag ist spezifiziert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.2 E_BA_DataClass

Die Enumeration dient der Spezifizierung von Werten.

Syntax

```
TYPE E_BA_DataClass:
(
  Invalid         := 0,
  Unknown         := 1,
  Null            := 2,

  eAnalog         := 10,
  eBinary         := 11,
  eMultistate     := 12
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
Unknown	Unbekannt.
Null	Nicht spezifiziert / nicht gesetzt.
eAnalog	Analog-Objekt.
eBinary	Binär-Objekt.
eMultistate	Multistate-Objekt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.3 E_BA_DataType

Die Enumeration dient der Spezifizierung von SPS-Datentypen.

Syntax

```
TYPE E_BA_DataType :
(
  Invalid        := 0,
  Undefined      := 1,

  eBool          := 10,
  eBit           := 11,
  eByte          := 12,
  eWord          := 13,
  eDWord         := 14,
  eLWord         := 15,
  eSInt          := 16,
  eInt           := 17,
  eDInt          := 18,
  eLInt          := 19,
  eUSInt         := 20,
  eUInt          := 21,
  eUDInt         := 22,
  eULInt         := 23,
  eReal          := 24,
  eLReal         := 25,
  eString        := 26,
  eWString       := 27,
  eTime          := 28,
  eDate          := 29,
  eDateTime      := 30,
```

```
  eTimeOfDay    := 31,

  eEnumeration  := 32,

  eStructure    := 33,
  eInterface    := 34,
) BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
Undefined	Nicht spezifiziert / nicht gesetzt.
eBool	Variable vom Datentyp BOOL.
eBit	Variable vom Datentyp BIT.
eByte	Variable vom Datentyp BYTE.
eWord	Variable vom Datentyp WORD.
eDWord	Variable vom Datentyp DWORD.
eLWord	Variable vom Datentyp LWORD.
eSInt	Variable vom Datentyp SINT.
eInt	Variable vom Datentyp INT.
eDInt	Variable vom Datentyp DINT.
eLInt	Variable vom Datentyp LINT.
eUSInt	Variable vom Datentyp USINT.
eUInt	Variable vom Datentyp UINT.
eUDInt	Variable vom Datentyp UDINT.
eULInt	Variable vom Datentyp ULINT.
eReal	Variable vom Datentyp REAL.
eLReal	Variable vom Datentyp LREAL.
eString	Variable vom Datentyp STRING.
eWString	Variable vom Datentyp WSTRING.
eTime	Variable vom Datentyp TIME
eDate	Variable vom Datentyp DATE
eDateTime	Variable vom Datentyp DATE_AND_TIME bzw. DT.
eTimeOfDay	Variable vom Datentyp TIME_OF_DAY bzw. TOD.
eEnumeration	Variable vom Datentyp ENUMERATION.
eStructure	Variable vom Datentyp STRUCT.
eInterface	Datentyp INTERFACE.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.4 Schedule

4.2.1.5.4.1 E_BA_SchedEntryState

Die Enumeration dient der Spezifizierung von Einträgen in Zeitschaltplänen.

Syntax

```
TYPE E_BA_SchedEntryState:
(
  Invalid        := 0,

  eUndefined     := 1,
  eValue         := 2,
```

```
  eNull          := 3
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eUndefined	Nicht spezifiziert.
eValue	Ein Wert ist gesetzt.
eNull	Nicht spezifiziert / nicht gesetzt.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.5 Trend

4.2.1.5.5.1 E_BA_LoggingType

Die Enumeration dient der Einordnung der Speicherung von Trenddaten.

Syntax

```
TYPE E_BA_LoggingType:
(
  Invalid        := -1,

  ePolled        := 0,
  eCOV           := 1,
  eTriggered     := 2
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
ePolled	Daten werden zyklisch abgerufen und gespeichert.
eCOV	Daten werden bei Änderung (plus Hysterese) gespeichert.
eTriggered	Daten werden nach Aufruf gespeichert.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.5.5.2 E_BA_TrendEntryType

Die Enumeration gibt den Typ des Trend-Eintrags an. Sie ist die Erweiterung des E_BA_DataClass [▶ 58].

Syntax

```
TYPE E_BA_TrendEntryType:
(
  Invalid        := E_BA_DataClass.Invalid,

  eBinary        := E_BA_DataClass.eBinary,
  eAnalog        := E_BA_DataClass.eAnalog,
  eMultistate    := E_BA_DataClass.eMultistate,

  eEvent         := 20
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eBinary	Eintrag mit einem binären Wert.
eAnalog	Eintrag mit einem analogen Wert.
eMultistate	Eintrag mit einem Multistate-Wert.
eEvent	Ereignis Eintrag.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.6 Units

4.2.1.6.1 E_BA_Unit

Die Enumeration dient der Spezifizierung von Einheiten.

Syntax

```
TYPE E_BA_Unit:
(
  Invalid                                 := -1,

  eArea_SquareMeters                      := 0,
  eArea_SquareFeet                        := 1,
  eElectrical_Milliamperes                := 2,
  eElectrical_Amperes                     := 3,
  eElectrical_Ohms                        := 4,
  eElectrical_Volts                       := 5,
  eElectrical_Kilovolts                   := 6,
  eElectrical_Megavolts                   := 7,
  eElectrical_VoltAmperes                 := 8,
  eElectrical_KilovoltAmperes             := 9,
  eElectrical_MegavoltAmperes             := 10,
  eElectrical_VoltAmperesReactive         := 11,
  eElectrical_KilovoltAmperesReactive     := 12,
  eElectrical_MegavoltAmperesReactive     := 13,
  eElectrical_DegreesPhase                := 14,
  eElectrical_PowerFactor                 := 15,
  eEnergy_Joules                          := 16,
  eEnergy_Kilojoules                      := 17,
  eEnergy_WattHours                       := 18,
  eEnergy_KilowattHours                   := 19,
  eEnergy_Btus                            := 20,
  eEnergy_Therms                          := 21,
  eEnergy_TonHours                        := 22,
  eEnthalpy_JoulesPerKilogramDryAir       := 23,
  eEnthalpy_BtusPerPoundDryAir            := 24,
  eFrequency_CyclesPerHour                := 25,
  eFrequency_CyclesPerMinute              := 26,
  eFrequency_Hertz                        := 27,
  eHumidity_GramsOfWaterPerKilogramDryAir := 28,
  eHumidity_PercentRelativeHumidity       := 29,
  eLength_Millimeters                     := 30,
  eLength_Meters                          := 31,
  eLength_Inches                          := 32,
  eLength_Feet                            := 33,
  eLight_WattsPerSquareFoot               := 34,
  eLight_WattsPerSquareMeter              := 35,
  eLight_Lumens                           := 36,
  eLight_Luxes                            := 37,
  eLight_FootCandles                      := 38,
  eMass_Kilograms                         := 39,
  eMass_PoundsMass                        := 40,
  eMass_Tons                              := 41,
  eMassFlow_KilogramsPerSecond            := 42,
  eMassFlow_KilogramsPerMinute            := 43,
  eMassFlow_KilogramsPerHour              := 44,
  eMassFlow_PoundsMassPerMinute           := 45,
  eMassFlow_PoundsMassPerHour             := 46,
```

```
ePower_Watts                                := 47,
ePower_Kilowatts                            := 48,
ePower_Megawatts                            := 49,
ePower_BtusPerHour                          := 50,
ePower_Horsepower                           := 51,
ePower_TonsRefrigeration                    := 52,
ePressure_Pascals                           := 53,
ePressure_Kilopascals                       := 54,
ePressure_Bars                              := 55,
ePressure_PoundsForcePerSquareInch          := 56,
ePressure_CentimetersOfWater                := 57,
ePressure_InchesOfWater                     := 58,
ePressure_MillimetersOfMercury              := 59,
ePressure_CentimetersOfMercury              := 60,
ePressure_InchesOfMercury                   := 61,
eTemperature_DegreesCelsius                 := 62,
eTemperature_DegreesKelvin                  := 63,
eTemperature_DegreesFahrenheit              := 64,
eTemperature_DegreeDaysCelsius              := 65,
eTemperature_DegreeDaysFahrenheit           := 66,
eTime_Years                                 := 67,
eTime_Months                                := 68,
eTime_Weeks                                 := 69,
eTime_Days                                  := 70,
eTime_Hours                                 := 71,
eTime_Minutes                               := 72,
eTime_Seconds                               := 73,
eVelocity_MetersPerSecond                   := 74,
eVelocity_KilometersPerHour                 := 75,
eVelocity_FeetPerSecond                     := 76,
eVelocity_FeetPerMinute                     := 77,
eVelocity_MilesPerHour                      := 78,
eVolume_CubicFeet                           := 79,
eVolume_CubicMeters                         := 80,
eVolume_ImperialGallons                     := 81,
eVolume_Liters                              := 82,
eVolume_UsGallons                           := 83,
eVolumetricFlow_CubicFeetPerMinute          := 84,
eVolumetricFlow_CubicMetersPerSecond        := 85,
eVolumetricFlow_ImperialGallonsPerMinute    := 86,
eVolumetricFlow_LitersPerSecond             := 87,
eVolumetricFlow_LitersPerMinute             := 88,
eVolumetricFlow_UsGallonsPerMinute          := 89,
eOther_DegreesAngular                       := 90,
eOther_DegreesCelsiusPerHour                := 91,
eOther_DegreesCelsiusPerMinute              := 92,
eOther_DegreesFahrenheitPerHour             := 93,
eOther_DegreesFahrenheitPerMinute           := 94,
eOther_NoUnits                              := 95,
eOther_PartsPerMillion                      := 96,
eOther_PartsPerBillion                      := 97,
eOther_Percent                              := 98,
eOther_PercentPerSecond                     := 99,
eOther_PerMinute                            := 100,
eOther_PerSecond                            := 101,
eOther_PsiPerDegreeFahrenheit               := 102,
eOther_Radians                              := 103,
eOther_RevolutionsPerMinute                 := 104,
eCurrency_Currency1                         := 105,
eCurrency_Currency2                         := 106,
eCurrency_Currency3                         := 107,
eCurrency_Currency4                         := 108,
eCurrency_Currency5                         := 109,
eCurrency_Currency6                         := 110,
eCurrency_Currency7                         := 111,
eCurrency_Currency8                         := 112,
eCurrency_Currency9                         := 113,
eCurrency_Currency10                        := 114,
eArea_SquareInches                          := 115,
eArea_SquareCentimeters                     := 116,
eEnthalpy_BtusPerPound                      := 117,
eLength_Centimeters                         := 118,
eMassFlow_PoundsMassPerSecond               := 119,
eTemperature_DeltaDegreesFahrenheit         := 120,
eTemperature_DeltaDegreesKelvin             := 121,
eElectrical_Kilohms                         := 122,
eElectrical_Megohms                         := 123,
eElectrical_Millivolts                      := 124,
eEnergy_KilojoulesPerKilogram               := 125,
eEnergy_Megajoules                          := 126,
```

```
eEntropy_JoulesPerDegreeKelvin             := 127,
eEntropy_JoulesPerKilogramDegreeKelvin     := 128,
eFrequency_Kilohertz                       := 129,
eFrequency_Megahertz                       := 130,
eFrequency_PerHour                         := 131,
ePower_Milliwatts                          := 132,
ePressure_Hectopascals                     := 133,
ePressure_Millibars                        := 134,
eVolumetricFlow_CubicMetersPerHour         := 135,
eVolumetricFlow_LitersPerHour              := 136,
eOther_KilowattHoursPerSquareMeter         := 137,
eOther_KilowattHoursPerSquareFoot          := 138,
eOther_MegajoulesPerSquareMeter            := 139,
eOther_MegajoulesPerSquareFoot             := 140,
eOther_WattsPerSquareMeterDegreeKelvin     := 141,
eVolumetricFlow_CubicFeetPerSecond         := 142,
eOther_PercentObscurationPerFoot           := 143,
eOther_PercentObscurationPerMeter          := 144,
eElectrical_Milliohms                      := 145,
eEnergy_MegawattHours                      := 146,
eEnergy_KiloBtus                           := 147,
eEnergy_MegaBtus                           := 148,
eEnthalpy_KilojoulesPerKilogramDryAir      := 149,
eEnthalpy_MegajoulesPerKilogramDryAir      := 150,
eEntropy_KilojoulesPerDegreeKelvin         := 151,
eEntropy_MegajoulesPerDegreeKelvin         := 152,
eForce_Newton                              := 153,
eMassFlow_GramsPerSecond                   := 154,
eMassFlow_GramsPerMinute                   := 155,
eMassFlow_TonsPerHour                      := 156,
ePower_KiloBtusPerHour                     := 157,
eTime_HundredthsSeconds                    := 158,
eTime_Milliseconds                         := 159,
eTorque_NewtonMeters                       := 160,
eVelocity_MillimetersPerSecond             := 161,
eVelocity_MillimetersPerMinute             := 162,
eVelocity_MetersPerMinute                  := 163,
eVelocity_MetersPerHour                    := 164,
eVolumetricFlow_CubicMetersPerMinute       := 165,
eAcceleration_MetersPerSecondPerSecond     := 166,
eElectrical_AmperesPerMeter                := 167,
eElectrical_AmperesPerSquareMeter          := 168,
eElectrical_AmpereSquareMeters             := 169,
eElectrical_Farads                         := 170,
eElectrical_Henrys                         := 171,
eElectrical_OhmMeters                      := 172,
eElectrical_Siemens                        := 173,
eElectrical_SiemensPerMeter                := 174,
eElectrical_Teslas                         := 175,
eElectrical_VoltsPerDegreeKelvin           := 176,
eElectrical_VoltsPerMeter                  := 177,
eElectrical_Webers                         := 178,
eLight_Candelas                            := 179,
eLight_CandelasPerSquareMeter              := 180,
eTemperature_DegreesKelvinPerHour          := 181,
eTemperature_DegreesKelvinPerMinute        := 182,
eOther_JouleSeconds                        := 183,
eOther_RadiansPerSecond                    := 184,
eOther_SquareMetersPerNewton               := 185,
eOther_KilogramsPerCubicMeter              := 186,
eOther_NewtonSeconds                       := 187,
eOther_NewtonsPerMeter                     := 188,
eOther_WattsPerMeterPerDegreeKelvin        := 189,
eMicro_Siemens                             := 190,
eCubic_FeetPerHour                         := 191,
eUs_GallonsPerHour                         := 192,
eKilometers                                := 193,
eMicrometers                               := 194,
eGrams                                     := 195,
eMilligrams                                := 196,
eMilliliters                               := 197,
eMillilitersPerSecond                      := 198,
eDecibels                                  := 199,
eDecibelsMillivolt                         := 200,
eDecibelsVolt                              := 201,
eMillisiemens                              := 202,
eWatt_HoursReactive                        := 203,
eKilowattHoursReactive                     := 204,
eMegawattHoursReactive                     := 205,
eMillimetersOfWater                        := 206,
```

```
  ePer_Mille                              := 207,
  eGrams_PerGram                          := 208,
  eKilograms_PerKilogram                  := 209,
  eGrams_PerKilogram                      := 210,
  eMilligrams_PeGram                      := 211,
  eMilligrams_PeKilogram                  := 212,
  eGrams_PerMilliliter                    := 213,
  eGrams_PerLiter                         := 214,
  eMilligrams_PerLiter                    := 215,
  eMicrograms_PerLiter                    := 216,
  eGrams_PerCubicMeter                    := 217,
  eMilligrams_PerCubicMeter               := 218,
  eMicrograms_PerCubicMeter               := 219,
  eNanograms_PerCubicMeter                := 220,
  eGrams_PerCubicCentimeter               := 221,
  eBecquerels                             := 222,
  eKilobecquerels                         := 223,
  eMegabecquerels                         := 224,
  eGray                                   := 225,
  eMilligray                              := 226,
  eMicrogray                              := 227,
  eSieverts                               := 228,
  eMillisieverts                          := 229,
  eMicrosieverts                          := 230,
  eMicrosievertsPerHour                   := 231,
  eDecibels_A                             := 232,
  eNephelometric_TurbidityUnit            := 233,
  ePH                                     := 234,
  eGrams_PerSquareMeter                   := 235,
  eMinutes_PerDegreeKelvin                := 236
)INT;
End_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eArea_SquareMeters	Quadratmeter
eArea_SquareFeet	Quadratfuß
eElectrical_Milliamperes	Milliampere
eElectrical_Amperes	Ampere
eElectrical_Ohms	Ohm
eElectrical_Volts	Volt
eElectrical_Kilovolts	Kilovolt
eElectrical_Megavolts	Megavolt
eElectrical_VoltAmperes	Voltampere
eElectrical_KilovoltAmperes	Kilovoltampere
eElectrical_MegavoltAmperes	Megavoltampere
eElectrical_VoltAmperesReactive	Voltampere-reaktiv
eElectrical_KilovoltAmperesReactive	Kilovoltampere-reaktiv
eElectrical_MegavoltAmperesReactive	Megavoltampere-reaktiv
eElectrical_DegreesPhase	Phasenlage in Grad
eElectrical_PowerFactor	Leistungsfaktor cos φ
eEnergy_Joules	Joule
eEnergy_Kilojoules	Kilojoule
eEnergy_WattHours	Wattstunden
eEnergy_KilowattHours	Kilowattstunden
eEnergy_Btus	BTU (British Thermal Unit)
eEnergy_Therms	Therm

Name	Beschreibung
eEnergy_TonHours	Tonnenstunden
eEnthalpy_JoulesPerKilogramDryAir	Joule pro Kg Trockenluft
eEnthalpy_BtusPerPoundDryAir	BTU pro Pfund Trockenluft
eFrequency_CyclesPerHour	Zyklen pro Stunde
eFrequency_CyclesPerMinute	Zyklen pro Minute
eFrequency_Hertz	Hertz
eHumidity_GramsOfWaterPerKilogramDryAir	Gramm Wasser pro Kilogramm Trockenluft
eHumidity_PercentRelativeHumidity	Prozent relative Feuchte
eLength_Millimeters	Millimeter
eLength_Meters	Meter
eLength_Inches	Zoll
eLength_Feet	Fuß
eLight_WattsPerSquareFoot	Watt pro Quadratfuß
eLight_WattsPerSquareMeter	Watt pro Quadratmeter
eLight_Lumens	Lumen
eLight_Luxes	Lux
eLight_FootCandles	Footcandle (angelsächsische Maßeinheit für die Beleuchtungsstärke)
eMass_Kilograms	Kilogramm
eMass_PoundsMass	Pfund
eMass_Tons	Tonnen
eMassFlow_KilogramsPerSecond	Kilogramm pro Sekunde
eMassFlow_KilogramsPerMinute	Kilogramm pro Minute
eMassFlow_KilogramsPerHour	Kilogramm pro Stunde
eMassFlow_PoundsMassPerMinute	Pfund pro Minute
eMassFlow_PoundsMassPerHour	Pfund pro Stunde
ePower_Watts	Watt
ePower_Kilowatts	Kilowatt
ePower_Megawatts	Megawatt
ePower_BtusPerHour	BTU pro Stunde
ePower_Horsepower	Pferdestärke
ePower_TonsRefrigeration	Kühlkapazität - Energie, um eine Tonne Eis in 24h schmelzen zu lassen.
ePressure_Pascals	Pascal
ePressure_Kilopascals	Kilo-Pascal
ePressure_Bars	Bar
ePressure_PoundsForcePerSquareInch	Pfund pro Quadratfuß
ePressure_CentimetersOfWater	Zentimeter Wassersäule

Name	Beschreibung
ePressure_InchesOfWater	Zoll Wassersäule
ePressure_MillimetersOfMercury	Millimeter Quecksilbersäule
ePressure_CentimetersOfMercury	Zentimeter Quecksilbersäule
ePressure_InchesOfMercury	Zoll Quecksilbersäule
eTemperature_DegreesCelsius	Grad Celsius
eTemperature_DegreesKelvin	Kelvin
eTemperature_DegreesFahrenheit	Grad Fahrenheit
eTemperature_DegreeDaysCelsius	Gradtage Celsius
eTemperature_DegreeDaysFahrenheit	Gradtage Fahrenheit
eTime_Years	Jahre
eTime_Months	Monate
eTime_Weeks	Wochen
eTime_Days	Tage
eTime_Hours	Stunden
eTime_Minutes	Minuten
eTime_Seconds	Sekunden
eVelocity_MetersPerSecond	Meter pro Sekunde
eVelocity_KilometersPerHour	Kilometer pro Stunde
eVelocity_FeetPerSecond	Fuß pro Sekunde
eVelocity_FeetPerMinute	Fuß pro Stunde
eVelocity_MilesPerHour	Meilen pro Stunde
eVolume_CubicFeet	Kubikfuß
eVolume_CubicMeters	Kubikmeter
eVolume_ImperialGallons	Britische Gallone
eVolume_Liters	Liter
eVolume_UsGallons	US-Gallone
eVolumetricFlow_CubicFeetPerMinute	Kubikfuß pro Minute
eVolumetricFlow_CubicMetersPerSecond	Kubikmeter pro Sekunde
eVolumetricFlow_ImperialGallonsPerMinute	Britische Gallone pro Minute
eVolumetricFlow_LitersPerSecond	Liter pro Sekunde
eVolumetricFlow_LitersPerMinute	Liter pro Minute
eVolumetricFlow_UsGallonsPerMinute	US-Gallone pro Minute
eOther_DegreesAngular	Gradmaß Raumwinkel
eOther_DegreesCelsiusPerHour	Grad Celsius pro Stunde

Name	Beschreibung
eOther_DegreesCelsiusPerMinute	Grad Celsius pro Minute
eOther_DegreesFahrenheitPerHour	Grad Fahrenheit pro Stunde
eOther_DegreesFahrenheitPerMinute	Grad Fahrenheit pro Minute
eOther_NoUnits	Einheitenlos
eOther_PartsPerMillion	Teile pro Million
eOther_PartsPerBillion	Teile pro Milliarde
eOther_Percent	Prozent
eOther_PercentPerSecond	Prozent pro Sekunde
eOther_PerMinute	Pro Minute
eOther_PerSecond	Pro Sekunde
eOther_PsiPerDegreeFahrenheit	Psi pro Grad Fahrenheit
eOther_Radians	Bogenmaß rad Winkel
eOther_RevolutionsPerMinute	Umdrehungen pro Minute
eCurrency_CurrencyN	Währung
eArea_SquareInches	Quadratzoll
eArea_SquareCentimeters	Quadratzentimeter
eEnthalpy_BtusPerPound	BTU pro Pfund
eLength_Centimeters	Zentimeter
eMassFlow_PoundsMassPerSecond	Pfund pro Sekunde
eTemperature_DeltaDegreesFahrenheit	Temperaturunterschied in Fahrenheit
eTemperature_DeltaDegreesKelvin	Temperaturunterschied in Kelvin
eElectrical_Kilohms	Kiloohm
eElectrical_Megohms	Megaohm
eElectrical_Millivolts	Millivolt
eEnergy_KilojoulesPerKilogram	Kilojoule
eEnergy_Megajoules	Megajoule
eEntropy_JoulesPerDegreeKelvin	Joule pro Kelvin
eEntropy_JoulesPerKilogramDegreeKelvin	Joule pro kg und Kelvin
eFrequency_Kilohertz	Kilohertz
eFrequency_Megahertz	Megahertz
eFrequency_PerHour	Pro Stunde
ePower_Milliwatts	Milliwatt
ePressure_Hectopascals	Hektopascal
ePressure_Millibars	Millibar
eVolumetricFlow_CubicMetersPerHour	Kubikmeter pro Stunde
eVolumetricFlow_LitersPerHour	Liter pro Stunde

Name	Beschreibung
eOther_KilowattHoursPerSquareMeter	Kilowattstunden pro Quadratmeter
eOther_KilowattHoursPerSquareFoot	Kilowattstunden pro Quadratfuß
eOther_MegajoulesPerSquareMeter	Megajoule pro Quadratmeter
eOther_MegajoulesPerSquareFoot	Megajoule pro Quadratfuß
eOther_WattsPerSquareMeterDegreeKelvin	Watt pro Quadratmeter und Kelvin
eVolumetricFlow_CubicFeetPerSecond	Kubikfuß pro Sekunde
eOther_PercentObscurationPerFoot	Prozent Verdunkelung pro Fuß
eOther_PercentObscurationPerMeter	Prozent Verdunkelung pro Meter
eElectrical_Milliohms	Milliohm
eEnergy_MegawattHours	Megawattstunden
eEnergy_KiloBtus	KiloBTU
eEnergy_MegaBtus	MegaBTU
eEnthalpy_KilojoulesPerKilogramDryAir	Kilojoule pro kg Trockenluft
eEnthalpy_MegajoulesPerKilogramDryAir	Megajoule pro kg Trockenluft
eEntropy_KilojoulesPerDegreeKelvin	Kilojoule pro Kelvin
eEntropy_MegajoulesPerDegreeKelvin	Megajoule pro Kelvin
eForce_Newton	Newton
eMassFlow_GramsPerSecond	Gramm pro Sekunde
eMassFlow_GramsPerMinute	Gramm pro Minute
eMassFlow_TonsPerHour	Tonnen pro Stunde
ePower_KiloBtusPerHour	Kilo-BTU pro Stunde
eTime_HundredthsSeconds	Hundertstelsekunden
eTime_Milliseconds	Millisekunden
eTorque_NewtonMeters	Newton-Meter
eVelocity_MillimetersPerSecond	Millimeter pro Sekunde
eVelocity_MillimetersPerMinute	Millimeter pro Minute
eVelocity_MetersPerMinute	Meter pro Minute
eVelocity_MetersPerHour	Meter pro Stunde
eVolumetricFlow_CubicMetersPerMinute	Kubikmeter pro Minute
eAcceleration_MetersPerSecondPerSecond	Meter pro Quadratsekunde
eElectrical_AmperesPerMeter	Ampere pro Meter

Name	Beschreibung
eElectrical_AmperesPerSquareMeter	Ampere pro Quadratmeter
eElectrical_AmpereSquareMeters	Ampere-Quadratmeter
eElectrical_Farads	Farad
eElectrical_Henrys	Henry
eElectrical_OhmMeters	Ohmmeter
eElectrical_Siemens	Siemens
eElectrical_SiemensPerMeter	Siemens pro Meter
eElectrical_Teslas	Tesla
eElectrical_VoltsPerDegreeKelvin	Volt pro Kelvin
eElectrical_VoltsPerMeter	Volt pro Meter
eElectrical_Webers	Weber
eLight_Candelas	Candela
eLight_CandelasPerSquareMeter	Candela pro Quadratmeter
eTemperature_DegreesKelvinPerHour	Kelvin pro Stunde
eTemperature_DegreesKelvinPerMinute	Kelvin pro Minute
eOther_JouleSeconds	Joule-Sekunde (Drehimpuls)
eOther_RadiansPerSecond	Radiant pro Sekunde
eOther_SquareMetersPerNewton	Quadratmeter pro Newton
eOther_KilogramsPerCubicMeter	Kilogramm pro Kubikmeter
eOther_NewtonSeconds	Newton-Sekunde (Impuls)
eOther_NewtonsPerMeter	Newton pro Meter
eOther_WattsPerMeterPerDegreeKelvin	Watt pro Meter und Kelvin
eMicro_Siemens	Mikrosiemens
eCubic_FeetPerHour	Kubikfuß pro Stunde
eUs_GallonsPerHour	US-Gallonen pro Stunde
eKilometers	Kilometer
eMicrometers	Mikrometer
eGrams	Gramm
eMilligrams	Milligramm
eMilliliters	Milliliter
eMillilitersPerSecond	Milliliter pro Sekunde
eDecibels	Dezibel
eDecibelsMillivolt	Dezibel Millivolt
eDecibelsVolt	Dezibel Volt
eMillisiemens	Millisiemens
eWatt_HoursReactive	Wattstunden-reaktiv
eKilowattHoursReactive	Kilowattstunden-reaktiv
eMegawattHoursReactive	Megawattstunden-reaktiv
eMillimetersOfWater	Millimeter Wassersäule
ePer_Mille	Promille

Name	Beschreibung
eGrams_PerGram	Gramm pro Gramm
eKilograms_PerKilogram	Kilogramm pro Kilogramm
eGrams_PerKilogram	Gramm pro Kilogramm
eMilligrams_PeGram	Milligramm pro Gramm
eMilligrams_PeKilogram	Milligramm pro Kilogramm
eGrams_PerMilliliter	Gramm pro Milliliter
eGrams_PerLiter	Gramm pro Liter
eMilligrams_PerLiter	Milligramm pro Liter
eMicrograms_PerLiter	Mikrogramm pro Liter
eGrams_PerCubicMeter	Gramm pro Kubikmeter
eMilligrams_PerCubicMeter	Milligramm pro Kubikmeter
eMicrograms_PerCubicMeter	Mikrogramm pro Kubikmeter
eNanograms_PerCubicMeter	Nanogramm pro Kubikmeter
eGrams_PerCubicCentimeter	Gramm pro Kubikzentimeter
eBecquerels	Becquerel
eKilobecquerels	Kilobecquerel
eMegabecquerels	Megabecquerel
eGray	Gray (Energiedosis)
eMilligray	Milligray (Energiedosis)
eMicrogray	Mikrogray (Energiedosis)
eSieverts	Sievert
eMillisieverts	Millisievert
eMicrosieverts	Mikrosievert
eMicrosievertsPerHour	Mikrosievert pro Stunde
eDecibels_A	Dezibel (a)
eNephelometric_TurbidityUnit	Nephelometrischer Trübungswert NTU
ePH	pH-Wert
eGrams_PerSquareMeter	Gramm pro Quadratmeter
eMinutes_PerDegreeKelvin	Minuten pro Kelvin

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.7 Universal

4.2.1.7.1 E_BA_Action

Auswahl des Wirksinns eines Reglers.

Syntax

```
TYPE E_BA_Action:
(
  Invalid        := -1,
  eDirect        := 0,
```

```
  eReverse       := 1
)INT;
End_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender
eDirect	Direkter Wirksinn (Kühlen)
eReverse	Indirekter Wirksinn (Heizen)

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.7.2 E_BA_Language

Die Enumeration dient der Spezifizierung von Sprachen.

Syntax

```
TYPE E_BA_Language:
(
  Invalid     := 0,

  eEnglish    := 1,
  eGerman     := 2,
);
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender
eEnglish	Englisch
eGerman	Deutsch

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.7.3 E_BA_MeasuringElement

Die Enumeration dient der Spezifizierung von Temperatur-Messelementen.

Syntax

```
TYPE E_BA_MeasuringElement :
(
  Undefined            := 0,

  eNI100               := 1,
  eNI120               := 2,
  eNI1000              := 3,
  eNI1000_LS           := 4,
  eNTC1K8              := 5,
  eNTC1K8_TK           := 6,
  eNTC2K2              := 7,
  eNTC3K               := 8,
  eNTC5K               := 9,
  eNTC10K              := 10,
  eNTC10KPRE           := 11,
  eNTC10K_3204         := 12,
  eNTC10KTYP2          := 13,
  eNTC10KTYP3          := 14,
  eNTC10KDALE          := 15,
  eNTC10K3A221         := 16,
  eNTC20K              := 17,
  eNTC100K             := 18,
```

```
  ePoti_Resolution_01  := 19,
  ePoti_Resolution_1   := 20,
  eOutput_10_5000      := 21,
  eOutput_10_1200      := 22,
  ePT100               := 23,
  ePT200               := 24,
  ePT500               := 25,
  ePT1000              := 26,
);
END_TYPE
```

Name	Beschreibung
Undefined	Nicht definiert
eNI100	NI100
eNI120	NI120
eNI1000	NI1000
eNI1000_LS	RSNI1000 (NI1000 nach Landis&Staefa-Charakteristik: 1000 Ω bei 0 °C und 1500 Ω bei 100 °C)
eNTC1K8	NTC1K8
eNTC1K8_TK	NTC1K8_TK
eNTC2K2	NTC2K2
eNTC3K	NTC3K
eNTC5K	NTC5K
eNTC10K	NTC10K
eNTC10KPRE	NTC10KPRE
eNTC10K_3204	NTC10K_3204
eNTC10KTYP2	NTC10KTYP2
eNTC10KTYP3	NTC10KTYP3
eNTC10KDALE	NTC10KDALE
eNTC10K3A221	NTC10K3A221
eNTC20K	NTC20K
eNTC100K	NTC100K
ePoti_Resolution_01	Poti, Auflösung 0,1 Ω
ePoti_Resolution_1	Poti, Auflösung 1 Ω
E_Output_10_5000	Ausgabe 10,0 Ω - 5000,0 Ω
E_Output_10_1200	Ausgabe 10,0 Ω - 1200,0 Ω
ePT100	PT100
ePT200	PT200
ePT500	PT500
ePT1000	PT1000

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.7.4 E_BA_Polarity

Die Enumeration dient der Spezifizierung der Polarität (z.B. Öffner / Schließer).

Syntax

```
TYPE E_BA_Polarity:
(
  Invalid        := -1,

  eNormal        := 0,
```

```
  eReverse       := 1
);
End_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender
eNormal	Schließer
eReverse	Öffner

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.7.5 E_BA_Reliability

Die Enumeration gibt Rückschlüsse auf die Zuverlässigkeit von Sensoren oder Messdaten.

Syntax

```
TYPE E_BA_Reliability:
(
  Invalid                := 0,

  eNoFaultDetected       := 1,
  eNoSensor              := 1,
  eOverRange             := 2,
  eUnderRange            := 3,
  eOpenLoop              := 4,
  eShortedLoop           := 5,
  eNoOutput              := 6,
  eUnreliableOther       := 7,
  eProcessError          := 8,
  eMultiStateFault       := 9,
  eConfigurationError    := 10,
  eCommunicationFailure  := 12,
  eMemberFault           := 13
);
END_TYPE
```

Name	Beschreibung
Invalid	Keine Bedeutung für den Anwender.
eNoFaultDetected	NO_FAULT_DETECTED Es wurde kein Fehler, der in dieser Enumeration beschrieben ist, erkannt.
eNoSensor	NO_SENSOR Es ist kein Sensor mit dem Eingangsobjekt verbunden.
eOverRange	OVER_RANGE Der gemessene Wert liegt oberhalb des normalen Messbereichs.
eUnderRange	UNDER_RANGE Der gemessene Wert liegt unterhalb des normalen Messbereichs.
eOpenLoop	OPEN_LOOP Es wird ein Wert erkannt, der auf einen Drahtbruch schließen lässt.
eShortedLoop	SHORTED_LOOP Es wird ein Wert erkannt, der auf einen Kurzschluss schließen lässt.
eNoOutput	NO_OUTPUT Es ist kein Ausgabegerät mit dem Ausgangsobjekt verbunden.
eUnreliableOther	Intern: anderer Plausibilitätsfehler.
eProcessError	PROCESS_ERROR Ein Prozessfehler ist aufgetreten.
eMultiStateFault	MULTI_STATE_FAULT

Name	Beschreibung
	Der FAULT_STATE, FAULT_LIFE_SAFETY oder FAULT_CHARACTERSTRING Fehleralgorithmus hat einen Fehlerzustand ermittelt.
eConfigurationError	CONFIGURATION_ERROR Die Properties des Objektes sind nicht in einem konsistenten Zustand.
eCommunicationFailure	COMMUNICATION_FAILURE Kommunikationsfehler
eMemberFault	Fehler Teilnehmer

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.1.7.6 E_BA_ToggleMode

Mit der Enumeration wird die Funktion einer booleschen Änderung interpretiert.

Syntax

```
TYPE E_BA_ToggleMode:
(
  Invalid        := 0,

  eSwitch        := 1,
  ePushButton    := 2
)BYTE;
END_TYPE
```

Name	Beschreibung
Invalid	Ungültig, hat keine Bedeutung.
eSwitch	Der Ausgabewert behält seinen Wert bei Änderung. Das Objekt hat die Funktion eines Schalters angenommen.
ePushButton	Der Ausgabewert ändert seinen Wert für einen Zyklus und nimmt anschließend automatisch den alten Zustand wieder an. Das Objekt hat die Funktion eines Tasters angenommen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2 Typen

4.2.2.1 Kalender

4.2.2.1.1 ST_BA_CalendarEntry

Struktur zur Angabe eines Kalendereintrages.

Syntax

```
TYPE ST_BA_CalendarEntry :
STRUCT
  eType             : E_BA_DateValChoice;
  uDate             : U_BA_DateVal;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
eType	E_BA_DateValChoice [▶ 54]	Angabe Bereichsauswahl
uDate	U_BA_DateVal [▶ 77]	Angabe des Zeitraumes

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.2 Datum und Zeit

4.2.2.2.1 ST_BA_Date

Struktur für die Beschreibung eines Datums.

Syntax

```
TYPE ST_BA_Date :
STRUCT
  nYear          : BYTE          := 16#FF;
  eMonth         : E_BA_Month    := E_BA_Month.Unspecified;
  nDay           : E_BA_Day      := E_BA_Day.Unspecified;
  eDayOfWeek     : E_BA_Weekday := E_BA_Weekday.Unspecified;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
nYear	BYTE	Jahresangabe, bei der ab dem Jahr 1900 gezählt wird.
eMonth	E_BA_Month [▶ 56]	Monatsangabe
nDay	E_BA_Day [▶ 55]	Angabe des Tages im Monat
eDayOfWeek	E_BA_Weekday [▶ 57]	Angabe des Wochentages

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.2.2 ST_BA_DateRange

Struktur zur Beschreibung eines Datumsbereiches.

Syntax

```
TYPE ST_BA_DateRange :
STRUCT
  stDateFrom        : ST_BA_Date;
  stDateTo          : ST_BA_Date;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
stDateFrom	ST_BA_Date [▶ 75]	Beginn eines Zeitraums
stDateTo	ST_BA_Date [▶ 75]	Ende eines Zeitraums

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.2.3 ST_BA_DateTime

Struktur zur Beschreibung eines Datums inklusive Zeitangabe.

Syntax

```
TYPE ST_BA_DateTime :
STRUCT
  stDate            : ST_BA_Date;
  stTime            : ST_BA_Time;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
stDate	ST_BA_Date [▶ 75]	Angabe eines Datums
stTime	ST_BA_Time [▶ 76]	Angabe der Tageszeit

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.2.4 ST_BA_Time

Struktur zur Angabe der Tageszeit.

Syntax

```
TYPE ST_BA_Time :
STRUCT
  nHour               : BYTE := 16#FF;
  nMinute             : BYTE := 16#FF;
  nSecond             : BYTE := 16#FF;
  nHundredths         : BYTE := 16#FF;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
nHour	BYTE	Angabe der Stunde
nMinute	BYTE	Angabe der Minute
nSecond	BYTE	Angabe der Sekunde
nHundredths	BYTE	Angabe der Hundertstel-Sekunde

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.2.5 ST_BA_WeekNDay

Struktur zur Angabe eines Wochentages.

Syntax

```
TYPE ST_BA_WeekNDay :
STRUCT
  eMonth          : E_BA_Month      := E_BA_Month.Unspecified;
  eWeekOfMonth    : E_BA_Week       := E_BA_Week.Unspecified;
```

```
  eWeekday        : E_BA_Weekday    := E_BA_Weekday.Unspecified;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
eMonth	E_BA_Month [▶ 56]	Angabe des Monats
eWeekOfMonth	E_BA_Week [▶ 56]	Angabe des Woche innerhalb eines Monats
eWeekday	E_BA_Weekday [▶ 57]	Angabe des Tages innerhalb einer Woche

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.2.6 U_BA_DateVal

Der Datentyp UNION bildet einen Datumswert ab, welcher verschiedene Ausprägungen haben kann (*Date*, *Range*, *WeekNDay*).

Der Typ des Werts wird mittels Choice-Enum (E_BA_DateValChoice [▶ 54]) im Kontext (z.B. ST_BA_CalendarEntry [▶ 74] oder ST_BA_ClassValue [▶ 81]) spezifiziert.

Alle anderen Elemente beginnen an derselben Adresse im Speicherbereich und werden mitbeschrieben. Deren Inhalt jedoch ist dann in der Regel nicht sinnvoll.

Syntax

```
TYPE U_BA_DateVal :
UNION
  stDate          : ST_BA_Date;
  stDateRange     : ST_BA_DateRange;
  stWeekDay       : ST_BA_WeekDay;
END_UNION
END_TYPE
```

Name	Typ	Beschreibung
stDate	ST_BA_Date [▶ 75]	Eingabe als Datum
stDateRange	ST_BA_DateRange [▶ 75]	Eingabe als Datumsbereich
stWeekDay	ST_BA_WeekNDay [▶ 76]	Eingabe als Wochentag

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.3 Event

4.2.2.3.1 ST_BA_EventTransitions

Struktur zur Angabe eines Zustandswechsels von Events.

Syntax

```
TYPE ST_BA_EventTransitions :
STRUCT
  bToOffNormal    : BOOL;
  bToFault        : BOOL;
```

```
  bToNormal       : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bToOffNormal	BOOL	Wechsel in den Alarmzustand.
bToFault	BOOL	Wechsel in den Zustand „unzulässiger Wert“.
bToNormal	BOOL	Wechsel in den Normalzustand.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.4 Schedule

4.2.2.4.1 ST_BA_SchedEntry

Struktur zur Angabe eines Zeitschaltplan-Eintrages.

Syntax

```
TYPE ST_BA_SchedEntry :
STRUCT
  eState          : E_BA_SchedEntryState;
  stTime          : ST_BA_Time;
  uValue          : U_BA_ClassValue;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
eState	E_BA_SchedEntryState [▶ 59]	Angabe des Eingabestatus
stTime	ST_BA_Time [▶ 76]	Angabe der Tageszeit
uValue	U_BA_ClassValue [▶ 82]	Angabe des Schaltwertes

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.5 ST_BA_Byte

Struktur zur Darstellung von Bits in einem Byte.

Syntax

```
TYPE ST_BA_Byte :
STRUCT
  bBit1    : BIT;
  bBit2    : BIT;
  bBit3    : BIT;
  bBit4    : BIT;
  bBit5    : BIT;
  bBit6    : BIT;
  bBit7    : BIT;
  bBit8    : BIT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bBit1…bBit8	BIT	Darstellung der einzelnen Bits.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.6 ST_BA_EnumInfo

Um Enumerationen auch textuell näher zu beschreiben, kann man jedem Wert einer Enumeration in einer globalen Liste ein Element der Struktur *ST_BA_EnumInfo* zuweisen. Dieses geschieht beispielsweise in der globalen Variablenliste BAComn_EnumDE [▶ 83].

Der Wert, den die Enumeration inne hat verweist dabei auf das Element der Liste, welches das Element der Enumeration näher beschreibt.

Syntax

```
TYPE ST_BA_EnumInfo :
STRUCT
  sName          : STRING;
  sDescription   : T_MaxString;
  sShortcut      : STRING(16);
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
sName	STRING	Name
sDescription	T_MaxString	Beschreibung
sShortcut	STRING(16)	Abkürzung

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.7 ST_BA_StatusFlags

Struktur über die möglichen Betriebsstatus eines Objektes.

Syntax

```
TYPE ST_BA_StatusFlags :
STRUCT
  bInAlarm         : BOOL;
  bFault           : BOOL;
  bOverridden      : BOOL;
  bOutOfService    : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bInAlarm	BOOL	Zeigt einen Alarmzustand an (offnormal).
bFault	BOOL	Zeigt ein Zuverlässigkeitsproblem an (fault).
bOverridden	BOOL	Zeigt das manuelle Überschreiben an.
bOutOfService	BOOL	Zeigt den out-of-service-Modus an.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.8 ST_BA_Version

Struktur zur Angabe der Versionsnummer, siehe dazu Versionierung.

Syntax

```
TYPE ST_BA_Version : ARRAY [1 .. 4] OF UDINT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.9 Trend

4.2.2.9.1 ST_BA_TrendEntry

Struktur zur Beschreibung eines Trendeintrages.

Syntax

```
TYPE ST_BA_TrendEntry :
STRUCT
  dtTime    : ST_BA_DateTime;
  eType     : E_BA_TrendEntryType     := E_BA_TrendEntryType.Invalid;
  stState   : ST_BA_StatusFlags;
  uValue    : U_BA_TrendEntryValue;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
dtTime	ST_BA_DateTime [▶ 76]	Angabe der Zeit
eType	E_BA_TrendEntryType [▶ 60]	Angabe des Trendaufzeichnungstyps: Binär-, Analog- oder Multistatewerte.
stState	ST_BA_StatusFlags [▶ 79]	Mögliche Status des Trends: Alarm, fehlerhafter Wert, übersteuert, abgekoppelt.
uValue	U_BA_TrendEntryValue [▶ 81]	Angabe des Eintrags-Wertes inklusive des Status der Trend-Aufzeichnung: Start, Stop, bereinigt, unterbrochen.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.9.2 ST_BA_TrendEntryEvent

Struktur für die Beschreibung des Status der Trend-Aufzeichnung.

Syntax

```
TYPE ST_BA_TrendEntryEvent :
STRUCT
  bStart            : BIT;
  bStop             : BIT;
  bBufferPurged     : BIT;
  bInterrupted      : BIT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
bStart	BIT	Gestartet
bStop	BIT	Gestoppt
bBufferPurged	BIT	Aufzeichnung bereinigt
bInterrupted	BIT	Aufzeichnung unterbrochen

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.9.3 U_BA_TrendEntryValue

Struktur für die Wertangabe von Trendeinträgen.

Der Datentyp UNION erlaubt hierbei die sinnvolle Beschreibung eines der beinhaltenden Elemente.

Alle anderen Elemente beginnen an derselben Adresse im Speicherbereich und werden mitbeschrieben. Deren Inhalt jedoch ist dann in der Regel nicht sinnvoll.

Syntax

```
TYPE U_BA_TrendEntryValue EXTENDS U_BA_ClassValue:
UNION
  stEvent   : ST_BA_TrendEntryEvent;
END_UNION
END_TYPE
```

Name	Typ	Beschreibung
stEvent	ST_BA_TrendEntryEvent [▶ 80]	Angabe des Status der Trend-Aufzeichnung.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.2.2.9.4 ST_BA_ClassValue

Struktur für die Angabe eines Objektwertes.

```
TYPE ST_BA_ClassValue :
STRUCT
  uValue   : U_BA_ClassValue;
  eClass   : E_BA_DataClass;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
uValue	U_BA_ClassValue [▶ 58]	Angabe des Wertes.
eClass	E_BA_DataClass [▶ 58]	Angabe der Werteart

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

U_BA_ClassValue [▶ 82]

4.2.2.9.5 U_BA_ClassValue

Struktur für die Eingabe von Werten.

Der Datentyp UNION erlaubt hierbei die sinnvolle Beschreibung eines der beinhaltenden Elemente.

Alle anderen Elemente beginnen an derselben Adresse im Speicherbereich und werden somit mitbeschrieben.

Deren Inhalt jedoch ist dann in der Regel nicht sinnvoll.

Syntax

```
TYPE U_BA_ClassValue :
UNION
  bVal      : BOOL;
  rVal      : REAL;
  udiVal    : UDINT;
END_UNION
END_TYPE
```

Name	Typ	Beschreibung
bVal	BOOL	Binärwerte.
rVal	REAL	Analoge Fließkommawerte.
udiVal	UDINT	Multistatewert als UDINT.

Voraussetzungen

Entwicklungsumgebung	Erforderliche SPS-Bibliothek
TwinCAT3.1 4024.35	Tc2_BA2_Common ab V2.1.20.0

4.3 GVLs

4.3.1 BAComn_Global

```
{attribute 'global_init_slot' := '49800'}
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
// Datatype Ranges:
 {warning disable C0196}
     nMinByte                : BYTE   := 16#00;
     nMaxByte                : BYTE   := 16#FF;
     nMinInt                 : INT    := 16#8000;
     nMaxInt                 : INT    := 16#7FFF;
     nMinUInt                : UINT   := 16#0000;
     nMaxUInt                : UINT   := 16#FFFF;
     nMinDInt                : DINT   := 16#80000000;
     nMaxDInt                : DINT   := 16#7FFFFFFF;
     nMinUDInt               : UDINT  := 16#00000000;
     nMaxUDInt               : UDINT  := 16#FFFFFFFF;
     fMinReal                : REAL   := -3.402823E+38;
     fMaxReal                : REAL   := 3.402823E+38;
     tMinTime                : TIME   := TO_TIME(0);
     tMaxTime                : TIME   := TO_TIME(16#FFFFFFFF);
     tMinTOD                 : TOD    := TO_TOD(0);
     tMaxTOD                 : TOD    := TO_TOD(16#FFFFFFFF);
     tMinDATE                : DATE   := TO_DATE(0);
     tMaxDATE                : DATE   := TO_DATE(16#FFFFFFFF);
     tMinDT                  : DT     := TO_DT(0);
     tMaxDT                  : DT     := TO_DT(16#FFFFFFFF);
 {warning restore C0196}

// I/O:
     nIO_RawMin              : INT     := 0;
     nIO_RawMax              : INT     := nMaxInt;
     nIO_Raw0V               : INT     := 0;                  // Raw value for 0V
     nIO_Raw1V               : INT     := (nIO_RawMax / 10); // Raw value for 1V
     nIO_Raw2V               : INT     := (nIO_Raw1V * 2);    // Raw value for 2V
     nIO_Raw3V               : INT     := (nIO_Raw1V * 3);    // Raw value for 3V
     nIO_Raw5V               : INT     := (nIO_Raw1V * 5);    // Raw value for 5V
```

```
    nIO_Raw10V              : INT     := (nIO_Raw1V * 10);  // Raw value for 10V
END_VAR

// General:
VAR_GLOBAL CONSTANT
 {region 'Time'}
    nMilli2Sek              : UINT    := 1000;
    nSek2Min                : UINT    := 60;
    nMin2Hour               : UINT    := 60;

    n24Hour2Hour            : UDINT   := (24 * 60 * 60);
    n24Hour2Milli           : UDINT   := n24Hour2Hour * 1000;

    udiMaxSecInMilli        : UDINT   := (nMaxUDInt / nMilli2Sek);     // Max. capable value (in [s]
) in a UDINT
    udiMaxMinInMilli        : UDINT   := (udiMaxSecInMilli / nSek2Min);// Max. capable value (in [m]
) in a UDINT
 {endregion}
 {region 'Characters'}
    bChar_0                 : BYTE    := 16#30;
    bChar_1                 : BYTE    := 16#31;
    bChar_2                 : BYTE    := 16#32;
    bChar_3                 : BYTE    := 16#33;
    bChar_4                 : BYTE    := 16#34;
    bChar_5                 : BYTE    := 16#35;
    bChar_6                 : BYTE    := 16#36;
    bChar_7                 : BYTE    := 16#37;
    bChar_8                 : BYTE    := 16#38;
    bChar_9                 : BYTE    := 16#39;
    bChar_Plus              : BYTE    := 16#2B;
    bChar_Minus             : BYTE    := 16#2D;
    bChar_Dot               : BYTE    := 16#2E;
 {endregion}
 {region 'Type'}
    fCloseToZero            : REAL   := 0.00001;   // Comparison value to prevent a division by zero
 {endregion}
 {region 'ADS'}
    tAmsNetID_Loopback      : T_AmsNetIdArr   := [ 127,0,0,1,1,1 ];
    sSymbolSeparator        : STRING(1)       := '.';
 {endregion}
END_VAR
```

4.3.2 BAComn_Param

```
{attribute 'qualified_only'}
VAR_GLOBAL CONSTANT
  {region 'Tokenizer'}
    nStrTokenizer_BufferSize  : UINT    := 32;
    nStrTokenizer_MaxLevel    : BYTE    := 5;
  {endregion}
END_VAR
```

Name	Typ	Beschreibung
nStrTokenizer_BufferSize	UINT	Anzahl der Einträge.
nStrTokenizer_MaxLevel	BYTE	Maximale Tiefe der Token-Hierarchie.

4.3.3 Enumerationen

4.3.3.1 BAComn_EnumDE

```
{attribute 'qualified_only'}
VAR_GLOBAL

aUnits    : ARRAY[E_BA_Unit.First .. E_BA_Unit.Last] OF ST_BA_EnumInfo := [

  (*  eArea_SquareMeters                  *) (sName := 'Square Meters',                       sDesc
ription := 'Fläche',                                    sShortcut := 'm²'),
  (*  eArea_SquareFeet                    *) (sName := 'Square Feet',                         sDesc
ription := '',                                          sShortcut := ''),
  (*  eElectrical_Milliamperes            *) (sName := 'Milliamperes',                        sDesc
ription := 'Strom',                                     sShortcut := 'mA'),
```

```
  (*  eElectrical_Amperes                 *) (sName := 'Amperes',                                sDesc
ription := 'Strom',                                      sShortcut := 'A'),
  (*  eElectrical_Ohms                    *) (sName := 'Ohms',                                   sDesc
ription := 'Elektrischer Widerstand',                    sShortcut := 'O'),
  (*  eElectrical_Volts                   *) (sName := 'Volts',                                  sDesc
ription := 'Elektrische Spannung',                       sShortcut := 'V'),
  (*  eElectrical_Kilovolts               *) (sName := 'Kilovolts',                              sDesc
ription := 'Elektrische Spannung',                       sShortcut := 'kV'),
  (*  eElectrical_Megavolts               *) (sName := 'Megavolts',                              sDesc
ription := 'Elektrische Spannung',                       sShortcut := 'MV'),
  (*  eElectrical_VoltAmperes             *) (sName := 'Volt Amperes',                           sDesc
ription := 'Elektrische Scheinleistung',                 sShortcut := 'VA'),
  (*  eElectrical_KilovoltAmperes         *) (sName := 'Kilovolt Amperes',                       sDesc
ription := 'Elektrische Scheinleistung',                 sShortcut := 'kVA'),
  (*  eElectrical_MegavoltAmperes         *) (sName := 'Megavolt Amperes',                       sDesc
ription := 'Elektrische Scheinleistung',                 sShortcut := 'MVA'),
  (*  eElectrical_VoltAmperesReactive     *) (sName := 'Volt Amperes Reactive',                  sDesc
ription := 'Elektrische Blindleistung',                  sShortcut := 'var'),
  (*  eElectrical_KilovoltAmperesReactive *) (sName := 'Kilovolt Amperes Reactive',              sDesc
ription := 'Elektrische Blindleistung',                  sShortcut := 'kvar'),
  (*  eElectrical_MegavoltAmperesReactive *) (sName := 'Megavolt Amperes Reactive',              sDesc
ription := 'Elektrische Blindleistung',                  sShortcut := 'Mvar'),
  (*  eElectrical_DegreesPhase            *) (sName := 'Degrees Phase',                          sDesc
ription := '',                                           sShortcut := ''),
  (*  eElectrical_PowerFactor             *) (sName := 'Power Factor',                           sDesc
ription := 'phi Leistungsfaktor',                        sShortcut := 'cos'),
  (*  eEnergy_Joules                      *) (sName := 'Joules',                                 sDesc
ription := 'Energie',                                    sShortcut := 'J'),
  (*  eEnergy_Kilojoules                  *) (sName := 'Kilojoules',                             sDesc
ription := 'Energie',                                    sShortcut := 'kJ'),
  (*  eEnergy_WattHours                   *) (sName := 'Watt Hours',                             sDesc
ription := 'Energie',                                    sShortcut := 'Wh'),
  (*  eEnergy_KilowattHours               *) (sName := 'Kilowatt Hours',                         sDesc
ription := 'Energie',                                    sShortcut := 'kWh'),
  (*  eEnergy_Btus                        *) (sName := 'Btus',                                   sDesc
ription := '',                                           sShortcut := 'btus'),
  (*  eEnergy_Therms                      *) (sName := 'Therms',                                 sDesc
ription := '',                                           sShortcut := ''),
  (*  eEnergy_TonHours                    *) (sName := 'Ton Hours',                              sDesc
ription := '',                                           sShortcut := ''),
  (*  eEnthalpy_JoulesPerKilogramDryAir   *) (sName := 'Joules per Kilogram Dry Air',            sDesc
ription := 'Enthalpie',                                  sShortcut := 'J/kg'),
  (*  eEnthalpy_BtusPerPoundDryAir        *) (sName := 'Btus per Pound Dry Air',                 sDesc
ription := '',                                           sShortcut := ''),
  (*  eFrequency_CyclesPerHour            *) (sName := 'Cycles per Hour',                        sDesc
ription := 'Schalthäufigkeit (Zyklen pro Stunde)',       sShortcut := '1/h'),
  (*  eFrequency_CyclesPerMinute          *) (sName := 'Cycles per Minute',                      sDesc
ription := 'Schalthäufigkeit (Zyklen pro Minute)',       sShortcut := '1/min'),
  (*  eFrequency_Hertz                    *) (sName := 'Hertz',                                  sDesc
ription := 'Frequenz',                                   sShortcut := 'Hz'),
  (*  eHumidity_GramsOfWaterPerKilogramDryAir*) (sName := 'Grams of Water per Kilogram Dry Air', sDe
scription := 'Wassergehalt',                               sShortcut := 'g/kg tr. Luft'),
  (*  eHumidity_PercentRelativeHumidity   *) (sName := 'Percent Relative Humidity',              sDesc
ription := 'Relative Luftfeuchtigkeit',                  sShortcut := '% r. F.'),
  (*  eLength_Millimeters                  *) (sName := 'Millimeters',                            sDesc
ription := 'Länge',                                      sShortcut := 'mm'),
  (*  eLength_Meters                      *) (sName := 'Meters',                                 sDesc
ription := 'Länge',                                      sShortcut := 'm'),
  (*  eLength_Inches                      *) (sName := 'Inches',                                 sDesc
ription := 'Zoll',                                       sShortcut := ''),
  (*  eLength_Feet                        *) (sName := 'Feet',                                   sDesc
ription := 'Fuss',                                       sShortcut := ''),
  (*  eLight_WattsPerSquareFoot           *) (sName := 'Watts per Square Foot',                  sDesc
ription := '',                                           sShortcut := ''),
  (*  eLight_WattsPerSquareMeter          *) (sName := 'Watts per Square Meter',                 sDesc
ription := 'Spezifische Leistung',                       sShortcut := 'W/m²'),
  (*  eLight_Lumens                       *) (sName := 'Lumens',                                 sDesc
ription := 'Lichtstrom',                                 sShortcut := 'lm'),
  (*  eLight_Luxes                        *) (sName := 'Luxes',                                  sDesc
ription := 'Beleuchtungsstärke',                         sShortcut := 'lx'),
  (*  eLight_FootCandles                  *) (sName := 'Foot Candles',                           sDesc
ription := '',                                           sShortcut := ''),
  (*  eMass_Kilograms                     *) (sName := 'Kilograms',                              sDesc
ription := 'Masse',                                      sShortcut := 'kg'),
  (*  eMass_PoundsMass                    *) (sName := 'Pounds Mass',                            sDesc
ription := '',                                           sShortcut := ''),
  (*  eMass_Tons                          *) (sName := 'Tons',                                   sDesc
ription := 'Masse',                                      sShortcut := 't'),
  (*  eMassFlow_KilogramsPerSecond        *) (sName := 'Kilograms per Second',                   sDesc
ription := 'Massenstrom',                                sShortcut := 'kg/s'),
```

```
  (*  eMassFlow_KilogramsPerMinute          *) (sName := 'Kilograms per Minute',                sDesc
ription := 'Massenstrom',                                sShortcut := 'kg/min'),
  (*  eMassFlow_KilogramsPerHour            *) (sName := 'Kilograms per Hour',                  sDesc
ription := 'Massenstrom',                                sShortcut := 'kg/h'),
  (*  eMassFlow_PoundsMassPerMinute         *) (sName := 'Pounds Mass per Minute',              sDesc
ription := '',                                           sShortcut := ''),
  (*  eMassFlow_PoundsMassPerHour           *) (sName := 'Pounds Mass per Hour',                sDesc
ription := '',                                           sShortcut := ''),
  (*  ePower_Watts                          *) (sName := 'Watts',                               sDesc
ription := 'Leistung',                                   sShortcut := 'W'),
  (*  ePower_Kilowatts                      *) (sName := 'Kilowatts',                           sDesc
ription := 'Leistung',                                   sShortcut := 'kW'),
  (*  ePower_Megawatts                      *) (sName := 'Megawatts',                           sDesc
ription := 'Leistung',                                   sShortcut := 'MW'),
  (*  ePower_BtusPerHour                    *) (sName := 'Btus per Hour',                       sDesc
ription := '',                                           sShortcut := ''),
  (*  ePower_Horsepower                     *) (sName := 'Horsepower',                          sDesc
ription := 'Leistung',                                   sShortcut := 'PS'),
  (*  ePower_TonsRefrigeration              *) (sName := 'Tons Refrigeration',                  sDesc
ription := '',                                           sShortcut := ''),
  (*  ePressure_Pascals                     *) (sName := 'Pascals',                             sDesc
ription := 'Druck',                                      sShortcut := 'Pa'),
  (*  ePressure_Kilopascals                 *) (sName := 'Kilopascals',                         sDesc
ription := 'Druck',                                      sShortcut := 'kPa'),
  (*  ePressure_Bars                        *) (sName := 'Bars',                                sDesc
ription := 'Druck',                                      sShortcut := 'bar'),
  (*  ePressure_PoundsForcePerSquareInch    *) (sName := 'Pounds Force per Square Inch',        sDesc
ription := '',                                           sShortcut := ''),
  (*  ePressure_CentimetersOfWater          *) (sName := 'Centimeters of Water',                sDesc
ription := 'Druck',                                      sShortcut := ''),
  (*  ePressure_InchesOfWater               *) (sName := 'Inches of Water',                     sDesc
ription := '',                                           sShortcut := ''),
  (*  ePressure_MillimetersOfMercury        *) (sName := 'Millimeters of Mercury',              sDesc
ription := 'Hg Druck',                                   sShortcut := 'mm'),
  (*  ePressure_CentimetersOfMercury        *) (sName := 'Centimeters of Mercury',              sDesc
ription := 'Druck',                                      sShortcut := ''),
  (*  ePressure_InchesOfMercury             *) (sName := 'Inches of Mercury',                   sDesc
ription := '',                                           sShortcut := ''),
  (*  eTemperature_DegreesCelsius           *) (sName := 'Degrees Celsius',                     sDesc
ription := 'Temperatur',                                 sShortcut := '°C'),
  (*  eTemperature_DegreesKelvin            *) (sName := 'Degrees Kelvin',                      sDesc
ription := 'Temperatur (thermodynamisch)',               sShortcut := 'K'),
  (*  eTemperature_DegreesFahrenheit        *) (sName := 'Degrees Fahrenheit',                  sDesc
ription := '',                                           sShortcut := ''),
  (*  eTemperature_DegreeDaysCelsius        *) (sName := 'Degree Days Celsius',                 sDesc
ription := 'Gradtage',                                   sShortcut := 'K d/c'),
  (*  eTemperature_DegreeDaysFahrenheit     *) (sName := 'Degree Days Fahrenheit',              sDesc
ription := '',                                           sShortcut := ''),
  (*  eTime_Years                           *) (sName := 'Years',                               sDesc
ription := 'Zeit',                                       sShortcut := 'a'),
  (*  eTime_Months                          *) (sName := 'Months',                              sDesc
ription := 'Zeit',                                       sShortcut := 'Monat'),
  (*  eTime_Weeks                           *) (sName := 'Weeks',                               sDesc
ription := 'Zeit',                                       sShortcut := 'Woche'),
  (*  eTime_Days                            *) (sName := 'Days',                                sDesc
ription := 'Zeit',                                       sShortcut := 'd'),
  (*  eTime_Hours                           *) (sName := 'Hours',                               sDesc
ription := 'Zeit',                                       sShortcut := 'h'),
  (*  eTime_Minutes                         *) (sName := 'Minutes',                             sDesc
ription := 'Zeit',                                       sShortcut := 'min'),
  (*  eTime_Seconds                         *) (sName := 'Seconds',                             sDesc
ription := 'Zeit',                                       sShortcut := 's'),
  (*  eVelocity_MetersPerSecond             *) (sName := 'Meters per Second',                   sDesc
ription := 'Geschwindigkeit',                            sShortcut := 'm/s'),
  (*  eVelocity_KilometersPerHour           *) (sName := 'Kilometers per Hour',                 sDesc
ription := 'Geschwindigkeit',                            sShortcut := 'km/h'),
  (*  eVelocity_FeetPerSecond               *) (sName := 'Feet per Second',                     sDesc
ription := '',                                           sShortcut := ''),
  (*  eVelocity_FeetPerMinute               *) (sName := 'Feet per Minute',                     sDesc
ription := '',                                           sShortcut := ''),
  (*  eVelocity_MilesPerHour                *) (sName := 'Miles per Hour',                      sDesc
ription := '',                                           sShortcut := ''),
  (*  eVolume_CubicFeet                     *) (sName := 'Cubic Feet',                          sDesc
ription := '',                                           sShortcut := ''),
  (*  eVolume_CubicMeters                   *) (sName := 'Cubic Meters',                        sDesc
ription := 'Volumen',                                    sShortcut := 'm³'),
  (*  eVolume_ImperialGallons               *) (sName := 'Imperial Gallons',                    sDesc
ription := 'Brit. Gallonen',                             sShortcut := ''),
  (*  eVolume_Liters                        *) (sName := 'Liters',                              sDesc
ription := 'Volumen',                                    sShortcut := 'l'),
```

```
  (*  eVolume_UsGallons                    *) (sName := 'US Gallons',                          sDesc
ription := 'US-Gallonen',                                sShortcut := ''),
  (*  eVolumetricFlow_CubicFeetPerMinute   *) (sName := 'Cubic Feet per Minute',               sDesc
ription := 'Volumenstrom',                               sShortcut := 'cf/min'),
  (*  eVolumetricFlow_CubicMetersPerSecond *) (sName := 'Cubic Meters per Second',             sDesc
ription := 'Volumenstrom',                               sShortcut := 'm³/s'),
  (*  eVolumetricFlow_ImperialGallonsPerMinute*) (sName := 'Imperial Gallons per Minute',      sDesc
ription := '',                                           sShortcut := ''),
  (*  eVolumetricFlow_LitersPerSecond      *) (sName := 'Liters per Second',                   sDesc
ription := 'Volumenstrom',                               sShortcut := 'l/s'),
  (*  eVolumetricFlow_LitersPerMinute      *) (sName := 'Liters per Minute',                   sDesc
ription := 'Volumenstrom',                               sShortcut := 'l/min'),
  (*  eVolumetricFlow_UsGallonsPerMinute   *) (sName := 'US Gallons per Minute',               sDesc
ription := '',                                           sShortcut := ''),
  (*  eOther_DegreesAngular                *) (sName := 'Degrees Angular',                     sDesc
ription := 'Winkelgrade',                                sShortcut := '°'),
  (*  eOther_DegreesCelsiusPerHour         *) (sName := 'Degrees Celsius per Hour',            sDesc
ription := 'Temperaturänderung pro Zeiteinheit',         sShortcut := '°C/h'),
  (*  eOther_DegreesCelsiusPerMinute       *) (sName := 'Degrees Celsius per Minute',          sDesc
ription := 'Temperaturänderung pro Zeiteinheit',         sShortcut := '°C/min'),
  (*  eOther_DegreesFahrenheitPerHour      *) (sName := 'Degrees Fahrenheit per Hour',         sDesc
ription := '',                                           sShortcut := ''),
  (*  eOther_DegreesFahrenheitPerMinute    *) (sName := 'Degrees Fahrenheit per Minute',       sDesc
ription := '',                                           sShortcut := ''),
  (*  eOther_NoUnits                       *) (sName := 'No Units',                            sDesc
ription := '',                                           sShortcut := ''),
  (*  eOther_PartsPerMillion               *) (sName := 'Parts per Million',                   sDesc
ription := 'Konzentration',                              sShortcut := 'ppm'),
  (*  eOther_PartsPerBillion               *) (sName := 'Parts per Billion',                   sDesc
ription := 'Konzentration',                              sShortcut := 'ppb'),
  (*  eOther_Percent                       *) (sName := 'Percent',                             sDesc
ription := 'Prozent',                                    sShortcut := '%'),
  (*  eOther_PercentPerSecond              *) (sName := 'Percent per Second',                  sDesc
ription := 'Änderungsgeschwindigkeit',                   sShortcut := '%/s'),
  (*  eOther_PerMinute                     *) (sName := 'Per Minute',                          sDesc
ription := 'Drehzahl',                                   sShortcut := '1/m'),
  (*  eOther_PerSecond                     *) (sName := 'Per Second',                          sDesc
ription := 'Drehzahl',                                   sShortcut := '1/s'),
  (*  eOther_PsiPerDegreeFahrenheit        *) (sName := 'Psi per Degree Fahrenheit',           sDesc
ription := '',                                           sShortcut := ''),
  (*  eOther_Radians                       *) (sName := 'Radians',                             sDesc
ription := 'Bogenmaß',                                   sShortcut := 'rad'),
  (*  eOther_RevolutionsPerMinute          *) (sName := 'Revolutions per Minute',              sDesc
ription := 'Umdrehungen pro Minute',                     sShortcut := '1/min'),
  (*  eCurrency_Currency1                  *) (sName := 'Currency 1',                          sDesc
ription := 'Währung',                                    sShortcut := '€'),
  (*  eCurrency_Currency2                  *) (sName := 'Currency 2',                          sDesc
ription := 'Währung',                                    sShortcut := 'DM'),
  (*  eCurrency_Currency3                  *) (sName := 'Currency 3',                          sDesc
ription := 'Währung',                                    sShortcut := ''),
  (*  eCurrency_Currency4                  *) (sName := 'Currency 4',                          sDesc
ription := 'Währung',                                    sShortcut := ''),
  (*  eCurrency_Currency5                  *) (sName := 'Currency 5',                          sDesc
ription := 'Währung',                                    sShortcut := ''),
  (*  eCurrency_Currency6                  *) (sName := 'Currency 6',                          sDesc
ription := 'Währung',                                    sShortcut := ''),
  (*  eCurrency_Currency7                  *) (sName := 'Currency 7',                          sDesc
ription := 'Währung',                                    sShortcut := ''),
  (*  eCurrency_Currency8                  *) (sName := 'Currency 8',                          sDesc
ription := 'Währung',                                    sShortcut := ''),
  (*  eCurrency_Currency9                  *) (sName := 'Currency 9',                          sDesc
ription := 'Währung',                                    sShortcut := ''),
  (*  eCurrency_Currency10                 *) (sName := 'Currency 10',                         sDesc
ription := 'Währung',                                    sShortcut := ''),
  (*  eArea_SquareInches                   *) (sName := 'Square Inches',                       sDesc
ription := '',                                           sShortcut := ''),
  (*  eArea_SquareCentimeters              *) (sName := 'Square Centimeters',                  sDesc
ription := 'Fläche',                                     sShortcut := 'cm²'),
  (*  eEnthalpy_BtusPerPound               *) (sName := 'Btus per Pound',                      sDesc
ription := '',                                           sShortcut := ''),
  (*  eLength_Centimeters                  *) (sName := 'Centimeters',                         sDesc
ription := 'Länge',                                      sShortcut := 'cm'),
  (*  eMassFlow_PoundsMassPerSecond        *) (sName := 'Pounds Mass per Second',              sDesc
ription := '',                                           sShortcut := ''),
  (*  eTemperature_DeltaDegreesFahrenheit  *) (sName := 'Delta Degrees Fahrenheit',            sDesc
ription := '',                                           sShortcut := ''),
  (*  eTemperature_DeltaDegreesKelvin      *) (sName := 'Delta Degrees Kelvin',                sDesc
ription := 'Temperaturdifferenz',                        sShortcut := 'delta K'),
  (*  eElectrical_Kilohms                  *) (sName := 'Kilohms',                             sDesc
ription := 'Spezifischer elektrischer Widerstand',       sShortcut := 'kOhm'),
```

```
  (*  eElectrical_Megohms                 *) (sName := 'Megohms',                                sDesc
ription := 'Spezifischer elektrischer Widerstand',            sShortcut := 'MOhm'),
  (*  eElectrical_Millivolts              *) (sName := 'Millivolts',                             sDesc
ription := 'Elektrische Spannung',                            sShortcut := 'mV'),
  (*  eEnergy_KilojoulesPerKilogram       *) (sName := 'Kilojoules per Kilogram',               sDesc
ription := 'Enthalpie',                                       sShortcut := 'kJ/kg'),
  (*  eEnergy_Megajoules                  *) (sName := 'Megajoules',                             sDesc
ription := 'Energie',                                         sShortcut := 'MJ'),
  (*  eEntropy_JoulesPerDegreeKelvin      *) (sName := 'Joules per Degree Kelvin',              sDesc
ription := 'Entropie',                                        sShortcut := 'J/K'),
  (*  eEntropy_JoulesPerKilogramDegreeKelvin*) (sName := 'Joules per Kilogram Degree Kelvin',   sDes
cription := 'Spezifische Entropie',                            sShortcut := 'J/(kg K)'),
  (*  eFrequency_Kilohertz                *) (sName := 'Kilohertz',                              sDesc
ription := 'Frequenz',                                        sShortcut := 'kHz'),
  (*  eFrequency_Megahertz                *) (sName := 'Megahertz',                              sDesc
ription := 'Frequenz',                                        sShortcut := 'MHz'),
  (*  eFrequency_PerHour                  *) (sName := 'Per Hour',                               sDesc
ription := 'Drehzahl',                                        sShortcut := '1/h'),
  (*  ePower_Milliwatts                   *) (sName := 'Milliwatts',                             sDesc
ription := 'Leistung',                                        sShortcut := 'mW'),
  (*  ePressure_Hectopascals              *) (sName := 'Hectopascals',                           sDesc
ription := 'Druck',                                           sShortcut := 'hPa'),
  (*  ePressure_Millibars                 *) (sName := 'Millibars',                              sDesc
ription := 'Druck',                                           sShortcut := 'mbar'),
  (*  eVolumetricFlow_CubicMetersPerHour  *) (sName := 'Cubic Meters per Hour',                  sDesc
ription := 'Volumenstrom',                                    sShortcut := 'm³/h'),
  (*  eVolumetricFlow_LitersPerHour       *) (sName := 'Liters per Hour',                        sDesc
ription := 'Volumenstrom',                                    sShortcut := 'l/h'),
  (*  eOther_KilowattHoursPerSquareMeter  *) (sName := 'Kilowatt Hours per Square Meter',       sDesc
ription := 'Energiebedarfskennwert',                          sShortcut := 'kWh/m²'),
  (*  eOther_KilowattHoursPerSquareFoot   *) (sName := 'Kilowatt Hours per Square Foot',        sDesc
ription := '',                                                sShortcut := ''),
  (*  eOther_MegajoulesPerSquareMeter     *) (sName := 'Megajoules per Square Meter',           sDesc
ription := 'Energiebedarfskennwert',                          sShortcut := 'MJ/m²'),
  (*  eOther_MegajoulesPerSquareFoot      *) (sName := 'Megajoules per Square Foot',            sDesc
ription := '',                                                sShortcut := ''),
  (*  eOther_WattsPerSquareMeterDegreeKelvin*) (sName := 'Watts per Square Meter Degree Kelvin',sDes
cription := 'Wärmedurchgangskoeffizient',                      sShortcut := 'W/(m² K)'),
  (*  eVolumetricFlow_CubicFeetPerSecond  *) (sName := 'Cubic Feet per Second',                  sDesc
ription := 'Volumenstrom',                                    sShortcut := 'cf/sec'),
  (*  eOther_PercentObscurationPerFoot    *) (sName := 'Percent Obscuration per Foot',          sDesc
ription := '',                                                sShortcut := ''),
  (*  eOther_PercentObscurationPerMeter   *) (sName := 'Percent Obscuration per Meter',         sDesc
ription := 'Verdunkelung (Rauchmelder)',                      sShortcut := '%/m'),
  (*  eElectrical_Milliohms               *) (sName := 'Milliohms',                              sDesc
ription := 'Spezifischer elektrischer Widerstand',            sShortcut := 'mOhm'),
  (*  eEnergy_MegawattHours               *) (sName := 'Megawatt Hours',                         sDesc
ription := 'Energie',                                         sShortcut := 'MWh'),
  (*  eEnergy_KiloBtus                    *) (sName := 'Kilo Btus',                              sDesc
ription := '',                                                sShortcut := ''),
  (*  eEnergy_MegaBtus                    *) (sName := 'Mega Btus',                              sDesc
ription := '',                                                sShortcut := ''),
  (*  eEnthalpy_KilojoulesPerKilogramDryAir*) (sName := 'Kilojoules per Kilogram Dry Air',      sDesc
ription := 'Enthalpie',                                       sShortcut := 'kJ/kg tr. Luft'),
  (*  eEnthalpy_MegajoulesPerKilogramDryAir*) (sName := 'Megajoules per Kilogram Dry Air',      sDesc
ription := 'Enthalpie',                                       sShortcut := 'MJ/kg tr. Luft'),
  (*  eEntropy_KilojoulesPerDegreeKelvin  *) (sName := 'Kilojoules per Degree Kelvin',          sDesc
ription := 'Entropie',                                        sShortcut := 'kJ/K'),
  (*  eEntropy_MegajoulesPerDegreeKelvin  *) (sName := 'Megajoules per Degree Kelvin',          sDesc
ription := 'Entropie',                                        sShortcut := 'MJ/K'),
  (*  eForce_Newton                       *) (sName := 'Newton',                                 sDesc
ription := 'Kraft',                                           sShortcut := 'N'),
  (*  eMassFlow_GramsPerSecond            *) (sName := 'Grams per Second',                       sDesc
ription := 'Massenstrom',                                     sShortcut := 'g/s'),
  (*  eMassFlow_GramsPerMinute            *) (sName := 'Grams per Minute',                       sDesc
ription := 'Massenstrom',                                     sShortcut := 'g/min'),
  (*  eMassFlow_TonsPerHour               *) (sName := 'Tons per Hour',                          sDesc
ription := 'Massenstrom',                                     sShortcut := 't/h'),
  (*  ePower_KiloBtusPerHour              *) (sName := 'Kilo Btus per Hour',                     sDesc
ription := '',                                                sShortcut := ''),
  (*  eTime_HundredthsSeconds             *) (sName := 'Hundredths Seconds',                     sDesc
ription := 'Zeit',                                            sShortcut := '10-² s'),
  (*  eTime_Milliseconds                  *) (sName := 'Milliseconds',                           sDesc
ription := 'Zeit',                                            sShortcut := 'ms'),
  (*  eTorque_NewtonMeters                *) (sName := 'Torque Newton Meters',                   sDesc
ription := 'Drehmoment',                                      sShortcut := 'Nm'),
  (*  eVelocity_MillimetersPerSecond      *) (sName := 'Millimeters per Second',                 sDesc
ription := 'Geschwindigkeit',                                 sShortcut := 'mm/s'),
  (*  eVelocity_MillimetersPerMinute      *) (sName := 'Millimeters per Minute',                 sDesc
ription := 'Geschwindigkeit',                                 sShortcut := 'mm/min'),
```

```
  (*  eVelocity_MetersPerMinute            *) (sName := 'Meters per Minute',                    sDesc
ription := 'Geschwindigkeit',                                sShortcut := 'm/min'),
  (*  eVelocity_MetersPerHour              *) (sName := 'Meters per Hour',                      sDesc
ription := 'Geschwindigkeit',                                sShortcut := 'm/h'),
  (*  eVolumetricFlow_CubicMetersPerMinute *) (sName := 'Cubic Meters per Minute',              sDesc
ription := 'Volumenstrom',                                   sShortcut := 'm³/min'),
  (*  eAcceleration_MetersPerSecondPerSecond*) (sName := 'Meters per Second per Second',        sDes
cription := 'Beschleunigung',                                 sShortcut := 'm/s²'),
  (*  eElectrical_AmperesPerMeter          *) (sName := 'Amperes per Meter',                    sDesc
ription := 'Strom pro Länge',                                sShortcut := 'A/m'),
  (*  eElectrical_AmperesPerSquareMeter    *) (sName := 'Amperes per Square Meter',             sDesc
ription := 'Strom pro Fläche',                               sShortcut := 'A/m²'),
  (*  eElectrical_AmpereSquareMeters       *) (sName := 'Ampere Square Meters',                 sDesc
ription := 'Strom mal Fläche',                               sShortcut := 'Am²'),
  (*  eElectrical_Farads                   *) (sName := 'Farads',                               sDesc
ription := 'Elektrische Kapazität',                          sShortcut := 'F'),
  (*  eElectrical_Henrys                   *) (sName := 'Henrys',                               sDesc
ription := 'Induktivität',                                   sShortcut := 'H'),
  (*  eElectrical_OhmMeters                *) (sName := 'Ohm Meters',                           sDesc
ription := 'Spezifischer elektrischer Widerstand',           sShortcut := 'Ohmm'),
  (*  eElectrical_Siemens                  *) (sName := 'Siemens',                              sDesc
ription := 'Elektrischer Leitwert',                          sShortcut := 'S'),
  (*  eElectrical_SiemensPerMeter          *) (sName := 'Siemens per Meter',                    sDesc
ription := 'Elektrische Leitfähigkeit',                      sShortcut := 'S/m'),
  (*  eElectrical_Teslas                   *) (sName := 'Teslas',                               sDesc
ription := 'Magnetische Flussdichte',                        sShortcut := 'T'),
  (*  eElectrical_VoltsPerDegreeKelvin     *) (sName := 'Volts per Degree Kelvin',              sDesc
ription := 'Thermoelektrische Spannung',                     sShortcut := 'V/K'),
  (*  eElectrical_VoltsPerMeter            *) (sName := 'Volts per Meter',                      sDesc
ription := 'Elektrische Feldstärke',                         sShortcut := 'V/m'),
  (*  eElectrical_Webers                   *) (sName := 'Webers',                               sDesc
ription := 'Magnetischer Fluss',                             sShortcut := 'Wb'),
  (*  eLight_Candelas                      *) (sName := 'Candelas',                             sDesc
ription := 'Lichtstärke',                                    sShortcut := 'cd'),
  (*  eLight_CandelasPerSquareMeter        *) (sName := 'Candelas per Square Meter',            sDesc
ription := 'Leuchtdichte',                                   sShortcut := 'cd/m²'),
  (*  eTemperature_DegreesKelvinPerHour    *) (sName := 'Degrees Kelvin per Hour',              sDesc
ription := 'Temperaturänderung pro Zeit',                    sShortcut := 'K/h'),
  (*  eTemperature_DegreesKelvinPerMinute  *) (sName := 'Degrees Kelvin per Minute',            sDesc
ription := 'Temperaturänderung pro Zeit',                    sShortcut := 'K/min'),
  (*  eOther_JouleSeconds                  *) (sName := 'Joule Seconds',                        sDesc
ription := 'Drehimpuls',                                     sShortcut := 'Js'),
  (*  eOther_RadiansPerSecond              *) (sName := 'Radians per Second',                   sDesc
ription := 'Winkelgeschwindigkeit',                          sShortcut := 'rad/s'),
  (*  eOther_SquareMetersPerNewton         *) (sName := 'Square Meters per Newton',             sDesc
ription := 'Kraftverteilung',                                sShortcut := 'm²/N'),
  (*  eOther_KilogramsPerCubicMeter        *) (sName := 'Kilograms per Cubic Meter',            sDesc
ription := 'Dichte',                                         sShortcut := 'kg/m³'),
  (*  eOther_NewtonSeconds                 *) (sName := 'Newton Seconds',                       sDesc
ription := 'Impuls',                                         sShortcut := 'Ns'),
  (*  eOther_NewtonsPerMeter               *) (sName := 'Newtons per Meter',                    sDesc
ription := 'Oberflächenspannung',                            sShortcut := 'N/m'),
  (*  eOther_WattsPerMeterPerDegreeKelvin  *) (sName := 'Watts per Meter per Degree Kelvin',    sDesc
ription := 'Wärmeleitfähigkeit',                             sShortcut := 'W/m K'),
  (*  eMicro_Siemens                       *) (sName := 'Micro Siemens',                        sDesc
ription := 'Elektrischer Leitwert',                          sShortcut := 'µS'),
  (*  eCubic_FeetPerHour                   *) (sName := 'Cubic Feet per Hour',                  sDesc
ription := 'Durchsatz',                                      sShortcut := 'cf/h'),
  (*  eUs_GallonsPerHour                   *) (sName := 'US Gallons per Hour',                  sDesc
ription := 'Durchsatz',                                      sShortcut := 'G/h'),
  (*  eKilometers                          *) (sName := 'Kilometers',                           sDesc
ription := 'Länge',                                          sShortcut := 'km'),
  (*  eMicrometers                         *) (sName := 'Micrometers',                          sDesc
ription := 'Länge',                                          sShortcut := 'µm'),
  (*  eGrams                               *) (sName := 'Grams',                                sDesc
ription := 'Masse',                                          sShortcut := 'g'),
  (*  eMilligrams                          *) (sName := 'Milligrams',                           sDesc
ription := 'Masse',                                          sShortcut := 'mg'),
  (*  eMilliliters                         *) (sName := 'Milliliters',                          sDesc
ription := 'Volumen',                                        sShortcut := 'ml'),
  (*  eMillilitersPerSecond                *) (sName := 'Milliliters per Second',               sDesc
ription := 'Volumenstrom',                                   sShortcut := 'ml/s'),
  (*  eDecibels                            *) (sName := 'Decibels',                             sDesc
ription := 'Schalldruckpegel',                               sShortcut := 'dB'),
  (*  eDecibelsMillivolt                   *) (sName := 'Decibels Millivolt',                   sDesc
ription := 'Spannungspegel (bez.auf 1V)',                    sShortcut := 'dBV'),
  (*  eDecibelsVolt                        *) (sName := 'Decibels Volt',                        sDesc
ription := 'Spannungspegel (bez.auf 1mV)',                   sShortcut := 'dBmV'),
  (*  eMillisiemens                        *) (sName := 'Millisiemens',                         sDesc
ription := 'Elektrischer Leitwert',                          sShortcut := 'mS'),
```

```
  (*  eWatt_HoursReactive                 *) (sName := 'Watt Hours Reactive',                 sDesc
ription := 'Elektrische Blindarbeit',                           sShortcut := 'Whr'),
  (*  eKilowattHoursReactive              *) (sName := 'Kilowatt Hours Reactive',             sDesc
ription := 'Elektrische Blindarbeit',                           sShortcut := 'kWhr'),
  (*  eMegawattHoursReactive              *) (sName := 'Megawatt Hours Reactive',             sDesc
ription := 'Elektrische Blindarbeit',                           sShortcut := 'MWhr'),
  (*  eMillimetersOfWater                 *) (sName := 'Millimeters of Water',                sDesc
ription := 'Druck',                                             sShortcut := 'mmWS'),
  (*  ePer_Mille                          *) (sName := 'Per Mille',                           sDesc
ription := 'Promille',                                          sShortcut := '‰'),
  (*  eGrams_PerGram                      *) (sName := 'Grams per Gram',                      sDesc
ription := 'Massenanteil',                                      sShortcut := 'g/g'),
  (*  eKilograms_PerKilogram              *) (sName := 'Kilograms per Kilogram',              sDesc
ription := 'Massenanteil',                                      sShortcut := 'kg/kg'),
  (*  eGrams_PerKilogram                  *) (sName := 'Grams per Kilogram',                  sDesc
ription := 'Massenanteil',                                      sShortcut := 'g/kg'),
  (*  eMilligrams_PeGram                  *) (sName := 'Milligrams per Gram',                 sDesc
ription := 'Massenanteil',                                      sShortcut := 'mg/g'),
  (*  eMilligrams_PeKilogram              *) (sName := 'Milligrams per Kilogram',             sDesc
ription := 'Massenanteil',                                      sShortcut := 'mg/kg'),
  (*  eGrams_PerMilliliter                *) (sName := 'Grams per Milliliter',                sDesc
ription := 'Massenkonzentration',                               sShortcut := 'g/ml'),
  (*  eGrams_PerLiter                     *) (sName := 'Grams per Liter',                     sDesc
ription := 'Massenkonzentration',                               sShortcut := 'g/l'),
  (*  eMilligrams_PerLiter                *) (sName := 'Milligrams per Liter',                sDesc
ription := 'Massenkonzentration',                               sShortcut := 'mg/l'),
  (*  eMicrograms_PerLiter                *) (sName := 'Micrograms per Liter',                sDesc
ription := 'Massenkonzentration',                               sShortcut := 'µg/l'),
  (*  eGrams_PerCubicMeter                *) (sName := 'Grams per Cubic Meter',               sDesc
ription := 'Massenkonzentration',                               sShortcut := 'g/m3'),
  (*  eMilligrams_PerCubicMeter           *) (sName := 'Milligrams per Cubic Meter',          sDesc
ription := 'Massenkonzentration',                               sShortcut := 'mg/m3'),
  (*  eMicrograms_PerCubicMeter           *) (sName := 'Micrograms per Cubic Meter',          sDesc
ription := 'Massenkonzentration',                               sShortcut := 'µg/m3'),
  (*  eNanograms_PerCubicMeter            *) (sName := 'Nanograms per Cubic Meter',           sDesc
ription := 'Massenkonzentration',                               sShortcut := 'ng/m3'),
  (*  eGrams_PerCubicCentimeter           *) (sName := 'Grams per Cubic Centimeter',          sDesc
ription := 'Massenkonzentration',                               sShortcut := 'g/cm3'),
  (*  eBecquerels                         *) (sName := 'Becquerels',                          sDesc
ription := 'Aktivität (radioaktiver Stoff)',                    sShortcut := 'Bq'),
  (*  eKilobecquerels                     *) (sName := 'Kilobecquerels',                      sDesc
ription := 'Aktivität (radioaktiver Stoff)',                    sShortcut := 'kBq'),
  (*  eMegabecquerels                     *) (sName := 'Megabecquerels',                      sDesc
ription := 'Aktivität (radioaktiver Stoff)',                    sShortcut := 'MBq'),
  (*  eGray                               *) (sName := 'Gray',                                sDesc
ription := 'Energiedosis (ionisierende Strahlung)',             sShortcut := 'Gy'),
  (*  eMilligray                          *) (sName := 'Milligray',                           sDesc
ription := 'Energiedosis (ionisierende Strahlung)',             sShortcut := 'mGy'),
  (*  eMicrogray                          *) (sName := 'Microgray',                           sDesc
ription := 'Energiedosis (ionisierende Strahlung)',             sShortcut := 'µGy'),
  (*  eSieverts                           *) (sName := 'Sieverts',                            sDesc
ription := 'Äquivalenzdosis (gewichtete Strahlendosis)',        sShortcut := 'Sv'),
  (*  eMillisieverts                      *) (sName := 'Millisieverts',                       sDesc
ription := 'Äquivalenzdosis (gewichtete Strahlendosis)',        sShortcut := 'mSv'),
  (*  eMicrosieverts                      *) (sName := 'Microsieverts',                       sDesc
ription := 'Äquivalenzdosis (gewichtete Strahlendosis)',        sShortcut := 'µSv'),
  (*  eMicrosievertsPerHour               *) (sName := 'Microsieverts per Hour',              sDescr
iption := 'Äquivalenzdosisleistung',                            sShortcut := 'µSv/h'),
  (*  eDecibels_A                         *) (sName := 'Decibels A',                          sDesc
ription := 'A - bewerteter Schalldruckpegel',                   sShortcut := 'dB(a)'),
  (*  eNephelometric_TurbidityUnit        *) (sName := 'Nephelometric Turbidity Unit',        sDesc
ription := 'Trübung (Wasserqualität)',                          sShortcut := 'NTU'),
  (*  ePH                                 *) (sName := 'PH',                                  sDesc
ription := 'Wasserstoffionen-Konzentration',                    sShortcut := ''),
  (*  eGrams_PerSquareMeter               *) (sName := 'Grams per Square Meter',              sDesc
ription := 'Flächengewicht',                                    sShortcut := 'g/m2'),
  (*  eMinutes_PerDegreeKelvin            *) (sName := 'Minutes per Degree Kelvin',           sDesc
ription := 'Zeitänderung pro Temperatureinheit',                sShortcut := 'min/K')
  ];

aMeasuringElement  : ARRAY[E_BA_MeasuringElement.First .. E_BA_MeasuringElement.Last] OF ST_BA_EnumI
nfo := [

  (*  eNI100                              *) (sName := 'NI100',                               sDesc
ription := 'Nickel 100',                                        sShortcut := ''),
  (*  eNI120                              *) (sName := 'NI120',                               sDesc
ription := 'Nickel 120',                                        sShortcut := ''),
  (*  eNI1000                             *) (sName := 'NI1000',                              sDesc
ription := 'Nickel 1000 (Standard)',                            sShortcut := ''),
  (*  eNI1000_LS                          *) (sName := 'NI1000 LS',                           sDesc
```

```
ription := 'Nickel 1000 (Landis & Staefa)',                    sShortcut := ''),
  (*  eNTC1K8                          *) (sName := 'NTC1K8',                          sDesc
ription := 'NTC 1,8 kOhms',                                    sShortcut := ''),
  (*  eNTC1K8_TK                       *) (sName := 'NTC1K8 TK',                       sDesc
ription := 'NTC 1,8 kOhms (TK)',                               sShortcut := ''),
  (*  eNTC2K2                          *) (sName := 'NTC2K2',                          sDesc
ription := 'NTC 2,2 kOhms',                                    sShortcut := ''),
  (*  eNTC3K                           *) (sName := 'NTC3K',                           sDesc
ription := 'NTC 3 kOhms',                                      sShortcut := ''),
  (*  eNTC5K                           *) (sName := 'NTC5K',                           sDesc
ription := 'NTC 5 kOhms',                                      sShortcut := ''),
  (*  eNTC10K                          *) (sName := 'NTC10K',                          sDesc
ription := 'NTC 10 kOhms',                                     sShortcut := ''),
  (*  eNTC10KPRE                       *) (sName := 'NTC10KPRE',                       sDesc
ription := 'NTC 10 kOhms (Precon)'                             sShortcut := ''),
  (*  eNTC10K_3204                     *) (sName := 'NTC10K 3204',                     sDesc
ription := 'NTC 10 kOhms 3204',                                sShortcut := ''),
  (*  eNTC10KTYP2                      *) (sName := 'NTC10K TYP2',                     sDesc
ription := 'NTC 10 kOhms TYP2',                                sShortcut := ''),
  (*  eNTC10KTYP3                      *) (sName := 'NTC10K TYP3',                     sDesc
ription := 'NTC 10 kOhms TYP3',                                sShortcut := ''),
  (*  eNTC10KDALE                      *) (sName := 'NTC10K DALE',                     sDesc
ription := 'NTC 10 kOhms DALE',                                sShortcut := ''),
  (*  eNTC10K3A221                     *) (sName := 'NTC10K 3A221',                    sDesc
ription := 'NTC 10 kOhms 3A221',                               sShortcut := ''),
  (*  eNTC20K                          *) (sName := 'NTC20K',                          sDesc
ription := 'NTC 20 kOhms',                                     sShortcut := ''),
  (*  eNTC100K                         *) (sName := 'NTC100K',                         sDesc
ription := 'NTC 100 kOhms',                                    sShortcut := ''),
  (*  ePoti_Resolution_01              *) (sName := 'Poti (Resolution 0,1 Ohms)',      sDesc
ription := 'Potentiometer (Resolution 0,1 Ohms)',              sShortcut := ''),
  (*  ePoti_Resolution_1               *) (sName := 'Poti (Resolution 1 Ohm)',         sDesc
ription := 'Potentiometer (Resolution 1 Ohm)',                 sShortcut := ''),
  (*  eOutput_10_5000                  *) (sName := 'Output (10 to 5000 Ohms)',        sDesc
ription := 'Output (10 to 5000 Ohms)',                         sShortcut := ''),
  (*  eOutput_10_1200                  *) (sName := 'Output (10 to 1200 Ohms)',        sDesc
ription := 'Output (10 to 1200 Ohms)',                         sShortcut := ''),
  (*  ePT100                           *) (sName := 'PT100',                           sDesc
ription := 'PT 100',                                           sShortcut := ''),
  (*  ePT200                           *) (sName := 'PT200',                           sDesc
ription := 'PT 200',                                           sShortcut := ''),
  (*  ePT500                           *) (sName := 'PT500',                           sDesc
ription := 'PT 500',                                           sShortcut := ''),
  (*  ePT1000                          *) (sName := 'PT1000',                          sDesc
ription := 'PT 1000',                                          sShortcut := '')
  ];

aToggleMode    : ARRAY[E_BA_ToggleMode.First .. E_BA_ToggleMode.Last] OF ST_BA_EnumInfo := [

  (*  eSwitch                          *) (sName := 'Schalter',                        sDesc
ription := 'Einrastender Schalter',                            sShortcut := ''),
  (*  ePushButton                      *) (sName := 'Taster',                          sDesc
ription := 'Nicht-einrastender Schalter',                      sShortcut := '')
  ];

aAction      : ARRAY[E_BA_Action.First .. E_BA_Action.Last] OF ST_BA_EnumInfo := [

  (*  eDirect                          *) (sName := 'Direkt',                          sDesc
ription := 'Gleichläufiger Wirksinn',                          sShortcut := ''),
  (*  eReverse                         *) (sName := 'Indirekt',                        sDesc
ription := 'Gegenläufiger Wirksinn',                           sShortcut := '')
  ];

aEventState    : ARRAY[E_BA_EventState.First .. E_BA_EventState.Last] OF ST_BA_EnumInfo := [

  (*  eNormal                        *) (sName := 'Normal',                          sDesc
ription := '',                                                 sShortcut := ''),
  (*  eFault                         *) (sName := 'Fehler',                          sDesc
ription := '',                                                 sShortcut := ''),
  (*  eOffnormal                     *) (sName := 'Unnormal',                        sDesc
ription := '',                                                 sShortcut := ''),
  (*  eLowLimit                      *) (sName := 'Unter Grenzwert',                 sDesc
ription := 'Unterschreitung des definierten Grenzwert',        sShortcut := ''),
  (*  eHighLimit                     *) (sName := 'Über Grenzwert',                  sDesc
ription := 'Überschreitung des definierten Grenzwert',         sShortcut := '')
  ];

aReliability    : ARRAY[E_BA_Reliability.First .. E_BA_Reliability.Last] OF ST_BA_EnumInfo := [

  (*  eNoFaultDetected               *) (sName := 'Kein Fehler',                     sDesc
```

```
ription := '',                                                  sShortcut := ''),
  (*  eNoSensor                        *) (sName := 'Sensorfehler',                          sDesc
ription := 'TODO',                                              sShortcut := ''),
  (*  eOverRange                       *) (sName := 'Über Bereich',                          sDesc
ription := 'Überschreitung des definierten Rohwert-Bereichs',  sShortcut := ''),
  (*  eUnderRange                      *) (sName := 'Unter Bereich',                         sDesc
ription := 'Unterschreitung des definierten Rohwert-Bereichs', sShortcut := ''),
  (*  eOpenLoop                        *) (sName := 'Drahtbruch',                            sDesc
ription := '',                                                  sShortcut := ''),
  (*  eShortedLoop                     *) (sName := 'Kurzschluß',                            sDesc
ription := '',                                                  sShortcut := ''),
  (*  eNoOutput                        *) (sName := 'Kein Ausgang',                          sDesc
ription := 'TODO',                                              sShortcut := ''),
  (*  eUnreliableOther                 *) (sName := 'Sonstiges',                             sDesc
ription := '',                                                  sShortcut := ''),
  (*  eProcessError                    *) (sName := 'Prozessfehler',                         sDesc
ription := 'TODO',                                              sShortcut := ''),
  (*  eMultiStateFault                 *) (sName := 'Multistate-
Fehler',               sDescription := 'TODO',                                          sShortcu
t := ''),
  (*  eConfigurationError              *) (sName := 'Konfigurationsfehler',                  sDesc
ription := '',                                                  sShortcut := ''),
  (*              *) (),
  (*  eCommunicationFailure            *) (sName := 'Kommunikationsfehler',                  sDesc
ription := '',                                                  sShortcut := ''),
  (*  eMemberFault                     *) (sName := 'Fehler Teilnehmer',                     sDesc
ription := 'TODO',                                              sShortcut := '')
  ];


aPolarity    : ARRAY[E_BA_Polarity.First .. E_BA_Polarity.Last] OF ST_BA_EnumInfo := [

  (*  eNormal                          *) (sName := 'Normal',                                sDesc
ription := 'Direkte Polarität',                                 sShortcut := ''),
  (*  eReverse                         *) (sName := 'Invertiert',                            sDesc
ription := 'Indirekte Polarität',                               sShortcut := '')
  ];

aByteMappingMode  : ARRAY[E_BA_ByteMappingMode.First .. E_BA_ByteMappingMode.Last] OF ST_BA_EnumInfo
 := [

  (*  eIndex1N             *) (sName := 'Index 1-
N',                   sDescription := 'TODO',                                  sShortcut := ''),
  (*  eBinary1N            *) (sName := 'Binary 1-
N',                    sDescription := 'TODO',                                   sShortcut := ''),
  (*  eIndexUpDown         *) (sName := 'Index Up-
Down',                 sDescription := 'TODO',                                   sShortcut := ''),
  (*  eBinaryUpDown        *) (sName := 'Binary Up-
Down',                 sDescription := 'TODO',                                  sShortcut := ''),
  (*  eBinaryDecimal       *) (sName := 'Binary Decimal',                       sDescription := 'TODO
',                                     sShortcut := '')
  ];

aPIDMode    : ARRAY[E_BA_PIDMode.First .. E_BA_PIDMode.Last] OF ST_BA_EnumInfo := [

  (*  eP1ID                *) (sName := 'P1ID',                              sDescription := 'Vorgelag
erter P-Anteil beeinflusst I- und D-Anteil', sShortcut := ''),
  (*  ePID                 *) (sName := 'PID',                               sDescription := 'Addition
 von P-, I- und D-Anteilen (Standard)',      sShortcut := '')
  ];

aLoggingType    : ARRAY[E_BA_LoggingType.First .. E_BA_LoggingType.Last] OF ST_BA_EnumInfo := [

  (*  ePolled              *) (sName := 'Polled',                                  sDescription := 'Gen
eriert Log-Einträge in konstanten Intervallen',  sShortcut := ''),
  (*  eCOV                 *) (sName := 'COV',                                     sDescription := 'Gen
eriert Log-Einträge bei Wertänderung',           sShortcut := ''),
  (*  eTriggered           *) (sName := 'Triggered',                               sDescription := 'Gen
eriert Log-Einträge ereignisgesteuert',          sShortcut := '')
  ];


aLanguage    : ARRAY[E_BA_Language.First .. E_BA_Language.Last] OF ST_BA_EnumInfo := [

  (*  eEnglish             *) (sName := 'Englisch',             sDescription := 'Englisch (1033)',
                               sShortcut := 'EN'),
  (*  eGerman              *) (sName := 'Deutsch',              sDescription := 'Deutsch (1031)',
                               sShortcut := 'DE')
```

```
  ];
END_VAR
```

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