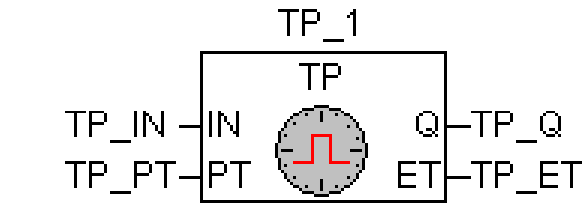
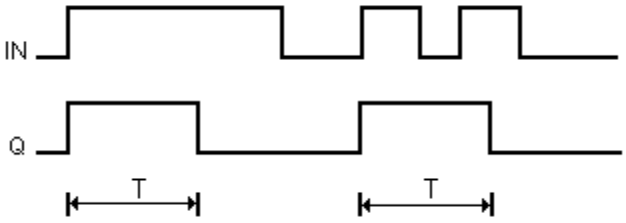
Softwarebeschreibung

Advant Controller 31
Dezentrale, intelligente
Automatisierungstechnik

Bausteinbibliothek
Serie 40/50

TP S40





Inhalt

Die Funktionsbaustein-Bibliothek Base_S40

Wesentliche Unterschiede der ABB-Bibliothek 907 AC 1131 zur VE-Bibliothek der 907 PC 331

Alle Bausteine und Funktionen werden nach der IEC-Norm in Operatoren, Funktionen und Funktionsblöcke unterteilt. (Die Definition dazu befindet sich im Handbuch der Programmiersoftware.)

- Alle in der Tabelle »Bausteinübersicht« aufgeführten Operatoren sind Bestandteil der Programmiersoftware.
- In der Standardbibliothek sind alle Bausteine enthalten, die im Programmierhandbuch beschrieben sind.
- Die ABB-Bibliothek für die SPS-Serien 40 und 50 enthält alle ABB-spezifischen Bausteine (Funktionen und Funktionsblöcke, jedoch keine Operatoren).
  Hinweis:
  Die Bausteine der ABB-Bibliothek laufen nicht im Simulationsmodus.

Um die ABB-spezifischen Bausteine zu verwenden, muß im Bibliotheksfenster die ABB-Bibliothek (Base_S40_Vxx) mit eingebunden werden.

Besonderheiten der ABB-Bibliothek

In der ABB-Bibliothek sind alle Herstellerfunktionen und -Funktionsblöcke integriert. Die Zuordnung der Bausteintypen ist in der Tabelle »Bausteinübersicht« aufgeführt.

Die Bausteine wurden in ihrer Funktionalität identisch zu den bekannten Bausteinen der 07 KR 51 - EBS umgesetzt. Dennoch sind beim Aufruf und der Auswahl der Bausteine einige Besonderheiten zu beachten.

Funktionen

Die Bausteine im FUP sind identisch aufgebaut wie die Bausteine der bisherigen Programmiersoftware. Teilweise haben sich Ein-/Ausgangsbezeichnungen geändert, da nicht alle Zeichen zugelassen sind. Die Reihenfolge der Ein- und Ausgänge ist in allen Programmiersprachen identisch. Im Bibliotheksfenster kann man die Belegung der Bausteine einsehen (sowohl Reihenfolge, als auch Variablentyp).

Funktionsblöcke

Ein Hauptmerkmal der Funktionsblöcke ist, daß beim Aufruf eine Instanz vergeben werden muß. Dabei ist zu beachten, **daß Instanznamen nicht mehrfach vergeben werden dürfen**.

Beispiel:

Name	Typ
VRZ1	VRZ
VRZ2	VRZ
.	.
.	.

Der Instanzname kann beliebig definiert werden. Der Typ ist vorgegeben und entspricht dem Namen des Funktionsblockes.

Besonderheiten einzelner Bausteine

1) Negation der Ein- und Ausgänge:

   Das Negieren ist in der 907 AC 1131 nur für Bitoperanden möglich (Variablentypen: BOOL, BYTE, WORD, DWORD). Die bekannten Operanden Wortmerker %MW (MW) und Doppelwortmerker %MD (MD) sind vom Variablentyp INT bzw. DINT.

2) Dopplung der Eingänge:

   Die Operatoren (ADD, DIV, MUL, SUB) können beliebig oft verwendet und gedoppelt werden.

3) Dopplung der Ein- und Ausgänge:

   Bausteine: z.B. PACK, UNPACK

   Die Anzahl der Dopplungen ist im FUP-Editor nicht veränderbar und fest auf 4, 8 oder 16 eingestellt. Dementsprechend wird der Eingang n belegt und der Baustein angezeigt.

4) Einfügen von ABB AWL mit Inline Pragma:

   ABB Anweisungsliste kann als Inline Pragma in ein Projekt eingefügt werden.

   Dazu wird ein Programmbaustein in der IEC-Sprache AWL angelegt. Ein Inline Pragma hat folgende Syntax:

```
LD 0  ; notwendig, wenn der Programmbaustein nur aus Inline Pragma besteht
{S40Inline
.
.                    ABB ASCII-AWL
.
.
}
```

   Bausteine: DRUCK, EMAS

   Die Bausteine DRUCK und EMAS stehen nur in diesem Format zur Verfügung.

   Achtung:
   Im Pragma wird keine Syntaxprüfung durchgeführt !!!

5) Sonderlösungen:

   Bausteine: LIZU8, LIZU16, LIZU32, LIZU64, LIZU256

   Die Zahl hinter LIZU steht für die Anzahl der direkten Konstanten in der Liste. Diese Zahl ist der Wert einer direkten Konstante (#8, #16, #32, #64, #256).

Bausteine: FKG2, FKG4, FKG16, FKG32, FKG256

Die Zahl hinter FKG steht für die Anzahl der Interpolationspunkte. Diese Zahl ist der Wert einer direkten Konstante (#2, #4, #16, #32, #256).

Bausteine: BMELD8, BMELD16, BMELD32, BMELD64, BMELD127

Die Zahl hinter BMELD steht für die Anzahl der Eingabewerte. Diese Zahl ist der Wert einer direkten Konstante (#8, #16, #32, #64, #127).

Bausteinübersicht, geordnet nach Aufrufnamen

Zeichenerklärung:

FBmV … Funktionsblock mit Vergangenheitswerten

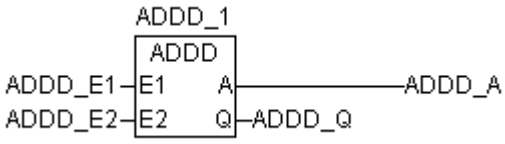
FBoV … Funktionsblock ohne Vergangenheitswerte

F … Funktion

Addition Doppelwort ADDD S40

Der Wert des Operanden am Eingang E1 wird zum Wert des Operanden am Eingang E2 addiert und das Ergebnis dem Operanden am Ausgang A zugewiesen.

Das Ergebnis wird auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt. Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein TRUE-Signal zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein FALSE-Signal zugewiesen.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	ADDD	Instanzname
E1	DINT	Summand 1
E2	DINT	Summand 2
A	DINT	Summe
Q	BOOL	Summe begrenzt

Beschreibung

Der Wert des Operanden am Eingang E1 wird zum Wert des Operanden am Eingang E2 addiert und das Ergebnis dem Operanden am Ausgang A zugewiesen.

Das Ergebnis wird auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt (Zahlenbereich: -2147483647 … 2147483647). Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein TRUE-Signal zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein FALSE-Signal zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
ADDD_1 : ADDD;
ADDD_E1 AT %MD2000.0: DINT;
ADDD_E2 AT %MD2000.1: DINT;
ADDD_A AT %MD2000.2: DINT;
ADDD_Q AT %MX0.0: BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
ADDD
E1
E2
A
Q
```

FUP:

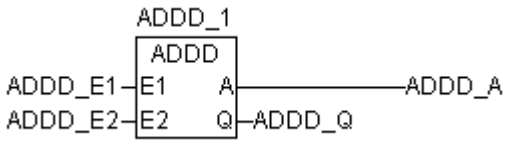


ABB AWL des Beispiels:

```
!BA 0
ADDD
MD0,0
MD0,1
MD0,2
M0,0
```

Funktionsaufruf in AWL

```
CAL   ADDD_1(E1 := ADDD_E1,
      E2 := ADDD_E2)

LD    ADDD_1.Q
ST    ADDD_Q
LD    ADDD_1.A
ST    ADDD_A
```

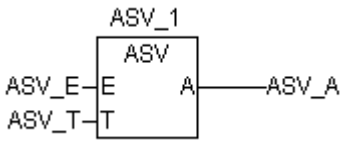
Funktionsaufruf in ST

```
ADDD_1   (E1 := ADDD_E1,
         E2 := ADDD_E2);

ADDD_Q:=ADDD_1.Q;
ADDD_A:=ADDD_1.A;
```

Ausschaltverzögerer ASV S40

Die TRUE/FALSE-Flanke des Eingangs E wird um die Zeitdauer T verzögert und als TRUE/FALSE-Flanke am Ausgang A ausgegeben.

Geht der Eingang E vor Ablauf der Zeit T wieder auf TRUE-Pegel, so bleibt der Ausgang A auf TRUE-Pegel.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	ASV	Instanzname
E	BOOL	Eingangssignal
T	TIME	Verzögerungszeit
A	BOOL	Verzögertes Signal, Operanden M, A (nicht E, S)

Beschreibung

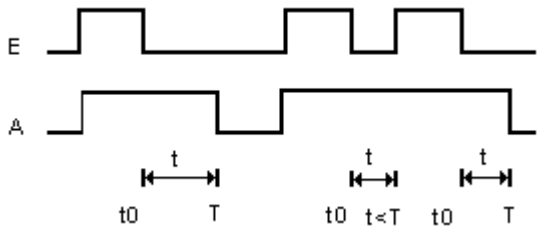
Die TRUE/FALSE-Flanke des Eingangs E wird um die Zeitdauer T verzögert und als TRUE/FALSE-Flanke am Ausgang A ausgegeben.

Geht der Eingang E vor Ablauf der Zeit T wieder auf TRUE-Pegel, so bleibt der Ausgang A auf TRUE-Pegel.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Sinnvoller Bereich für T: > 1 Zykluszeit

Die Eingänge und der Ausgang sind weder doppelbar noch invertierbar.



Allgemeines Verhalten

• Gestartete Zeitwerke werden vom Betriebssystem der SPS bearbeitet und sind deshalb vollkommen unabhängig von der Bearbeitung des SPS-Programms. Erst nach Ablauf des Zeitwerks erfolgt eine entsprechende Meldung des Betriebssystems an den zugehörigen Zeitbaustein im SPS-Programm.

• Die Bearbeitung eines Zeitwerks im Betriebssystem der SPS wird durch folgende Befehle beeinflußt: Alle laufenden Zeitwerke werden gestoppt und initialisiert, wenn einer der folgenden Fehler auftritt:

- SPS-Programm abbrechen
- RUN/STOP-Schalter von RUN -> STOP
- Warm- oder Kaltstart

Beispiel

Deklaration:

```
ASV_1: ASV;
ASV_E AT %MX0.0: BOOL;
ASV_T AT %MD4001.0 : TIME := t#2s500ms; (* 2500ms*)
ASV_A AT %MX0.1: BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
ASV
E
T
A
```

FUP:

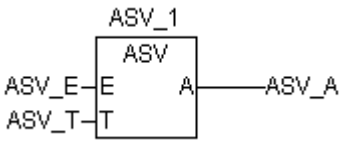


ABB AWL des Beispiels:

```
!BA 0
ASV
M0,0
KD1,0                ; 2500
M0,1
```

Funktionsaufruf in AWL

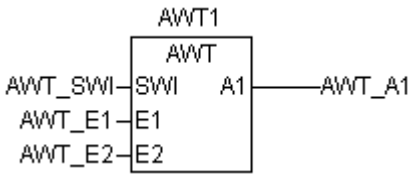
```
CAL   ASV_1(E := ASV_E,
            T := ASV_T)

LD    ASV_1.A
ST    ASV_A
```

Funktionsaufruf in ST

```
ASV_1   (E := ASV_E,
         T := ASV_T);

ASV_A   := ASV_1.A;
```

Auswahltor, Wort AWT S40

In Abhängigkeit der Eingangsschalterwerte (SWI) kopiert AWT Eingang1 (E1) oder Eingang2 (E2) in den Ausgang (A1).



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	AWT	Instanzname
SWI	BOOL	Umschalteingang
E1	INT	Wort-Eingang für SWI = FALSE
E2	INT	Wort-Eingang für SWI = TRUE
A1	INT	Wort-Ausgang

Beschreibung

FALSE-Signal am binären Eingang SWI weist dem Wortoperanden am Ausgang A1 den Wert des Wortoperanden am Eingang E1 zu.

TRUE-Signal am binären Eingang SWI weist dem Wortoperanden am Ausgang A1 den Wert des Wortoperanden am Eingang E2 zu.

Beispiel

Deklaration:

```
AWT1 : AWT;
AWT_SWI AT %MX0.0 : BOOL;
AWT_E1 AT %MW1000.0 : INT;
AWT_E2 AT %MW1000.1 : INT;
AWT_A1 AT %MW1000.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
AWT
SWI
E1
E2
A1
```

FUP:

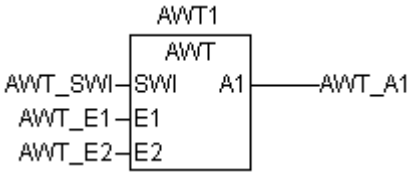


ABB AWL des Beispiels:

```
!BA 0
AWT
M0,0
MW0,0
MW0,1
MW0,2
```

Funktionsaufruf in AWL

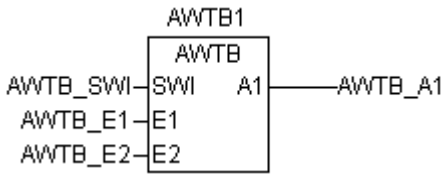
```
CAL   AWT_1(SWI := AWT_SWI,
            E1 := AWT_E1,
            E2 := AWT_E2)

LD    AWT_1.A1
ST    AWT_A1
```

Funktionsaufruf in ST

```
AWT_1    (SWI := AWT_SWI,
          E1 := AWT_E1,
          E2 := AWT_E2);

AWT_A1   := AWT_1.A1;
```

Auswahltor Binär AWTB S40

In Abhängigkeit der Eingangsschalterwerte (SWI) kopiert AWTB Eingang1 (E1) oder Eingang2 (E2) in den Ausgang (A1).



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	AWTB	Instanzname
SWI	BOOL	Umschalteingang
E1	BOOL	Binäreingang für SWI = FALSE
E2	BOOL	Binäreingang für SWI = TRUE
A1	BOOL	Binärausgang

Beschreibung

FALSE-Signal am binären Eingang SWI weist dem Operanden am Ausgang A1 den Status des Operanden am Eingang E1 zu.

TRUE-Signal am binären Eingang SWI weist dem Operanden am Ausgang A1 den Status des Operanden am Eingang E2 zu.

Beispiel

Deklaration:

```
AWTB1 : AWTB;
AWTB_SWI AT %MX0.0 : BOOL;
AWTB_E1 AT %MX0.1 : BOOL;
AWTB_E2 AT %MX0.2 : BOOL;
AWTB_A1 AT %MX0.3 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
AWTB
SWI
E1
E2
A1
```

FUP:

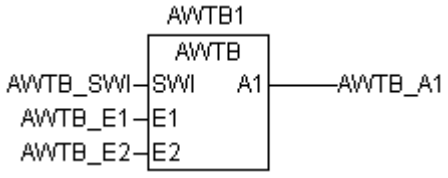


ABB AWL des Beispiels:

```
!BA 0
AWTB
M0,0
M0,1
M0,2
M0,3
```

Funktionsaufruf in AWL

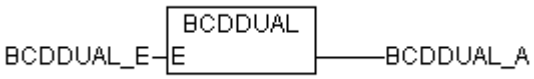
```
CAL   AWTB_1(SWI := AWTB_SWI,
              E1 := AWTB_E1,
              E2 := AWTB_E2)

LD    AWTB_1.A1
ST    AWTB_A1
```

Funktionsaufruf in ST

```
AWTB_1   (SWI := AWTB_SWI,
          E1 := AWTB_E1,
          E2 := AWTB_E2);

AWTB_A   := AWTB_1.A1;
```

BCD nach DUAL-Wandlung, Wort BCDDUAL S40

Die positive, BCD-codierte Zahl am Eingang E wird in eine Dualzahl gewandelt und dem Operanden am Ausgang A zugewiesen.



Bausteintyp

Funktion

Parameter

E	INT	BCD-codierte Zahl
(A)	INT	Dualzahl

Beschreibung

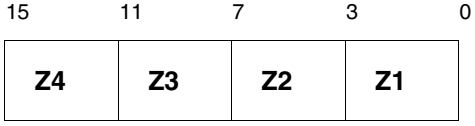
Die positive, BCD-codierte Zahl am Eingang E wird in eine Dualzahl gewandelt und dem Operanden am Ausgang A zugewiesen.

Ein- und Ausgang sind weder doppelbar noch negierbar.

Definition:

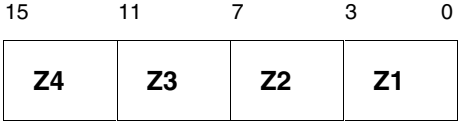
Die Wertigkeit der Ziffern einer BCD-codierten Zahl und einer Hexadezimal-Zahl sind wie folgt definiert:

BCD-ZAHL



Zahlenwert:
Z1 * 1
Z2 * 10
Z3 * 100
Z4 * 1000
$0 \leq Zi \leq 9$

HEX-ZAHL



Zahlenwert:
Z1 * 1
Z2 * 16
Z3 * 256
Z4 * 4096
$0 \leq Zi \leq F$

Anmerkung:

Der Baustein akzeptiert zusätzlich am BCD-Eingang auch Ziffern, für die gilt:

$0 \leq Zi \leq F$

Beispiel 1

BCD-ZAHL

15	11	7	3	0
1	**2**	**3**	**4**	

BIT ->

HEX-ZAHL

15	11	7	3	0
0	**4**	**D**	**2**	

Z1 = 4 * 1	=	4	Z1 = 2 * 1	= 2
Z2 = 3 * 10	=	30	Z2 = 13 * 16	= 208
Z3 = 2 * 100	=	200	Z3 = 4 * 256	= 1024
Z4 = 1 * 1000	=	1000	Z4 = 0 * 4096	= 0
		1234		1234

Beispiel 2

BCD-ZAHL

15	11	7	3	0
A	**2**	**F**	**4**	

BIT ->

HEX-ZAHL

15	11	7	3	0
2	**8**	**7**	**2**	

Z1 = 4 * 1	=	4	Z1 = 2 * 1	= 2
Z2 = 15 * 10	=	150	Z2 = 7 * 16	= 112
Z3 = 2 * 100	=	200	Z3 = 8 * 256	= 2048
Z4 = 10 * 1000	=	10000	Z4 = 2 * 4096	= 8192
		10354		10354

Darstellung einer negativen BCD-Zahl

Eine negative BCD-Zahl kann in der SPS durch getrennte Darstellung des Wertes und des Vorzeichens dargestellt werden. Dabei wird der Wert der BCD-Zahl in einer Wortvariablen und die Information über das Vorzeichen in einer Binärvariablen abgelegt.

Beispiel

Deklaration:

```
BCDDUAL_E AT %MW1002.0 : INT;
BCDDUAL_A AT %MW1002.1 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
BCDDUAL
E
A
```

FUP:

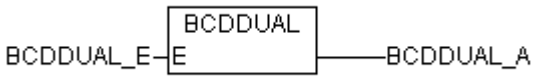


ABB AWL des Beispiels:

```
!BA 0
BCDDUAL
MW2,0
MW2,1
```

Funktionsaufruf in AWL

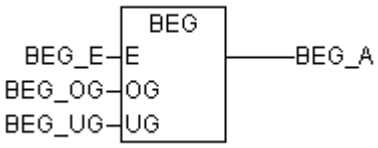
```
LD      BCD_E
BCDDUAL
ST      BCD_A
```

Funktionsaufruf in ST

```
BCD_A:=BCDDUAL(BCD_E);
```

Begrenzer, Wort BEG S40

Der Wert des Operanden am Eingang E wird auf den Bereich zwischen der oberen und der unteren Grenze begrenzt.



Bausteintyp

Funktion

Parameter

E	INT	Eingangswert
OG	INT	Obere Grenze
UG	INT	Untere Grenze
(A)	INT	Begrenzter Wert

Beschreibung

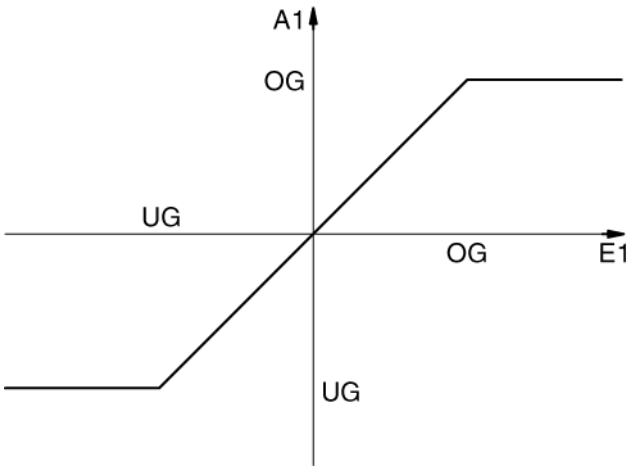
Der Wert des Operanden am Eingang E wird auf den Bereich zwischen der oberen und der unteren Grenze begrenzt.

Die obere Grenze wird durch den Operanden am Eingang OG, die untere Grenze durch den Operanden am Eingang UG vorgegeben.

Es gilt:

A = UG für E < UG
A = E für UG ≤ E ≤ OG
A = OG für E > OG

Die Eingänge und der Ausgang sind weder doppelbar noch negierbar.



Beispiel

Deklaration:

```
BEG_E AT %MW1000.0 : INT;
BEG_OG AT %MW3001.0 : INT := 20000;
BEG_UG AT %MW3001.1 : INT := 10000;
BEG_A AT %MW1000.1 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
BEG
E
OG
UG
A
```

FUP:

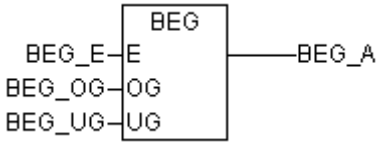


ABB AWL des Beispiels:

```
!BA 0
BEG
MW0,0
KW1,0          ; 20000
KW1,1          ; 10000
MW0,1
```

Funktionsaufruf in AWL

```
LD    BEG_E
BEG   BEG_OG, BEG_UG
ST    BEG_A
```

Funktionsaufruf in ST

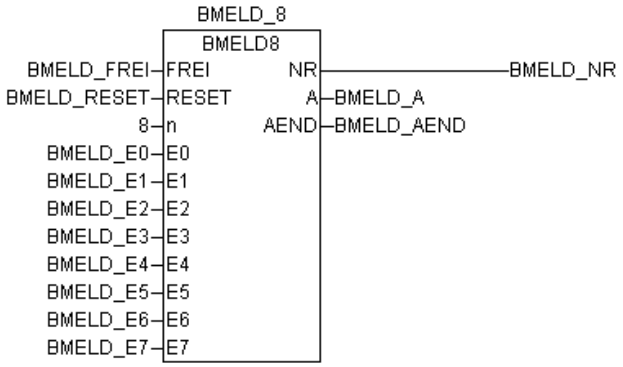
```
BEG_A    := BEG(BEG_E, BEG_OG, BEG_UG);
```

Binärwert-Änderungsmelder BMELD(8..127) S40

Der Baustein überwacht die am doppelbaren Eingang E0 anliegenden Binärwerte auf Änderung.

Die BMELD-Nummer gibt die max. Anzahl der Eingangswerte an. Es stehen folgende Änderungsmelder zur Verfügung:

BMELD8	Binärwert-Änderungsmelder mit max. 8 Eingangswerten
BMELD16	Binärwert-Änderungsmelder mit max. 16 Eingangswerten
BMELD32	Binärwert-Änderungsmelder mit max. 32 Eingangswerten
BMELD127	Binärwert-Änderungsmelder mit max. 127 Eingangswerten



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	BMELD(8..32)	Instanzname
FREI	BOOL	Baustein-Freigabe
RESET	BOOL	Reset
N	INT	Anzahl Eingangswerte
E0 .. En-1	BOOL	Eingangswerte
NR	INT	Nummer des Eingangswertes
A	INT	aktueller Eingangswert
AEND	BOOL	Änderung erkannt

Beschreibung

Der Baustein überwacht die am doppelbaren Eingang E0 … En-1 anliegenden Binärwerte auf Änderung.

Die Ein- und Ausgänge sind nicht negierbar/invertierbar.

Erkennung einer Änderung

Bei jeder Bearbeitung des Bausteins werden der Reihe nach die aktuellen Eingangswerte an den Eingängen E0 … En-1 mit den Vergangenheitswerten (Eingangswerte der vorherigen Bausteinbearbeitung) verglichen. Wird an einem der Eingänge E0 … En-1 eine Änderung erkannt, so wird:

- dies am Ausgang AEND signalisiert,
- die Nummer des Eingangs, an dem die Änderung festgestellt wurde, am Ausgang NR ausgegeben,
- der sich geänderte Eingangswert am Ausgang A ausgegeben.

Pro Bearbeitung des Bausteins wird nur die Änderung an einem Eingang erkannt. Wird eine Änderung erkannt, so werden bei der nächsten Bearbeitung des Bausteins die Eingänge überwacht, die auf den Eingang folgen, an dem zuvor die Änderung festgestellt wurde.

Initialisierung der Vergangenheitswerte

Bei der ersten Bearbeitung nach der SPS-Initialisierung (FREI = TRUE) bzw. Freigabe der Bearbeitung nachdem sie gesperrt war (FREI wechselt von FALSE nach TRUE), werden alle aktuellen Eingangswerte einmalig als Vergangenheitswerte übernommen, und alle Ausgänge werden auf den Wert 0 gesetzt. Diese initialisierten Vergangenheitswerte stellen nun die Ausgangsbasis zur Erkennung von Änderungen dar.

FREI **BOOL**

Mit dem Eingang FREI wird die Bearbeitung des Bausteins freigegeben.

FREI = FALSE → Baustein wird nicht bearbeitet
FREI = TRUE → Bearbeitung des Bausteins ist freigegeben

Ist FREI = FALSE, dann werden auch die Ausgänge des Bausteins nicht mehr aktualisiert.

RESET **BOOL**

Mit dem Eingang RESET kann der Baustein zurückgesetzt werden (Reset).

RESET = FALSE → kein Reset
RESET = TRUE → Reset des Bausteins

Reset bedeutet:

- Übernahme der aktuellen Werte an den Eingängen E0 … En-1 als Vergangenheitswerte.
- Alle Ausgänge werden auf den Wert 0 bzw. FALSE gesetzt.

n **INT**

Am Eingang n wird die Anzahl der zu überwachenden Werte an den Eingängen E0 … En-1 angegeben.

Bereich für n: $1 \leq n \leq$ max. Anzahl (8..127)

E0 .. En-1 **BOOL**

An den Eingängen E0 … En-1 werden die auf Änderung zu überwachenden Operanden angegeben.

NR **INT**

Am Ausgang NR wird die laufende Nummer des Eingangs E0 … En-1 ausgegeben, an dem eine Änderung festgestellt wurde. Wird bei der Bearbeitung des Bausteins keine Änderung festgestellt, so wird am Ausgang NR weiterhin die Nummer des Eingangs ausgegeben, der sich zuletzt geändert hat.

Es gilt die Zuordnung:

Änderung festgestellt an E0 → NR = 0
Änderung festgestellt an E1 → NR = 1
Änderung festgestellt an En-1 → NR = n-1

A **BOOL**

Wird an einem der Eingänge E0 … En-1 eine Änderung erkannt, so wird der Eingangswert, der sich geändert hat, dem Ausgang A zugewiesen. Wird bei der Bearbeitung des Bausteins keine Änderung an den Eingängen E0 … En-1 festgestellt, so wird am Ausgang A weiterhin der Wert des Eingangs ausgegeben, der sich zuletzt geändert hat.

AEND **BOOL**

Am Ausgang AEND wird angezeigt, ob eine Änderung an den Eingängen E0 … En-1 erkannt wurde.

AEND = FALSE→ keine Änderung festgestellt
AEND = TRUE → Änderung festgestellt

Beispiel

Deklaration:

```
BMELD_8 : BMELD8;
BMELD_FREI AT %MX0.0 : BOOL;
BMELD_RESET AT %MX0.1 : BOOL;
BMELD_E0 AT %MX1.0 : BOOL;
BMELD_E1 AT %MX1.1 : BOOL;
BMELD_E2 AT %MX1.2 : BOOL;
BMELD_E3 AT %MX1.3 : BOOL;
BMELD_E4 AT %MX1.4 : BOOL;
BMELD_E5 AT %MX1.5 : BOOL;
BMELD_E6 AT %MX1.6 : BOOL;
BMELD_E7 AT %MX1.7 : BOOL;
BMELD_NR AT %MW1010.0 : INT;
BMELD_A AT %MX0.2 : BOOL;
BMELD_AEND AT %MX0.3 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
BMELD
FREI
RESET
#n
E0
E1
E2
E3
E4
E5
E6
E7
NR
A
AEND
```

FUP:

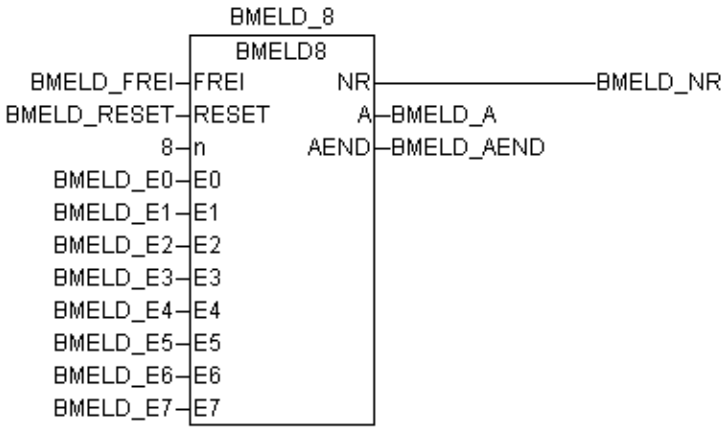


ABB AWL des Beispiels:

```
!BA 0
BMELD
M0,0
M0,1
M1,0
M1,1
M1,2
M1,3
M1,4
M1,5
M1,6
M1,7
MW10,0
M0,2
M0,3
```

Funktionsaufruf in AWL

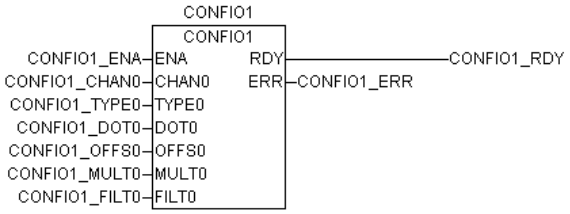
```
CAL   BMELD_1(FREI := BMELD_FREI,
              RESET := BMELD_RESET, n := 8,
              E0 := BMELD_E0, E1 := BMELD_E1,
              E2 := BMELD_E2, E3 := BMELD_E3,
              E4 := BMELD_E4, E5 := BMELD_E5,
              E6 := BMELD_E6, E7 := BMELD_E7 )

LD    BMELD_1.NR
ST    BMELD_NR
LD    BMELD_1.A
ST    BMELD_A
LD    BMELD_1.AEND
ST    BMELD_AEND
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
BMELD_1   ( FREI := BMELD_FREI,
            RESET := BMELD_RESET, n := 8,
            E0 := BMELD_E0, E1 := BMELD_E1,
            E2 := BMELD_E2, E3 := BMELD_E3,
            E4 := BMELD_E4, E5 := BMELD_E5,
            E6 := BMELD_E6, E7 := BMELD_E7 );

BMELD_NR    := BMELD_1.NR;
BMELD_A     := BMELD_1.A;
BMELD_AEND  := BMELD_1.AEND;
```

Konfiguration Analog Kanal CONFIO(1..8) S40



Der Funktionsbaustein CONFIO1 dient zum

- Konfigurieren des Typs (Spannung, Strom oder BALCO500/NI1000/PT100/PT1000) eines analogen Kanals auf den AC31-Erweiterungen.

- Ändern der Filterzeit des analogen Eingangs.

- Ändern der Einheit des Anzeigewertes.

- Verriegeln oder Entriegeln der Konfiguration aller analogen Kanäle einer analogen Erweiterung.

CONFIO1	Konfiguration 1 analoger Kanal
CONFIO4	Konfiguration 4 analoge Kanäle
CONFIO8	Konfiguration 8 analoge Kanäle

Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	CONFIO(1..8)	Instanzname
ENA	BOOL	Baustein-Freigabe
CHAN0..8	INT	Kanal-Erkennung
TYPE0..8	INT	Typ des analogen Kanals
DOT0..8	INT	Punktposition des Anzeigewertes
OFFS0..8	INT	Offset-Wert für den Anzeigewert
MULT0..8	INT	Wert für die Multiplikation des Anzeigewertes
FILT0..8	INT	Filterzeit
RDY	BOOL	Bearbeitung der Konfiguration ist abgeschlossen
ERR	BOOL	Ein Fehler wurde festgestellt.

Beschreibung

Der Funktionsbaustein CONFIO dient zum:

- Konfigurieren des Typs (Spannung, Strom oder BALCO500/NI1000/PT100/PT1000) eines analogen Kanals auf den AC31-Erweiterungen.
- Ändern der Filterzeit des analogen Eingangs
- Ändern der Einheit des Anzeigewertes.
- Verriegeln oder Entriegeln der Konfiguration aller analogen Kanäle einer analogen Erweiterung

Alternativ zum Drucktaster auf der Frontplatte der analogen Erweiterung wird der Funktionsbaustein zu Einstellung der Konfiguration des analogen Kanals im Anwenderprogramm verwendet.

Die erfolgte Konfiguration wird in einem internen EEPROM der analogen Erweiterung gespeichert.

Die Einheit des Anzeigewertes wird entsprechend der folgenden Formel eingestellt:

Anzeigewert = int-Wert * MULT0 / 32767 + OFFS0

Es wird die zuletzt konfigurierte Kanalnummer einer analogen Erweiterung angezeigt.

ENA BOOL

Der Funktionsbaustein wird mit einen steigenden Flanke FALSE->TRUE am Eingang ENA bearbeitet.

CHAN0 INT

Der analoge Kanal, der konfiguriert werden soll, wird hier direkt angegeben.

Zum Beispiel %IW1000.0 für den analogen Eingang 0 auf der analogen Erweiterung mit der Adresse 0, %QW1065.01 für den analogen Ausgang 1 auf der analogen Erweiterung mit der Adresse 65.

TYPE0 **INT**

Typ des analogen Signals:

0 : der Kanal wird auf +/- 10 V eingestellt

1 : der Kanal wird auf 0-20mA eingestellt

2 : der Kanal wird auf 4-20mA eingestellt

3 : der Kanal wird auf Pt100 eingestellt

4 : der Kanal wird auf Pt1000 eingestellt

5 : der Kanal wird auf Pt100, dreiadrig eingestellt

6 : der Kanal wird auf Pt1000, dreiadrig eingestellt

8 : der Kanal wird auf Ni1000 eingestellt

9 : der Kanal wird auf BALCO500 eingestellt

14 : Entriegeln der Konfiguration über den Drucktaster an der Frontplatte der analogen Erweiterung mit der Adresse xx, wenn CHAN0 %IW10xx.yy oder %QW10xx.yy ist.

15 : Verriegeln der Konfiguration über den Drucktaster an der Frontplatte der analogen Erweiterung mit der Adresse xx, wenn CHAN0 %IW10xx.yy oder %QW10xx.yy ist.

Die Konfiguration über den Drucktaster wird nach Einschalten der Spannungsversorgung automatisch entriegelt.

DOT0 **INT**

Position des Punktes auf der Anzeige:

0 : 4 Ziffern werden ohne Punkt angezeigt
Beispiel: Wert=1234, Anzeige: 1234

1 : 4 Ziffern werden mit Punkt an Position 1 angezeigt
Beispiel: Wert=1234, Anzeige: 123.4

2 : 4 Ziffern werden mit Punkt an Position 2 angezeigt
Beispiel: Wert=1234, Anzeige: 12.34

3 : 4 Ziffern werden mit Punkt an Position 3 angezeigt
Beispiel: Wert=1234, Anzeige: 1.234

Bei DOT0 < 0 oder DOT0 > 3 ist das Bit ERR = TRUE und die Funktion wird nicht bearbeitet.

OFFS0 **INT**

Wert des Offsets:

$-32767 \leq OFFS0 \leq 32767$

MULT0 **INT**

Wert der Multiplikation:

$-32767 \leq MULT0 \leq 32767$

Bei MULT0 = 0 werden die Parameter OFFS0 und DOT0= nicht verwendet. In diesem Fall wird die werkseitige Einheit eingestellt.

Mit dem Parameter MULT0 kann ein Kanalwert auf der Anzeige ausgegeben werden.

FILT0 **INT**

Filterzeit:

0 : interner Filter, wie in der Dokumentation der analogen Erweiterung beschrieben

1 - 127 : Anzahl Integrationen

160 : schnelle Refreshzeit (50 ms anstatt der Standardeinstellung 120 ms)

192 : 60 Hz-Filter

224 : 50 Hz-Filter

Dieser Parameter beeinflußt alle Kanäle einer Erweiterung.

RDY **BOOL**

Dieses Bit wird während der Bearbeitung der Funktion auf FALSE gesetzt.

ERR **BOOL**

Dieses Bit wird während eines Zyklus auf TRUE gesetzt (das Bit RDY wird gleichzeitig auch auf TRUE gesetzt).

Ein Fehler wird festgestellt wenn:

- ein Parameterwert falsch ist
- der analoge Kanal nicht existiert
- ein Kommunikationsproblem zwischen Zentraleinheit und analoger Erweiterung besteht

Hinweis: Von der Funktion CONFIO1 wird ein Vergangenheitswert verwendet

Beispiel

Deklaration:

```
CONFIO1 : CONFIO1;
CONFIO1_ENA AT %MX0.0 : BOOL;
CONFIO1_CHAN0 AT %MW1000.0 : INT;
CONFIO1_TYPE0 AT %MW1000.1 : INT;
CONFIO1_DOT0 AT %MW1000.2 : INT;
CONFIO1_OFFS0 AT %MW1000.3 : INT;
CONFIO1_MULT0 AT %MW1000.4 : INT;
CONFIO1_FILT0 AT %MW1000.5 : INT;
CONFIO1_RDY AT %MX0.1 : BOOL;
CONFIO1_ERR AT %MX0.2 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
CONFIO
#1
ENA
CHAN0
TYPE0
DOT0
OFFS0
MULT0
FILT0
RDY
ERR
```

FUP:

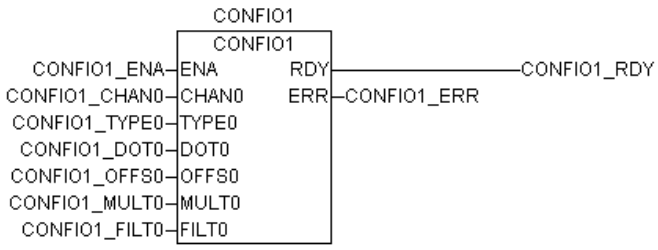


ABB AWL des Beispiels:

```
!BA 0
CONFIO
#1
M0,0
MW0,0
MW0,1
MW0,2
MW0,3
MW0,4
MW0,5
M0,1
M0,2
```

Funktionsaufruf in AWL

```
CAL   CONFIO1(ENA := CONFIO1_ENA,
              CHAN0 := CONFIO1_CHAN0,
              TYPE0 := CONFIO1_TYPE0,
              DOT0 := CONFIO1_DOT0,
              OFFS0 := CONFIO1_OFFS0,
              MULT0 := CONFIO1_MULT0,
              FILT0 := CONFIO1_FILT0)

LD    CONFIO1.RDY
ST    CONFIO1_RDY
LD    CONFIO1.ERR
ST    CONFIO1_ERR
```

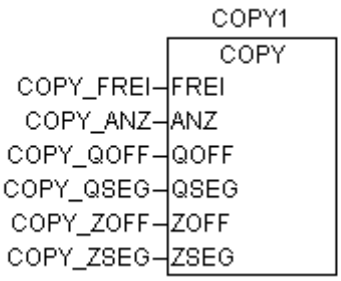
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
CONFIO1   (ENA := CONFIO1_ENA,
           CHAN0 := CONFIO1_CHAN0,
           TYPE0 := CONFIO1_TYPE0,
           DOT0 := CONFIO1_DOT0,
           OFFS0 := CONFIO1_OFFS0,
           MULT0 := CONFIO1_MULT0,
           FILT0 := CONFIO1_FILT0);

CONFIO1_RDY   := CONFIO1.RDY;
CONFIO1_ERR   := CONFIO1.ERR ;
```

Kopieren von Speicherbereichen COPY S40

Der Baustein kopiert n Worte aus einem Quell-Speicherbereich in einen Ziel-Speicherbereich.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	COPY	Instanzname
FREI	BOOL	Baustein-Freigabe
ANZ	INT	Anzahl (n) der zu kopierenden Worte
QOFF	INT	Offset-Adresse des Anfangs des Quellbereichs
QSEG	INT	Segment-Adresse des Anfangs des Quellbereichs
ZOFF	INT	Offset-Adresse des Anfangs des Zielbereichs
ZSEG	INT	Segment-Adresse des Anfangs des Zielbereichs

Beschreibung

Der Funktionsbaustein kopiert n Worte aus einem Quell-Speicherbereich in einen Ziel-Speicherbereich.

Der Inhalt des Quell-Speicherbereichs wird dabei nicht verändert.

Der Anfang des Quell- und des Ziel-Speicherbereichs wird an den Bausteineingängen jeweils durch die Offset- und Segment-Adresse angegeben.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar/invertierbar.

FREI **BOOL**

Baustein-Freigabe
FREI = FALSE → der Baustein wird nicht bearbeitet
FREI = TRUE → der Baustein wird bearbeitet

ANZ **INT**

Anzahl n der zu kopierenden Worte.

Es gilt: $0 \leq n \leq +8000_H$
n = 0: Kein Kopiervorgang
n = 8000H: Ein ganzes Segment (64 kByte) wird kopiert

QOFF **INT**

Offset-Adresse des Quellbereich-Anfangs.

QSEG **INT**

Segment-Adresse des Quellbereich-Anfangs.

ZOFF **INT**

Offset-Adresse des Zielbereich-Anfangs.

ZSEG **INT**

Segment-Adresse des Zielbereich-Anfangs.

Beispiel

Deklaration:

```
COPY1 : COPY;
COPY_FREI AT %MX0.0 : BOOL;
COPY_ANZ AT %MW3002.0 : INT := 16;
COPY_QOFF AT %MW3002.1 : INT := 16#1C00; (*MW00,00*)
COPY_QSEG AT %MW3002.2 : INT := 0;
COPY_ZOFF AT %MW3002.3 : INT := 16#1C40; (*MW03,00*)
COPY_ZSEG AT %MW3002.4 : INT := 0;
```

Übersetzung in ABB AWL:

```
!BA 0
COPY
FREI
ANZ
QOFF
QSEG
ZOFF
ZSEG
```

FUP:

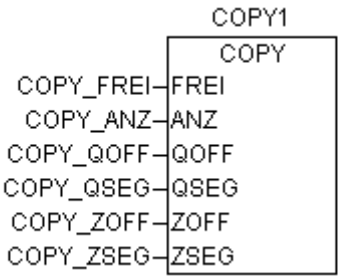


ABB AWL des Beispiels:

```
!BA 0
COPY
M0,0
KW2,0          ; 16
KW2,1          ; #H1C00
KW2,2          ; 0
KW2,3          ; #H1C40
KW2,4          ; 0
```

Funktionsaufruf in AWL

```
CAL   COPY1(FREI := COPY_FREI,
      ANZ := COPY_ANZ,
      QOFF := COPY_QOFF,
      QSEG := COPY_QSEG,
      ZOFF := COPY_ZOFF,
      ZSEG := COPY_ZSEG)
```

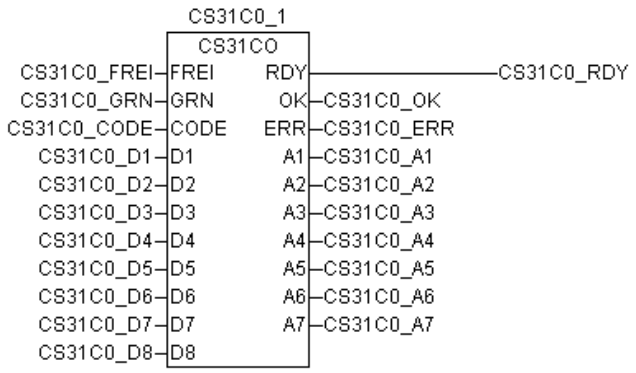
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
COPY1 (FREI := COPY_FREI,
      ANZ := COPY_ANZ,
      QOFF := COPY_QOFF,
      QSEG := COPY_QSEG,
      ZOFF := COPY_ZOFF,
      ZSEG := COPY_ZSEG);
```

AC31-Module konfigurieren CS31CO S40

Der Baustein dient zum Konfigurieren der AC31-Vorortmodule. Der Baustein kann sowohl Konfigurationsparameter an die Vorortmodule schicken, als auch deren aktuell eingestellte Konfiguration abfragen.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	CS31CO	Instanzname
FREI	BOOL	Freigabe (FALSE/TRUE-Flanke) zur Bearbeitung des Bausteins
GRN	INT	Gruppennummer des Vorortmoduls, auf den sich der Auftrag bezieht
CODE	INT	Kennung des durchzuführenden Auftrages
D1	INT	Erster Parameter des Auftrages
.	.	.
.	.	.
D8	INT	Achter Parameter des Auftrages
RDY	BOOL	Bearbeitung des Auftrages ist abgeschlossen
OK	BOOL	Auftrag konnte korrekt bearbeitet werden
ERR	INT	Fehlermeldung/Statusmeldung
A1	INT	Erster Parameter der Antwort
.	.	.
.	.	.
A7	INT	Siebter Parameter der Antwort

Beschreibung

Der Baustein dient zum Konfigurieren der AC31-Vorortmodule. Der Baustein kann sowohl Konfigurationsparameter an die Vorortmodule schicken, als auch deren aktuell eingestellte Konfiguration abfragen.

Außer der Konfiguration der AC31-Vorortmodule kann der Baustein noch weitere Aufträge bearbeiten (siehe Liste der Aufträge).

Die Freigabe zur einmaligen Bearbeitung eines Auftrages erfolgt durch eine FALSE/TRUE-Flanke am Eingang FREI.

Am Eingang CODE wird die gewünschte Auftragskennung angegeben.

Die für den Auftrag erforderlichen Parameter werden an den Eingängen D1 … D8 projektiert.

An den Ausgängen RDY, OK und ERR werden Statusmeldungen signalisiert.

Die Antwortdaten des Auftrages stehen an den Ausgängen A1 … A7 zur Verfügung.

Die Bearbeitung des Auftrages kann mehrere SPS-Zyklen dauern.

FREI BOOL

Über den Eingang FREI wird die Bearbeitung des Bausteins gesteuert.

FREI = FALSE:
Alle Bausteinausgänge werden auf den Wert »FALSE« gesetzt.
Dies gilt aber nicht während einer gerade laufenden Auftragsabwicklung, d. h. die Bearbeitung eines gerade laufenden Auftrages wird durch FREI = FALSE nicht beeinflußt.

FREI = FALSE/TRUE-Flanke:
Die Bearbeitung des Auftrages wird freigegeben.
Während der Bearbeitung des Auftrages wird der Eingang FREI nicht mehr ausgewertet.

FREI = TRUE:
Der Baustein wird nicht bearbeitet, d. h. der Baustein verändert seine Ausgänge nicht mehr. Dies gilt aber nicht während einer gerade laufenden Auftragsabwicklung.

GRN INT

Gruppennummer, mit der das Vorortmodul vom Automatisierungsprogramm aus angesprochen wird.

Bereich: 0 … 63

Beispiel:
Beim Binäreingang E 12,08 ist »12« die Gruppennummer und »08« die Kanalnummer.

CODE INT

Am Eingang CODE wird die Kennung des auszuführenden Auftrages angegeben (siehe Liste der Aufträge auf der nächsten Seite).

D1…D8 INT

An den Eingängen D1 … D8 werden die für den Auftrag erforderlichen Parameter vorgegeben. Die Anzahl der Parameter hängt vom durchzuführenden Auftrag ab. Es gibt auch Aufträge, die keine Parameter benötigen (siehe dazu Liste der Aufträge auf der nächsten Seite).

RDY BOOL

Der Ausgang RDY signalisiert, daß die Bearbeitung des laufenden Auftrages abgeschlossen ist. Der Ausgang RDY macht keine Aussage darüber, ob die Bearbeitung des Auftrages mit Erfolg durchgeführt werden konnte oder nicht. Der Ausgang RDY ist deshalb immer zusammen mit dem Ausgang OK zu betrachten.

RDY = TRUE und OK = TRUE:
Die Bearbeitung des Auftrages wurde fehlerfrei abgewickelt. Ein neuer Auftrag kann mit einer FALSE/TRUE-Flanke am Eingang FREI gestartet werden.

RDY = TRUE und OK = FALSE:
Bei der Bearbeitung des Auftrages wurde ein Fehler festgestellt. Am Ausgang ERR steht die zugehörige Fehlerkennung zur Verfügung. Ein neuer Auftrag kann mit einer FALSE/TRUE-Flanke am Eingang FREI gestartet werden.

RDY = FALSE:
Die Bearbeitung eines freigegebenen Auftrages ist noch nicht abgeschlossen (Auftrag läuft noch) oder mit FREI = FALSE wurde der Ausgang RDY zurückgesetzt.

OK BOOL

Der Ausgang OK signalisiert, ob der Auftrag erfolgreich abgewickelt wurde, oder ob bei der Bearbeitung ein Fehler festgestellt wurde. Im Fehlerfall wird am Ausgang ERR eine Fehlernummer angezeigt. Der Ausgang OK ist erst gültig, wenn der Auftrag abgeschlossen ist d. h. wenn RDY = TRUE ist.

Es gilt:

Wenn RDY = TRUE und
OK = TRUE: Der Auftrag wurde erfolgreich bearbeitet.
OK = FALSE: Bei der Bearbeitung des Auftrages wurde ein Fehler festgestellt.

ERR INT

Am Ausgang ERR werden Status- und Fehlerkennungen ausgegeben. Die Statuskennungen werden während der Bearbeitung eines Auftrages ausgegeben, um zu signalisieren in welchem Bearbeitungsstadium sich der Auftrag gerade befindet. Nach der Freigabe eines Auftrages werden Statuskennungen also nur solange signalisiert, wie RDY = FALSE ist.

Die Fehlerkennungen werden nach Abschluß der Auftragsbearbeitung ausgegeben, falls ein Fehler aufgetreten ist. Fehlerkennungen werden also erst dann signalisiert, wenn

RDY = TRUE und
OK = FALSE

Fehlerkennungen

ERR = 1: Es wurde am Eingang CODE eine unzulässige Auftragskennung angegeben.

ERR = 2: Es wurden an den Eingängen D1 … D8 falsche Parameter angegeben (z. B. eine Gruppennummer, zu der es kein Vorortmodul am CS31-Systembus gibt).

ERR = 3: Das angesprochene AC31-Vorortmodul akzeptiert den Auftrag nicht.

Statuskennungen

ERR = 8: Der Baustein wartet, weil momentan gerade ein Auftrag eines anderen Auftraggebers bearbeitet wird.

ERR = 10: Der Auftrag wurde an den Empfänger abgeschickt und der Baustein wartet auf dessen Antwort.

A1…A7 INT

An den Ausgängen A1 … A7 steht nach Abschluß der Auftragsbearbeitung die Antwort zur Verfügung. Die Anzahl der Antwortparameter hängt vom ausgeführten Auftrag ab (siehe Liste der Aufträge).

Liste der Aufträge

Die Abwicklung eines Auftrages besteht aus:
- dem Verschicken des Auftrages und
- der Bereitstellung der OK-Antwort bzw. Nicht-OK-Antwort

Die OK-Antwort wird beim jeweiligen Auftrag beschrieben.
Die Nicht-OK-Antwort der einzelnen Aufträge sieht immer wie folgt aus:

RDY: TRUE
OK: FALSE
ERR:
1. unzulässige Auftragskennung
2. falscher Parameter, z. B. Gruppennummer, zu der es kein Vorortmodul gibt
3. Vorortmodul akzeptiert den Auftrag nicht

A1 … A7: 0

- **Aktualisierung der maximalen Anzahl erkannter Vorortmodule**

  Im Eingang INT EW 07,15 steht u. a. die maximale Anzahl der bisher erkannten Vorortmodule. Die momentane wirkliche Anzahl der vorhandenen Vorortmodule kann geringer sein. Mit diesem Kommando wird dieser Wert aktualisiert. Die vorhandenen Module werden gezählt und der Wert wird abgelegt. Der Anwender kann diesen Wert im SPS-Programm (EW 07,15, Bit 8 … 15) abfragen.

  – Auftrag
  GRN: 255 (Master-SPS mit Bus)
  CODE: 132
  D1 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1 … A7: 0

- **Abfrage, ob bei einem Eingang die Drahtbruchüberwachung ein- oder ausgeschaltet ist**

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 32
  D1: Kanalnummer
  D2 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1: 47. Drahtbruchüberwachung EIN
  32. Drahtbruchüberwachung AUS
  A2 … A7: 0

- **Abfrage, ob bei einem Ausgang die Drahtbruchüberwachung ein- oder ausgeschaltet ist**

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 33
  D1: Kanalnummer
  D2 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1: 47. Drahtbruchüberwachung EIN
  32. Drahtbruchüberwachung AUS
  A2 … A7: 0

- **Drahtbruchüberwachung eines Eingangs ein- bzw. ausschalten**

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 224. Drahtbruchüberwachung EIN
  160. Drahtbruchüberwachung AUS
  D1: Kanalnummer
  D2 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1 … A7: 0

- **Drahtbruchüberwachung eines Ausgangs ein- bzw. ausschalten**

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 225. Drahtbruchüberwachung EIN
  161. Drahtbruchüberwachung AUS
  D1: Kanalnummer
  D2 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1 … A7: 0

- **Abfrage, ob ein Kanal als Eingang oder als Eingang/Ausgang konfiguriert ist**

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 34
  D1: Kanalnummer
  D2 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1: 34. Eingang
  35. Eingang/Ausgang
  A2 … A7: 0

- **Konfiguration eines Kanals als Eingang oder Eingang/Ausgang**

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 162. Eingang
  163. Eingang/Ausgang
  D1: Kanalnummer
  D2 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1 … A7: 0

- **Abfragen der Eingangsverzögerung eines Kanals**

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 38
  D1: Kanalnummer
  D2 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1: Eingangsverzögerung:
  2. 2 ms
  4. 4 ms
  :
  30. 30 ms
  32. 32 ms
  A2 … A7: 0

- **Einstellen der Eingangsverzögerung eines Kanals**

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 166
  D1: Kanalnummer
  D2: Eingangsverzögerung
  2. 2 ms
  4. 4 ms
  :
  30. 30 ms
  32. 32 ms

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1 … A7: 0

- **Fehler auf Vorortmodul quittieren**

  Mit diesem Kommando werden die auf dem gewählten Vorortmodul registrierten Fehlermeldungen zurückgesetzt. Ein Rücksetzen ist nur möglich, wenn die Fehlerursache nicht mehr wirksam ist.

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 232
  D1: kleinste Kanalnummer auf dem Modul:
  0. kleinste Kanalnummer auf dem Modul ist 0 (<7)
  8. kleinste Kanalnummer auf dem Modul ist 8 (>7)
  D2: Modultyp:
  0. Binäre Eingabe
  1. Analoge Eingabe
  2. Binäre Ausgabe
  3. Analoge Ausgabe
  4. Binäre Ein-/Ausgabe
  5. Analoge Ein-/Ausgabe
  Hinweis:
  Bit: gerade Zahl (0, 2, 4)
  Wort: ungerade Zahl (1, 3, 5)
  D3 … D8: nicht benutzt

  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1 … A7: 0

- **Fehler auf Vorortmodul quittieren und Konfigurationswerte auf Standardeinstellung (Default) zurücksetzen**
  Zusätzlich zum Auftrag »Fehler auf Vorortmodul quittieren« werden alle konfigurierbaren Einstellungen auf die Standardeinstellung zurückgesetzt.

  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 233
  D1: erste Kanalnummer auf dem Modul:
  0. erste Kanalnummer auf dem Modul ist 0 (<7)
  8. erste Kanalnummer auf dem Modul ist 8 (>7)
  D2: Modultyp:
  0. Binäre Eingabe
  1. Analoge Eingabe
  2. Binäre Ausgabe
  3. Analoge Ausgabe
  4. Binäre Ein-/Ausgabe

5. Analoge Ein-/Ausgabe
Hinweis:
Bit: gerade Zahl (0, 2, 4)
Wort: ungerade Zahl (1, 3, 5)

D3 … D8: nicht benutzt

– OK-Antwort
RDY: TRUE
OK: TRUE
A1 … A7: 0

- **Abfragen der Konfiguration eines analogen Eingangs**
  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 42
  D1: Kanalnummer
  D2 … D8: nicht benutzt
  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1: 50. Eingang 0 … 20 mA
  49. Eingang 4 … 20 mA
  A2 … A7: 0

- **Abfragen der Konfiguration eines analogen Ausgangs**
  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 43
  D1: Kanalnummer
  D2 … D8: nicht benutzt
  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1: 50. Ausgang 0 … 20 mA
  49. Ausgang 4 … 20 mA
  51. Ausgang + 10V
  A2 … A7: 0

- **Konfiguration eines analogen Eingangs**
  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 170
  D1: Kanalnummer
  D2: 50. Eingang 0 … 20 mA
  49. Eingang 4 … 20 mA
  D3 … D8: nicht benutzt
  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1 … A7: 0

- **Konfiguration eines analogen Ausgangs**
  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 171
  D1: Kanalnummer
  D2: 50. Ausgang 0 … 20 mA
  49. Ausgang 4 … 20 mA
  51. Ausgang +10 V
  D3 … D8: nicht benutzt
  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1 … A7: 0

- **Abfragen der Buskonfiguration**
  Die Busanschaltung der Master-SPS verfügt über eine Liste, in der bestimmte Daten der Vorortmodule abgelegt sind. Die Vorortmodule sind in dieser Liste in der Reihenfolge numeriert, in welcher sie am CS31-Systembus aufgefunden wurden. Bei diesem Kommando muß die interne Nummer der Module angegeben werden. Als Antwort erhält man die unter dieser Nummer gespeicherte Gruppennummer sowie Statusinformationen zu dem entsprechenden Modul.
  – Auftrag
  GRN: wird nicht ausgewertet
  CODE: 80
  D1: Nummer aus der Modulliste (1 … 31)
  D2 … D8: nicht benutzt
  – OK-Antwort
  RDY: TRUE
  OK: TRUE
  A1: Status des Vorortmoduls:
  Bit 0 … 3: Anzahl der Prozeßdaten-Bytes (Binärmodul) bzw. Worte (Wortmodul), die das Modul dem Master sendet.
  Bit 4 ... 7: Anzahl der Prozeßdaten-Bytes (Binärmodul) bzw. Worte (Wortmodul), die der Master dem Modul sendet.
  A2: Gruppennummer (0 … 63)
  A3: Bit 0: 0. kleinste Kanalnummer <7
  1. kleinste Kanalnummer >7
  Bit 1: 0. Binärmodul
  1. Wortmodul
  A4 … A7: 0

- **1 … 6 Bytes lesen**
  – Auftrag
  GRN: Gruppennummer 0 … 63
  CODE: 49. 1 Byte lesen
  50. 2 Bytes lesen
  51. 3 Bytes lesen
  52. 4 Bytes lesen
  53. 5 Bytes lesen
  54. 6 Bytes lesen
  D1: erste Kanalnummer auf dem Modul:
  0. erste Kanalnummer auf dem Modul ist 0 (<7)
  8. erste Kanalnummer auf dem Modul ist 8 (>7)

D2: Modultyp:
0. Binäre Eingabe
1. Analoge Eingabe
2. Binäre Ausgabe
3. Analoge Ausgabe
4. Binäre Ein-/Ausgabe
5. Analoge Ein-/Ausgabe
Hinweis:
Bit: gerade Zahl (0, 2, 4)
Wort: ungerade Zahl (1, 3, 5)
D3: Byte-Anfangsadresse (Low Byte)
D4: Byte-Anfangsadresse (High Byte)
D5 … D8: nicht benutzt

– OK-Antwort
RDY: TRUE
OK: TRUE
A1: Wert des 1. Byte
A2: Wert des 2. Byte oder 0
A3: Wert des 3. Byte oder 0
A4: Wert des 4. Byte oder 0
A5: Wert des 5. Byte oder 0
A6: Wert des 6. Byte oder 0
A7: 0

- **1 Bit eines Byte lesen**

– Auftrag
GRN: Gruppennummer 0 … 63
CODE: 63
D1: erste Kanalnummer auf dem Modul:
0. erste Kanalnummer auf dem Modul ist 0 (<7)
8. erste Kanalnummer auf dem Modul ist 8 (>7)
D2: Modultyp:
0. Binäre Eingabe
1. Analoge Eingabe
2. Binäre Ausgabe
3. Analoge Ausgabe
4. Binäre Ein-/Ausgabe
5. Analoge Ein-/Ausgabe
Hinweis:
Bit: gerade Zahl (0, 2, 4)
Wort: ungerade Zahl (1, 3, 5)
D3: Byte-Anfangsadresse (Low Byte)
D4: Byte-Anfangsadresse (High Byte)
D5: Bitposition innerhalb des Byte 0 … 7
D6 ... D8: nicht benutzt

– OK-Antwort
RDY: TRUE
OK: TRUE
A1: Bitwert (0 oder 1)
A2 … A7: 0

- **1 … 4 Bytes schreiben**

– Auftrag
GRN: Gruppennummer 0 … 63
CODE: 65. 1 Byte schreiben
66. 2 Bytes schreiben
67. 3 Bytes schreiben
68. 4 Bytes schreiben
D1: erste Kanalnummer auf dem Modul:
0. erste Kanalnummer auf dem Modul ist 0 (<7)
8. erste Kanalnummer auf dem Modul ist 8 (>7)
D2: Modultyp:
0. Binäre Eingabe
1. Analoge Eingabe
2. Binäre Ausgabe
3. Analoge Ausgabe
4. Binäre Ein-/Ausgabe
5. Analoge Ein-/Ausgabe
Hinweis:
Bit: gerade Zahl (0, 2, 4)
Wort: ungerade Zahl (1, 3, 5)
D3: Byte-Anfangsadresse (Low Byte)
D4: Byte-Anfangsadresse (High Byte)
D5: Wert des 1. Byte
D6: Wert des 2. Byte oder nicht benutzt
D7: Wert des 3. Byte oder nicht benutzt
D8: Wert des 4. Byte oder nicht benutzt

– OK-Antwort
RDY: TRUE
OK: TRUE
A1 … A7: 0

- **1 Bit eines Byte schreiben**

– Auftrag
GRN: Gruppennummer 0 … 63
CODE: 79
D1: erste Kanalnummer auf dem Modul:
0. erste Kanalnummer auf dem Modul ist 0 (<7)
8. erste Kanalnummer auf dem Modul ist 8 (>7)
D2: Modultyp:
0. Binäre Eingabe
1. Analoge Eingabe
2. Binäre Ausgabe
3. Analoge Ausgabe
4. Binäre Ein-/Ausgabe
5. Analoge Ein-/Ausgabe
Hinweis:
Bit: gerade Zahl (0, 2, 4)
Wort: ungerade Zahl (1, 3, 5)
D3: Byte-Anfangsadresse (Low Byte)
D4: Byte-Anfangsadresse (High Byte)
D5: Bitposition innerhalb des Byte 0 … 7
D6: Bitwert (0 oder 1)
D7 … D8: nicht benutzt

– OK-Antwort
RDY: TRUE
OK: TRUE
A1 … A7: 0

Beispiel

Deklaration:

```
CS31CO_1 : CS31CO;
CS31CO_FREI AT %MX0.0 : BOOL;
CS31CO_GRN AT %MW1001.0 : INT;
CS31CO_CODE AT %MW1001.1 : INT;
CS31CO_D1 AT %MW1002.0 : INT;
CS31CO_D2 AT %MW1002.1 : INT;
CS31CO_D3 AT %MW1002.2 : INT;
CS31CO_D4 AT %MW1002.3 : INT;
CS31CO_D5 AT %MW1002.4 : INT;
CS31CO_D6 AT %MW1002.5 : INT;
CS31CO_D7 AT %MW1002.6 : INT;
CS31CO_D8 AT %MW1002.7 : INT;
CS31CO_RDY AT %MX0.1 : BOOL;
CS31CO_OK AT %MX0.2 : BOOL;
CS31CO_ERR AT %MW1001.2 : INT;
CS31CO_A1 AT %MW1002.8 : INT;
CS31CO_A2 AT %MW1002.9 : INT;
CS31CO_A3 AT %MW1002.10 : INT;
CS31CO_A4 AT %MW1002.11 : INT;
CS31CO_A5 AT %MW1002.12 : INT;
CS31CO_A6 AT %MW1002.13 : INT;
CS31CO_A7 AT %MW1002.14 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
CS31CO
FREI
GRN
CODE
D1
D2
D3
D4
D5
D6
D7
D8
RDY
OK
ERR
A1
A2
A3
A4
A5
A6
A7
```

FUP:

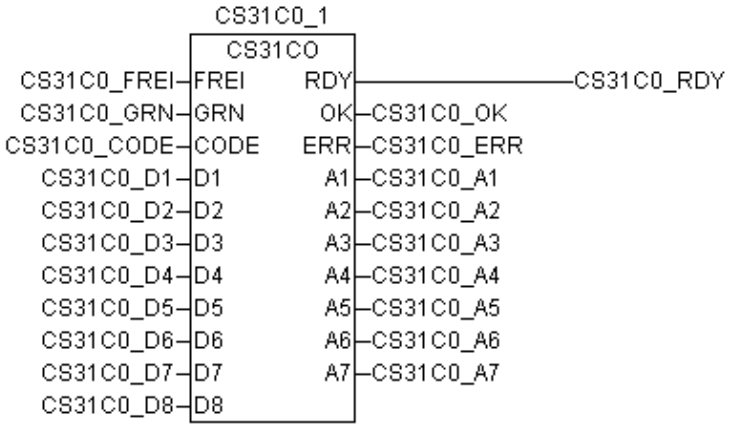


ABB AWL des Beispiels:

```
!BA 0
CS31CO
M0,0
MW1,0
MW1,1
MW2,0
MW2,1
MW2,2
MW2,3
MW2,4
MW2,5
MW2,6
MW2,7
M0,1
M0,2
MW1,2
MW2,8
MW2,9
MW2,10
MW2,11
MW2,12
MW2,13
MW2,14
```

Funktionsaufruf in AWL

```
CAL   CS31CO1(FREI := CSCO_FREI,
      GRN := CSCO_GRN,
      CODE := CSCO_CODE,
      D1 := CSCO_D1,
      D2 := CSCO_D2,
      D3 := CSCO_D3,
      D4 := CSCO_D4,
      D5 := CSCO_D5,
      D6 := CSCO_D6,
      D7 := CSCO_D7,
      D8 := CSCO_D8)

LD    CS31CO1.OK
ST    CS31CO_OK
LD    CS31CO1.ERR
ST    CS31CO_ERR
LD    CS31CO1.A1
ST    CS31CO_A1
LD    CS31CO1.A2
ST    CS31CO_A2
LD    CS31CO1.A3
ST    CS31CO_A3
LD    CS31CO1.A4
ST    CS31CO_A4
LD    CS31CO1.A5
ST    CS31CO_A5
LD    CS31CO1.A6
ST    CS31CO_A6
LD    CS31CO1.A7
ST    CS31CO_A7
LD    CS31CO1.RDY
ST    CS31CO_RDY
```

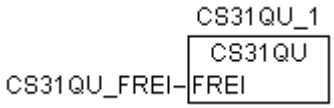
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
CS31CO1     (FREI := CSCO_FREI,
            GRN := CSCO_GRN,
            CODE := CSCO_CODE,
            D1 := CSCO_D1,
            D2 := CSCO_D2,
            D3 := CSCO_D3,
            D4 := CSCO_D4,
            D5 := CSCO_D5,
            D6 := CSCO_D6,
            D7 := CSCO_D7,
            D8 := CSCO_D8);

CS31CO_OK   := CS31CO1.OK;
CS31CO_ERR  := CS31CO1.ERR;
CS31CO_A1   := CS31CO1.A1;
CS31CO_A2   := CS31CO1.A2;
CS31CO_A3   := CS31CO1.A3;
CS31CO_A4   := CS31CO1.A4;
CS31CO_A5   := CS31CO1.A5;
CS31CO_A6   := CS31CO1.A6;
CS31CO_A7   := CS31CO1.A7;
CS31CO_RDY  := CS31CO1.RDY;
```

Fehlerquittierung an AC31-Modulen CS31QU S40

Mit diesem Baustein können Fehlermeldungen von AC31-Vorortmodulen automatisch quittiert werden.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	CS31QU	Instanzname
FREI	BOOL	Freigabe der Bausteinbearbeitung

Beschreibung

Mit diesem Funktionsbaustein können FK3- und FK4-Fehlermeldungen von AC31-Vorortmodulen quittiert werden. Fehlermeldungen werden auf den AC31-Vorortmodulen solange gespeichert, bis sie quittiert werden. Selbst wenn der Fehler beseitigt ist, steht die Fehlermeldung auf dem Modul bis zur Quittierung an und wird auch solange an das Automatisierungsgerät gemeldet.

Mit einem TRUE-Signal am Eingang FREI wird die Bearbeitung des Bausteins freigegeben. Dieser führt dann solange Quittierungen von AC31-Fehlern durch, bis der Eingang FREI von TRUE nach FALSE wechselt.

Die Quittierung eines Fehlers auf einem AC31-Modul kann mehrere SPS-Zyklen dauern.

Ist der Baustein freigegeben, so prüft er ständig, ob ein AC31-Fehler der Klasse 3 oder 4 vorliegt und quittiert diesen.

Beispiele:

1. **Es liegt ein AC31-Fehler der Klasse 3 vor:**
   Der Baustein quittiert den Fehler auf dem AC31-Vorortmodul, das den Fehler meldet und löscht auch die Fehlermeldung auf der SPS, d. h. der Fehlermerker M 255,13 / MX255.13 wird zurückgesetzt, und die Leuchtdiode FK3 wird ausgeschaltet.

   Beispiel für einen FK3-Fehler:
   - Ein Vorortmodul wird vom CS31-Systembus abgetrennt.

2. **Es liegt ein AC31-Fehler der Klasse 4 vor:**
   Der Baustein quittiert den Fehler auf dem AC31-Vorortmodul, das den Fehler meldet und löscht auch die Fehlermeldung auf der SPS, d. h. der Fehlermerker M 255,14 / MX255.14 wird zurückgesetzt.

   Beispiel für einen FK4-Fehler:
   Ein Vorortmodul meldet Drahtbruch.

Empfehlung:

Den Eingang FREI z. B: über einen Quittiertaster ansteuern.

Beispiel

Deklaration:

```
CS31QU_1 : CS31QU;
CS31QU_FREI AT %IX62.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
CS31QU
FREI
```

FUP:

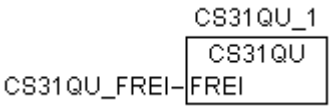


ABB AWL des Beispiels:

```
!BA 0
CS31QU
E62,0
```

Funktionsaufruf in AWL

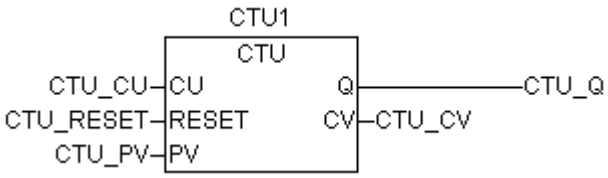
```
CAL     CS31QU_1(FREI := CS31QU_FREI)
```

Funktionsaufruf in ST

```
CS31QU_1(FREI := CS31QU_FREI);
```

Vorwärts-Zähler CTU S40

Der Funktionsbaustein dient zum Zählen von Impulsen.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	CTU	Instanzname
CU	BOOL	Impulseingang
RESET	BOOL	Zähler-Rücksetzeingang
PV	INT	Obere Zählergrenze
Q	BOOL	Grenzwertmelder
CV	INT	Zählerwert

Beschreibung

Jede positive Flanke (FALSE->TRUE-Flanke) am Eingang CU erhöht den momentanen Wert am Ausgang CV um 1.

CU BOOL

Das Impulssignal wird dem Eingang CU zugeordnet. Jeweils die positive Flanke des Impulses wird ausgewertet.

R BOOL

Ein 1-Signal am Eingang R setzt den Zählerstand auf den Wert 0. Der Reset-Eingang R hat die höchste Priorität.

PV INT

Die obere Grenze des Zählers wird am Eingang PV angegeben.

Q BOOL

Am Ausgang Q wird angezeigt, ob der Zählerwert höher ist als der Wert am Eingang PV oder nicht.

CV >= PV -> Q = TRUE

CV < PV -> Q = FALSE

CV INT

Am Ausgang CV steht der aktuelle Zählerstand zur Verfügung. Erreicht der Zählerstand die positive bzw. negative Grenze des Zahlenbereichs, so wird der Zählerstand auf diesen Wert begrenzt.

Zahlenbereich

Untere Grenze: 0

Obere Grenze: $7FFF_H$ +32767

Beispiel

Deklaration:

```
CTU1: CTU;
CTU_CU AT %MX1.0: BOOL;
CTU_RESET AT %MX0.0 : BOOL;
CTU_PV %MW3002.0 : INT := 1000;
CTU_Q AT %MX1.1 : BOOL;
CTU_CV AT %MW1002.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
CTU
CU
RESET
PV
Q
CV
```

FUP:

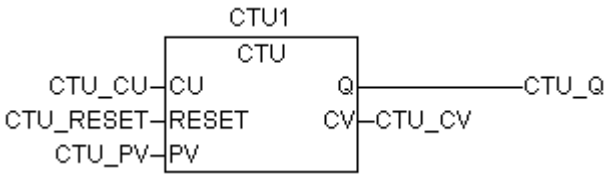


ABB AWL des Beispiels:

```
!BA 0
CTU
M1,0
M0,0
KW2,0                ; 1000
M1,1
MW2,2
```

Funktionsaufruf in AWL

```
CAL   CTU_1(CU := CTU_CU,
      RESET := CTU_RESET,
      PV := CTU_PV)

LD    CTU_1.Q
ST    CTU_Q
LD    CTU_1.CV
ST    CTU_CV
```

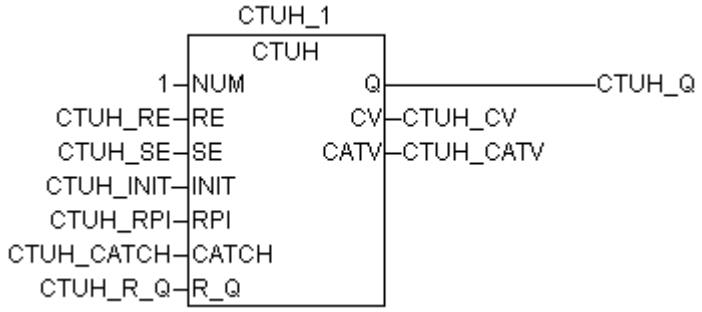
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
CTU     ( CU= CTU_CU,
        RESET:= CTU_RESET,
        PV := CTU_PV);

CTU_Q   := CTU_1.Q;
CTU_CV  := CTU_.CV;
```

Schneller Zähler CTUH S40

Der Funktionsbaustein CTUH ermöglicht das Zählen von schnellen Impulsen mit Geräten der Serie 40 und 50.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	CTUH	Instanzname
NUM	INT	Zähler-Betriebsart
RE	BOOL	Zähler-Rücksetzeingang
SE	BOOL	Zähler-Setzeingang
INIT	INT	Setzwert
RPI	BOOL	Rücksetzpunkt-Melder
CATCH	BOOL	Erfassung des Zählerstands
R_Q	BOOL	Rücksetzen Bit-Überlauf
Q	BOOL	Überlauf
CV	INT	Zählerstand
CATV	INT	Erfaßter Zählerstand

Beschreibung

Geräte der Serie 40..50 verfügen über zwei schnelle Zähler, die in den folgenden **Betriebsarten** verwendet werden können:

- C1 : Zählen am Eingang %IX62.00

Zählerstart: bei positiver Flanke (FALSE->TRUE)
Darstellung: 1 INT (16 Bit)
Überlauf: beim Wechsel von -1 nach 0.
Erfassung: bei positiver Flanke des Eingangs %IX62.02

- C2 : Zählen am Eingang %IX62.01

Zählerstart: bei positiver Flanke (FALSE->TRUE)
Darstellung: 1 INT (16 Bit)
Überlauf: beim Wechsel von -1 nach 0.
Erfassung: bei positiver Flanke des Eingangs %IX62.03

- Inkremental-Encoder: Zählen an den Eingängen %IX62.00 und %IX62.01

Zählerstart: bei positiver Flanke (FALSE/TRUE)
Darstellung: 1 INT (16 Bit)
Überlauf: beim Wechsel von -1 nach 0 oder von 0 nach -1
Erfassung: bei positiver Flanke des Eingangs %IX62.02

Ist einer der Kanäle defekt (z.B. ein Eingang nicht angeschlossen) wechselt der Wert zwischen +1 und –1.

NUM DIREKTE KONSTANTE

Die Zähler-Betriebsart wird am Eingang NUM festgelegt.

1 = Zähler-Betriebsart C1
2 = Zähler-Betriebsart C2
3 = Inkremental-Encoder
>3 = der Baustein wird nicht bearbeitet

R BOOL

Ein TRUE-Signal am Eingang R setzt den Zählerstand und das Erfassungsregister auf den Wert 0. Der Reset-Eingang R hat die höchste Priorität.
Ist R = TRUE sind CV = 0 und CATV = 0

S BOOL

Ein TRUE-Signal am Eingang S lädt den Zählerstand mit dem an Eingang INIT anliegenden Vorgabewert.
Ist S = TRUE dann ist CV = INIT

INIT **INT**

Am Eingang INIT wird der Vorgabewert angegeben.

RPI **BOOL**

Ein TRUE-Signal am Eingang RPI bestätigt das Erfassen des Zählerstandes und das Rücksetzen des Zählers während der Erfassung. Der Eingang RPI hat höhere Priorität als CATCH.

RPI = TRUE Die Erfassung ist auf allen Zählern gültig.

Liegt an %IX62,02 oder %IX62,03 eine positive Flanke an, erfolgt eine harte Erfassung des Zählers. Der Zähler wird auf 0 zurückgesetzt.

CATCH **BOOL**

Ein TRUE-Signal am Eingang CATCH bestätigt die Erfassung des Zählerstandes.

CATCH = 0 Erfassung nicht gültig.

CATCH = 1 Die Erfassung ist auf allen Zählern gültig.

Liegt an %IX62,02 oder %IX62,03 eine positive Flanke an, erfolgt eine harte Erfassung des Zählers. Der Zähler wird nicht auf 0 zurückgesetzt.

R_Q **BOOL**

Ein TRUE-Signal am Eingang R_Q setzt den Überlauf auf den Wert FALSE zurück.

Ist R_Q = TRUE dann ist Q = FALSE

Q **BOOL**

Der Überlauf wird an Ausgang Q angegeben.

Q = TRUE wenn CV von –1 nach 0 oder von 0 nach –1 wechselt.

CV **INT**

Am Ausgang CV steht der aktuelle Zählerstand zur Verfügung.

CATV **INT**

Ist CATCH = TRUE, steht am Ausgang CATV der aktuelle Zählerstand zur Verfügung.

Zahlenbereich:

Integer Wort (16 Bit)

Untere Grenze: 8000_H - 32768

Obere Grenze: $7FFF_H$ + 32767

Beispiel

Deklaration:

```
CTUH_1: CTUH;
CTUH_RE AT %MX1.0: BOOL;
CTUH_SE AT %MX1.1: BOOL;
CTUH_INIT AT %MW3002.0: INT := 100;
CTUH_RPI AT %MX1.2: BOOL;
CTUH_CATCH AT %MX1.3: BOOL;
CTUH_R_Q AT %MX1.4: BOOL;
CTUH_Q AT %MX1.5: BOOL;
CTUH_CV AT %MW1010.0: INT;
CTUH_CATV AT %MW1010.1: INT;
```

Übersetzung in ABB AWL:

```
!BA 0
CTUH
#NUM
RE
SE
INIT
RPI
CATCH
R_Q
Q
CV
CATV
```

FUP:

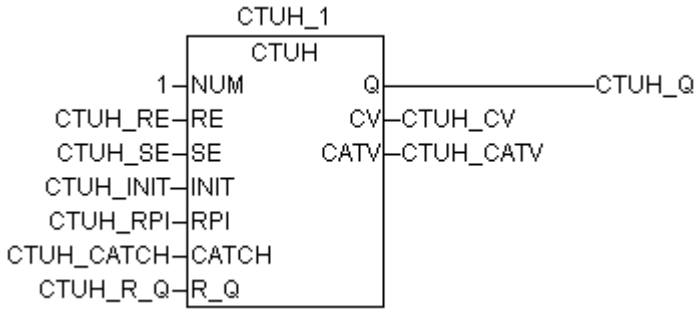


ABB AWL des Beispiels:

```
!BA 0
CTUH
#1
M1,0
M1,1
KW2,0                ; 100
M1,2
M1,3
M1,4
M1,5
MW10,0
MW10,1
```

Funktionsaufruf in AWL

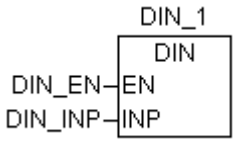
```
CAL   CTUH1( NUM := 01,
             RE := CTUH_RE,
             SE := CTUH_SE,
             INIT := CTUH_INIT,
             RPI := CTUH_RPI,
             CATCH := CTUH_CATCH,
             R_Q := CTUH_R_Q)

LD    CTUH1.Q
ST    CTUH_Q
LD    CTUH1.CV
ST    CTUH_CV
LD    CTUH1.CATV
ST    CTUH_CATV
```

Funktionsaufruf in ST

```
CTUH1      ( NUM := 01,
             RE := CTUH_RE,
             SE := CTUH_SE,
             INIT := CTUH_INIT,
             RPI := CTUH_RPI,
             CATCH := CTUH_CATCH,
             R_Q := CTUH_R_Q);

CTUH_Q     := CTUH1.Q;
CTUH_CV    := CTUH1.CV;
CTUH_CATV  := CTUH1.CATV;
```

Direkte Eingänge lesen DIN S40

Der Funktionsbaustein DIN liest EINEN direkten Eingang der Zentraleinheit und deren Erweiterungen ein. Am Eingang INP wird der zu lesende direkte Eingang angegeben.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	DIN	Instanzname
EN	BOOL	Freigabe
INP	BOOL	Direkter Eingang

Beschreibung

Diese Funktion kann nur mit Zentraleinheiten der Serie 40..50 verwendet werden.
Der Funktionsbaustein DIN liest EINEN direkten Eingang der Zentraleinheit und deren Erweiterungen ein. Am Eingang INP wird der zu lesende direkte Eingang angegeben.

Dieser Funktionsbaustein ist hilfreich:

- Bei einer langen Zykluszeit.

- Bei einer hohen Auslastung der Zentraleinheit.

Am Anfang jedes Programmzyklus erstellt der SPS-Verarbeiter automatisch ein aktuelles Prozeßabbild der Eingänge.

Innerhalb eines Programmzyklus kann der Funktionsbaustein DIN den am Eingang INP angegebenen physikalischen Eingangswert laden. Dies kann in Verbindung mit speziellen Applikationen erforderlich sein, um Signaländerungen an den Eingängen öfter als einmal pro Programmzyklus zu erkennen und zu verarbeiten.

- Wenn sich der zu lesende Eingang auf der Zentraleinheit befindet:

Der neue Eingangswert kann sofort eingelesen werden.

- Wenn sich der zu lesende Eingang auf einer zentralen Erweiterungseinheit befindet:

Die DIN-Funktion liest den aktuellen Eingangswert ein und aktualisiert gleichzeitig den Erweiterungsbus. Der Eingangswert steht pro zentralem Erweiterungsmodul min 1-2 ms verzögert zur Verfügung.

Wenn DIN in einer Interrupt-Routine benutzt wird (z. B: zyklischer Interrupt), wird nicht der augenblickliche Wert, sondern der vorhergehende Wert, (der vorhergehenden Interrupt-Routine) eingelesen.

Hinweis:
Vorort-Eingänge werden mit jeder Zykluszeit aktualisiert. Der Funktionsbaustein DIN kann nicht für Vorortmodule verwendet werden.

Beispiel

Deklaration:

```
DIN_1 : DIN;
DIN_EN AT %MX0.0 : BOOL;
DIN_INP AT %IX62.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
DI
EN
INP
```

FUP:

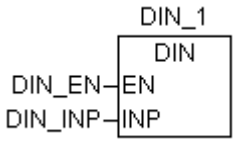


ABB AWL des Beispiels:

```
!BA 0
DI
M0,0
E62,0
```

Funktionsaufruf in AWL

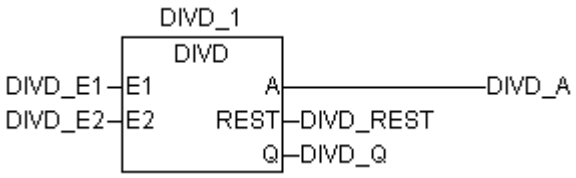
```
CAL   DIN_1(EN := DIN_EN,
      INP := DIN_INP)
```

Funktionsaufruf in ST

```
DIN_1   (EN := DIN_EN,
        INP := DIN_INP);
```

Division Doppelwort DIVD S40

Der Wert des Operanden am Eingang E1 wird durch den Wert des Operanden am Eingang E2 dividiert und das Ergebnis dem Operanden am Ausgang A, der Rest dem Operanden am Ausgang REST zugewiesen. Wenn ein Rest entsteht, wird das Ergebnis stets abgerundet. Liegt das Ergebnis außerhalb des zulässigen Zahlenbereichs, wird es auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt. Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein TRUE-Signal und dem Ausgang REST der Wert 0 zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein FALSE-Signal zugewiesen.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	DIVD	Instanzname
E1	DINT	Dividend
E2	DINT	Divisor
A	DINT	Ergebnis (Quotient)
REST	DINT	Rest
Q	BOOL	Ergebnis begrenzt

Beschreibung

Der Wert des Operanden am Eingang E1 wird durch den Wert des Operanden am Eingang E2 dividiert und das Ergebnis dem Operanden am Ausgang A, der Rest dem Operanden am Ausgang REST zugewiesen. Wenn ein Rest entsteht, wird das Ergebnis stets abgerundet. Liegt das Ergebnis außerhalb des zulässigen Zahlenbereichs, wird es auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt (Zahlenbereich: -2147483647 (8000 0001_H) … 2147483647 (7FFF $FFFF_H$)). Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein TRUE-Signal und dem Ausgang REST der Wert 0 zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein FALSE-Signal zugewiesen.

Somit wird auch eine Division durch »Null« am Binärausgang Q signalisiert.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Begrenzung des Divisors

Der Divisor E2 ist auf den Bereich von -32767...+32767 begrenzt.

Restbehandlung

Entsteht bei der Division ein Rest, so steht dieser am Doppelwortausgang REST zur Verfügung. Bei der Division wird das Ergebnis stets abgerundet, falls ein Rest entsteht.

Beispiel: 3 : 3 = 1 Rest 0
4 : 3 = 1 Rest 1
5 : 3 = 1 Rest 2
6 : 3 = 2 Rest 0

Da der Rest am Ausgang REST zur Verfügung steht, kann der Anwender diesen mit dem Divisor vergleichen und das Ergebnis am Ausgang A nach Bedarf selbst runden.

Beispiel:
Rest > Divisor/2
→ Ergebnis an A aufrunden

Division durch »Null«

Hat der Divisor den Wert »Null«, so wird dem Ausgang A der positive bzw. negative Grenzwert des Zahlenbereiches zugewiesen.

Für die Division durch »Null« gilt:

A = -2147483647 (8000 0001_H) falls Dividend negativ
A = +2147483647 (7FFF $FFFF_H$) falls Dividend positiv

REST = 0 Ausgang für Rest

Q = TRUE Ausgang für Signalisierung, daß der Wert am Ausgang A begrenzt wurde

Unzulässiger Ergebniswert

Entsteht bei der Division als Ergebnis der unzulässige Wert 8000 0000_H, so wird dieser auf den zulässigen Grenzwert 8000 0001_H (-2147483647) korrigiert, der Binärausgang Q auf den Wert TRUE und der Ausgang REST auf den Wert 0 gesetzt.

Beispiel

Deklaration:

```
DIVD_1 : DIVD;
DIVD_E1 AT %MD2002.0 : DINT;
DIVD_E2 AT %MD2002.1 : DINT;
DIVD_A AT %MD2002.2 : DINT;
DIVD_REST AT %MD2002.3 : DINT;
DIVD_Q AT %MX0.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
DIVD
E1
E2
A
REST
Q
```

FUP:

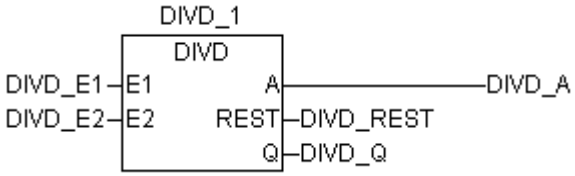


ABB AWL des Beispiels:

```
!BA 0
DIVD
MD2,0
MD2,1
MD2,2
MD2,3
M0,0
```

Funktionsaufruf in AWL

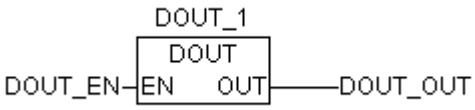
```
CAL   DIVD1(E1 := DIVD_E1,
      E2 := DIVD_E2)

LD    DIVD1.REST
ST    DIVD_REST
LD    DIVD1.Q
ST    DIVD_Q
LD    DIVD1.A
ST    DIVD_A
```

Funktionsaufruf in ST

```
DIVD1       (E1 := DIVD_E1,
            E2 := DIVD_E2);

DIVD_REST   :=DIVD1.REST;
DIVD_Q      :=DIVD1.Q;
DIVD_A      :=DIVD1.A;
```

Direkte Ausgänge schreiben DOUT S40

Der Funktionsbaustein DOUT schreibt EINEN direkten Ausgang der Zentraleinheit und deren Erweiterungen. Am Eingang OUT wird der zu lesende direkte Ausgang angegeben.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	DOUT	Instanzname
EN	BOOL	Freigabe
OUT	BOOL	Direkter Ausgang

Beschreibung

Der Funktionsbaustein DOUT schreibt EINEN direkten Ausgang der Zentraleinheit oder deren Erweiterungen. Am Ausgang OUT wird der zu schreibende direkte Ausgang angegeben.

Dieser Funktionsbaustein ist hilfreich:
- Bei einer langen Zykluszeit.
- Bei einer hohen Auslastung der Zentraleinheit.

Am Ende jedes Programmzyklus gibt der SPS-Verarbeiter automatisch ein Prozeßabbild der direkten Ausgänge aus, das während des Programmzyklus aktualisiert wurde. Innerhalb eines Programmzyklus kann der Funktionsbaustein DOUT den am Parameter OUT angegebenen physikalischen Ausgang setzen. Dies kann in Verbindung mit speziellen Applikationen erforderlich sein, um Ausgangs-Signaländerungen öfter als einmal pro Programmzyklus zur Verfügung zu haben.

- Wenn sich der zu schreibende Ausgang auf der Zentraleinheit befindet:

Der neue Ausgangswert kann sofort geschrieben werden.

- Wenn sich der zu schreibende Ausgang auf einer Erweiterungseinheit befindet:

Der neue Ausgangswert kann mit einer Verzögerung von max. 2 ms geschrieben werden.

Hinweis:
Dezentrale Vorort-Ausgänge werden mit jedem Zyklus aktualisiert. Der Funktionsbaustein DOUT kann nicht für dezentrale Vorortmodule verwendet werden.

Beispiel

Deklaration:

```
DOUT_1 : DOUT;
DOUT_EN AT %MX0.0 : BOOL;
DOUT_OUT AT %QX62.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
DO
EN
OUT
```

FUP:

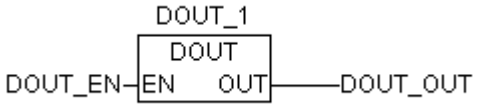


ABB AWL des Beispiels:

```
!BA 0
DO
M0,0
A62,0
```

Funktionsaufruf in AWL

```
CAL   DOUT_1(EN := DOUT_EN)

LD    DOUT_1.OUT
ST    DOUT_OUT
```

Funktionsaufruf in ST

```
DOUT_1      (EN := DOUT_EN);

DOUT_OUT    := DOUT_1.OUT;
```

Ausgabe von ASCII-Zeichen DRUCK S40

Unter Verwendung des Funktionsbausteins DRUCK kann eine Zentraleinheit der Serie 40..50 ASCII-Meldungen über die serielle Schnittstelle RS232 senden.

Jede Meldung hat eine Kennung und kann ASCII-Texte und Operandenwerte enthalten.

Die Text- und Operandenkennungen, die ausgegeben werden sollen, werden vom Baustein DRUCK direkt im Anwenderprogramm der SPS gespeichert. Durch die Vergabe einer Formatkennung werden die auszugebenden Zahlenwerte für vielfältige Darstellungsformen aufbereitet.

Der Baustein DRUCK steht nur in AWL und ST zur Verfügung!

```
{S40Inline
!BA 0
DRUCK
FREI
TXNR
SSK
RDY
TX
}
```

Bausteintyp

ABB AWL-Programmbaustein in Pragma

Parameter

	ABB AWL-Typ	Beschreibung
FREI	BOOL	Freigabesignal für die Ausgabe eines Textes (FALSE/TRUE-Flanke)
SSK	INT	Kennung der seriellen Schnittstelle
TXNR	INT	Nummer des auszugebenden Textes
RDY	BOOL	Ready
TX	Texte	Doppelbare Texte/Operanden

Beschreibung

WICHTIGER HINWEIS:
Initialisierung der seriellen Schnittstelle

Bevor der Baustein DRUCK mit der seriellen Schnittstelle kommunizieren kann, muß er mit dem Baustein SINT initialisiert werden.

Kommunikation mehrerer DRUCK-Bausteine mit derselben seriellen Schnittstelle:

Mehrere DRUCK-Bausteine können die selbe serielle Schnittstelle benutzen. Wenn die serielle Schnittstelle von einem der DRUCK-Bausteine belegt ist, warten die anderen DRUCK-Bausteine automatisch, bis die Schnittstelle wieder frei ist. Haben mehrere DRUCK-Bausteine gleichzeitig Zugriff auf die selbe Schnittstelle, hängt die Priorität des Zugriffs von der Reihenfolge ab, in der die DRUCK-Bausteine im Anwenderprogramm aufgerufen werden. Der erste DRUCK-Baustein der sich am Anfang des Anwenderprogrammes befindet erhält den ersten Zugriff. Der Programmablauf muß mit einer geeigneten gegenseitigen Verriegelung der DRUCK-Bausteine projektiert werden.

Kommunikation eines DRUCK und eines EMAS-Bausteins mit derselben seriellen Schnittstelle:

Ein DRUCK-Baustein und ein EMAS-Baustein (Empfang von ASCII-Meldungen) können dieselbe serielle Schnittstelle benutzen, ohne daß besondere Vorkehrungen zu treffen sind.

FREI BOOL

Wenn der Baustein bereit ist (RDY = TRUE) und am Eingang FREI eine FALSE/TRUE-Flanke auftritt, wird der am Eingang TXNR anliegende Text über die am Eingang SSK definierte serielle Schnittstelle ausgegeben.

Erfolgt eine FALSE/TRUE-Flanke am Eingang FREI, obwohl der Ausgang RDY gleich FALSE ist (d.h. der Baustein ist noch nicht bereit für eine neue Übertragung), so wird die FALSE/TRUE-Flanke ignoriert. Solange das RDY-Signal auf FALSE ist, kann daher keine neue Textübertragung gestartet werden.

SSK INT

Zentraleinheiten können eine oder zwei serielle RS232-Schnittstellen haben. Am Eingang SSK wird die Nummer der Schnittstelle angegeben, über die der Text ausgegeben werden soll:

SSK = 1 für COM1
SSK = 2 für COM2

TXNR INT

Am Eingang TXNR wird die Nummer des auszugebenden Textes angegeben: 1 ≤ TXNR ≤ 99

Die Nummer des auszugebenden Textes muß solange am Eingang TXNR anstehen, bis der Baustein das Ende der Textübertragung mit einem TRUE-Signal an seinem Ausgang RDY anzeigt.

RDY BOOL

Beim Programmzyklus, bei dem der Baustein zum ersten Mal aufgerufen wird und während der Zeit, in der ein Text ausgegeben wird, ist RDY gleich FALSE. Solange RDY gleich FALSE ist, kann keine neue Textausgabe aktiviert werden und alle am Eingang FREI anstehende FALSE/TRUE-Flanken werden ignoriert und gehen verloren.

Nach dem ersten Aufruf des Bausteins bzw. nach Beendigung einer Textausgabe ist RDY gleich TRUE und der Baustein ist wieder bereit für eine neue Textausgabe.

FREI	RDY	TX NR	SSK	BEDEUTUNG
FALSE/ TRUE-Flanke	TRUE	2	1	Der Text mit der Nummer 2 wird über die Schnittstelle 1 ausgegeben
FALSE/ TRUE-Flanke	FALSE	2	1	FALSE/TRUE-Flanke wird ignoriert, da der Baustein nicht bereit ist
Keine FALSE/ TRUE-Flanke	X	X	X	Keine Ausgabe eines neuen Textes

=> Das RDY-Signal kann zum Beispiel am Eingang FREI benutzt werden, um eine neue Textübertragung zu aktivieren.

TX ALLE

Auszugebende Texte und Operanden werden direkt an den TX-Eingängen geschrieben. Der Aufbau von Meldungen ist weiter unten beschrieben.

Ablage von Texten in der Zentraleinheit:

Anzahl: 1...99 Meldungen

Länge: Max. 256 Zeichen
(wegen Sendebufferlänge = 256)

- Jedes Textzeichen zählt als 1 Zeichen
- Jede Formatkennung zählt als 3 Zeichen
- Jeder Bit-Operand zählt als 1 Zeichen
- Jeder Wort-Operand zählt als 1 bis 6 Zeichen *
- Jeder Doppelwort-Operand zählt als 10 bis 11 Zeichen *

* abhängig vom Format

Die SPS überprüft beim Programmstart die Texte auf Überschreitung der maximalen Länge.

Syntax von Texten:

Ein Text für den DRUCK-Baustein besteht aus:

- der Text-Nummer
- einem oder mehreren Teiltexten (optional)
- Operanden mit Formatkennung (optional)

#n: Nummer des einzugebenden Textes als direkte Konstante (1..99).

#" : Anfangs-Kennung für Texteingabe.

“# : Ende-Kennung für Texteingabe.

Teiltext: Alle ASCII-Zeichen (Hexadezimal-Code von 00 bis 7F).

Operand: BOOL, INT oder DINT-Operanden, deren Werte je nach Anzeigeformat ausgegeben werden.
=> die Operanden müssen im Format ABB AWL geschrieben werden!

Formatkennung:

Die SPS kann die Zahlenwerte auf verschiedene Arten auf einem Bildschirm oder Drucker ausgeben. Die Formatkennung gibt den Typ des Operanden und das Anzeigeformat seines Wertes an.

Sie besteht aus drei Ziffern und wird direkt vor der Ausgabe des Operanden projektiert. Die 1. Ziffer von links gibt den Operandentyp an. Es gibt drei Operandentypen:

BOOL: 1
INT: 2
DINT: 3

Die Ziffern 2 und 3 definieren das Anzeigeformat.

Beispiele:

- Formatkennung 103
Ziffer 1: 1 : BOOL-Operand
Ziffern 2 und 3: 03 : Anzeigeformat 03 (s. Tabelle)

- Formatkennung 204
Ziffer 1: 2 : INT-Operand
Ziffern 2 und 3: 04 : Anzeigeformat 04 (s. Tabelle)

- Formatkennung 341
Ziffer 1: 3 : DINT-Operand
Ziffern 2 und 3: 41 : Anzeigeformat 41 (s. Tabelle)

Mögliche Anzeigeformate:

In der folgenden Tabelle sind alle möglichen Anzeigeformate aufgelistet:

- Auf Wort-Datentypen anwendbare Kennungen:
01 bis 16, 21 bis 26, 33 bis 36, 42 bis 51 und 99

- Auf Doppelwort-Datentypen anwendbare Kennungen: 05 bis 16 und 41 bis 62

Es gibt Formate
– mit führenden Nullen und
– mit führenden Nullen, die durch Leerzeichen ersetzt werden. Diese sind in der untenstehenden Tabelle durch – dargestellt.

Formatkennung: Ausgabe ASCII

Zahlenbeispiel=0012345678

01	x	8
02	xx	78
03	xxx	678
04	xxxx	5678
05	xxxxx	45678

Zahlenbeispiel=1102215

06	xxxx,x	0221,5
07	xxx,xx	022,15
08	xx,xxx	02,215
09	x,xxxx	0,2215
10	,xxxxx	,02215

Zahlenbeispiel=00331

11	+/- xxxxx	+00331
12	+/- xxxx,x	+0033,1
13	+/- xxx,xx	+003,31
14	+/- xx,xxx	+00,331
15	+/- x,xxxx	+0,0331
16	+/- ,xxxxx	+,00331

Zahlenbeispiel=00234

21	xxxxx	--234

Zahlenbeispiel=00347

22	xxxx,x	--347
23	xxx,xx	--3,47
24	xx,xxx	--,347
25	x,xxxx	-,0347
26	,xxxxx	,00347

Zahlenbeispiel=00347

33	+/- xxx,xx	+--3,47
34	+/- xx,xxx	+--,347
35	+/- x,xxxx	+-,0347
36	+/- ,xxxxx	+,00347

Zahlenbeispiel=0012345678

41	xxxxxxxxxx	0012345678

Zahlenbeispiel=0011223344

42	xxxxxxxxx,x	001122334,4
43	xxxxxxxx,xx	00112233,44
44	xxxxxxx,xxx	0011223,344
45	xxxxxx,xxxx	001122,3344
46	xxxxx,xxxxx	00112,23344
47	xxxx,xxxxxx	0011,223344
48	xxx,xxxxxxx	001,1223344
49	xx,xxxxxxxx	00,11223344
50	x,xxxxxxxxx	0,011223344
51	,xxxxxxxxxx	,0011223344

Zahlenbeispiel=0055667788

52	+/- xxxxxxxxxx	+0055667788
53	+/- xxxxxxxxx,x	+005566778,8
54	+/- xxxxxxxx,xx	+00556677,88
55	+/- xxxxxxx,xxx	+0055667,788
56	+/- xxxxxx,xxxx	+005566,7788
57	+/- xxxxx,xxxxx	+00556,67788
58	+/- xxxx,xxxxxx	+0055,667788
59	+/- xxx,xxxxxxx	+005,5667788
60	+/- xx,xxxxxxxx	+00,55667788
61	+/- x,xxxxxxxxx	+0,055667788
62	+/- ,xxxxxxxxxx	+,0055667788

Sonderformat:
Ausgabe des HEX-Wertes eines Wort-Operanden:

Der Wert eines INT-Operanden wird direkt als Hexadezimalwert ausgegeben. Es erfolgt also vor der Ausgabe keine Wandlung des Wertes in ASCII-Darstellung.

98 Es wird nur das LOW BYTE (8 Bit) des Wort-Operanden ausgegeben

99 Es wird zuerst das LOW BYTE und anschließend das HIGH BYTE des Wort-Operanden ausgegeben.

Achtung: Dieses Sonderformat ist nur beim Datentyp "WORT" zulässig.
Zulässiges Format: 298 und 299
Unzulässiges Format: 198, 199, 398 und 399

- Eingabe von Texten:

Die folgenden Teile des Gesamttextes werden als eigenständige Operanden behandelt:
- die Text-Nummer z.B. # 1
- ein Teiltext z.B. #" Text1 "#
- eine Formatkennung z.B. # 203
- ein Operand z.B. MW002,03
- ein weiterer Teiltext z.B. #" Text2 "#

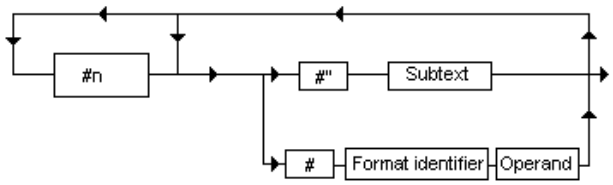
Eingabe von Sonderzeichen zur Bildschirm bzw. Druckersteuerung:

Zur Gliederung des Textes bei der Ausgabe auf einen Bildschirm oder Drucker sind Steuerzeichen wie z.B. "Zeilenvorschub" <LF> oder "Zeilen-Anfang" <CR> erforderlich. Diese Sonderzeichen können innerhalb eines Teiltextes an einer beliebigen Stelle eingefügt werden. Beim Programmiersystem erfolgt die Eingabe dieser Sonderzeichen durch:

\Zahlenwert des Zeichens

Der Zahlenwert des Zeichens wird als dreistellige Dezimalzahl angegeben.

Beispiel:



Auf einem Drucker soll folgende Ausgabe gemacht werden:

Erste Zeile
Leerzeile
Zweite Zeile

Dazu ist eine Projektierung des folgenden Textes notwendig:

Erste Zeile <CR> <LF> <LF> Zweite Zeile

Es gilt:

<CR> = 013
<LF> = 010

Die Eingabe des Textes in das Programmiersystem sieht wie folgt aus:

#"Erste Zeile\013\010\010Zweite Zeile"#

Hinweise:

- Die Zeichen mit ASCII-Code >20H (=32D) müssen über die Tastatur eingegeben werden. So muß beispielsweise anstatt \033 das diesem Code entsprechende Zeichen "!" eingegeben werden.

- Zeichen, die keine Sonderzeichen sind und auch nicht über die Tastatur eingegeben werden können, werden wie folgt erzeugt:
Drücken und gedrückt halten der Taste <ALT> und Eingabe des Zahlencodes (Dezimalcode) auf der numerischen Tastatur, loslassen der Taste <ALT>.

- Das Zeichen mit dem ASCII-Code 255 ist innerhalb der Programmiersoftware reserviert und darf nicht verwendet werden.

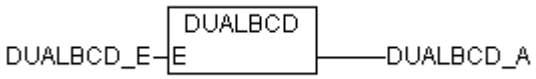
- Beispiel:

Text 1: Die Maschine ist bereit
Text 2: Die Maschine ist nicht bereit
Text 3: Pegel beträgt (%MW1001.1=>MW01,01)m, Temperatur beträgt (%MW1001,0=>MW01,00)°C.

In der Anweisungsliste sieht das Programm wie folgt aus:

```
LD 0
{S40Inline
!BA0
DRUCK
M0,0
MW00,00
MW00,01
A00,00
#01
#"\010\013 Die Maschine ist bereit: "#
#02
#"\010\013 Die Maschine ist nicht bereit. "#
#03
#"\010\013 Pegel beträgt "#
#202
MW01,01
#"m, Temperatur beträgt "#
#203
MW01,00
#"°C. "#
}
```

Dual nach BCD-Wandlung, Wort DUALBCD S40



Die Dualzahl am Eingang E wird in eine BCD codierte Zahl gewandelt und dem Operanden am Ausgang A zugewiesen.

Die Dualzahl ist in 16 Bit dargestellt und muß im Bereich $0 \leq E \leq 270F_H$ (entspricht BCD 9999) liegen. Liegt sie außerhalb dieses Bereichs, so wird die BCD-Zahl auf 9999 begrenzt. Die BCD-Zahl wird in einem 16-Bit-INT abgelegt.

Bausteintyp

Funktion

Parameter

E	INT	Dualzahl
(A)	INT	BCD-codierte Zahl

Beschreibung

Die Dualzahl am Eingang E wird in eine BCD codierte Zahl gewandelt und dem Operanden am Ausgang A zugewiesen.

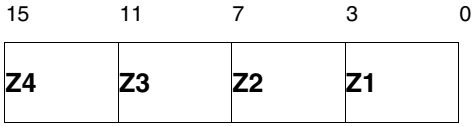
Die Dualzahl ist in 16 Bit dargestellt und muß im Bereich $0 \leq E \leq 270F_H$ (entspricht BCD 9999) liegen. Liegt sie außerhalb dieses Bereichs, so wird die BCD-Zahl auf 9999 begrenzt. Die BCD-Zahl wird in einem 16-Bit-INT abgelegt.

Ein- und Ausgang sind weder doppelbar noch negierbar.

Definition:

Die Wertigkeit der Ziffern einer Hexadezimalzahl und einer BCD-codierten Zahl sind wie folgt definiert:

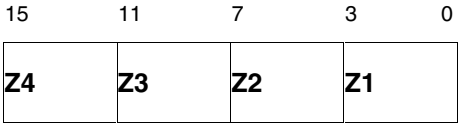
HEX-ZAHL



Zahlenwert:
Z1 * 1
Z2 * 16
Z3 * 256
Z4 * 4096
$0 \leq Zi \leq F$

BIT

BCD-ZAHL



Zahlenwert:
Z1 * 1
Z2 * 10
Z3 * 100
Z4 * 1000
$0 \leq Zi \leq 9$

Beispiel

HEX-ZAHL

15	11	7	3	0
0	**4**	**D**	**2**	

BIT ->

BCD-ZAHL

15	11	7	3	0
1	**2**	**3**	**4**	

Z1 = 2 * 1	=	2
Z2 = 13 * 16	=	208
Z3 = 4 * 256	=	1024
Z4 = 0 * 4096	=	0
		1234

Z1 = 4 * 1	=	4
Z2 = 3 * 10	=	30
Z3 = 2 * 100	=	200
Z4 = 1 * 1000	=	1000
		1234

Wandlung einer negativen Dualzahl in eine BCD-Zahl

Eine negative Dualzahl mit einem Betrag kleiner als $270F_H$ kann in eine BCD-Zahl gewandelt werden, wobei Wert und Vorzeichen der BCD-Zahl in jeweils einem Merker abgelegt werden.

Wandlung einer Dualzahl mit Betrag größer als $270F_H$

Für S40..50 nicht möglich.

Beispiel

Deklaration:

```
DUALBCD_E AT %MW1002.1 : INT;
DUALBCD_A AT %MW1002.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
BINBCD
E
A
```

FBD:

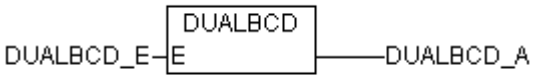


ABB AWL des Beispiels:

```
!BA 0
BINBCD
MW2,1
MW2,2
```

Funktionsaufruf in AWL

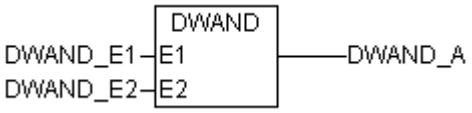
```
LD      DUAL_E
DUALBCD
ST      DUAL_A
```

Funktionsaufruf in ST

```
DUAL_A:= DUALBCD(DUAL_E);
```

UND-Verknüpfung, Doppelwort DWAND S40

Der Funktionsbaustein bildet bitweise die UND-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.



Bausteintyp

Funktion

Parameter

E1	DINT	Operand 1
E2	DINT	Operand 2
A	DINT	Ergebnis der UND-Verknüpfung

Beschreibung

Der Funktionsbaustein bildet bitweise die UND-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
DWAND_E1 AT %MD2000.0 : DINT;
DWAND_E2 AT %MD2000.1 : DINT;
DWAND_A AT %MD2000.2 : DINT;
```

Übersetzung in ABB AWL:

```
!BA 0
DWAND
E1
E2
A
```

FBD:

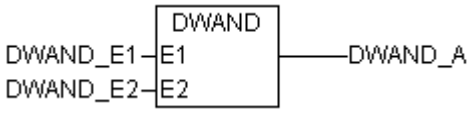


ABB AWL des Beispiels:

```
!BA 0
DWAND
MD0,0
MD0,1
MD0,2
```

Funktionsaufruf in AWL

```
LD      DWAND_E1
DWAND   DWAND_E2
ST      DWAND_A
```

Funktionsaufruf in ST

```
DWAND_A    := DWAND(E1 := DWAND_E1,
              E2 := DWAND_E2);
```

ODER-Verknüpfung, Doppelwort DWOR S40

Der Funktionsbaustein bildet bitweise die ODER-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.

Bausteintyp

Funktion

Parameter

E1	DINT	Operand 1
E2	DINT	Operand 2
A	DINT	Ergebnis der ODER-Verknüpfung

Beschreibung

Der Funktionsbaustein bildet bitweise die ODER-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
DWOR_E1 AT %MD2000.0 : DINT;
DWOR_E2 AT %MD2000.1 : DINT;
DWOR_A AT %MD2000.2 : DINT;
```

Übersetzung in ABB AWL:

```
!BA 0
DWOR
E1
E2
A
```

FBD:

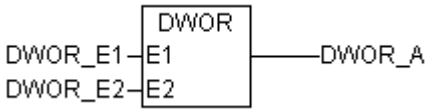


ABB AWL des Beispiels:

```
!BA 0
DWOR
MD0,0
MD0,1
MD0,2
```

Funktionsaufruf in AWL

```
LD      DWOR_E1
DWOR    DWOR_E2
ST      DWOR_A
```

Funktionsaufruf in ST

```
DWOR_A    := DWOR(E1 := DWOR_E1,
             E2 := DWOR_E2);
```

Doppelwort nach Wort-Wandlung DWW S40

Der Wert des Doppelwort-Operanden am Eingang E wird in eine Wortgröße gewandelt und das Ergebnis dem Wortoperanden am Ausgang A1 zugewiesen.

Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	DWW	Instanzname
E	DINT	Zu wandelnde Doppelwortgröße
A1	INT	Ergebnis der Wandlung, Wortgröße
A2	BOOL	Ergebnis begrenzt

Beschreibung

Der Wert des Doppelwort-Operanden am Eingang E1 wird in eine Wortgröße gewandelt, und das Ergebnis dem Wortoperanden am Ausgang A1 zugewiesen.

Das Ergebnis wird auf den maximalen bzw. minimalen Zahlenbereich begrenzt.

max. Zahlenbereich: +32767 ($7FFF_H$)
min. Zahlenbereich: -32767 (8001_H)

Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang A2 ein TRUE-Signal zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang A2 ein FALSE-Signal zugewiesen.

Der Eingang und die Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
DWW_1 : DWW
DWW_E AT %MD2000.0 : DINT;
DWW_A1 AT %MW1000.0 : INT;
DWW_A2 AT %MW1000.1 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
DWW
E
A1
A2
```

FBD:

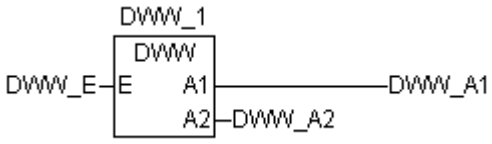


ABB AWL des Beispiels:

```
!BA 0
DWW
MD0,0
MW0,0
MW0,1
```

Funktionsaufruf in AWL

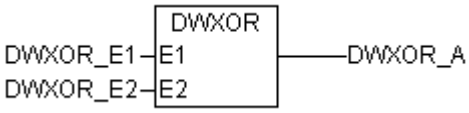
```
CAL   DWW_1(E := DWW_E)

LD    DWW_1.A2
ST    DWW_A2
LD    DWW_1.A1
ST    DWW_A1
```

Funktionsaufruf in ST

```
DWW_1     (E := DWW_E);

DWW_A2    := DWW_1.A2;
DWW_A1    := DWW_1.A1;
```

Exklusiv-ODER-Verknüpfung, Doppelwort DWXOR S40

Der Funktionsbaustein bildet bitweise die XOR-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.



Bausteintyp

Funktion

Parameter

E1	DINT	Operand 1
E2	DINT	Operand 2
A	DINT	Ergebnis der XOR-Verknüpfung

Beschreibung

Der Funktionsbaustein bildet bitweise die XOR-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
DWXOR_E1 AT %MD2000.0 : DINT;
DWXOR_E2 AT %MD2000.1 : DINT;
DWXOR_A AT %MD2000.2 : DINT;
```

Übersetzung in ABB AWL:

```
!BA 0
DWXOR
E1
E2
A
```

FBD:

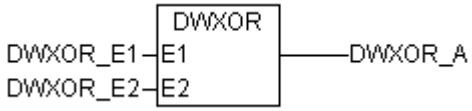


ABB AWL des Beispiels:

```
!BA 0
DWXOR
MD0,0
MD0,1
MD0,2
```

Funktionsaufruf in AWL

```
LD      DWXOR_E1
DWXOR   DWXOR_E2
ST      DWXOR_A
```

Funktionsaufruf in ST

```
DWXOR_A     := DWXOR(E1 := DWXOR_E1,
               E2 := DWXOR_E2);
```

Empfang von ASCII-Zeichen EMAS S40

Der Funktionsbaustein EMAS:

- empfängt Telegramme über eine serielle Schnittstelle der SPS,
- vergleicht diese Telegramme mit im Anwenderprogramm hinterlegten Vergleichstelegrammen
- und stellt bei Übereinstimmung die Nutzdaten des empfangenen Telegramms an den Bausteinausgängen zur Verfügung.

Der Baustein EMAS steht nur in AWL und ST zur Verfügung!

```
{S40Inline
!BA 0
EMAS
QUIT
SSK
#ANU
MEUN
RDY
TELN
MW0
VT0
}
```

Bausteintyp

ABB AWL-Programmbaustein in Pragma

Parameter

	ABB AWL-Typ	Beschreibung
QUIT	BOOL	Keine Freigabe zum Empfang von Telegrammen.
SSK	INT	Kennung der seriellen Schnittstelle
#ANU	Direkte Konstante	Anzahl der Ausgänge für Nutzinformationen im Format: #Nummer
MEUN	BOOL	Daten ungültig
RDY	BOOL	Bereit: Telegramm wurde empfangen
TELN	INT	Nummer des Vergleichstelegramms, mit dem das empfangene Telegramm übereinstimmt
MW0	INT	Nutzinformationen im ABB AWL-Format
VT0	Texte	Vergleichstelegramm, doppelbar

Beschreibung

Funktionsbaustein zur ASCII-Kommunikation.

Unter Verwendung des Funktionsbausteins EMAS kann eine AC31-Zentraleinheit ASCII-Meldungen über die serielle RS232-Schnittstelle empfangen.

Der Funktionsbaustein EMAS:
- empfängt Telegramme über eine serielle Schnittstelle der SPS,
- vergleicht diese Telegramme mit im Anwenderprogramm hinterlegten Vergleichstelegrammen
- und stellt bei Übereinstimmung die Nutzdaten des empfangenen Telegramms an den Bausteinausgängen zur Verfügung.

Die Empfangstelegramme werden durch einen Schnittstellentreiber von der seriellen Schnittstelle abgeholt und für EMAS zur weiteren Verarbeitung in einem BUFFER bereitgestellt. Der Treiber erkennt am Telegramm-Abschlußzeichen das Telegrammende. Dieses Telegramm-Abschlußzeichen wird im Baustein SINIT projektiert.

WICHTIGER HINWEIS:
Initialisierung der seriellen Schnittstelle.
Bevor der Baustein EMAS verwendet wird, muß er mit dem Baustein SINIT initialisiert werden.

Kommunikation mehrerer EMAS-Bausteine mit derselben seriellen Schnittstelle:

- EMAS-Bausteine eines Anwenderprogramms, die auf dieselbe serielle Schnittstelle zugreifen, sind so gegeneinander zu verriegeln, daß immer nur ein EMAS-Baustein aktiv ist. Wird dies unterlassen, so können Telegramme vom falschen EMAS bearbeitet und für ungültig erklärt werden.

- Sind sowohl im Anwenderprogramm 1 als auch im Anwenderprogramm 2 EMAS-Bausteine vorhanden, die auf dieselbe serielle Schnittstelle zugreifen, so sind sie so gegeneinander zu verriegeln, daß immer nur ein EMAS-Baustein aktiv ist. Wird dies unterlassen, so können Telegramme vom falschen EMAS bearbeitet und für ungültig erklärt werden.

Ein Telegrammverlust kann durch eine gegenseitige Verriegelung der EMAS-Bausteine vermieden werden. Die Verriegelung muß so projektiert werden, daß immer nur derjenige EMAS-Baustein freigegeben wird, für den das über die Schnittstelle ankommende Telegramm bestimmt ist.

Kommunikation eines EMAS-Bausteins und eines DRUCK-Bausteins mit derselben seriellen Schnittstelle:

Ein EMAS-Baustein und ein DRUCK-Baustein können dieselbe serielle Schnittstelle benutzen, ohne daß besondere Vorkehrungen zu treffen sind.

QUIT BOOL

Der Eingang QUIT steuert den Empfang von Telegrammen und dient außerdem zur Quittierung im Fehlerfall.

QUIT = FALSE: Freigabe zum Empfang von Telegrammen.

QUIT = TRUE: Keine Freigabe zum Empfang von Telegrammen. Quittierung nach dem Empfang eines ungültigen Telegramms.

Wird beim Vergleich eines empfangenen Telegramms mit keinem der hinterlegten Vergleichstelegramme Übereinstimmung festgestellt, dann geht der EMAS automatisch in den Betriebszustand "Fehler". In diesem Fall wird vom EMAS solange kein neues Telegramm mehr bearbeitet, bis der Fehler mit einem 1-Signal am Eingang QUIT quittiert und anschließend (nächster Zyklus) der Empfang von Telegrammen mit einem 0-Signal am Eingang QUIT wieder freigegeben wird.

SSK INT

Am Eingang SSK (Schnittstellenkennung) wird die Nummer der Schnittstelle angegeben, über die der Baustein seine Telegramme empfängt.
Es gilt:
COM1: Nummer = 1
COM2: Nummer = 2

#ANU DIREKTE KONSTANTE

Am Eingang #ANU (Anzahl Nutzinformationen) wird die Anzahl der Ausgänge MW0 angegeben, an denen der Baustein die empfangenen Nutzinformationen zur Verfügung stellt. Die Angabe erfolgt als direkte Konstante.
D.h. Nummer = 10 => #ANU = #10 oder #H0A

MEUN BOOL

Am Ausgang MEUN (Merker ungültig) wird angezeigt, ob die Daten an den Ausgängen MW gültig oder ungültig sind.

Wird ein Telegramm ordnungsgemäß empfangen und bearbeitet, so werden die Daten an den Ausgängen MW für gültig erklärt.
Stimmt das empfangene Telegramm mit keinem abgelegten Vergleichstelegramm überein oder kann das empfangene Telegramm nicht ordnungsgemäß bearbeitet werden, so werden die Daten an den Ausgängen MW für ungültig erklärt.

MEUN = FALSE -> Daten an den Ausgängen MW sind gültig
MEUN = TRUE -> Daten an den Ausgängen MW sind ungültig

RDY BOOL

Am Ausgang RDY (Ready) wird angezeigt, daß ein Telegramm empfangen und bearbeitet wurde.
Der Ausgang RDY macht keine Aussage darüber, ob ein gültiges oder ein ungültiges Telegramm empfangen wurde.

RDY = FALSE -> Es wurde noch kein Telegramm empfangen
RDY = TRUE -> Ein Telegramm wurde empfangen und bearbeitet

QUIT	MEUN	RDY	Bedeutung
TRUE	FALSE	FALSE	Durch QUIT=TRUE ist der EMAS gesperrt. Die Ausgänge MEUN und RDY werden dabei permanent auf FALSE gesetzt.
FALSE	TRUE	FALSE	EMAS ist zum Empfang freigegeben, es wurde aber noch kein Telegramm empfangen und ausgewertet.
FALSE	TRUE	TRUE	EMAS hat ein gültiges Telegramm empfangen und ist bereit um ein neues Telegramm zu empfangen.
FALSE	TRUE	TRUE	EMAS hat ein ungültiges Telegramm empfangen. Um ein neues Telegramm empfangen zu können, ist eine Quittierung am Eingang QUIT erforderlich. QUIT: FALSE->TRUE-Flanke
FALSE/ TRUE-Flanke	FALSE	TRUE	Quittierung nach dem Empfang eines ungültigen Telegramms. Nach der Quittierung wird der EMAS durch QUIT=FALSE wieder freigegeben.

TELN INT

Wird ein gültiges Telegramm empfangen, so wird am Ausgang TELN (Telegramm-Nummer) die Nummer des zugehörigen Vergleichstelegramms ausgegeben.

MW0 INT

Der Ausgang MW0 ist doppelbar. Durch die Parameter MW0 werden die Nutzdaten ausgegeben, die im gerade empfangenen Telegramm übermittelt werden. Diese Nutzdaten können Zahlenwerte oder beliebige Zeichen sein. Dies hängt davon ab, welche Art von Platzhaltern im Telegramm projektiert wurden. Die Nutzdaten eines Telegramms werden beginnend mit dem ersten MW-Parameter abgelegt und zwar in der Reihenfolge, wie sie im Vergleichstelegramm projektiert wurden. Es müssen so viele Ausgänge MW0 vorgesehen werden, daß sie für das Telegramm mit den meisten Nutzdaten ausreichen. **=> die Operanden müssen im Format ABB AWL geschrieben werden!**

VT0 ALLE in AWL

An den Eingängen VT0 werden die im SPS-Programm zu hinterlegenden Vergleichstelegramme angegeben. Der Baustein kann 1 bis 99 Telegramme bearbeiten. Ein Telegramm belegt zwei Eingänge, wobei an einem Eingang die Telegramm-Nummer und am darauf folgenden Eingang der eigentliche Telegrammtext angegeben wird. Die genaue Syntax und Handhabung der Vergleichstelegramme ist weiter unten erläutert.
Der Aufbau der Meldungen ist weiter unten beschrieben und am Ende wird ein Beispiel gegeben.

Detailbeschreibung der Vergleichstelegramme

Direkt hinter dem Baustein EMAS sind 1...99 Vergleichstelegramme abgelegt.

Die Vergleichstelegramme dienen zur Identifikation
- der empfangenen aktuellen Telegramme
- und der in den empfangenen Telegrammen enthaltenen Nutzdaten.

Die abgelegten Vergleichstelegramme (V-Telegramm) haben jeweils zur Kennung eine Telegramm-Nummer. Jedes V-Telegramm darf bis zu 255 Zeichen lang sein.

Die Vergleichstelegramme bestehen aus:
- ASCII-Zeichen, die nur zur Identifikation des empfangenen Telegramms dienen,
- Platzhaltern für die zu empfangenden und an die Bausteinausgänge auszugebenden Nutzinformationen.

Bei den Platzhaltern für die Nutzinformationen unterscheidet der Funktionsbaustein EMAS zwischen Platzhaltern für Ziffern und Platzhaltern für Zeichen.

Platzhalter für Ziffern:
(ein # pro Ziffer)

Platzhalter für Zeichen:
* (ein * pro Zeichen/Byte)

Platzhalter für Ziffern:
Für jeden Ziffern-Platzhalter (*) des Vergleichstelegramms erwartet der EMAS im zu empfangenden Telegramm genau eine ASCII-codierte Dezimalziffer. Maximal 5 Ziffern-Platzhalter bilden eine Platzhalter-Gruppe. Solch eine Gruppe von Ziffern-Platzhaltern repräsentiert den Zahlenwert einer maximal 5-stelligen Dezimahlzahl.

Für das Vorzeichen der Dezimahlzahl wird kein Platzhalter vergeben, da dieses von EMAS automatisch berücksichtigt wird. EMAS weist jeweils den Zahlenwert, der zu einer Platzhalter-Gruppe gehört, einem Nutzinfo-Ausgang zu.

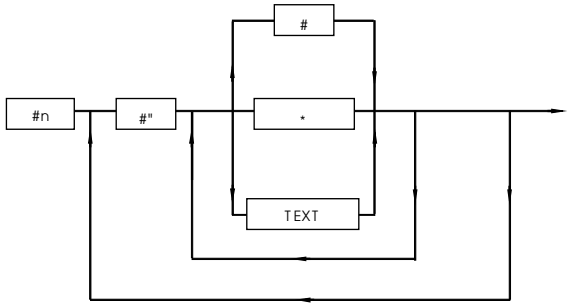
Z.B.:

Dezimalzahl	Platzhalter
1234	####
+1234	####
-1234	####

Der Baustein EMAS überprüft die empfangene Dezimahlzahl auf Ihren Wertigkeitsbereich. In der SPS können nur Zahlen im Bereich +32767 verarbeitet werden. Überschreitet die empfangene Dezimalzahl den Wertigkeitsbereich, so wird vom Baustein EMAS automatisch der jeweilige max. Grenzwert eingesetzt. Bei einer positiven Zahl +32767, bei einer negativen Zahl –32767.

Platzhalter für Zeichen:
Für jeden Zeichen-Platzhalter (*) des Vergleichstelegramms erwartet der EMAS im zu empfangenden Telegramm jeweils ein Zeichen/Byte. Dies können ASCII-Zeichen von Buchstaben aber auch alle anderen HEX-Werte von 0...FF sein.
Die Länge einer Zeichen-Platzhaltergruppe ist maximal 255. In diesem Fall würde das ganze Vergleichstelegramm nur aus Zeichen-Platzhaltern bestehen. Die empfangenen Zeichen/Bytes werden vom EMAS ohne Veränderung und fortlaufend seinen Nutzinfo-Ausgängen MW0 zugewiesen.

Syntaxdiagramm: Aufbau der Vergleichstelegramme



#n: Fortlaufende Telegramm-Nummer
(direkte Konstante 1...99)

#" : Anfangs-Kennung für Texteingabe

“# : Ende-Kennung für Texteingabe

* : Platzhalter für Zeichen/Byte

: Platzhalter für Ziffern

TEXT: Alle ASCII-Zeichen 01 bis FF, außer * und #

Eingabe der Vergleichstelegramme

- Jedes Vergleichstelegramm besteht aus:
  der Telegramm-Nummer
  dem Telegramm-Text

Die Telegramm-Nummer und der Telegramm-Text sind jeweils eigene Operanden. Deshalb belegen die Telegramm-Nummer und der Telegramm-Text beim FUP-Symbol des EMAS-Bausteins jeweils einen eigenen Eingang.
Für ein Vergleichstelegramm werden also zwei Eingänge benötigt.

Beispiel:

Erster TEXT-Eingang: #1
(Nr. des ersten Vergleichstelegramms)
Zweiter TEXT-Eingang: #"DRUCK###
KENNUNG**** (Text des ersten Vergleichstelegramms)

- Außer den ASCII-Zeichen für * und # sind im Telegramm-Text alle ASCII-Zeichen möglich.

- Bei der Eingabe von speziellen ASCII-Zeichen wie z.B. "Zeilenanfang" <CR> ist folgendes zu beachten: Die Eingabe von Sonderzeichen erfolgt durch:

  \Zahlenwert des Zeichens

Der Zahlenwert des Zeichens wird dabei als dreistellige Dezimalzahl angegeben.

Beispiel für den Baustein EMAS:

Folgender Telegrammtext soll eingegeben werden:
Temperatur <CR> Kessel 1
Es gilt: <CR> = 013
Die Eingabe des Textes in das Programmiersystem sieht wie folgt aus:

  #"Temperatur\013Kessel 1

Hinweis:

- Zeichen mit ASCII-Codes >20H oder >32D müssen über die Tastatur eingegeben werden. So muß beispielsweise anstatt \033 das diesem Code entsprechende "!" eingegeben werden.
- Zeichen, die keine Sonderzeichen sind und auch nicht über die Tastatur eingegeben werden können, werden wie folgt erzeugt:
Drücken und gedrückt halten der Taste <ALT> und Eingabe des Zahlencodes (Dezimalcode) auf der numerischen Tastatur, loslassen der Taste <ALT>.
- Das Zeichen mit dem ASCII-Code 255 ist innerhalb der Programmiersoftware reserviert und darf nicht verwendet werden.

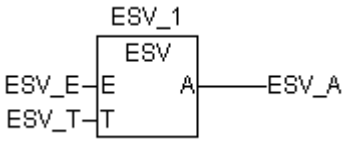
In der Anweisungsliste sieht das Programm wie folgt aus:

```
LD 0
{S40Inline
!BA 0
EMAS
M00,00
MW00,00
#1
AW00,00
AW00,01
#1
#"###"#
}
```

Einschaltverzögerung ESV S40

Die FALSE/TRUE-Flanke am Eingang E wird um die Zeitdauer T verzögert und als FALSE/TRUE-Flanke am Ausgang A ausgegeben.

Geht der Eingang E vor Ablauf der Zeit T wieder auf FALSE-Pegel, so bleibt der Ausgang A auf FALSE-Pegel.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	ESV	Instanzname
E	BOOL	Eingangssignal
T	TIME	Verzögerungszeit
A	BOOL	Verzögertes Signal, Operanden M, A (nicht E, S)

Beschreibung

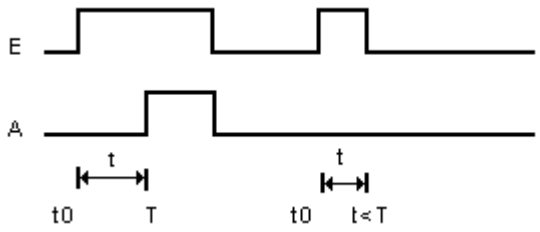
Die FALSE/TRUE-Flanke am Eingang E wird um die Zeitdauer T verzögert und als FALSE/TRUE-Flanke am Ausgang A ausgegeben.

Geht der Eingang E vor Ablauf der Zeit T wieder auf FALSE-Pegel, so bleibt der Ausgang A auf FALSE-Pegel.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Sinnvoller Bereich für T: > 1 Zykluszeit

Die Eingänge und der Ausgang sind weder doppelbar noch invertierbar.



Allgemeines Verhalten

• Gestartete Zeitwerke werden vom Betriebssystem der SPS bearbeitet und sind deshalb vollkommen unabhängig von der Bearbeitung des SPS-Programms. Erst nach Ablauf des Zeitwerks erfolgt eine entsprechende Meldung des Betriebssystems an den zugehörigen Zeitbaustein im SPS-Programm.

• Die Bearbeitung eines Zeitwerks im Betriebssystem der SPS wird durch folgende Befehle beeinflußt: Alle laufenden Zeitwerke werden gestoppt und initialisiert, wenn einer der folgenden Fehler auftritt:

- SPS-Programm abbrechen
- RUN/STOP-Schalter von RUN -> STOP
- Warm- oder Kaltstart

Beispiel

Deklaration:

```
ESV_1: ESV;
ESV_E AT %MX0.0 : BOOL;
ESV_T AT %MD4002.0 : DINT := t#3s; (* 3000 ms*)
ESV_A AT %MX0.1 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
ESV
EI
A
Q
```

FBD:

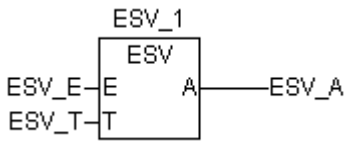


ABB AWL des Beispiels:

```
!BA 0
ESV
M0,0
KD2,0                ; 3000
M0,1
```

Funktionsaufruf in AWL

```
CAL   ESV_1(E := ESV_E,
      T := ESV_T )

LD    ESV_1.A
ST    ESV_A
```

Funktionsaufruf in ST

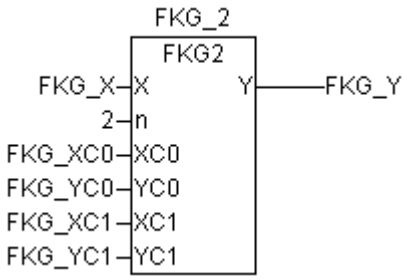
```
ESV_1     (E := ESV_E,
           T := ESV_T );

ESV_A     := ESV1.A;
```

Funktionsgeber FKG(2..256) S40

In einem x/y-Koordinatensystem wird durch n Koordinaten-Punkte X0/Y0…Xn-1/Yn-1 ein Polygonzug festgelegt. Für jeden Wert am Eingang X gibt der Baustein am Ausgang Y den zugeordneten y-Wert des Polygonzuges aus.

Die FKG-Nummer gibt die max. Anzahl der Stützstellen an. Es stehen folgende Funktionsgeber zur Verfügung:

FKG2	Funktionsgenerator mit max. 2 Stützstellen
FKG4	Funktionsgenerator mit max. 4 Stützstellen
FKG16	Funktionsgenerator mit max. 16 Stützstellen
FKG32	Funktionsgenerator mit max. 32 Stützstellen
FKG64	Funktionsgenerator mit max. 64 Stützstellen
FKG256	Funktionsgenerator mit max. 256 Stützstellen



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	FKG(2..256)	Instanzname
X	INT	Eingang für den x-Wert des Polygonzuges
n	INT	Anzahl der Stützstellen
XC0..XCn-1	INT	Eingang für x-Werte der Stützstellen
YC0..YCn-1	INT	Eingang für y-Werte der Stützstellen
Y	INT	Ausgang für den y-Wert des Polygonzuges

Beschreibung

Die FKG-Nummer gibt die max. Anzahl der Stützstellen an. Es stehen folgende Funktionsgeber zur Verfügung:

FKG2	Funktionsgenerator mit max. 2 Stützstellen
FKG4	Funktionsgenerator mit max. 4 Stützstellen
FKG16	Funktionsgenerator mit max. 16 Stützstellen
FKG32	Funktionsgenerator mit max. 32 Stützstellen
FKG64	Funktionsgenerator mit max. 64 Stützstellen
FKG256	Funktionsgenerator mit max. 256 Stützstellen

In einem x/y-Koordinationsystem wird durch n Koordinaten-Punkte X0/Y0…Xn-1/Yn-1 ein Polygonzug festgelegt. Für jeden Wert am Eingang x gibt der Baustein am Ausgang y den zugeordneten y-Wert des Polygonzuges aus.

Für die x-Koordinaten gilt:

$X0 < X1 < X2 \ldots < Xn-1$
$2 \leq n \leq$ FKG-Nummer

Beispiel:
Bei FKG16 gilt: $2 \leq n \leq 16$

Der Baustein interpoliert linear zwischen den Stützstellen. Der so erhaltene Polygonzug stellt den Zusammenhang zwischen der Eingangsgröße x und der Ausgangsgröße y dar.

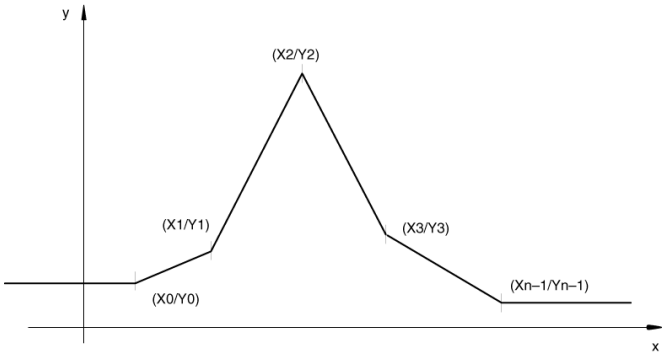
Für die Interpolation zwischen zwei Stützstellen gilt:

$$y = \frac{(x - X_{i-1}) * (Y_i - Y_{i-1})}{X_i - X_{i-1}} + Y_{i-1}$$

Hinweis:
Bei der Division wird stets abgerundet, d.h. ein Rest bei der Division wird nicht berücksichtigt.

Für den Bereich außerhalb der Stützstellen gilt:

für x < X0 ist y = Y0
für x > Xn-1 ist y = Yn-1



X **INT**

Am Eingang X wird die laufende x-Koordinate vorgegeben. Der Baustein bestimmt dann die durch den Polygonzug zugeordnete y-Koordinate.

n **INT**

Am Eingang n wird die Anzahl der Stützstellen vorgegeben, die zur Festlegung des Polygonzuges erforderlich sind.

XC0-…-XCn-1 **INT**

An den Eingängen XC0 … XCn-1 werden die x-Koordinaten der n Stützstellen angegeben.

YC0-…-YCn-1 **INT**

An den Eingängen YC0 … YCn-1 werden die y-Koordinaten der n Stützstellen angegeben.

Y **INT**

Am Ausgang Y wird die y-Koordinate ausgegeben, die der vorgegebenen x-Koordinate durch den Polygonzug zugeordnet ist.

Beispiel

Deklaration:

```
FKG_2: FKG;
FKG_X AT %MW1000.0 : INT;
FKG_XC0 AT %MW3002.0 : INT := 0;
FKG_YC0 AT %MW3002.1 : INT := 10000;
FKG_XC1 AT %MW3002.2 : INT := 0;
FKG_YC1 AT %MW3002.3 : INT := 20000;
FKG_Y AT %MW1000.1 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
FKG
X
#(2 * n)
XC0
YC0
XC1
YC1
Y
```

FBD:

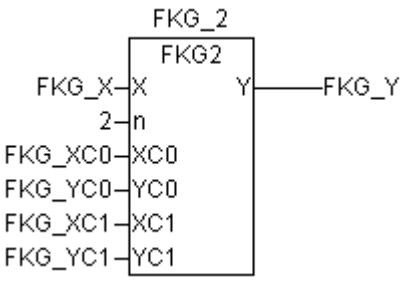


ABB AWL des Beispiels:

```
!BA 0
FKG
MW0,0
#4                    (* 2 * n *)
KW2,0
KW2,1
KW2,2
KW2,3
MW0,1
```

Funktionsaufruf in AWL

```
CAL     FKG_1(X := FKG2_X, n := 2, XC0 :=
FKG2_XC0, YC0 := FKG2_YC0, XC1 := FKG2_XC1,
YC1 := FKG2_YC1)

LD      FKG_1.Y
ST      FKG2_Y
```

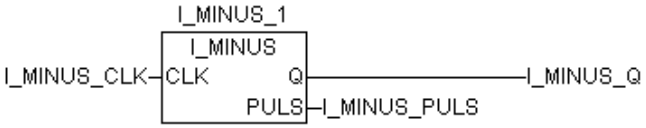
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
FKG_1(X := FKG2_X, n := 2, XC0 := FKG2_XC0,
YC0 := FKG2_YC0, XC1 := FKG2_XC1, YC1 :=
FKG2_YC1);

FKG2_Y:=FKG_1.Y;
```

Erkennung fallende Flanke I_MINUS S40

Eine negative Flanke (TRUE/FALSE) am Eingang CLK erzeugt am Ausgang PULS einen Impuls von der Dauer eines SPS-Programmzyklus.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

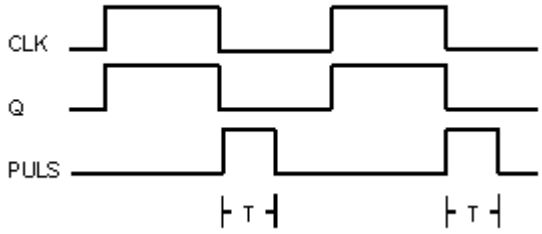
Instanz	I_MINUS	Instanzname
CLK	BOOL	Eingang für TRUE/FALSE-Flanke
Q	BOOL	Ausgang zur Abfrage des direkten Merkers, Operand M (nicht E,S)
PULS	BOOL	Impuls-Ausgang, Operanden M, A (nicht E, S)

Beschreibung

Eine negative Flanke (TRUE/FALSE) am Eingang CLK erzeugt am Ausgang PULS einen Impuls von der Dauer eines SPS-Programmzyklus.

Ausgang Q wird zur Flankenerkennung verwendet. Dieser Merker **darf** im SPS-Programm **nicht nochmals verwendet werden**.

Dauer des Impulses:
Von der Erkennung der TRUE/FALSE-Flanke durch das Verknüpfungselement bis zur erneuten Bearbeitung dieses Verknüpfungselements im nächsten Programmzyklus.



Beispiel

Deklaration:

```
I_MINUS_1: I_MINUS;
I_MINUS_CLK AT %IX62.0: BOOL;
I_MINUS_Q AT %MX80.0: BOOL;
I_MINUS_PULS AT %MX0.0: BOOL;
```

Übersetzung in ABB AWL:

```
! CLK
=S Q
!N CLK
& Q
=R Q
= PULS
```

FBD:

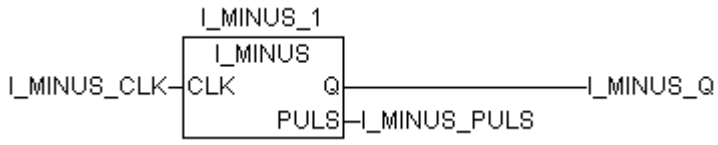


ABB AWL des Beispiels:

```
! E62,0
=S M80,0
!N E62,0
& M80,0
=R M80,0
=M0,0
```

Funktionsaufruf in AWL

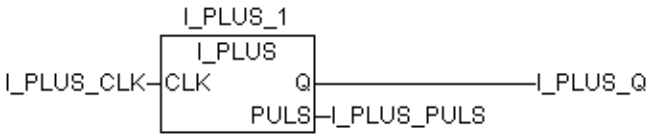
```
CAL   I_MINUS_1(CLK := I_MINUS-CLK)

LD    I_MINUS_1.Q
ST    I_MINUS_Q
LD    I_MINUS_1.PULS
ST    I_MINUS_PULS
```

Funktionsaufruf in ST

```
I_MINUS_1(CLK := I_MINUS_CLK, );

I_MINUS_Q     := I_MINUS_1.Q;
I_MINUS_PULS  := I_MINUS_1.PULS;
```

Erkennung steigende Flanke I_PLUS S40

Eine positive Flanke (FALSE/TRUE) am Eingang CLK erzeugt am Ausgang PULS einen Impuls von der Dauer eines SPS-Programmzyklus.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	I_PLUS	Instanzname
CLK	BOOL	Eingang für FALSE/TRUE-Flanke
Q	BOOL	Ausgang zur Abfrage des direkten Merkers, Operand M (nicht E,S)
PULS	BOOL	Impuls-Ausgang, Operanden M, A (nicht E, S)

Beschreibung

Eine positive Flanke (FALSE/TRUE) am Eingang CLK erzeugt am Ausgang PULS einen Impuls von der Dauer eines SPS-Programmzyklus.

Ausgang Q wird zur Flankenerkennung verwendet. Dieser Merker **darf** im SPS-Programm **nicht nochmals verwendet werden**.

Dauer des Impulses:
Von der Erkennung der FALSE/TRUE-Flanke durch das Verknüpfungselement bis zur erneuten Bearbeitung dieses Verknüpfungselements im nächsten Programmzyklus.

Beispiel

Deklaration:

```
I_PLUS_1: I_PLUS;
I_PLUS_CLK AT %IX62.1: BOOL;
I_PLUS_Q AT %MX80.1: BOOL;
I_PLUS_PULS AT %MX0.1: BOOL;
```

Übersetzung in ABB AWL:

```
!N CLK
=R Q
! CLK
&N Q
=S Q
= PULS
```

FBD:

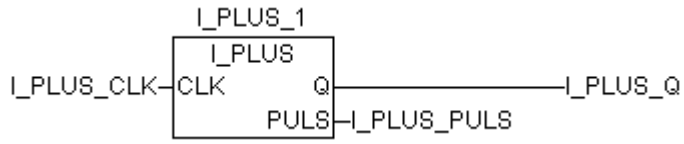


ABB AWL des Beispiels:

```
!N E62,1
=R M80,1
! E62,1
&N M80,1
=S M80,1
= M0,1
```

Funktionsaufruf in AWL

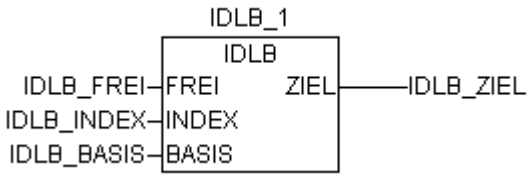
```
CAL   I_PLUS_1(CLK := I_PLUS_CLK)

LD    I_PLUS_1.Q
ST    I_PLUS_Q
LD    I_PLUS_1.PULS
ST    I_PLUS_PULS
```

Funktionsaufruf in ST

```
I_PLUS_1(CLK := I_PLUS_CLK, );

I_PLUS_Q    := I_PLUS_1.Q;
I_PLUS_PULS := I_PLUS_1.PULS;
```

Binärvariable indiziert lesen IDLB S40

Der Baustein dient zum indizierten Lesen von Binärvariablen.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	IDLB	Instanzname
FREI	BOOL	Baustein-Freigabe FREI = FALSE: Baustein wird nicht bearbeitet FREI = TRUE: Der Wert der Quellvariable wird gelesen und der Zielvariable zugewiesen.
INDEX	INT	Der Index und die Basisvariable ergeben die Quellvariable
BASIS	BOOL	Basisadresse der Binärvariable
ZIEL	BOOL	Zielvariable, Operanden M, A (nicht E, S)

Beschreibung

Der Funktionsbaustein dient zum indizierten Lesen von Binärvariablen.

Die zu lesende Quellvariable ergibt sich durch Indizierung der Basisvariablen.

Der Wert der gelesenen Quellvariable wird der Zielvariable zugewiesen.

Bestimmung von Gruppen- und Kanal-Nummer des Quell-Merkers (Quellvariable) aus dem Basis-Merker und dem Index INDEX.

Der Quell-Merker heißt:
M (G_Basis + A) , (K_Basis + B)

mit:

G_Basis: Gruppen-Nummer des Basis-Merkers
K_Basis: Kanal-Nummer des Basis-Merkers

Formel:

$$\frac{\text{INDEX}}{16} = A \quad \text{Rest } B$$

Gruppen-Nr. des Quell-Merkers:
Gruppen-Nr. des Basis-Merkers +A

Kanal-Nr. des Quell-Merkers:
Kanal-Nr. des Basis-Merkers +B

Beispiel:

Basisvariable: M00,00 AT %MX000.00

INDEX = 10

-> A = 10: 16 -> A = 0, Rest B = 10

->Quellvariable:
%MX(000+A).(00+B) =
%MX(000+0).(00+10) =
%MX000.10

Weitere Beispiele:

Basisvariable	INDEX	Quellvariable
%MX000.00	0	%MX000.00
%MX000.00	2	%MX000.02
%MX000.00	16	%MX001.00
%MX000.02	18	%MX001.04

FREI **BOOL**

Freigabe für den Baustein

FREI = FALSE -> Baustein wird nicht bearbeitet

FREI = TRUE -> Der Wert der Quellvariable wird gelesen und der Zielvariable ZIEL zugewiesen.

INDEX **INT**

Am Eingang INDX wird der Indexwert angegeben. Aus dem Index INDEX und der Basisvariable ergibt sich die Quellvariable (Berechnung siehe oben).

Wertebereich: -16383 < INDEX < +16383

BASIS **BOOL**

Am Eingang BASIS wird die Basisvariable angegeben. Aus dem Index INDEX und der Basisvariable ergibt sich die Quellvariable (Berechnung siehe oben).

ZIEL **BOOL**

Am Ausgang ZIEL wird die Zielvariable angegeben. Der Zielvariable ZIEL wird der Wert der ausgewählten Quellvariable zugewiesen.

Beispiel

Deklaration:

```
IDLB_1: IDLB;
IDLB_FREI AT %MX0.0: BOOL;
IDLB_INDEX AT %MW1000.0: INT;
IDLB_BASIS AT %MX20.0: BOOL;
IDLB_ZIEL AT %MX0.1: BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
IDLB
FREI
INDEX
BASIS
ZIEL
```

FBD:

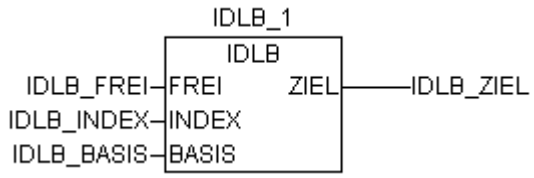


ABB AWL des Beispiels:

```
!BA 0
IDLB
M0,0
MW0,0
M20,0
M0,1
```

Funktionsaufruf in AWL

```
CAL   IDLB_1(FREI := IDLB_FREI,
              INDEX := IDLB_INDEX,
              BASIS := IDLB_BASIS)

LD    IDLB_1.ZIEL
ST    IDLB_ZIEL
```

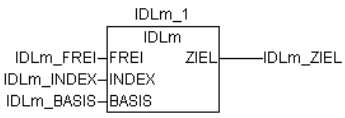
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
IDLB_1      (FREI := IDLB_FREI,
             INDEX := IDLB_INDEX,
             BASIS := IDLB_BASIS);

IDLB_ZIEL:=IDLB_1.ZIEL;
```

Wortvariable indiziert lesen IDLm S40

Der Baustein dient zum indizierten Lesen von Wortvariablen.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	IDLm	Instanzname
FREI	BOOL	Baustein-Freigabe FREI = FALSE: Baustein wird nicht bearbeitet FREI = TRUE: Der Wert der Quellvariable wird gelesen und der Zielvariable zugewiesen.
INDEX	INT	Der Index und die Basisvariable ergeben die Quellvariable
BASIS	INT	Basisvariable
ZIEL	INT	Zielvariable

Beschreibung

Der Baustein dient zum indizierten Lesen von Wortvariablen.

Die zu lesende Quellvariable ergibt sich durch Indizierung der Basisvariable. Der Wert der gelesenen Quellvariable wird der Zielvariable zugewiesen.

Die Eingänge und Ausgänge sind weder doppelbar noch invertierbar noch negierbar.

Bestimmung von Gruppen- und Kanal-Nummer des Quell-Merkers (Quellvariable) aus dem Basis-Merker und dem Index.

Der Quell-Merker heißt:
MW (G_Basis + A) , (K_Basis + B)

mit:

G_Basis: Gruppen-Nummer des Basis-Merkers
K_Basis: Kanal-Nummer des Basis-Merkers

Formel:

$$\frac{\text{INDEX}}{16} = A \quad \text{Rest } B$$

Gruppen-Nr. des Quell-Merkers:
Gruppen-Nr. des Basis-Merkers +A

Kanal-Nr. des Quell-Merkers:
Kanal-Nr. des Basis-Merkers +B

Beispiel:

Basisvariable: MW00,00 AT %MW1000.00

INDEX = 10

-> A = 10: 16 -> A = 0, Rest B = 10

-> Quellvariable:
%MW(1000+A).(00+B) =
%MW(1000+0).(00+10) =
%MW1000.10

Weitere Beispiele:

Basisvariable	INDEX	Quellvariable
%MW1000.00	0	%MW1000.00
%MW1000.00	2	%MW1000.02
%MW100.00	16	%MW1001.00
%MW1000.02	18	%MW1001.04

FREI **BOOL**

Baustein-Freigabe

FREI = FALSE
→ Baustein wird nicht bearbeitet

FREI = TRUE
→ Der Wert der Quellvariable wird gelesen und der Zielvariable ZIEL zugewiesen.

INDEX **INT**

Am Eingang INDEX wird der Indexwert angegeben. Aus dem Index und der Basisvariable ergibt sich die Quellvariable.

Wertebereich: -32767 ≤ INDEX ≤ +32767

Falls INDEX außerhalb des Bereiches liegt, wird der Funktionsblock nicht bearbeitet.

BASIS **INT**

Am Eingang BASIS wird die Adresse der Basisvariable angegeben. Aus dem Index und der Basisvariablen ergibt sich die Quellvariable.

ZIEL **INT**

Am Ausgang ZIEL wird die Zielvariable angegeben. Der Zielvariable ZIEL wird der Wert der ausgewählten Quellvariable zugewiesen.

Beispiel

Deklaration:

```
IDLm_1 : IDLm;
IDLm_FREI AT %MX0.0 : BOOL;
IDLm_INDEX AT %MW1002.0 : INT;
IDLm_BASIS AT %MW1020.0 : INT;
IDLm_ZIEL AT %MW1002.1 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
IDL
FREI
INDEX
BASIS
ZIEL
```

FBD:

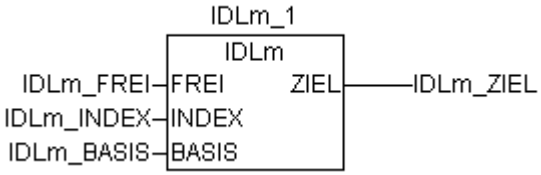


ABB AWL des Beispiels:

```
!BA 0
IDL
M0,0
MW2,0
MW20,0
MW2,1
```

Funktionsaufruf in AWL

```
CAL   IDLM_1(FREI := IDLM_FREI,
              INDEX := IDLM_INDEX,
              BASIS := IDLM_BASIS)

LD    IDLM_1.ZIEL
ST    IDLM_ZIEL
```

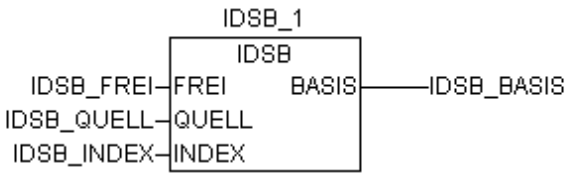
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
IDLM_1      (FREI := IDLM_FREI,
             INDEX := IDLM_INDEX,
             BASIS := IDLM_BASIS);

IDLM_ZIEL   := IDLM_1.ZIEL;
```

Binärvariable indiziert schreiben IDSB S40

Der Baustein dient zum indizierten Schreiben von Binärvariablen.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	IDSB	Instanzname
FREI	BOOL	Baustein-Freigabe FREI = FALSE: Baustein wird nicht bearbeitet FREI = TRUE: Der Wert der Quellvariable wird gelesen und der Zielvariable zugewiesen.
QUELL	BOOL	Quellvariable
INDEX	INT	Aus dem Index und der Basisvariable ergibt sich die aktuelle Zielvariable.
BASIS	BOOL	Basisvariable, Operanden M, A (nicht E, S)

Beschreibung

Der Baustein dient zum indizierten Schreiben von Binärvariablen.
Bei Freigabe des Bausteins wird der Wert der Quellvariable gelesen und der Zielvariable zugewiesen. Die Zielvariable wird durch Indizierung der Basisvariable bestimmt.

Die Eingänge und Ausgänge sind weder doppelbar noch invertierbar noch negierbar.

Bestimmung von Gruppen- und Kanal-Nummer des Ziel-Merkers (Zielvariable) aus dem Basis-Merker und dem Index.

Der Ziel-Merker heißt:
M (G_Basis + A) , (K_Basis + B)

mit:

G_Basis: Gruppen-Nummer des Basis-Merkers
K_Basis: Kanal-Nummer des Basis-Merkers

Formel:

$$\frac{\text{INDEX}}{16} = A \quad \text{Rest } B$$

Gruppen-Nr. des Ziel-Merkers:
Gruppen-Nr. des Basis-Merkers +A

Kanal-Nr. des Ziel-Merkers:
Kanal-Nr. des Basis-Merkers +B

Beispiel:

Basisvariable: M00,00 AT %MX000.00

INDEX = 10

-> A = 10: 16 -> A = 0, Rest B = 10

-> Zielvariable:
%MX(000+A).(00+B) =
%MX(000+0).(00+10) =
%MX000.10

Weitere Beispiele:

Basisvariable	INDEX	Zielvariable
%MX000.00	0	%MX000.00
%MX000.00	2	%MX000.02
%MX000.00	16	%MX001.00
%MX000.02	18	%MX001.04

FREI **BOOL**

Freigabe für den Baustein
FREI = FALSE → Baustein wird nicht bearbeitet

FREI = TRUE → Der Wert der Quellvariable wird gelesen und der Zielvariable zugewiesen.

QUELL **BOOL**

Am Eingang QUELL wird die Quellvariable angegeben. Der Wert dieser Variable wird gelesen und der Zielvariable zugewiesen.

INDEX **INT**

Am Eingang INDEX wird der Indexwert angegeben. Aus dem Index und der Basisvariable ergibt sich die Quellvariable.
Wertebereich: -16383 ≤ INDEX ≤ +16383
Falls INDEX außerhalb des Bereiches liegt, wird der Funktionsblock nicht bearbeitet.

BASIS **BOOL**

Am Ausgang BASIS wird die Basisvariable angegeben.

Aus dem Index INDEX und der Basisvariable ergibt sich die Zielvariable (Berechnung siehe oben).

Beispiel

Deklaration:

```
IDSB_1 : IDSB;
IDSB_FREI AT %MX0.0 : BOOL;
IDSB_QUELL AT %MX1.0 : INT;
IDSB_INDEX AT %MW1001.1 : INT;
IDSB_BASIS AT %MX20.0 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
IDSB
FREI
QUELL
INDEX
BASIS
```

FBD:

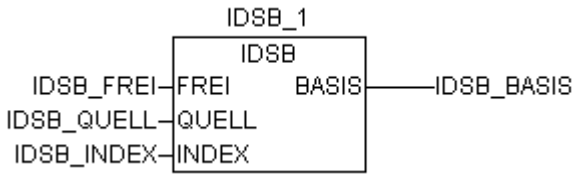


ABB AWL des Beispiels:

```
!BA 0
IDSB
M0,0
M1,0
MW1,1
M20,0
```

Funktionsaufruf in AWL

```
CAL   IDSB_1(FREI := IDSB_FREI,
              QUELL := IDSB_QUELL,
              INDEX := IDSB_INDEX)

LD    IDSB_1.BASIS
ST    IDSB_BASIS
```

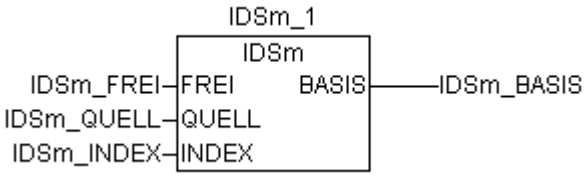
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
IDSB_1      (FREI := IDSB_FREI,
             QUELL := IDSB_QUELL,
             INDEX := IDSB_INDEX);

IDSB_BASIS  := IDSB_1.BASIS;
```

Wortvariable indiziert schreiben IDSM S40

Der Baustein dient zum indizierten Schreiben von Wortvariablen.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	IDSm	Instanzname
FREI	BOOL	Baustein-Freigabe FREI = FALSE: Baustein wird nicht bearbeitet FREI = TRUE: Der Wert der Quellvariable wird gelesen und der Zielvariable zugewiesen.
QUELL	INT	Quellvariable
INDEX	INT	Der Index und die Basisvariable ergeben die Quellvariable
BASIS	INT	Basisvariable

Beschreibung

Der Baustein dient zum indizierten Schreiben von Wortvariablen.
Bei Freigabe des Bausteins wird der Wert der Quellvariable gelesen und der Zielvariable zugewiesen. Die Zielvariable wird durch Indizierung der Basisvariable bestimmt.

Die Eingänge und Ausgänge sind weder doppelbar noch invertierbar noch negierbar.

Bestimmung von Gruppen- und Kanal-Nummer des Ziel-Merkers (Zielvariable) aus dem Basis-Merker und dem Index.

Der Ziel-Merker heißt:
MW (G_Basis + A) , (K_Basis + B)

mit:

G_Basis: Gruppen-Nummer des Basis-Merkers
K_Basis: Kanal-Nummer des Basis-Merkers

Formel:

INDEX
---------- = A Rest B

16

Gruppen-Nr. des Ziel-Merkers:
Gruppen-Nr. des Basis-Merkers +A

Kanal-Nr. des Ziel-Merkers:
Kanal-Nr. des Basis-Merkers +B

Beispiel:

Basisvariable: MW00,00 AT %MW1000.00

INDEX = 10

-> A = 10: 16 -> A = 0, Rest B = 10

-> Zielvariable:
%MW(1000+A).(00+B) =
%MW(1000+0).(00+10) =
%MW1000.10

Weitere Beispiele:

Basisvariable	INDEX	Zielvariable
%MW1000.00	0	%MW1000.00
%MW1000.00	2	%MW1000.02
%MW1000.00	16	%MW1001.00
%MW1000.02	18	%MW1001.04

FREI **BOOL**

Freigabe für den Baustein
FREI = FALSE → Baustein wird nicht bearbeitet

FREI = TRUE → Der Wert der Quellvariable wird gelesen und der Zielvariable zugewiesen.

QUELL **INT**

Am Eingang QUELL wird die Quellvariable angegeben. Der Wert dieser Variable wird gelesen und der Zielvariable zugewiesen.

INDEX **INT**

Am Eingang INDEX wird der Indexwert angegeben. Aus dem Index und der Basisvariable ergibt sich die Quellvariable.
Wertebereich: -32767 ≤ INDEX ≤ +32767
Falls INDEX außerhalb des Bereiches liegt, wird der Funktionsblock nicht bearbeitet.

BASIS **INT**

Am Ausgang BASIS wird die Adresse der Basisvariable angegeben. Aus dem Index INDEX und der Basisvariable ergibt sich die Zielvariable (Berechnung siehe oben).

Beispiel

Deklaration:

```
IDSm_1 : IDSm
IDSm_FREI AT %MX0.0 : BOOL;
IDSm_QUELL AT %MW1020.0 : INT;
IDSm_INDEX AT %MW1002.1 : INT;
IDSm_BASIS AT %MW1002.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
IDSm
FREI
QUELL
INDEX
BASIS
```

FBD:

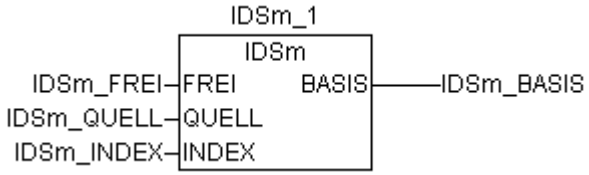


ABB AWL des Beispiels:

```
!BA 0
IDSm
M0,0
MW20,0
MW2,1
MW2,2
```

Funktionsaufruf in AWL

```
CAL   IDSM_1(FREI := IDSM_FREI,
              QUELL := IDSM_QUELL,
              INDEX := IDSM_INDEX)

LD    IDSM_1.BASIS
ST    IDSM_BASIS
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

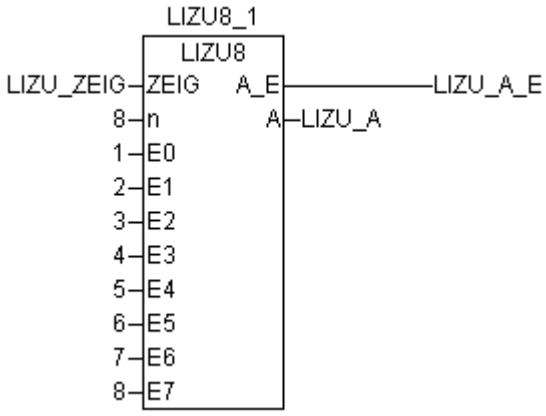
```
IDSM_1      ( FREI := IDSM_FREI,
              QUELL := IDSM_QUELL,
              INDEX := IDSM_INDEX);

IDSM_BASIS  := IDSM_1.BASIS;
```

Listenzuordner LIZU(8..256) S40

Der Baustein hat an seinen Eingängen eine Liste von Wortdaten. Mit einem Listenzeiger wählt er aus dieser Liste einen Wert aus und gibt sie an seinem Ausgang aus.

Die LIZU-Nummer gibt die max. Anzahl der Wortdaten an. Es stehen folgende Listenzuordner zur Verfügung:

LIZU8	Listenzuordner für max. 8 Wortdaten
LIZU16	Listenzuordner für max. 16 Wortdaten
LIZU32	Listenzuordner für max. 32 Wortdaten
LIZU64	Listenzuordner für max. 64 Wortdaten
LIZU256	Listenzuordner für max. 256 Wortdaten



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	LIZU(8..256)	Instanzname
ZEIG	INT	Zeiger auf die Liste der Wortdaten
E0..En-1	INT	Liste der direkten Konstanten
A_E	BOOL	0 ≤ ZEIG < n, d. h. Zeiger im zulässigen Bereich; Operanden M,A (nicht E, S)
A	INT	Ausgewählter Wortwert

Beschreibung

Der Baustein hat an seinen Eingängen eine Liste von Wortdaten. Mit einem Listenzeiger wählt er aus dieser Liste einen Wert aus und gibt sie an seinem Ausgang aus.

Die Ein- und Ausgänge sind nicht invertier- und doppelbar.

ZEIG INT

Am Eingang ZEIG wird der Zeiger auf den aus der Liste auszuwählenden Wortwert angegeben.

Es gilt die Zuordnung:

ZEIG = 0 → Wortwert an E0
ZEIG = 1 → Wortwert an E1
: : :
ZEIG = n-1 → Wortwert an En-1

Der Wert am Eingang ZEIG wird auf Zulässigkeit überwacht. Das Ergebnis dieser Bereichsüberprüfung wird am Ausgang A_E signalisiert.

Erlaubter Bereich:

0 ≤ ZEIG ≤ n-1

mit n: Anzahl der Eingänge E0..En-1.

Befindet sich der Wert am Eingang ZEIG außerhalb des erlaubten Bereichs, so erfolgt auch keine Zuweisung an den Ausgang A.

E0 .. En-1 INT

An den Eingängen E0 … En-1 werden die Wortwerte angegeben, aus denen mit dem Wert am Eingang ZEIG einer ausgewählt und am Ausgang A zugewiesen wird.

A_E BOOL

Am Ausgang A_E wird angezeigt, ob der Wert des Listen-Zeigers (Eingang ZEIG) sich im erlaubten Bereich befindet.

Erlaubter Bereich:

0 ≤ ZEIG ≤ n-1

mit n: Anzahl der Eingänge E0..En-1.

Es gilt:

ZEIG im erlaubten Bereich → A_E = TRUE
ZEIG im unzulässigen Bereich → A_E = FALSE

Hat der Listenzeiger einen unzulässigen Wert, so erfolgt auch keine Zuweisung eines Wertes an den

Ausgang A, weil kein Wortwert ausgewählt werden kann. Der Ausgang A wird in diesem Fall nicht aktualisiert.

A **INT**

Der Wert des ausgewählten Wortwertes wird dem Operanden am Ausgang A zugewiesen.

Beispiel

Deklaration:

```
LIZU8_1 : LIZU8;
LIZU_ZEIG AT %MW1000.0: INT;
LIZU_A_E AT %MX0.0 : BOOL;
LIZU_A AT %MW1000.1 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
LIZU
ZEIG
#n
#E0
#E1
#E2
#E3
#E4
#E5
#E6
#E7
A_E
A
```

FBD:

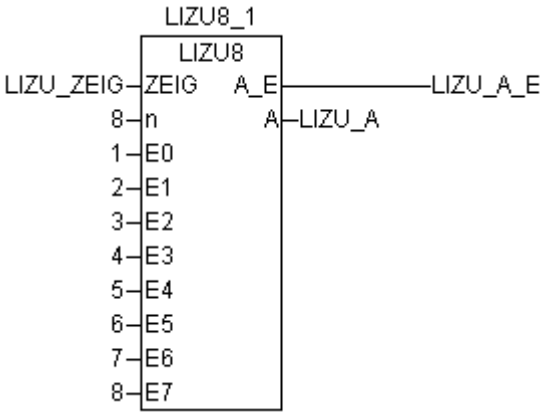


ABB AWL des Beispiels:

```
!BA 0
LIZU
MW0,0
#8
#1
#2
#3
#4
#5
#6
#7
#8
M0,0
MW0,1
```

Funktionsaufruf in AWL

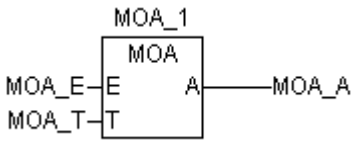
```
CAL   LIZU_8(ZEIG := LIZU8_ZEIG,
             n := 8,
             E0 := 1,
             E1 := 2,
             E2 := 3,
             E3 := 4,
             E4 := 5,
             E5 := 6,
             E6 := 7,
             E7 := 8)
LD    LIZU8_1.A_E
ST    LIZU_A_E
LD    LIZU8_1.A
ST    LIZU8_A
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
LIZU_8      ZEIG := LIZU8_ZEIG,
            n := 8,
            E0 := 1,
            E1 := 2,
            E2 := 3,
            E3 := 4,
            E4 := 5,
            E5 := 6,
            E6 := 7,
            E7 := 8);
LIZU8_A_E   := LIZU8_1.A_E;
LIZU8_A     := LIZU8_1.A;
```

Monostabiles Kippglied »Abbruch« MOA S40



Eine FALSE/TRUE-Flanke am Eingang E bewirkt eine FALSE/TRUE-Flanke am Ausgang A. Bleibt der Eingang E auf TRUE-Pegel, wird nach der Zeitdauer T eine TRUE/FALSE-Flanke am Ausgang A ausgegeben.

Geht der Eingang E vor Ablauf der Zeitdauer T wieder auf FALSE-Pegel zurück, so wird auch sofort der Ausgang A wieder auf FALSE-Pegel gesetzt.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	MOA	Instanzname
E	BOOL	Eingangssignal
T	TIME	Impulslänge
A	BOOL	Ausgangssignal, Operanden M, A (nicht E, S)

Beschreibung

Eine FALSE/TRUE-Flanke am Eingang E bewirkt eine FALSE/TRUE-Flanke am Ausgang A. Bleibt der Eingang E auf TRUE-Pegel, wird nach der Zeitdauer T eine TRUE/FALSE-Flanke am Ausgang A ausgegeben.

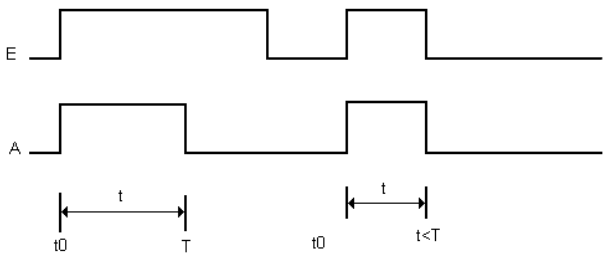
Geht der Eingang E vor Ablauf der Zeitdauer T wieder auf FALSE-Pegel zurück, so wird auch sofort der Ausgang A wieder auf FALSE-Pegel gesetzt.

Gültiger Zeitbereich: 5 ms .. 24,8 Tage.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Sinnvoller Bereich für T: > 1 Zykluszeit

Die Eingänge und der Ausgang sind weder doppelbar noch invertierbar.



Allgemeines Verhalten

• Gestartete Zeitwerke werden vom Betriebssystem der SPS bearbeitet und sind deshalb vollkommen unabhängig von der Bearbeitung des SPS-Programms. Erst nach Ablauf des Zeitwerks erfolgt eine entsprechende Meldung des Betriebssystems an den zugehörigen Zeitbaustein im SPS-Programm.

• Die Bearbeitung eines Zeitwerks im Betriebssystem der SPS wird durch folgende Befehle beeinflußt: Alle laufenden Zeitwerke werden gestoppt und initialisiert, wenn einer der folgenden Fehler auftritt:

- SPS-Programm abbrechen
- RUN/STOP-Schalter von RUN -> STOP
- Warm- oder Kaltstart

Beispiel

Deklaration:

```
MOA_1 : MOA;
MOA_E AT %MX0.0 : BOOL;
MOA_T %MD4001.0 : TIME := t#1m; (*60000 ms*)
MOA_A AT %MX0.1 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
MOA
E
T
A
```

FBD:

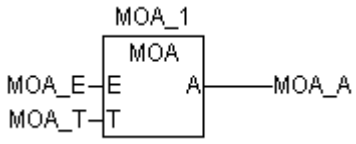


ABB AWL des Beispiels:

```
!BA 0
MOA
M0,0
KD1,0                    ; 60000
M0,1
```

Funktionsaufruf in AWL

```
CAL   MOA_1(E := MOA_E,
            T := MOA_T)

LD    MOA_1.A
ST    MOA_A
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

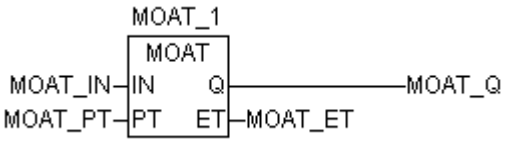
```
MOA_1    (E := MOA_E,
          T := MOA_T);

MOA_A    := MOA_1.A;
```

Monostabiles Kippglied »Abbruch« MOAT S40

Eine FALSE/TRUE-Flanke am Eingang E bewirkt eine FALSE/TRUE-Flanke am Ausgang A. Bleibt der Eingang E auf TRUE-Pegel, wird nach der Zeitdauer T eine TRUE/FALSE-Flanke am Ausgang A ausgegeben.

Der Ausgang ET zeigt die aktuelle Zeit an.

Geht der Eingang E vor Ablauf der Zeitdauer T wieder auf FALSE-Pegel zurück, so wird auch sofort der Ausgang A wieder auf FALSE-Pegel gesetzt.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	MOA	Instanzname
IN	BOOL	Eingangssignal
PT	TIME	Impulslänge
Q	BOOL	Ausgangssignal, Operanden M, A (nicht E, S)
ET	TIME	Zeitanzeige

Beschreibung

Eine FALSE/TRUE-Flanke am Eingang E bewirkt eine FALSE/TRUE-Flanke am Ausgang A. Bleibt der Eingang E auf TRUE-Pegel, wird nach der Zeitdauer T eine TRUE/FALSE-Flanke am Ausgang A ausgegeben.

Geht der Eingang E vor Ablauf der Zeitdauer T wieder auf FALSE-Pegel zurück, so wird auch sofort der Ausgang A wieder auf FALSE-Pegel gesetzt.

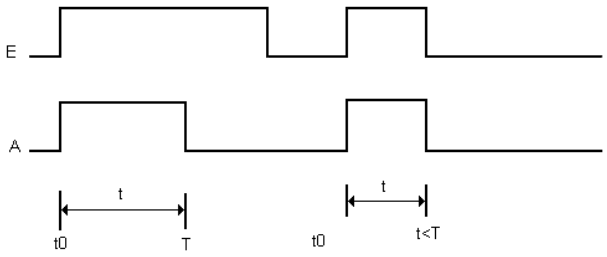
Die Istzeit wird am Ausgang ET angezeigt.

Gültiger Zeitbereich: 5 ms .. 24,8 Tage.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Sinnvoller Bereich für T: > 1 Zykluszeit

Die Eingänge und der Ausgang sind weder doppelbar noch invertierbar.



Allgemeines Verhalten

• Gestartete Zeitwerke werden vom Betriebssystem der SPS bearbeitet und sind deshalb vollkommen unabhängig von der Bearbeitung des SPS-Programms. Erst nach Ablauf des Zeitwerks erfolgt eine entsprechende Meldung des Betriebssystems an den zugehörigen Zeitbaustein im SPS-Programm.

• Die Bearbeitung eines Zeitwerks im Betriebssystem der SPS wird durch folgende Befehle beeinflußt: Alle laufenden Zeitwerke werden gestoppt und initialisiert, wenn einer der folgenden Fehler auftritt:

- SPS-Programm abbrechen
- RUN/STOP-Schalter von RUN -> STOP
- Warm- oder Kaltstart

Beispiel

Deklaration:

```
MOAT_1 : MOAT
MOAT_IN AT %MX0.0 : BOOL;
MOAT_PT AT %MD4002.0 : TIME := t#2s500ms; (*2500 ms*)
MOAT_Q AT %MX0.1 : BOOL;
MOAT_ET AT %MD2002.0 : TIME;
```

Übersetzung in ABB AWL:

```
!BA 0
MOAT
IN
PT
Q
ET
```

FBD:

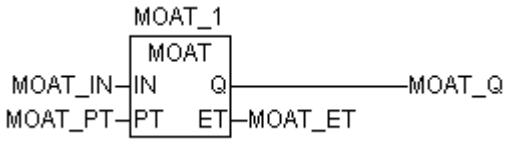


ABB AWL des Beispiels:

```
!BA 0
MOAT
M0,0
KD2,0                    ; 2500
M0,1
MD2,0
```

Funktionsaufruf in AWL

```
CAL   MOAT_1(E := MOAT_E,
              T := MOAT_T)

LD    MOAT_1.A
ST    MOAT_A
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

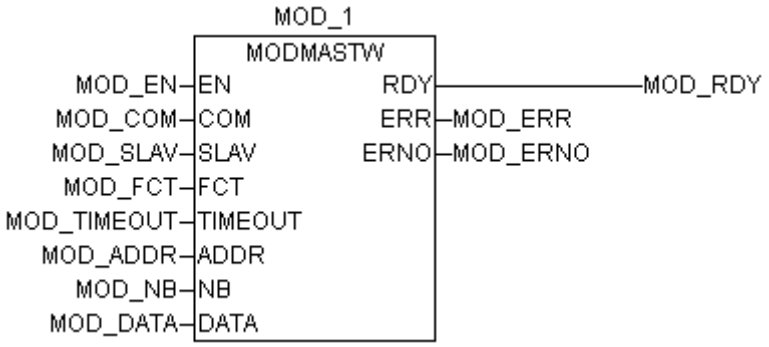
Funktionsaufruf in ST

```
MOAT_1    (E := MOAT_E,
           T := MOAT_T);

MOAT_A    := MOAT_1.A;
```

Betriebsart MODBUS Master MODMASTB/W S40

MODBUS Master Kommunikation:

MODBUSB	COM 1 voreingestellt und DATA ist vom Typ BOOL
MODBUSW	COM 1 voreingestellt und DATA ist vom Typ INT
MODMASTB	COM 1 oder COM 2 und DATA ist vom Typ BOOL
MODMASTW	COM 1 oder COM 2 und DATA ist vom Typ INT



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	MODBUS B/W MODMAST B/W	Instanzname
EN	BOOL	Freigabe der Bausteinbearbeitung
COM	INT	Schnittstellenkennung (COM1, COM2)
SLAVE	INT	Slave-Nummer (1...255)
FCT	INT	Funktions-Code
TIMEOUT	INT	Telegramm-Timeout in Millisekunden
ADDR	INT	Operanden-/Registeradresse im Slave
NB	INT	Anzahl der Daten
DATA	BOOL / INT	Daten zum Senden an den Slave oder zum Schreiben mit den Daten, die von einem Slave-MODBUS empfangen wurden.
RDY	BOOL	Ready-Meldung, Kommunikation läuft, Operanden M, A
ERR	BOOL	Fehlermeldung, Operanden M, A (nicht E, S)
ERNO	INT	Fehlerkennung

Beschreibung

Funktionsbaustein zur MODBUS-Master-Kommunikation.

Die Zentraleinheit ist ein Master an einem MODBUS-Netzwerk und kann mit anderen Geräten über das MODBUS-Protokoll kommunizieren.

Die Funktion MODBUS-MASTER der Zentraleinheit wird eingestellt durch:
- Systemkonstante KW 00,06 / %MW1000.6 = 100
- und eine Verbindung zwischen den Pins 7 und 6 am Stecker der seriellen Schnittstelle

In einem Anwenderprogramm können mehrere MODBUS/MODMAST-Funktionsblöcke verwendet werden.

Das MODBUS-Protokoll ist ein Master-Slave-Protokoll. Der Master sendet eine Anforderungstelegramm (Request) an einen Slave und wartet auf die Antwort (Replay). Das Timeout ist festgelegt. Im Slave können binäre oder numerische Daten geschrieben oder gelesen werden.

Der Datenbereich im Master wird durch die Adresse der ersten Variable gewählt. Die Größe des Bereichs ist zum Senden und Empfangen notwendig. Das Lesen oder Schreiben findet automatisch aus diesen Datenbereichen statt.

EN **BOOL**

Eine steigende Flanke am FREI-Eingang führt zur Ausgabe einer Anforderung an einen MODBUS-Slave, vorausgesetzt, daß der Funktionsblock dazu in der Lage ist (RDY = TRUE).

Wenn am FREI-Eingang eine steigende Flanke auftritt, obwohl der Ausgang RDY gleich FALSE ist (d. h. der Baustein ist nicht bereit für eine neue MODBUS-Kommunikation), so wird diese steigende Flanke ignoriert. Solange der Ausgang RDY FALSE ist, kann

daher keine neue MODBUS-Kommunikation gestartet werden.

COM **INT**

Am Eingang COM wird die Nummer der MODBUS-Schnittstelle vorgegeben.

COM = 1: COM1 (SPS)
COM = 2: COM2 (SPS)

SLAV **INT**

Adresse des Slave, der die Anforderung empfängt.

Wert: 0 < ADDR < 255

Falls die Adresse 0 ist (ADDR = 0), wird das Telegramm von allen Slaves am MODBUS-Netzwerk gelesen.

FCT **INT**

Die Funktion ist abhängig von den Parameter-Typen und davon, ob geschrieben oder gelesen wird.

Wert: 1 (01H): Lesen von n Bits
2 (02H): Lesen von n Bits
3 (03H): Schreiben von n Worten
4 (04H): Lesen von n Worten
5 (05H): Schreiben von einem Bit
6 (06H): Schreiben von einem Wort
7 (07H): Schnelles Lesen von 8 Bits
8 (08H): Diagnose/Initialisierung
15 (0FH): Schreiben von n Bits
16 (10H): Schreiben von n Worten

Die übrigen Funktionscodes werden von den Zentraleinheiten der Serie 40..50 nicht unterstützt. Tritt ein falscher Funktionscode auf, wird im Wort ERNO der Fehler 1 gesetzt.

TIME **INT**

Timeout für die Kommunikation (maximale Zeit für eine Antwort des MODBUS-Slave).

Der Wert wird in Millisekunden angegeben.

Zykluszeit (KD00,00 / %MD4000.0) < TIME < 32767

Wenn ein Timeout auftritt, liefert der Ausgang ERN den Wert 9.

ADDR **INT**

Adresse der Daten im Slave-Speicher, die zu lesen oder zu schreiben sind.

NB **INT**

Anzahl der Daten, die im Slave zu lesen oder zu schreiben sind.

Die Anzahl definiert ebenfalls die Größe des Datenbereichs im Master, von dem gesendet oder empfangen wird.

DATA **INT, BOOL**

INT: MODBUSW oder MODMASTW
BOOL:

DATA definiert die erste Variable des Datenbereichs im Master. Die Größe des Bereichs hängt vom Parameter NB ab.

Unterschiedliche Fälle sind in Abhängigkeit vom Funktionscode und dem Operanden möglich.

Lesen:

ADDR	DATA	Ergebnis
INT	INT	Identisch
BOOL	BOOL	Identisch
BOOL	INT	Gleich wie Funktionsbaustein PACK
INT	BOOL	Das erste Bit im Wort wird im ersten Bit der Gruppen-Nummer plaziert.

Beispiel: DATA = %MX00.07:
Das erste Bit des gelesenen Wortes wird in %MX 00.00 geschrieben.

Schreiben:

ADDR	DATA	Ergebnis
INT	INT	identisch
BOOL	BOOL	identisch
BOOL	INT	Gleich wie Funktionsbaustein PACK
INT	BOOL	Gleich wie Funktionsbaustein UNPACK

RDY **BOOL**

Der Ausgang RDY (Ready) zeigt an, ob eine MODBUS-Kommunikation läuft oder nicht. Solange eine Kommunikation läuft, ist der Ausgang gleich FALSE. Der Funktionsbaustein kann nur benutzt werden wenn RDY = TRUE ist.

ERR **BOOL**

Der Ausgang ERR zeigt einen Fehler an, der während der Kommunikation aufgetreten ist. Der Wort-Ausgang ERN liefert Einzelheiten zum Fehler.

Ist ERR = 1 -> Fehler,
ERR = 0 -> Kein Fehler oder Kommunikation läuft.

Der Fehler ist nach einer Zykluszeit eindeutig.

ERNO **INT**

Einzelheiten zum Fehler:
- 0 : Kein Fehler
- 1 : Unbekannter Funktionscode
- 2 : Adreßfehler
- 3 : Daten-Fehler
- 9 : Timeout
- 10 : Prüfsummen-Fehler

Der Fehler ist nach einer Zykluszeit eindeutig.

Operanden (symbolisch)	Operanden (IEC)	MODBUS-Adresse (HEX)	Operandenbezeichnung
E000_00 : E061_15	%IX000.00 : %IX061.15	0000_{HEX} : $03DF_{HEX}$	Eingänge binär, CS31-Bus
E062_00 : E068_15	%IX062.00 : %IX068.15	$03E0_{HEX}$: $044F_{HEX}$	Eingänge binär, lokal
A000_00 : A061_15	%QX000.00 : %QX061.15	1000_{HEX} : $13DF_{HEX}$	Ausgänge binär, CS31-Bus
A062_00 : A068_15	%QX062.00 : %QX068.15	$13E0_{HEX}$: $144F_{HEX}$	Ausgänge binär, lokal
M000_00 : M099_15	%MX0000.00 : %MX0099.15	2000_{HEX} : $263F_{HEX}$	Binär-Merker
M280_00 : M254_15	%MX0000.00 : %MX0254.15	2000_{HEX} : $2FEF_{HEX}$	Binär-Merker
M255_00 : M255_15	%MX0255.00 : %MX0255.15	$2FF0_{HEX}$: $2FFF_{HEX}$	Binär-Merker (System)
S000_00 : S125_15	%MX5000.00 : %MX5125.15	3000_{HEX} : $37DF_{HEX}$	Schritte
EW000_00 : EW061_15	%IW1000.00 : %IW1061.15	0000_{HEX} : $03DF_{HEX}$	Eingänge analog, CS31-Bus
EW200_00 : EW068_15	%IW1062.00 : %IW1068.15	$03E0_{HEX}$: $044F_{HEX}$	Eingänge analog, lokal
AW000_00 : AW061_15	%QW1000.00 : %QW1061.15	1000_{HEX} : $13DF_{HEX}$	Ausgänge analog, CS31-Bus

Operanden (symbolisch)	Operanden (IEC)	MODBUS-Adresse (HEX)	Operandenbezeichnung
AW062_00 : AW068_15	%QW1062.00 : %QW1068.15	$13E0_{HEX}$: $144F_{HEX}$	Ausgänge analog, lokal
MW000_00 : MW099_15	%MW1000.00 : %MW1099.15	2000_{HEX} : $263F_{HEX}$	Wort-Merker
MW230_00 : MW253_15	%MW1230.00 : %MW1253.15	$2E60_{HEX}$: $2FCF_{HEX}$	Wort-Merker
MW254_00 : MW255_15	%MW1254.00 : %MW1255.15	$2FD0_{HEX}$: $2FFF_{HEX}$	Wort-Merker (Fehlermeldung)
KW000_00 : KW000_15	%MW3000.00 : %MW3000.15	3000_{HEX} : $300F_{HEX}$	Wort-Konstanten (System)
KW001_00 : KW031_15	%MW3001.00 : %MW3031.15	3010_{HEX} : $31FF_{HEX}$	Wort-Konstanten
MD000_00 : MD007_15	%MD2000.00 : %MD2007.15	4000_{HEX} : $40FE_{HEX}$	Doppelwort-Merker
KD000_00 : KD000_15	%MD4000.00 : %MD4000.15	5000_{HEX} : $501E_{HEX}$	Doppelwort-Konstanten
KD000_01 : KD007_15	%MD4000.00 : %MD4007.15	5020_{HEX} : $50FE_{HEX}$	Doppelwort-Konstanten

Beispiel 1: MODBUSB - MODBUS-Master für COM1 und BOOL-Daten

Deklaration:

```
MOD_1: MODBUSB;
MOD_EN AT %MX13.0: BOOL;
MOD_SLAV AT %MW3002.1: INT := 10;
MOD_FCT AT %MW3002.2: INT := 3;
MOD_TIMEOUT AT %MW3002.3: INT := 5000;
MOD_ADDR AT %MW3002.4: INT := 16#2000; (*8192*)
MOD_NB AT %MW3002.5: INT := 16;
MOD_DATA_B AT %MX40.0: BOOL;
MOD_RDY AT %MX30.0: BOOL;
MOD_ERR AT %MX30.1: BOOL;
MOD_ERNO AT %MW1039.0: INT;
```

Übersetzung in ABB AWL:

```
!BA 0
CALLUP
EN
#H0300
#H0000
#H0000
#00009
SLAV
FCT
TIMEOUT
ADDR
NB
DATA
RDY
ERR
ERNO
```

FBD:

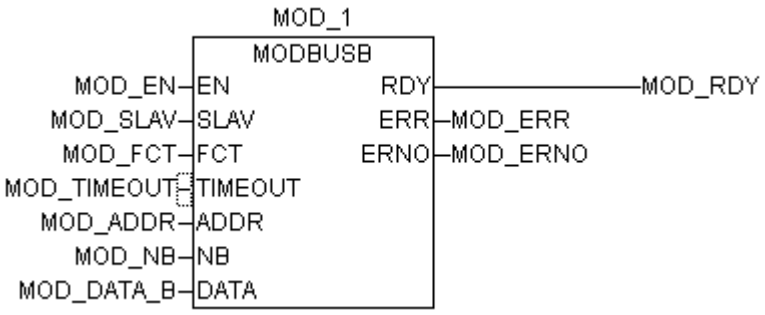


ABB AWL des Beispiels:

```
!BA 0
CALLUP
M0,0
#H0300
#H0000
#H0000
#00009
KW2,1          ; 10
KW2,2          ; 3
KW2,3          ; 5000
KW2,4          ; 8192
KW2,5          ; 16
M40,0
M30,0
M30,1
MW39,0
```

Funktionsaufruf in AWL

```
CAL   MOD_1(EN := MOD_EN,
          SLAV := MOD_SLAV,
          FCT := MOD_FCT,
          TIMEOUT := MOD_TIMEOUT,
          ADDR := MOD_ADDR,
          NB := MOD_NB,
          DATA := MOD_DATA_B)

LD    MOD_1.RDY
ST    MOD_RDY
LD    MOD_1.ERR
ST    MOD_ERR
LD    MOD_1.ERNO
ST    MOD_ERNO
```

Funktionsaufruf in ST

```
MOD_1      (EN := MOD_EN,
            SLAV := MOD_SLAV,
            FCT := MOD_FCT,
            TIMEOUT := MOD_TIMEOUT,
            ADDR := MOD_ADDR,
            NB := MOD_NB,
            DATA := MOD_DATA_B)

MOD_RDY    := MOD_1.RDY;
MOD_ERR    := MOD_1.ERR;
MOD_ERNO   := MOD_1.ERNO;
```

Beispiel 2: MODBUSW - MODBUS-Master für COM1 und INT-Daten

Deklaration:

```
MOD_1: MODBUSW;
MOD_EN AT %MX13.0: BOOL;
MOD_SLAV AT %MW3002.1: INT := 10;
MOD_FCT AT %MW3002.2: INT := 3;
MOD_TIMEOUT AT %MW3002.3: INT := 5000;
MOD_ADDR AT %MW3002.4: INT := 16#2000; (*8192*)
MOD_NB AT %MW3002.5: INT := 16;
MOD_DATA AT %MW1030.0: INT;
MOD_RDY AT %MX30.0: BOOL;
MOD_ERR AT %MX30.1: BOOL;
MOD_ERNO AT %MW1039.0: INT;
```

Übersetzung in ABB AWL:

```
!BA 0
CALLUP
EN
#H0300
#H0000
#H0000
#00009
SLAV
FCT
TIMEOUT
ADDR
NB
DATA
RDY
ERR
ERNO
```

FBD:

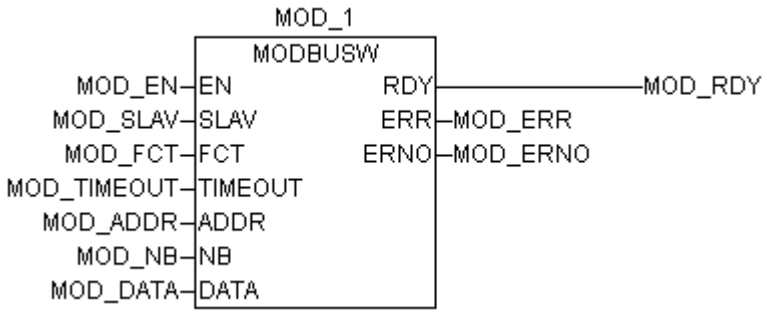


ABB AWL des Beispiels:

```
!BA 0
CALLUP
M0,0
#H0300
#H0000
#H0000
#00009
KW2,1          ; 10
KW2,2          ; 3
KW2,3          ; 5000
KW2,4          ; 8192
KW2,5          ; 16
MW30,0
M30,0
M30,1
MW39,0
```

Funktionsaufruf in AWL

```
CAL   MOD_1(EN := MOD_EN,
            SLAV := MOD_SLAV,
            FCT := MOD_FCT,
            TIMEOUT := MOD_TIMEOUT,
            ADDR := MOD_ADDR,
            NB := MOD_NB,
            DATA := MOD_DATA)

LD    MOD_1.RDY
ST    MOD_RDY
LD    MOD_1.ERR
ST    MOD_ERR
LD    MOD_1.ERNO
ST    MOD_ERNO
```

Funktionsaufruf in ST

```
MOD_1     (EN := MOD_EN,
           SLAV := MOD_SLAV,
           FCT := MOD_FCT,
           TIMEOUT := MOD_TIMEOUT,
           ADDR := MOD_ADDR,
           NB := MOD_NB,
           DATA := MOD_DATA)

MOD_RDY   := MOD_1.RDY;
MOD_ERR   := MOD_1.ERR;
MOD_ERNO  := MOD_1.ERNO;
```

Beispiel 3: MODMASTB - MODBUS-Master für COM1/COM2 und BOOL-Daten

Deklaration:

```
MOD_1: MODMASTB;
MOD_EN AT %MX13.0: BOOL;
MOD_COM AT %MW3002.0: INT := 1;
MOD_SLAV AT %MW3002.1: INT := 10;
MOD_FCT AT %MW3002.2: INT := 3;
MOD_TIMEOUT AT %MW3002.3: INT := 5000;
MOD_ADDR AT %MW3002.4: INT := 16#2000; (*8192*)
MOD_NB AT %MW3002.5: INT := 16;
MOD_DATA_B AT %MX40.0: INT;
MOD_RDY AT %MX30.0: BOOL;
MOD_ERR AT %MX30.1: BOOL;
MOD_ERNO AT %MW1039.0: INT;
```

Übersetzung in ABB AWL:

```
!BA 0
CALLUP
EN
#H0306
#H0000
#H0000
#00010
COM
SLAV
FCT
TIMEOUT
ADDR
NB
DATA
RDY
ERR
ERNO
```

FBD:

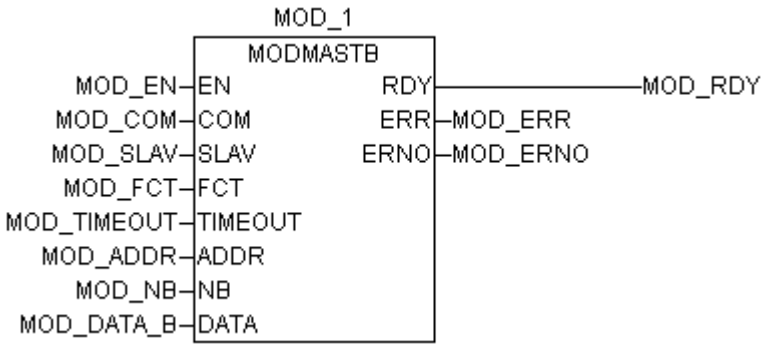


ABB AWL des Beispiels:

```
!BA 0
CALLUP
M0,0
#H0306
#H0000
#H0000
#00010
KW2,0          ; 1
KW2,1          ; 10
KW2,2          ; 3
KW2,3          ; 5000
KW2,4          ; 8192
KW2,5          ; 16
M40,0
M30,0
M30,1
MW39,0
```

Funktionsaufruf in AWL

```
CAL   MOD_1(EN := MOD_EN,
        COM := MOD_COM,
        SLAV := MOD_SLAV,
        FCT := MOD_FCT,
        TIMEOUT := MOD_TIMEOUT,
        ADDR := MOD_ADDR,
        NB := MOD_NB,
        DATA := MOD_DATA_B)

LD    MOD_1.RDY
ST    MOD_RDY
LD    MOD_1.ERR
ST    MOD_ERR
LD    MOD_1.ERNO
ST    MOD_ERNO
```

Funktionsaufruf in ST

```
MOD_1     (EN := MOD_EN,
           COM := MOD_COM,
           SLAV := MOD_SLAV,
           FCT := MOD_FCT,
           TIMEOUT := MOD_TIMEOUT,
           ADDR := MOD_ADDR,
           NB := MOD_NB,
           DATA := MOD_DATA_B)

MOD_RDY   := MOD_1.RDY;
MOD_ERR   := MOD_1.ERR;
MOD_ERNO  := MOD_1.ERNO;
```

Beispiel 4: MODMASTW - MODBUS-Master für COM1/COM2 und INT-Daten

Deklaration:

```
MOD_1: MODMASTW;
MOD_EN AT %MX13.0: BOOL;
MOD_COM AT %MW3002.0: INT := 1;
MOD_SLAV AT %MW3002.1: INT := 10;
MOD_FCT AT %MW3002.2: INT := 3;
MOD_TIMEOUT AT %MW3002.3: INT := 5000;
MOD_ADDR AT %MW3002.4: INT := 16#2000; (*8192*)
MOD_NB AT %MW3002.5: INT := 16;
MOD_DATA AT %MW1030.0: INT;
MOD_RDY AT %MX30.0: BOOL;
MOD_ERR AT %MX30.1: BOOL;
MOD_ERNO AT %MW1039.0: INT;
```

Übersetzung in ABB AWL:

```
!BA 0
CALLUP
EN
#H0306
#H0000
#H0000
#00010
COM
SLAV
FCT
TIMEOUT
ADDR
NB
DATA
RDY
ERR
ERNO
```

FBD:

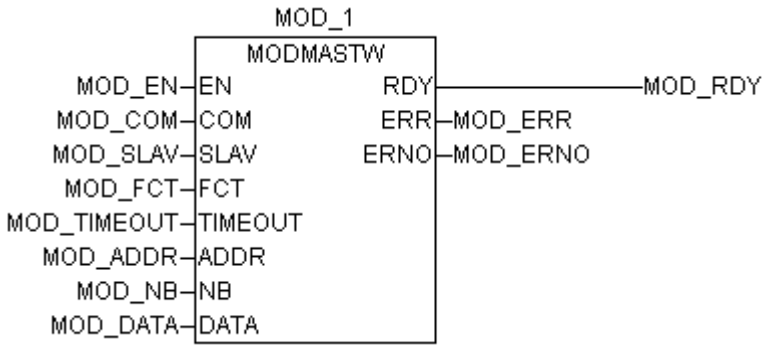


ABB AWL des Beispiels:

```
!BA 0
CALLUP
M0,0
#H0306
#H0000
#H0000
#00010
KW2,0          ; 1
KW2,1          ; 10
KW2,2          ; 3
KW2,3          ; 5000
KW2,4          ; 8192
KW2,5          ; 16
MW30,0
M30,0
M30,1
MW39,0
```

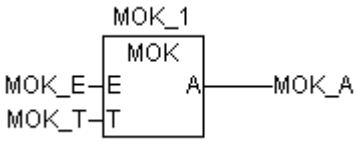
Funktionsaufruf in AWL

```
CAL   MOD_1(EN := MOD_EN,
            COM := MOD_COM,
            SLAV := MOD_SLAV,
            FCT := MOD_FCT,
            TIMEOUT := MOD_TIMEOUT,
            ADDR := MOD_ADDR,
            NB := MOD_NB,
            DATA := MOD_DATA)

LD    MOD_1.RDY
ST    MOD_RDY
LD    MOD_1.ERR
ST    MOD_ERR
LD    MOD_1.ERNO
ST    MOD_ERNO
```

Funktionsaufruf in ST

```
MOD_1     (EN := MOD_EN,
           COM := MOD_COM,
           SLAV := MOD_SLAV,
           FCT := MOD_FCT,
           TIMEOUT := MOD_TIMEOUT,
           ADDR := MOD_ADDR,
           NB := MOD_NB,
           DATA := MOD_DATA)

MOD_RDY   := MOD_1.RDY;
MOD_ERR   := MOD_1.ERR;
MOD_ERNO  := MOD_1.ERNO;
```

Monostabiles Kippglied »Konstant« MOK S40



Eine FALSE/TRUE-Flanke am Eingang E bewirkt eine FALSE/TRUE-Flanke am Ausgang A. Bleibt der Eingang E auf TRUE-Pegel, wird nach der Zeitdauer T eine TRUE/FALSE-Flanke am Ausgang A ausgegeben.

Eine zweite FALSE/TRUE-Flanke des Eingangs E vor Ablauf der Zeit T wird ignoriert.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	MOK	Instanzname
E	BOOL	Eingangssignal
T	TIMER	Impulslänge
A	BOOL	Ausgangssignal, Operanden M, A (nicht E, S)

Beschreibung

Eine FALSE/TRUE-Flanke am Eingang E bewirkt eine FALSE/TRUE-Flanke am Ausgang A. Bleibt der Eingang E auf TRUE-Pegel, wird nach der Zeitdauer T eine TRUE/FALSE-Flanke am Ausgang A ausgegeben.

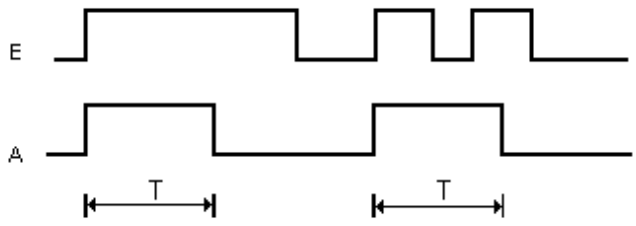
Eine zweite FALSE/TRUE-Flanke des Eingangs E vor Ablauf der Zeit T wird ignoriert.

Gültiger Zeitbereich: 5 ms .. 24,8 Tage

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Sinnvoller Bereich für T: > 1 Zykluszeit

Die Eingänge und der Ausgang sind weder doppelbar noch invertierbar.



Allgemeines Verhalten

• Gestartete Zeitwerke werden vom Betriebssystem der SPS bearbeitet und sind deshalb vollkommen unabhängig von der Bearbeitung des SPS-Programms. Erst nach Ablauf des Zeitwerks erfolgt eine entsprechende Meldung des Betriebssystems an den zugehörigen Zeitbaustein im SPS-Programm.

• Die Bearbeitung eines Zeitwerks im Betriebssystem der SPS wird durch folgende Befehle beeinflußt: Alle laufenden Zeitwerke werden gestoppt und initialisiert, wenn einer der folgenden Fehler auftritt:

- SPS-Programm abbrechen
- RUN/STOP-Schalter von RUN -> STOP
- Warm- oder Kaltstart

Beispiel

Deklaration:

```
MOK_1 : MOK;
MOK_E AT %MX0.0 : BOOL;
MOK_T AT %MD4002.1 : TIME := 16#1s; (*1000 ms*)
MOK_A AT %MX0.1 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
MOK
E
T
A
```

FBD:

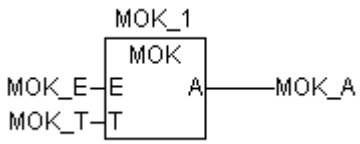


ABB AWL des Beispiels:

```
!BA 0
MOK
M0,0
KD2,1                    ; 1000
M0,1
```

Funktionsaufruf in AWL

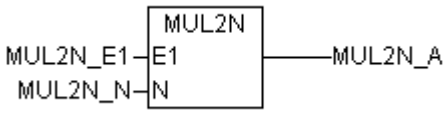
```
CAL   MOK_1(E := MOK_E,
            T := MOK_T)

LD    MOK_1.A
ST    MOK_A
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
MOK_1    (E := MOK_E,
          T := MOK_T);

MOK_A    := MOK_1.A;
```

Multiplikation mit 2 hoch N, Wort MUL2N S40



Der Wert des Operanden am Eingang E1 wird N-mal bitweise geschoben.

Ist der Wert am Eingang N positiv, wird nach links geschoben. Dies entspricht pro Schieben um eine Bitposition einer Multiplikation des aktuellen Wertes mit 2.

Ist der Wert am Eingang N negativ, wird nach rechts geschoben. Dies entspricht pro Schieben um eine Bitposition einer Division des aktuellen Wertes durch 2.

Das Ergebnis wird dem Operanden am Ausgang A1 zugewiesen.

Bausteintyp

Funktion

Parameter

E1	INT	Eingangsoperand
N	INT	Anzahl
A1	INT	Ergebnis

Beschreibung

Der Wert des Operanden am Eingang E1 wird N-mal bitweise geschoben.

Ist der Wert am Eingang N positiv, wird nach links geschoben. Dies entspricht pro Schieben um eine Bitposition einer Multiplikation des aktuellen Wertes mit 2.

Ist der Wert am Eingang N negativ, wird nach rechts geschoben. Dies entspricht pro Schieben um eine Bitposition einer Division des aktuellen Wertes durch 2.

Das Ergebnis wird dem Operanden am Ausgang A1 zugewiesen.

Die Eingänge und der Ausgang sind weder doppelbar noch negierbar.

Sinnvoller Bereich für N: $-14 \leq N \leq +14$

Bei N = 0 wird der Wert am Eingang E1 direkt an den Ausgang A1 gegeben.

Vorzeichen des Wertes am Eingang E1:

Das Vorzeichen des Wertes E1 wird durch die Schiebeoperation nicht beeinflußt. D. h. das Vorzeichen des Ausgangswertes ist immer identisch mit dem des Eingangswertes.

Links-Schieben (Multiplikation):

Beim Links-Schieben des Wertes am Eingang wird jeweils das frei werdende Bit 0 mit 0 aufgefüllt. Das Vorzeichenbit (Bit 15) wird dabei nicht verändert, da vorher auf den Grenzwert des Zahlenbereichs begrenzt wird.

Begrenzung des Wertes am Ausgang A1 beim Links-Schieben:

- Für positive Werte am Eingang E1 gilt: Ist Bit 14 mit einer »1« besetzt und sind aufgrund des Wertes am Eingang N noch Schiebeoperationen auszuführen, so werden diese nicht mehr ausgeführt. Statt dessen wird der Ausgang auf den Grenzwert des positiven Zahlenbereichs gesetzt. D.h. spätestens nach 14maligem Schieben ist in jedem Fall der Grenzwert erreicht.

  Grenzwert: Ausgang A1 = +32767 (7FFFH).

- Für negative Werte am Eingang E1 gilt: Ist Bit 14 mit einer »0« besetzt und sind aufgrund des Wertes am Eingang N noch Schiebeoperationen auszuführen, so werden diese nicht mehr ausgeführt. Statt dessen wird der Ausgang auf den Grenzwert des negativen Zahlenbereichs gesetzt. D.h. spätestens nach 14maligem Schieben ist in jedem Fall der Grenzwert erreicht.

  Grenzwert: Ausgang A1 = -32767 (8001H).

Rechts-Schieben (Division):

Beim Rechts-Schieben rückt jedes Bit eine Stelle nach rechts. Das Vorzeichenbit (Bit 15) behält dabei seinen Wert immer bei. Das frei werdende Bit (Bit 14) wird jeweils mit dem Wert des Vorzeichenbits aufgefüllt.

Begrenzung des Wertes am Ausgang beim Rechts-Schieben:

- Für positive Werte am Eingang E1 gilt: Ist nur noch Bit 0 mit einer »1« besetzt und sind aufgrund des Wertes am Eingang N noch Schiebeoperationen auszuführen, so wird der Ausgang auf den Wert 0 gesetzt. D.h. spätestens nach 14maligem Schieben ist in jedem Fall der Wert 0 erreicht.

  Ausgang A1 = 0.

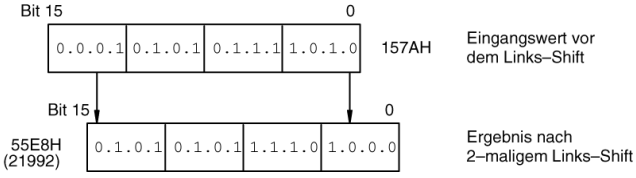
- Für negative Werte am Eingang E1 gilt: Sind aufgrund des Schiebevorgangs Bit 0 … Bit 15 mit einer »1« besetzt, so ist der Grenzwert (-1) erreicht. Weitere Schiebevorgänge bleiben ohne Wirkung. D. h. spätestens nach 15maligem Schieben ist in jedem Fall der Wert -1 erreicht.

  Ausgang A1 = -1 (FFFFH).

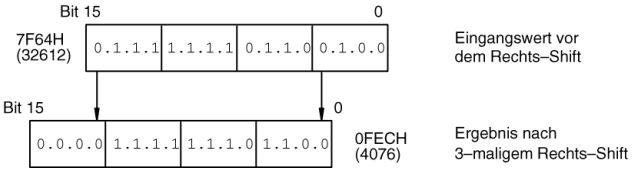
Die Eingänge und der Ausgang sind weder doppelbar noch negierbar.

Beispiele

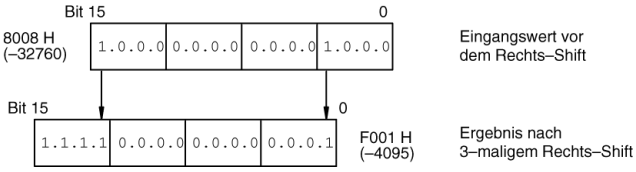
1. Eingangswert E1 = 5498 (157A hex)
   Exponent N = 2 -> 2 * Links-Schieben



2. Eingangswert E1 = 32612 (7F64 hex)
   Exponent N = -3 -> 3 * Rechts-Schieben



3. Eingangswert E1 = -32612 (8008 hex)
   Exponent N = -3 -> 3 * Rechts-Schieben



Beispiel

Deklaration:

```
MUL2N_E1 AT %MW1000.0 : INT;
MUL2N_N AT %MW1000.1 : INT;
MUL2N_A AT %MW1000.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
MUL2N
E1
N
A
```

FBD:

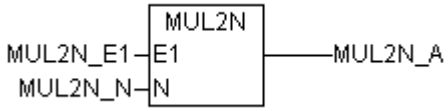


ABB AWL des Beispiels:

```
!BA 0
MUL2N
MW0,0
MW0,1
MW0,2
```

Funktionsaufruf in AWL

```
LD      MUL2N_E1
MUL2N MUL2N_N
ST      MUL2N_A
```

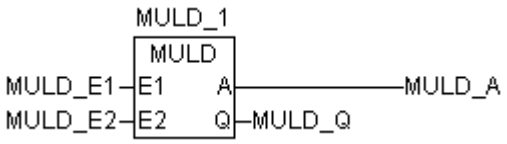
Funktionsaufruf in ST

```
MUL2N_A:=MUL2N(MUL2N_E1, MUL2N_N);
```

Multiplikation Doppelwort MULD S40

Der Wert des Operanden am Eingang E1 wird mit dem Wert des Operanden am Eingang E2 multipliziert und das Ergebnis dem Operanden am Ausgang A zugewiesen.

Das Ergebnis wird auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt. Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein TRUE-Signal zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein FALSE-Signal zugewiesen.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	MULD	Instanzname
E1	DINT	Multiplikand
E2	DINT	Multiplikator
A	DINT	Ergebnis (Produkt)
Q	BOOL	Ergebnis begrenzt, Operanden M, A (nicht E, S)

Beschreibung

Der Wert des Operanden am Eingang E1 wird mit dem Wert des Operanden am Eingang E2 multipliziert und das Ergebnis dem Operanden am Ausgang A zugewiesen.

Das Ergebnis wird auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt (Zahlenbereich: -2147483647 … 2147483647). Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein TRUE-Signal zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein FALSE-Signal zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
MULD_1 : MULD;
MULD_E1 AT %MD2000.0 : DINT;
MULD_E2 AT %MD2000.1 : DINT;
MULD_A AT %MD2000.2 : DINT;
MULD_Q AT %M0.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
MULD
E1
E2
A
Q
```

FBD:

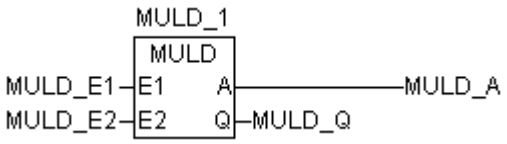


ABB AWL des Beispiels:

```
!BA 0
MULD
MD0,0
MD0,1
MD0,2
M0,0
```

Funktionsaufruf in AWL

```
CAL   MULD_1(E1 := MULD_E1,
      E2 := MULD_E2)

LD    MULD_1.Q
ST    MULD_Q
LD    MULD_1.A
ST    MULD_A
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

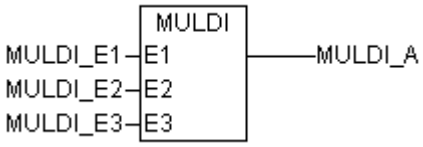
Funktionsaufruf in ST

```
MULD_1   (E1 := MULD_E1,
         E2 := MULD_E2);

MULD_A   := MULD_1.A;
MULD_Q   := MULD_1.Q;
```

Multiplikation mit Division MULDI S40

Der Wert des Operanden am Eingang E1 wird mit dem Wert des Operanden am Eingang E2 multipliziert, das Zwischenergebnis wird durch den Wert des Operanden am Eingang E3 dividiert und dann das Ergebnis dem Operanden am Ausgang A zugewiesen.

Das Ergebnis wird auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt.



Bausteintyp

Funktion

Parameter

E1	INT	Multiplikand
E2	INT	Multiplikator
E3	INT	Divisor
A	INT	Ergebnis

Beschreibung

Der Wert des Operanden am Eingang E1 wird mit dem Wert des Operanden am Eingang E2 multipliziert, das Zwischenergebnis wird durch den Wert des Operanden am Eingang E3 dividiert und dann das Ergebnis dem Operanden am Ausgang A zugewiesen.

Intern arbeitet der Baustein bei der Multiplikation und Division mit Doppelwortgenauigkeit (32 Bit). Erst bei der Zuweisung des Ergebnisses an den Ausgang A erfolgt die Begrenzung auf Wortgenauigkeit (16 Bit). Ist der Rest der Division >0,5 wird das Ergebnis aufgerundet. Entsteht bei der Division ein Zahlenüberlauf (z. B. Division durch Null), so wird dem Ausgang A der vorzeichenrichtige Grenzwert des Zahlenbereiches zugewiesen.

Das Ergebnis wird so begrenzt, daß der Wert 32767 nicht überschritten und der Wert -32767 nicht unterschritten wird.

Die Eingänge und der Ausgang sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
MULDI_E1 AT %MW1000.0 : INT;
MULDI_E2 AT %MW1000.1 : INT;
MULDI_E3 AT %MW1000.2 : INT;
MULDI_A AT %MW1000.3 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
MULDI
E1
E2
E3
A
```

FBD:

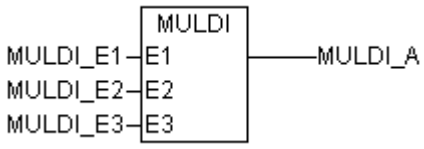


ABB AWL des Beispiels:

```
!BA 0
MULDI
MW0,0
MW0,1
MW0,2
MW0,3
```

Funktionsaufruf in AWL

```
LD      MULDI_E1
MULDI   MULDI_E2,MULDI_E3
ST      MULDI_A
```

Funktionsaufruf in ST

```
MULDI_A := MULDI   (MULDI_E1,
                    MULDI_E2,
                    MULDI_E3);
```

Negation Wort NEGW S40

Der Wert des Operanden am Eingang E wird negiert und das Ergebnis dem Operanden am Ausgang A zugewiesen.

Bausteintyp

Funktion

Parameter

E1	INT	Eingangswert
A	INT	negierter Wert

Beschreibung

Der Wert des Operanden am Eingang E wird negiert und das Ergebnis dem Operanden am Ausgang A zugewiesen.

Ein- und Ausgang sind weder doppelbar noch negierbar.

Zahlenbereich:

Untere Grenze: 8000_H - 32768

Obere Grenze $7FFF_H$ + 32767

Hinweis: Ist E1 = -32768 dann ist A = +32767.

Beispiel

Deklaration:

```
NEGW_E1 AT %MW1000.0 : INT;
NEGW_A AT %MW1000.1 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
NEGW
E1
A
```

FBD:

ABB AWL des Beispiels:

```
!BA 0
NEGW
MW0,0
MW0,1
```

Funktionsaufruf in AWL

```
LD      NEGW_E1
NEGW
ST      NEGW_A
```

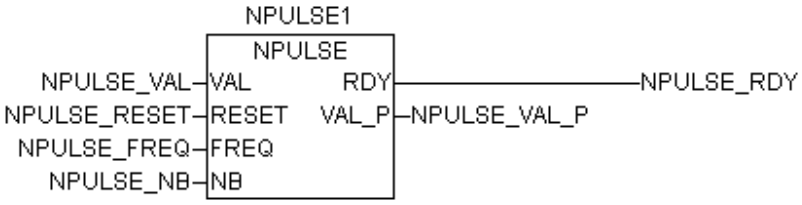
Funktionsaufruf in ST

```
NEGW_A:=NEGW(NEGW_E1);
```

Impulsgeber NPULSE S40

Der Funktionsbaustein erzeugt Impulsfolgen mit bestimmten Frequenzen und bestimmter Anzahl von Impulsen. Er eignet sich damit zur Ansteuerung von Schrittmotoren.

Die Impulsfolgen stehen am Ausgang A 62,00 / %QX62.00 zur Verfügung.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

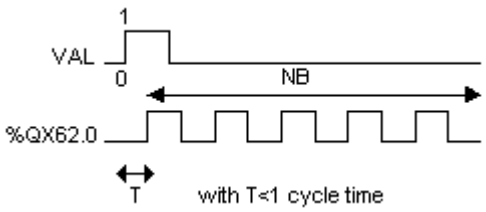
Instanz	NPULSE	Instanzname
VAL	BOOL	Freigabe
RESET	BOOL	Reset
FREQ	INT	Frequenz
NB	INT	Anzahl der Impulse
RDY	BOOL	Ready, Operanden M, A (nicht E, S)
VAL_P	INT	Anzahl der abgelaufenen Impulse

Beschreibung

Der Funktionsbaustein erzeugt Impulsfolgen mit bestimmten Frequenzen und bestimmter Anzahl von Impulsen. Er eignet sich damit zur Ansteuerung von Schrittmotoren.

Die Impulsfolgen stehen am Ausgang A 62,00 / %QX62.00 zur Verfügung.

Am Ausgang VAL_P kann die gerade erzeugte Anzahl Impulse abgelesen werden. Die Impulsausgabe wird durch eine positive Flanke am Eingang VAL gestartet. Die erste Periode startet mit dem binären Wert 1.



VAL **BOOL**

Freigabe für die Ausgabe der Impulsfolge am festen Ausgang A 62,00 / %QX62.0.

FALSE/TRUE-Flanke am Eingang VAL:
Der Impuls-Modus wird gesetzt.
A62,00 / %QX62.00 können nicht normal verwendet werden.

RESET **BOOL**

RESET = FALSE stoppt den Impulsegeber, der Ausgang A62,00 / %QX62.0 wird solange auf FALSE gesetzt, bis RESET = TRUE ist.

FREQ **INT**

Frequenz der Impulsfolge:

Impulsfrequenz: 10 Hz bis 2,604 kHz

Formel zur Frequenzberechnung:

Frequenz [Hz] = 1000000 / ((256 - FREQ)*384)

FREQ	Frequenz
0	10,173 Hz
1	11,212 Hz
2	10,253 Hz
10	10,586 Hz
100	16,693 Hz
254	1,302 kHz
255	2,604 kHz
< 0	10,173 Hz
> 255	2,604 kHz

NB **INT**

Anzahl der Impulse, die erzeugt werden sollen.

Bei NB < 0 erzeugt der Funktionsbaustein NPULSE solange Impulse, bis der Eingang RESET gesetzt wird.

RDY **BOOL**

Ready Bit

RDY = FALSE Impulse werden erzeugt

RDY = TRUE Impulsfolge ist beendet

VAL_P **INT**

Anzahl der gerade erzeugten Impulse.

Diese Anzahl wird vom Funktionsbaustein bei jeder Zykluszeit geschätzt. Sie repräsentiert nicht den exakten Wert.

Das interne Speicherbit %QX62.00 wird während der Impulserzeugung gesperrt.

Beispiel

Deklaration:

```
NPULSE1 : NPULSE;
NPULSE_VAL AT %MX0.0 : BOOL;
NPULSE_RESET AT %MX0.1 : BOOL;
NPULSE_FREQ AT %MW1000.0 : INT;
NPULSE_NB AT %MW1000.1 : INT;
NPULSE_RDY AT %MX0.2 : BOOL;
NPULSE_VAL_P AT %MW1000.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
NPULSE
VAL
RESET
FREQ
NB
RDY
VAL_P
```

FBD:

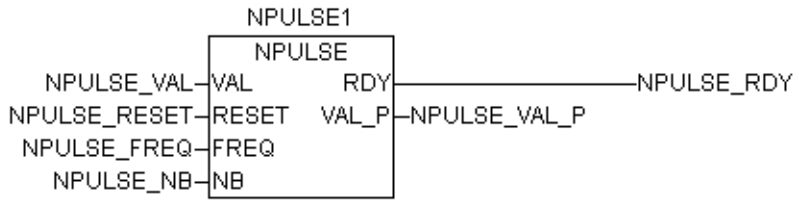


ABB AWL des Beispiels:

```
!BA 0
NPULSE
M0,0
M0,1
MW0,0
MW0,1
M0,2
MW0,2
```

Funktionsaufruf in AWL

```
CAL   NPULSE1(VAL := NPULSE_VAL,
              RESET := NPULSE_RESET,
              FREQ := NPULSE_FREQ,
              NB := NPULSE_NB)

LD    NPULSE1.RDY
ST    NPULSE_RDY
LD    NPULSE1.VAL_P
ST    NPULSE_VAL_P
```

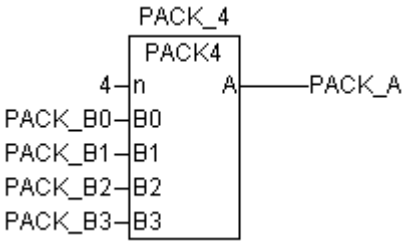
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
NPULSE1       (VAL := NPULSE_VAL,
               RESET := NPULSE_RESET,
               FREQ := NPULSE_FREQ,
               NB := NPULSE_NB);

NPULSE_RDY    := NPULSE1.RDY;
NPULSE_VAL_P  := NPULSE1.VAL_P;
```

Packen binärer Variablen in Wort PACK(4..16) S40

Der Baustein packt n Binärvariablen in eine Wortvariable.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	PACK	Instanzname
N	INT	Anzahl der zu packenden Binärvariablen
B0..B15	BOOL	zu packende Binärvariable
A	INT	Wortvariable

Beschreibung

Die PACK-Nummer gibt die max. Anzahl der zu packenden Bits an. Es stehen folgende PACK-Bausteine zur Verfügung:

PACK4	PACK für max. 2 Binär-Variablen
PACK8	PACK für max. 8 Binär-Variablen
PACK16	PACK für max. 16 Binär-Variablen

Der Baustein packt n Binärvariablen in eine Wortvariable.

n **INT**

Am Eingang n wird die Anzahl der zu packenden Binärvariablen angegeben.

Es gilt:

$1 \leq n \leq 4,(8, 16)$
n = 0 ist unzulässig! (Vorgabe = 1)

B0…B15 **BOOL**

An den Eingängen B0 … Bn-1 werden die zu packenden Binärvariablen vorgegeben.

A **INT**

Der Wert jeder Binärvariable an den Eingängen B0 … Bn-1 wird in das entsprechende Bit (Bit 0 … Bit 15) der Variable am Ausgang A geladen.

Zuordnung:

B0 → Bit0 der Ausgangsvariable
B1 → Bit1 der Ausgangsvariable
. .
B15 → Bit15 der Ausgangsvariable

Hinweis:
Werden vom Anwender weniger als 16 binäre Eingangsvariablen projektiert, so werden die nicht benötigten Bits der Ausgangsvariable auf den Wert FALSE gesetzt.

Beispiel

Deklaration:

```
PACK_16 : PACK;
PACK_B0 AT %MX0.0 : BOOL;
PACK_B1 AT %MX0.1 : BOOL;
PACK_B2 AT %MX0.2 : BOOL;
PACK_B3 AT %MX0.3 : BOOL;
PACK_A AT %MW1000.0 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
PACK
#n
B0
B1
B2
B3
A
```

FBD:

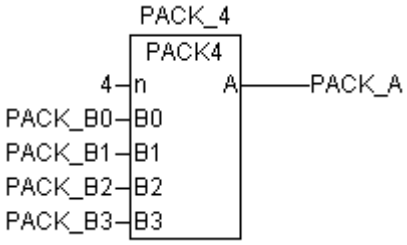


ABB AWL des Beispiels:

```
!BA 0
PACK
#4
M0,0
M0,1
M0,2
M0,3
MW0,0
```

Funktionsaufruf in AWL

```
CAL   PACK_1(n := 4,
              B0 := PACK_B0,
              B1 := PACK_B1,
              B2 := PACK_B2,
              B3 := PACK_B3)

LD    PACK_1.A
ST    PACK_A
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

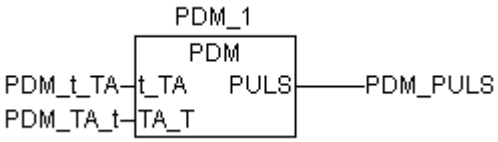
Funktionsaufruf in ST

```
PACK_1      (n := 16,
             B0 := PACK_B0,
             B1 := PACK_B1,
             B2 := PACK_B2,
             B3 := PACK_B3);

PACK_A:=PACK_1.A;
```

Puls-Dauer-Modulator PDM S40

Der Funktionsbaustein erzeugt an seinem Ausgang PULS ein pulsdauermoduliertes binäres Signal.

Am Eingang t_TA wird das Tastverhältnis und am Eingang TA_T wird die Periodendauer für das Ausgangssignal vorgegeben.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	PDM	Instanzname
t_TA	INT	Tastverhältnis
TA_T	INT	Periodendauer bezogen auf die Zykluszeit
PULS	BOOL	Pulsdauermoduliertes Signal

Beschreibung

Der Funktionsbaustein erzeugt an seinem Ausgang PULS ein pulsdauermoduliertes binäres Signal.

Am Eingang t_TA wird das Tastverhältnis und am Eingang TA_T wird die Periodendauer für das Ausgangssignal vorgegeben.

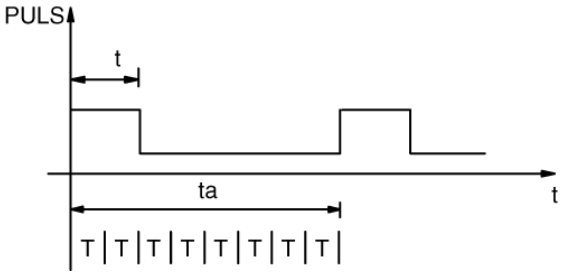
Die Eingänge und der Ausgang können weder gedoppelt noch negiert noch invertiert werden.

t_TA INT

Am Eingang t_TA wird das gewünschte Tastverhältnis für das Ausgangssignal PULS vorgegeben. Dabei ist TA die Periodendauer des Signals am Ausgang PULS, und t ist die Zeit innerhalb der Periodendauer TA, während der das Ausgangssignal TRUE-Pegel annimmt. Der Vorgabewert für das gewünschte Tastverhältnis am Eingang t_TA muß in skalierter Form angegeben werden. Dazu ist das gewünschte Tastverhältnis mit dem Wert 32767 zu multiplizieren und auf eine ganze Zahl zu runden. Der so erhaltene Zahlenwert wird dann am Eingang t_TA angegeben.

Randbedingung für t: t > T

D. h. die gewünschte Einschaltdauer des Ausgangssignals muß größer sein als die Zykluszeit des SPS-Programms (T).



Für die Vorgabe des Tastverhältnisses am Eingang t_TA gilt folgender Zusammenhang:

Der skalierte Wert am Eingang t_TA ergibt das Tastverhältnis am Ausgang PULS:

Eingang t_TA	Ausgang PULS
negativer Wert	0 (0 %)
0 (0 * 32767)	0 (0 %)
:	: :
:	: :
16384 (0,5 * 32767)	0,5 (50 %)
:	: :
:	: :
32767 (1 * 32767)	1 (100 %)

TA_T INT

Am Eingang TA_T wird die gewünschte Periodendauer TA für das Signal am Ausgang PULS angegeben. Die Periodendauer TA ist dabei auf die Zykluszeit T zu normieren.

Randbedingung für TA:

- TA muß ein ganzzahliges Vielfaches von T sein (TA = n * T)
- TA >> T > 0; je größer TA im Verhältnis zu T ist, desto genauer wird das gewünschte Tastverhältnis eingehalten

Beispiel:
TA > 10 * T → Ungenauigkeit des Tastverhältnisses am Ausgang PULS < 10 %.

Wird für TA_T der Wert TA_T < 0 vorgegeben, so ersetzt der Baustein diesen unsinnigen Wert automatisch durch den Wert 32767.

PULS **BOOL**

Am Ausgang PULS steht das pulsdauermodulierte Signal zur Verfügung.

Kombination des Bausteins PDM mit einem Regler:

Wird der Baustein PDM an den Ausgang eines Reglers angeschlossen um einen »schaltenden« Regler zu realisieren, so gelten folgende Randbedingungen:

- Periodendauer TA vom PDM = Abtastzeit des Reglers.
- Periode des Pulssignals muß synchron zur Periode der Abtastzeit des Reglers sein.

Diese Randbedingungen werden dadurch erfüllt, daß der Regler im gleichen SPS-Programm wie der PDM, aber innerhalb eines Laufzahlblocks, projektiert ist. Durch den Laufzahlblock wird die Abtastzeit des Reglers um ein ganzzahliges Vielfaches der Zykluszeit verlängert. Der Regler wird also innerhalb des Laufzahlblocks weniger oft bearbeitet als der PDM außerhalb des Laufzahlblocks.

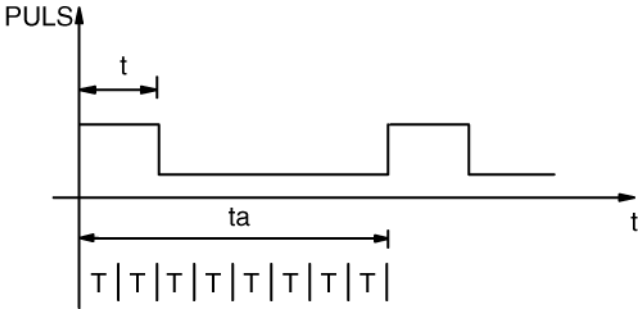
Beispiel

Gewünscht:

- Tastverhältnis: t_TA = 0,25 (25 %)
- Periodendauer: TA = 800 ms (nur ganzzahliges Vielfaches der SPS-Zykluszeit möglich)
- Zykluszeit: T = 100 ms

Vorzugebende Bausteinparameter:

- Wert am Eingang t_TA: 8192 (0,25 * 32767)
- Wert am Eingang TA_T: 8 (800 ms/100 ms)



Beispiel

Deklaration:

```
PDM_1 : PDM;
PDM_t_TA AT %MW1000.0 : INT;
PDM_TA_t AT %MW1000.1 : INT;
PDM_PULS AT %MX0.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
PDM
t_TA
TA_t
PULS
```

FBD:

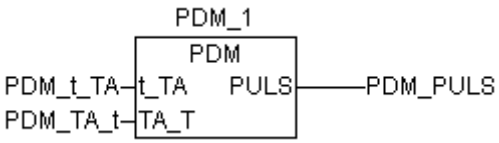


ABB AWL des Beispiels:

```
!BA 0
PDM
MW0,0
MW0,1
M0,0
```

Funktionsaufruf in AWL

```
CAL   PDM_1(t_TA := PDM_t_TA,
            TA_T := PDM_TA_t)

LD    PDM_1.PULS
ST    PDM_PULS
```

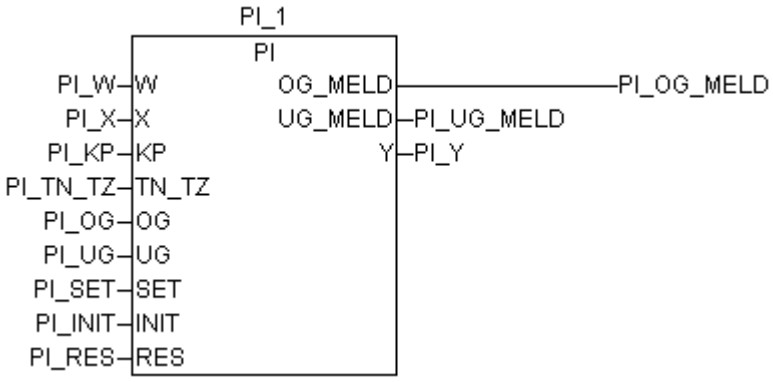
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
PDM_1      (t_TA := PDM_t_TA,
            TA_T := PDM_TA_t);

PDM_PULS:=PDM_1.PULS;
```

Proportional-Integral-Regler PI S40

Der PI-Regler ändert den Wert an seinem Ausgang Y (Stellgröße) solange, bis der Wert am Eingang X (Regelgröße) gleich dem Wert am Eingang W (Führungsgröße) ist.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	PI	Instanzname
W	INT	Führungsgröße (Sollwert)
X	INT	Regelgröße (Istwert)
KP	INT	Proportionalbeiwert, Angabe in Prozent
TN_TZ	INT	Nachstellzeit normiert auf SPS-Zykluszeit
OG	INT	obere Grenze für die Stellgröße Y
UG	INT	untere Grenze für die Stellgröße Y
SET	BOOL	Freigabe zum Setzen der Stellgröße Y auf Initialwert INIT
INIT	INT	Initialwert für die Stellgröße Y
RES	BOOL	Rücksetzen der Stellgröße Y auf den Wert 0
OG_MELD	BOOL	Oberer Grenzwert ist erreicht, Operanden M, A (nicht E, S)
UG_MELD	BOOL	Oberer Grenzwert ist erreicht, Operanden M, A (nicht E, S)
Y	INT	Ausgang für die Stellgröße Y

Beschreibung

Der PI-Regler ändert den Wert an seinem Ausgang Y (Stellgröße) solange, bis der Wert am Eingang X (Regelgröße) gleich dem Wert am Eingang W (Führungsgröße) ist.

Regelalgorithmus: einfache Rechteckregel

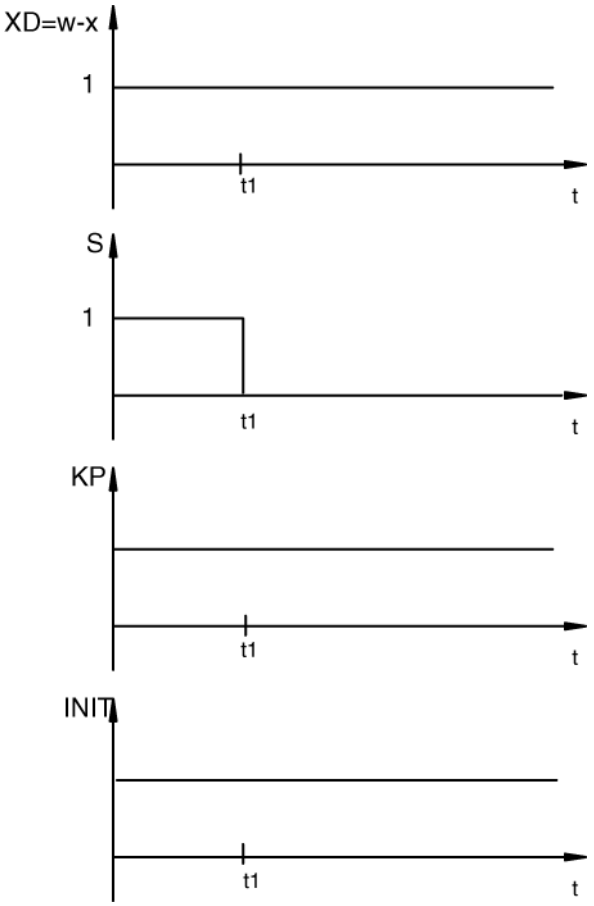
$$Y = \frac{KP}{100} * \frac{W-X}{TN\_TZ} + YI(z-1) + \frac{KP}{100} * (W-X)$$

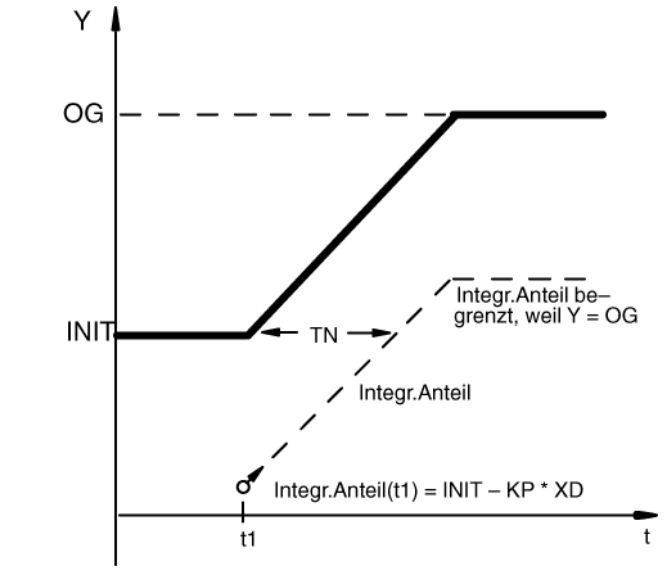
Dabei ist: YI(z-1) der Integralanteil aus dem vorhergehenden Programmzyklus

Übertragungsfunktion:

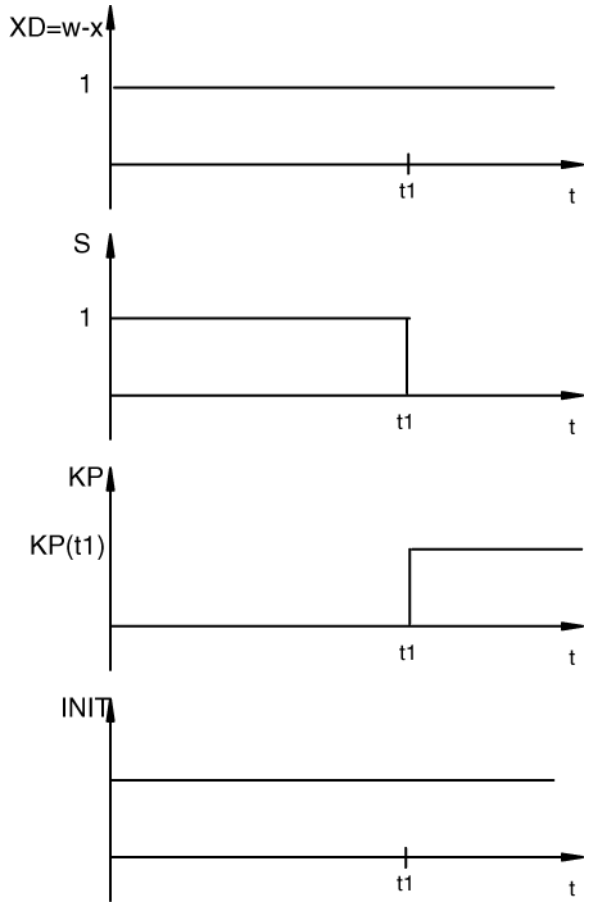
$$F(s) = KP * (1+ \frac{1}{s * TN})$$

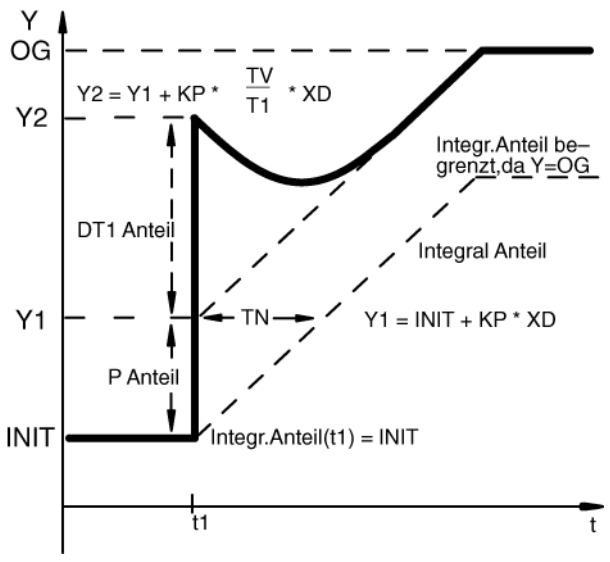
Die Ein- und Ausgänge sind weder doppelbar noch negierbar/invertierbar.





PI-Regler: Stoßfreier Übergang vom vorgegebenen Initialwert in den Regelbetrieb





PI-Regler: Stoßbehafteter Übergang vom vorgegebenen Initialwert in den Regelbetrieb

W **INT**

Die Führungsgröße (Sollwert) wird am Eingang W vorgegeben.

X **INT**

Die Regelgröße (Istwert) wird am Eingang X vorgegeben.

KP **INT**

Der Proportionalbeiwert wird am Eingang KP angegeben. Die Angabe dieses Wertes erfolgt in Prozent und kann positiv oder negativ sein.

Beispiel:

```
   1 =    1 Prozent
  55 =   55 Prozent
 100 =  100 Prozent
1000 = 1000 Prozent
-500 = -500 Prozent
```

- 1 Prozent heißt, daß der Baustein die Regeldifferenz mit dem Faktor 0,01 multipliziert (s. a. Regelalgorithmus).
- 100 Prozent heißt, daß der Baustein die Regeldifferenz mit dem Faktor 1 multipliziert (s. a. Regelalgorithmus).
- 1000 Prozent heißt, daß der Baustein die Regeldifferenz mit dem Faktor 10 multipliziert (s. a. Regelalgorithmus).

Proportionalbeiwerte größer als 1000 Prozent sind regelungstechnisch in der Regel nicht sinnvoll.

TN_TZ **INT**

Die Nachstellzeit TN wird auf die SPS-Zykluszeit TZ normiert und am Eingang TN_TZ angegeben.

Wertebereich: 0 <= TN_TZ <= 328

- Werden Werte vorgegeben, die außerhalb des zulässigen Wertebereichs liegen, so arbeitet die SPS grundsätzlich mit dem Wert 328.
- Eine große Nachstellzeit TN kann dadurch erreicht werden, daß auch die Zykluszeit T groß gewählt wird. Befindet sich der Baustein innerhalb eines Laufzahlblockes, so ist für ihn die Zykluszeit des Laufzahlblockes und nicht die Zykluszeit (KD 0,0) des SPS-Programmes maßgebend.

Begrenzung der Stellgröße Y

OG **INT**
UG **INT**

Der Ausgang Y (Stellgröße) des Reglers kann durch

- Vorgabe eines Grenzwertes am Eingang OG (Obere Grenze) auf einen Maximalwert begrenzt werden.
- Vorgabe eines Grenzwertes am Eingang UG (Untere Grenze) auf einen Minimalwert begrenzt werden.

Die obere und untere Grenze gilt ebenfalls für den reglerinternen I-Anteil. D. h. der I-Anteil kann nur Werte zwischen oberer und unterer Grenze annehmen. Erreicht die Stellgröße Y einen der beiden Grenzwerte, so wird der I-Anteil des Reglers nicht mehr verändert. Dadurch wird verhindert, daß der I-Anteil bei Begrenzung des Regler-Ausgangs Y wegläuft, regelungstechnisch sinnlose Werte annimmt und u. U. erst nach sehr langer Zeit wieder in den Arbeitsbereich zurückkehrt. Dieses Verhalten eines Reglers wird auch als »spezielle Anti-Reset-Windup-Maßnahme (ARW)« bezeichnet.

Setzen und Rücksetzen des Reglers

SET **BOOL**
INIT **INT**
RES **BOOL**

Setzen des Reglers auf einen Initialwert:

- Mit einem TRUE-Signal am Eingang SET (Set) wird der Ausgang Y des Reglers auf den am Eingang INIT angegebenen Initialwert gesetzt.
- Ein TRUE-Signal am Eingang RES (Reset) ist gleichbedeutend wie die Vorgabe des Initialwerts 0 (s. o.).

Stoßfreies Setzen/Rücksetzen

- Mit einem TRUE-Signal am binären Eingang SET (Set) wird der Ausgang Y des Reglers auf den am Eingang INIT angegebenen Initialwert gesetzt.
- Ein TRUE-Signal am Eingang RES (Reset) ist gleichbedeutend wie die Vorgabe des Initialwerts 0.

Dabei erfolgt intern im Regler ein Abgleich auf den Initialwert. Der Abgleich ist eine Verschiebung des Reglerausgangs vom momentanen Wert auf den gewünschten Initialwert. Der Regler arbeitet jetzt von diesem Initialwert aus genau so weiter, wie er es vor der Verschiebung im alten Arbeitspunkt getan hätte, d. h. stoßfrei. Der I-Anteil des Reglers wird dabei so festgelegt, daß die Summe aus P-Anteil und I-Anteil gerade den Initialwert ergibt.

Vorteil des stoßfreien Setzens:
- Der Regelvorgang ab dem neuen Initialwert erfolgt stoßfrei.

Nachteil des stoßfreien Setzens:
- Es gilt die Gleichung: I_Anteil = INIT - P_Anteil

Der I-Anteil wird dabei u. U. auf große Werte gesetzt, und es kann dann sehr lange dauern bis dieser regelungstechnisch »falsche« I-Anteil wieder abgebaut ist.

Stoßbehaftetes Setzen/Rücksetzen

- Mit einem TRUE-Signal am Eingang SET (Set) wird der Ausgang Y des Reglers auf den am Eingang INIT angegebenen Initialwert gesetzt.
- Ein TRUE-Signal am Eingang RES (Reset) ist gleichbedeutend wie die Vorgabe des Initialwerts 0.

Beim stoßbehafteten Setzen bzw. Rücksetzen des Reglers wird der I-Anteil gleich dem Initialwert gesetzt. Der P-Anteil muß dazu während des Setzvorgangs unterdrückt werden.

Es gilt: I-Anteil = INIT

Das stoßbehaftete Setzen auf einen Initialwert wird durch folgende Maßnahme während des Setzvorgangs erreicht:

- Vorgabe des Wertes 0 am Eingang KP.

Durch diese Maßnahme wird der P-Anteil des Reglers unwirksam. Der Regler-Ausgang Y nimmt im Setz-Zyklus den Initialwert an.
Nach dem Setz-Zyklus wird der P-Anteil wieder freigegeben. Der Regler-Ausgang Y macht vom Initialwert aus einen Sprung entsprechend dem P-Anteil des Reglers.

Vorteil des stoßbehafteten Setzens:
- Der I-Anteil wird nicht auf regelungstechnisch »falsche« Werte gesetzt.

Nachteil des stoßbehafteten Setzens:
- Keine Stoßfreiheit

Y **INT**

Am Ausgang Y wird die Stellgröße Y des Reglers ausgegeben.

OG_MELD **BOOL**

Am Ausgang OG_MELD wird signalisiert, ob der Wert am Ausgang Y den vorgegebenen oberen Grenzwert erreicht hat.

OG_MELD = FALSE Grenzwert ist nicht erreicht.
OG_MELD = TRUE Grenzwert ist erreicht.

UG_MELD **BOOL**

Am Ausgang UG_MELD wird signalisiert, ob der Wert am Ausgang Y den vorgegebenen unteren Grenzwert erreicht hat.

UG_MELD = FALSE Grenzwert ist nicht erreicht.
UG_MELD = TRUE Grenzwert ist erreicht.

Beispiel

Deklaration:

```
PI_1: PI;
PI_W AT %MW1000.0: INT;
PI_X AT %MW1000.1: INT;
PI_KP AT %MW3001.0: INT := 100;
PI_TN_TZ AT %MW3001.1: INT := 20;
PI_OG AT %MW3001.2: INT := 20000;
PI_UG AT %MW3001.3: INT := 0;
PI_SET AT %MX0.0: BOOL;
PI_INIT AT %MW3001.4: INT := 10000;
PI_RES AT %MX0.1: BOOL;
PI_OG_MELD AT %MX0.2: BOOL;
PI_UG_MELD AT %MX0.3: BOOL;
PI_Y AT %MW1000.2: INT;
```

Übersetzung in ABB AWL:

```
!BA 0
PI
W
X
KP
TN_TZ
INIT
SET
RES
OG
UG
Y
OG_MELD
UG_MELD
```

FBD:

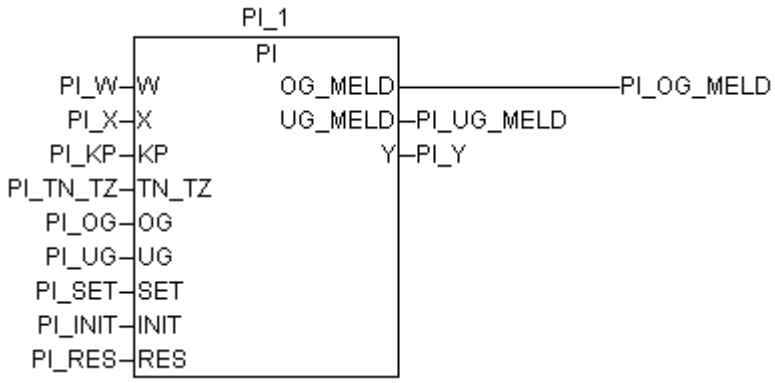


ABB AWL des Beispiels:

```
!BA 0
PI
MW0,0
MW1,1
KW1,0          ; 100
KW1,1          ; 20
M0,0
M0,1
KW1,4          ; 10000
M0,0
M0,1
KW1,2          ; 20000
KW1,3          ; 0
MW0,2
M0,2
M0,3
```

Funktionsaufruf in AWL

```
CAL    PI_1(W := PI_W, X := PI_X, KP := PI_KP,
TN_TZ := PI_TN_TZ, OG := PI_OG, UG := PI_UG,
SET := PI_SET, INIT := PI_INIT, RES := PI_RES)

LD     PI_1.OG_MELD
ST     PI_OG_MELD
LD     PI_1.UG_MELD
ST     PI_UG_MELD
LD     PI_1.Y
ST     PI_Y
```

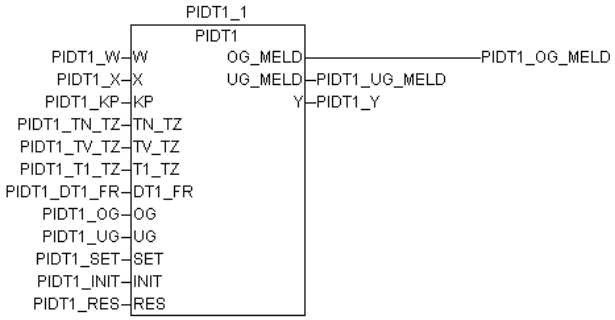
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
PI_1(W := PI_W, X := PI_X, KP := PI_KP,
TN_TZ := PI_TNTZ, OG := PI_OG,
UG := PI_UG, SET := PI_SET, INIT := PI_INIT,
RES := PI_RES);

PI_OG_MELD:=PI_1.OG_MELD;
PI_UG_MELD:=PI_1.UG_MELD;
PI_Y:=PI_1.Y;
```

PIDT1-Regler **PIDT1 S40**

Der PI-Regler ändert seinen Ausgang Y (Stellgröße) solange, bis der Eingang X (Regelgröße) gleich dem Eingang W (Führungsgröße) ist.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	PIDT1	Instanzname
W	INT	Führungsgröße (Sollwert)
X	INT	Regelgröße (Istwert)
KP	INT	Proportionalbeiwert, Angabe in Prozent
TN_TZ	INT	Nachstellzeit normiert auf SPS-Zykluszeit
TV_TZ	INT	Vorhaltezeit normiert auf SPS-Zykluszeit
T1_TZ	INT	Rücklaufzeit normiert auf SPS-Zykluszeit
DT1_FR	BOOL	Freigabe DT1-Anteil
OG	INT	obere Grenze für die Stellgröße Y
UG	INT	untere Grenze für die Stellgröße Y
SET	BOOL	Freigabe zum Setzen auf Initialwert INIT
INIT	INT	Initialwert für die Stellgröße Y
RES	INT	Rücksetzen der Stellgröße Y auf den Wert 0
OG_MELD	BOOL	Oberer Grenzwert ist erreicht, Operanden M, A (nicht E, S)
UG_MELD	BOOL	Unterer Grenzwert ist erreicht, Operanden M, A (nicht E, S)
Y	INT	Ausgang für die Stellgröße Y

Beschreibung

Der PI-Regler ändert seinen Ausgang Y (Stellgröße) solange, bis der Eingang X (Regelgröße) gleich dem Eingang W (Führungsgröße) ist.

Übertragungsfunktion:

$$F(s) = KP * \left(1 + \frac{1}{s * TN} + \frac{s * TV}{1 + (s\ T1)}\right)$$

Regelalgorithmus: Einfache Rechteckregel:

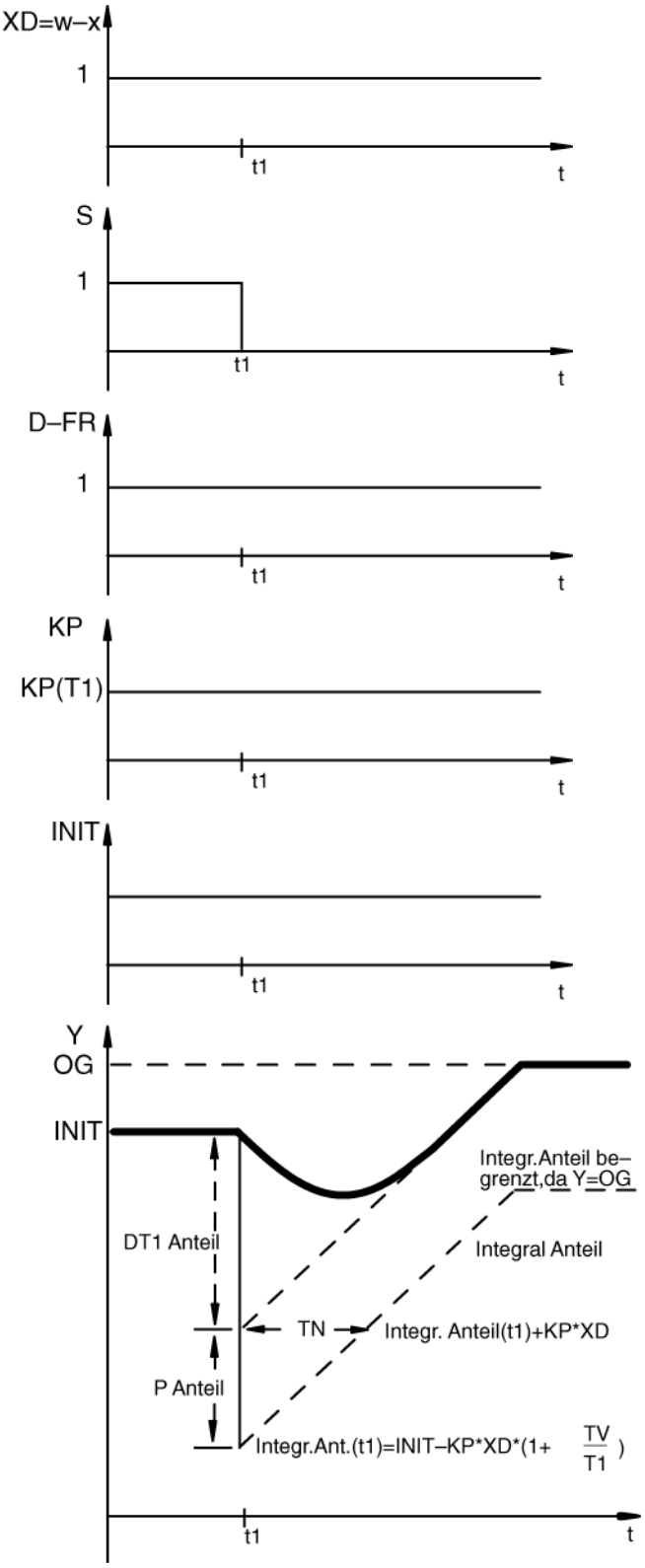
$$Y = \frac{KP*XD}{100} + \frac{KP}{100} * \frac{XD}{TN\_TZ} + YI(z-1) + \frac{T1\_TZ}{1+(T1\_TZ)} * (YDTI(z-1) + \frac{1}{T1\_TZ} * \frac{TV}{TZ} * \frac{KP}{100} * (XD - XD(z-1)))$$

Dabei ist:
YI(z-1): Der Integralanteil aus dem vorhergehenden Programmzyklus

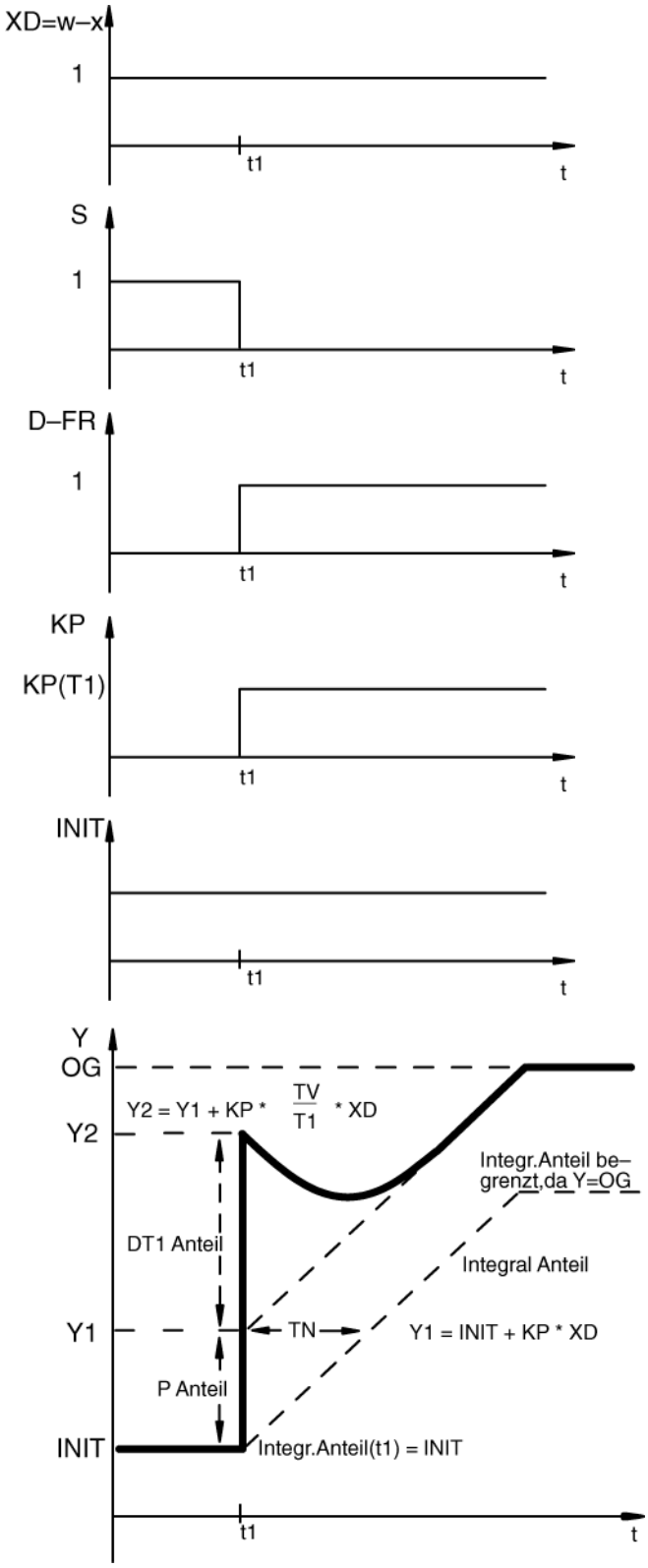
(YDTI(z-1) Der Differentialanteil aus dem vorhergehenden Programmzyklus

XD(z-1): Regeldifferenz aus dem vorhergehenden Programmzyklus

Die Ein- und Ausgänge sind weder doppelbar noch negierbar/invertierbar.



PIDT1-Regler: Stoßfreier Übergang vom vorgegebenen Initialwert in den Regelbetrieb



PIDT1-Regler: Stoßbehafteter Übergang vom vorgegebenen Initialwert in den Regelbetrieb

W **INT**

Die Führungsgröße (Sollwert) wird am Eingang W vorgegeben.

X **INT**

Die Regelgröße (Istwert) wird am Eingang X vorgegeben.

KP **INT**

Der Proportionalbeiwert wird am Eingang KP angegeben. Die Angabe dieses Wertes erfolgt in Prozent und kann positiv oder negativ sein.

Beispiel:

```
   1 =    1 Prozent
  55 =   55 Prozent
 100 =  100 Prozent
1000 = 1000 Prozent
-500 = -500 Prozent
```

- 1 Prozent heißt, daß der Baustein die Regeldifferenz mit dem Faktor 0,01 multipliziert (s. a. Regelalgorithmus).
- 100 Prozent heißt, daß der Baustein die Regeldifferenz mit dem Faktor 1 multipliziert (s. a. Regelalgorithmus).
- 1000 Prozent heißt, daß der Baustein die Regeldifferenz mit dem Faktor 10 multipliziert (s. a. Regelalgorithmus).

Proportionalbeiwerte größer als 1000 Prozent sind regelungstechnisch in der Regel nicht sinnvoll.

TN_TZ **INT**

Die Nachstellzeit TN wird auf die SPS-Zykluszeit TZ normiert und am Eingang TN_TZ angegeben.

Wertebereich: 0 <= TN_TZ <= 328

- Werden Werte vorgegeben, die außerhalb des zulässigen Wertebereichs liegen, so arbeitet die SPS grundsätzlich mit dem Wert 328.
- Eine große Nachstellzeit TN kann dadurch erreicht werden, daß auch die Zykluszeit T groß gewählt wird. Befindet sich der Baustein innerhalb eines Laufzahlblockes, so ist für ihn die Zykluszeit des Laufzahlblockes und nicht die Zykluszeit (KD 00,00 / MD4000.0) des SPS-Programmes maßgebend.

TV_TZ **INT**

Die Vorhaltezeit TV wird auf die SPS-Zykluszeit TZ normiert und am Eingang TV_TZ angegeben:
(0 <= TN_TZ <= 32767).

T1_TZ **INT**

Die Rücklaufzeit T1 wird auf die SPS-Zykluszeit TZ normiert und am Eingang T1_TZ angegeben:
($0 \leq$ TN_TZ $\leq$ 32767).

Die Rücklaufzeit ist die Zeit, in der der DT1-Anteil auf ca. 37 % seines Anfangswertes zurückgegangen ist.

Unzulässige Zeitangaben:
Jeder Zeitwert wird auf den max. positiven Wert 32767 gesetzt, falls der Zeitwert am Eingang versehentlich kleiner oder gleich »0« angegeben wird.

DT1_FR **BOOL**

Mit dem Eingang D-FR kann der DT1-Anteil des Reglers zu- bzw. abgeschaltet werden.

D_FR	Bedeutung
0	DT1-Anteil ist abgeschaltet → reiner PI-Regler
1	DT1-Anteil ist zugeschaltet → PIDT1-Regler

In folgenden Fällen ist es aus regelungstechnischer Sicht oft störend und nicht sinnvoll, daß der DT1-Anteil wirksam ist:

- bei Einschaltvorgängen
- bei großen Regeldifferenzen
- beim Setzen des Reglers auf einen vorgegebenen Initialwert
- beim Rücksetzen des Reglers auf den Wert 0

Man kann außerhalb des Reglers einen Vergleich von Führungsgröße und Regelgröße vornehmen. In Abhängigkeit von diesem Vergleich kann über den Eingang D-FR der DT1-Anteil gezielt eingeschaltet oder ausgeschaltet werden.

Das Einschalten kann dabei z. B. darauf beschränkt werden, daß sich die Regeldifferenz innerhalb einer gewünschten Bandbreite befindet. D. h. der DT1-Anteil ist nur dann im Eingriff, wenn die Regelgröße sich innerhalb einer bestimmten Bandbreite um den Sollwert bewegt. Verläßt die Regelgröße dieses Toleranzband, so wird der DT1-Anteil ausgeschaltet.

Begrenzung der Stellgröße Y

OG	**INT**
UG	**INT**

Der Ausgang Y (Stellgröße) des Reglers kann durch

- Vorgabe eines Grenzwertes am Eingang OG (Obere Grenze) auf einen Maximalwert begrenzt werden.
- Vorgabe eines Grenzwertes am Eingang UG (Untere Grenze) auf einen Minimalwert begrenzt werden.

Die obere und untere Grenze gilt ebenfalls für den reglerinternen I-Anteil. D. h. der I-Anteil kann nur Werte zwischen oberer und unterer Grenze annehmen. Erreicht die Stellgröße Y einen der beiden Grenzwerte, so wird der I-Anteil des Reglers nicht mehr verändert.

Dadurch wird verhindert, daß der I-Anteil bei Begrenzung des Regler-Ausgangs Y wegläuft, regelungstechnisch sinnlose Werte annimmt und u. U. erst nach sehr langer Zeit wieder in den Arbeitsbereich zurückkehrt. Dieses Verhalten eines Reglers wird auch als »spezielle Anti-Reset-Windup-Maßnahme (ARW)« bezeichnet.

Setzen und Rücksetzen des Reglers

SET	**BOOL**
INIT	**INT**
RES	**BOOL**

Setzen des Reglers auf einen Initialwert

- Mit einem TRUE-Signal am Eingang SET (Set) wird der Ausgang Y des Reglers auf den am Eingang INIT angegebenen Initialwert gesetzt.
- Ein 1-Signal am Eingang RES (Reset) ist gleichbedeutend wie die Vorgabe des Initialwerts 0 (s. o.).

Stoßfreies Setzen/Rücksetzen

- Mit einem TRUE-Signal am binären Eingang SET (Set) wird der Ausgang Y des Reglers auf den am Eingang INIT angegebenen Initialwert gesetzt.
- Ein 1-Signal am Eingang RES (Reset) ist gleichbedeutend wie die Vorgabe des Initialwerts 0.

Dabei erfolgt intern im Regler ein Abgleich auf den Initialwert. Der Abgleich ist eine Verschiebung des Reglerausgangs vom momentanen Wert auf den gewünschten Initialwert. Der Regler arbeitet jetzt von diesem Initialwert aus genau so weiter, wie er es vor der Verschiebung im alten Arbeitspunkt getan hätte, d. h. stoßfrei. Der I-Anteil des Reglers wird dabei so festgelegt, daß die Summe aus dem P-Anteil, I-Anteil und DT1-Anteil gerade den Initialwert ergibt.

Vorteil des stoßfreien Setzens:
- Der Regelvorgang ab dem neuen Initialwert erfolgt stoßfrei.

Nachteil des stoßfreien Setzens:
- Es gilt die Gleichung:
  I-Anteil = INIT - P-Anteil - DT1-Anteil

Der I-Anteil wird dabei u. U. auf große Werte gesetzt, und es kann dann sehr lange dauern bis dieser regelungstechnisch »falsche« I-Anteil wieder abgebaut ist.

Stoßbehaftetes Setzen/Rücksetzen

- Mit einem TRUE-Signal am Eingang SET (Set) wird der Ausgang Y des Reglers auf den am Eingang INIT angegebenen Initialwert gesetzt.
- Ein 1-Signal am Eingang RES (Reset) ist gleichbedeutend wie die Vorgabe des Initialwerts 0.
- Beim stoßbehafteten Setzen bzw. Rücksetzen des Reglers wird der I-Anteil gleich dem Initialwert gesetzt. Der P- und der DT1-Anteil müssen dazu während des Setzvorgangs unterdrückt werden.

Es gilt: I-Anteil = INIT

Das stoßbehaftete Setzen auf einen Initialwert wird durch folgende Maßnahmen während des Setzvorgangs erreicht:

- Ausschalten des DT1-Anteils über den Steuereingang D-FR und
- Vorgabe des Wertes 0 am Eingang KP

Durch diese Maßnahmen werden P-Anteil und DT1-Anteil beim Setzen des Reglers unwirksam.

Der Regler-Ausgang nimmt im Setz-Zyklus den Initialwert an.

Nach dem Setz-Zyklus werden P-Anteil und DT1-Anteil wieder freigegeben. Der Regler-Ausgang Y macht vom Initialwert aus einen Sprung entsprechend dem P- und DT1-Anteil des Reglers.

Vorteil des stoßbehafteten Setzens:
- Der I-Anteil wird nicht auf regelungstechnisch »falsche« Werte gesetzt.

Nachteil des stoßbehafteten Setzens:
- Keine Stoßfreiheit

Y	**INT**

Am Ausgang Y wird die Stellgröße Y des Reglers ausgegeben.

OG_MELD **BOOL**

Am Ausgang OG_MELD wird signalisiert, ob der Wert am Ausgang Y den vorgegebenen oberen Grenzwert erreicht hat.

OG_MELD = FALSE Grenzwert ist nicht erreicht.
OG_MELD = TRUE Grenzwert ist erreicht.

UG_MELD **BOOL**

Am Ausgang UG_MELD wird signalisiert, ob der Wert am Ausgang y den vorgegebenen unteren Grenzwert erreicht hat.

UG_MELD = FALSE Grenzwert ist nicht erreicht.
UG_MELD = TRUE Grenzwert ist erreicht.

Beispiel

Deklaration:

```
PIDT1_1 : PIDT1;
PIDT1_W AT %MW1000.0: INT;
PIDT1_X AT %MW1000.1: INT;
PIDT1_KP AT %MW3001.0: INT := 100;
PIDT1_TN_TZ AT %MW3001.1: INT := 20;
PIDT1_TV_TZ AT %MW3001.2 : INT := 0;
PIDT1_T1_TZ AT %MW3001.3 : INT := 0;
PIDT1_DT1_FR AT %MX0.4 : BOOL;
PIDT1_OG AT %MW3001.5: INT := 20000;
PIDT1_UG AT %MW3001.6: INT := 0;
PIDT1_SET AT %MX0.0: BOOL;
PIDT1_INIT AT %MW3001.7: INT := 10000;
PIDT1_RES AT %MX0.1: BOOL;
PIDT1_OG_MELD AT %MX0.2: BOOL;
PIDT1_UG_MELD AT %MX0.3: BOOL;
PIDT1_Y AT %MW1000.2: INT;
```

Übersetzung in ABB AWL:

```
!BA 0
PIDT1
W
X
KP
TN_TZ
TV_TZ
T1_TZ
DT1_FR
INIT
SET
RES
OG
UG
Y
OG_MELD
UG_MELD
```

FBD:

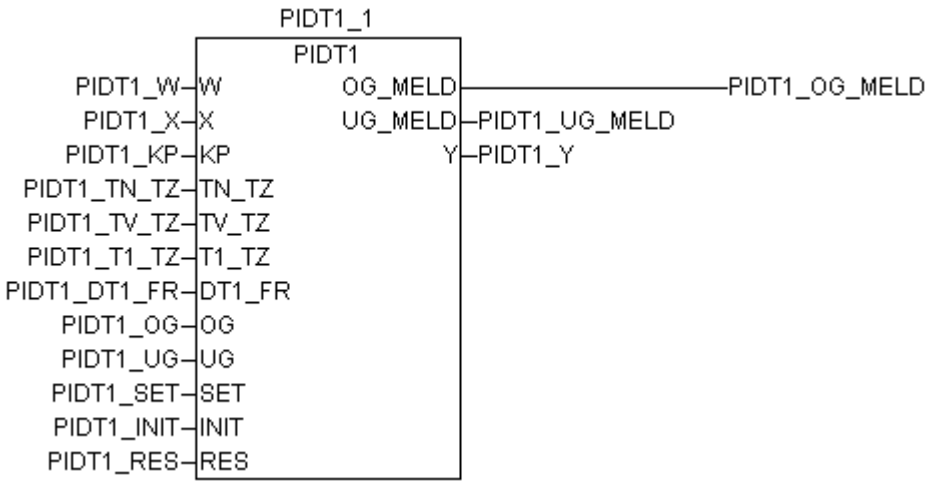


ABB AWL des Beispiels:

```
!BA 0
PI
MW0,0
MW1,1
KW1,0      ; 100
KW1,1      ; 20
KW1,2      ; 0
KW1,3      ; 0
M0,4
M0,0
M0,1
KW1,5      ; 10000
M0,0
M0,1
KW1,6      ; 20000
KW1,7      ; 0
MW0,2
M0,2
M0,3
```

Funktionsaufruf in AWL

```
CAL   PIDT1_1(W := PIDT1_W,
              X := PIDT1_X,
              KP := PIDT1_KP,
              TN_TZ := PIDT1_TNTN,
              TV_TZ := PIDT1_TVTZ,
              T1_TZ := PIDT1_T1_TZ,
              DT1_FR := PIDT1_DT1_FR,
              OG := PIDT1_OG,
              UG := PIDT1_UG,
              SET := PIDT1_SET,
              INIT := PIDT1_INIT,
              RES := PIDT1_RES)
LD    PIDT1_1.OG_MELD
ST    PIDT1_OG_MELD
LD    PIDT1_1.UG_MELD
ST    PIDT1_UG_MELD
LD    PIDT1_1.Y
ST    PIDT1_Y
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

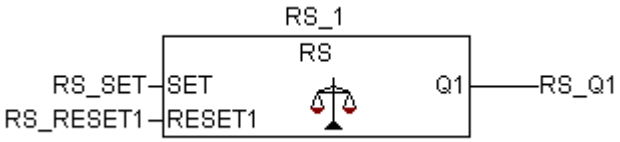
```
PIDT1_1          W := PIDT1_W,
                 X := PIDT1_X,
                 KP := PIDT1_KP,
                 TN_TZ := PIDT1_TNTN,
                 TV_TZ := PIDT1_TVTZ,
                 T1_TZ := PIDT1_T1_TZ,
                 DT1_FR := PIDT1_DT1_FR,
                 OG := PIDT1_OG,
                 UG := PIDT1_UG,
                 SET := PIDT1_SET,
                 INIT := PIDT1_INIT,
                 RES := PIDT1_RES);
PIDT1_OG_MELD    := IDT1_1.OG_MELD;
PIDT1_UG_MELD    := IDT1_1.UG_MELD;
PIDT1_Y          := IDT1_1.Y;
```

Speicher dominierend rücksetzen RS S40

Zustand TRUE am Eingang SET setzt den Operanden Q1 auf Zustand TRUE.

Zustand TRUE am Eingang RESET1 setzt den Operanden Q1 zurück auf Zustand FALSE.

=> dominierend RESET



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	RS	Instanzname
SET	BOOL	Setz-Eingang
RESET1	BOOL	Rücksetz-Eingang
Q1	BOOL	Flip-Flop-Ausgang

Beschreibung

Zustand TRUE am Eingang SET setzt den Operanden Q1 auf Zustand TRUE.

Zustand TRUE am Eingang RESET1 setzt den Operanden Q1 zurück auf Zustand FALSE.

Gleichzeitiger TRUE-Zustand am Eingang SET und RESET1 setzt den Operanden Q1 auf Zustand FALSE (dominierend Rücksetzen).

Zustand FALSE am Eingang SET oder RESET1 hat keinen Einfluß auf den Operanden Q1.

Beispiel

Deklaration:

```
RS_1 : RS;
RS_SET AT %MX0.0 : BOOL;
RS_RESET1 AT %MX0.1 : BOOL;
RS_Q1 AT %MX0.2 : BOOL;
```

Übersetzung in ABB AWL:

```
! SET
=S Q1
! RESET1
=R Q1
```

FBD:

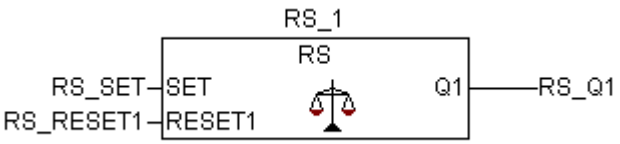


ABB AWL des Beispiels:

```
! M0,0
=S M0,2
! M0,1
=R M0,2
```

Funktionsaufruf in AWL

```
CAL   RS_1(SET = RS_SET,
           RESET1 := RS_RESET)

LD    RS_1.Q1
ST    RS_Q
```

Funktionsaufruf in ST

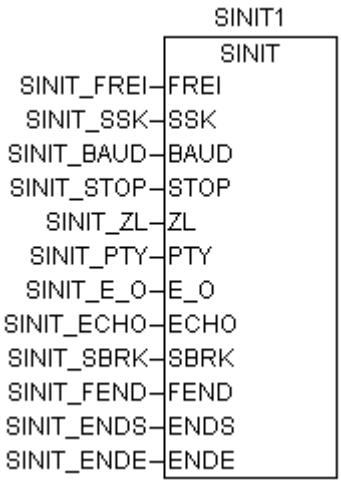
```
RS_1  (SET := RS_SET,
       RESET1 := RS_RESET);

RS_Q  := RS_1.Q1;
```

Initialisierung der seriellen Schnittstellen SINIT S40

Funktionsbaustein zur Initialisierung der seriellen Schnittstellen.

Der Funktionsbaustein SINIT wird bei jeder FALSE/TRUE-Flanke am Eingang FREI einmal bearbeitet. Er initialisiert die am Eingang SSK angegebene serielle Schnittstelle.

Die Schnittstelle kann vom SPS-Programm bedient werden (z.B. mit den Bausteinen DRUCK und EMAS).



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	SINIT	Instanzname
FREI	BOOL	Freigabe der Bausteinbearbeitung, FALSE->TRUE-Flanke
SSK	INT	Schnittstellenkennung (1 oder 2)
BAUD	INT	Baudrate; 300 ... 9600 Baud
STOP	INT	Anzahl Stoppbits; Eingang ist ohne Wirkung
ZL	INT	Zeichenlänge, 7 oder 8 Datenbits pro Zeichen
PTY	BOOL	Parität, Ein/Aus
E_O	BOOL	Parität gerade/ungerade
ECHO	BOOL	Echo, Ein/Aus
SBRK	BOOL	Sende Break-Zeichen
FEND	BOOL	Freigabe Textabschlußzeichen für Senderichtung
ENDS	INT	Textabschlußzeichen für Senderichtung
ENDE	INT	Textabschlußzeichen für Empfangsrichtung

Beschreibung

Funktionsbaustein zur Initialisierung der seriellen Schnittstellen.

Vor Benutzung einer dieser Schnittstellen muß diese initialisiert werden. Dazu steht der Funktionsbaustein SINIT zur Verfügung.

Der Funktionsbaustein SINIT wird bei jeder FALSE/TRUE-Flanke am Eingang FREI einmal bearbeitet.

Er initialisiert die am Eingang SSK angegebene serielle Schnittstelle (COM1, COM2).

FREI **BOOL**

Bei Vorgabe einer FALSE/TRUE-Flanke am Eingang FREI wird der Baustein einmal durchlaufen. Dadurch wird die serielle Schnittstelle, deren Nummer am Eingang SSK angegeben ist, initialisiert und die Schnittstelle ist danach betriebsbereit.

SSK **INT**

Am Eingang SSK wird die Nummer der zu initialisierenden Schnittstelle angegeben.

Es gilt:

COM1: Nummer = 1

COM2: Nummer = 2

BAUD INT

Am Eingang BAUD wird der Wert für die Baudrate angegeben.

Baudrate: 300 ... 9600 Baud

STOP INT

Die Anzahl der Stoppbits ist fest auf 1 eingestellt und nicht änderbar. Der am Eingang STOP angegebene Wert für die Anzahl der Stoppbits hat keine Bedeutung.

ZL INT

Am Eingang ZL wird die gewünschte Zeichenlänge angegeben. Zeichenlänge bedeutet Anzahl Daten-Bits pro Zeichen.

Möglich sind 7 oder 8 Daten-Bits pro Zeichen.

PTY BOOL

Am Eingang PTY wird angegeben, ob ein Zeichen mit oder ohne Paritätsbit übertragen wird.

PTY = 0 -> Übertragung ohne Paritätsbit

PTY = 1 -> Übertragung mit Paritätsbit

E_O BOOL

Am Eingang E/O wird angegeben, ob ein gerades oder ungerades Paritätsbit gewünscht wird.

E/O = 0 -> Paritätsbit ungerade

E/O = 1 -> Paritätsbit gerade

ECHO BOOL

Am Eingang ECHO wird angegeben, ob die über die betreffende Schnittstelle empfangenen Zeichen von der SPS reflektiert (geechot) werden sollen. Damit kann zum Beispiel der Sender eines Zeichens feststellen, ob dieses korrekt in der SPS angekommen ist.

ECHO = 0 -> kein Echo, Zeichen wird nicht reflektiert

ECHO = 1 -> Echo, Zeichen wird reflektiert

SBRK BOOL

Am Eingang SBRK (Send-Break-Character) kann der Zustand der Sendeleitung TxD beeinflußt werden.

SBRK = 0 -> Normalzustand der Sendeleitung TxD zur Übertragung von Zeichen

SBRK = 1 -> Sendeleitung TxD wird auf "0" gesetzt

FEND BOOL

Am Eingang FEND (Freigabe Endezeichen) wird angegeben, ob das am Eingang ENDS projektierte Textabschlußzeichen für die Senderichtung mit ausgegeben wird oder nicht.

FEND = 0 -> Textabschlußzeichen in Senderichtung wird nicht ausgegeben

FEND = 1 -> Textabschlußzeichen in Senderichtung wird ausgegeben

ENDS INT

Am Eingang ENDS kann ein frei wählbares Textabschlußzeichen für die Senderichtung angegeben werden. Dieses Abschlußzeichen wird dann automatisch an jeden Text (Telegramm) angehängt, den der Baustein DRUCK über die serielle Schnittstelle nach außen sendet. Voraussetzung ist allerdings die Freigabe von Eingang FEND.

Die Angabe des Textabschlußzeichens erfolgt als Zahlenwert.

Beispiel:

3 bzw. 03H bedeutet <ETX>

4 bzw. 04H bedeutet <EOT>

13 bzw. 0DH bedeutet <CR>

10 bzw. OAH bedeutet <LF>

32 bzw. 20H bedeutet <SP>

...

ENDE INT

Am Eingang ENDE kann ein frei wählbares Textabschlußzeichen für die Empfangsrichtung angegeben werden. Beim Empfang eines Telegramms über die serielle Schnittstelle erkennt die SPS an diesem Abschlußzeichen das Ende eines Telegramms. Die Angabe des Abschlußzeichens erfolgt in der gleichen Art und Weise wie beim Eingang ENDS.

Beispiel

Deklaration:

```
SINIT1 : SINIT;
SINIT_FREI AT %MX0.0 : BOOL;
SINIT_SSK AT %MW3001.1 : INT := 1;
SINIT_BAUD AT %MW3001.2 : INT := 9600;
SINIT_STOP AT %MW3001.3 : INT := 1;
SINIT_ZL AT %MW3001.4 : INT := 8;
SINIT_PTY AT %MX0.1 : BOOL;
SINIT_E_O AT %MX0.2 : BOOL;
SINIT_ECHO AT %MX0.3 : BOOL;
SINIT_SBRK AT %MX0.4 : BOOL;
SINIT_FEND AT %MX0.5 : BOOL;
SINIT_ENDS AT %MW3001.5 : INT := 13;
SINIT_ENDE AT %MW3001.6 : INT := 13;
```

Übersetzung in ABB AWL:

```
!BA 0
SINIT
FREI
SSK
BAUD
STOP
ZL
PTY
E_O
ECHO
SBRK
FEND
ENDS
ENDE
```

FBD:

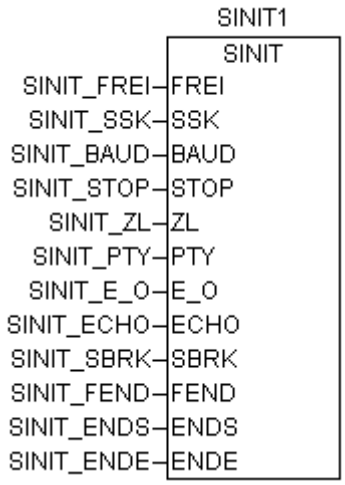


ABB AWL des Beispiels:

```
!BA 0
SINIT
M0,0
KW1,1          ; 1
KW1,2          ; 9600
KW1,3          ; 1
KW1,4          ; 8
M0,1
M0,2
M0,3
M0,4
M0,5
KW1,5          ; 13
KW1,6          ; 13
```

Funktionsaufruf in AWL

```
CAL   SINIT1(FREI := SINIT_FREI,
        SSK := SINIT_SSK,
        BAUD := SINIT_BAUD,
        STOP := SINIT_STOP,
        ZL := SINIT_ZL,
        PTY := SINIT_PTY,
        E_O := SINIT_E_O,
        ECHO := SINIT_ECHO,
        SBRK := SINIT_SBRK,
        FEND := SINIT_FEND,
        ENDS := SINIT_ENDS,
        ENDE := SINIT_ENDE)
```

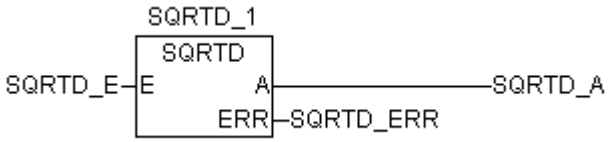
Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

Funktionsaufruf in ST

```
SINIT1      (FREI := SINIT_FREI,
             SSK := SINIT_SSK,
             BAUD := SINIT_BAUD,
             STOP := SINIT_STOP,
             ZL := SINIT_ZL,
             PTY := SINIT_PTY,
             E_O := SINIT_E_O,
             ECHO := SINIT_ECHO,
             SBRK := SINIT_SBRK,
             FEND := SINIT_FEND,
             ENDS := SINIT_ENDS,
             ENDE := SINIT_ENDE);
```

Quadratwurzel, Doppelwort SQRTD S40

Der Funktionsbaustein SQRTD bildet die Quadratwurzel des Wertes am Eingang E. Das Ergebnis steht am Ausgang A zur Verfügung und wird grundsätzlich auf eine ganze Zahl abgerundet. Der Wert am Eingang E muß eine positive Zahl sein. Ist der Wert am Eingang negativ, so wird am Ausgang A der Wert "Null" und am Ausgang ERR der Wert TRUE ausgegeben.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	SQRTD	Instanzname
E	DINT	Eingang
A	DINT	Quadratwurzel des Eingangswertes
ERR	BOOL	Fehler bei negativem Eingangswert, Operanden M, A (nicht E, S)

Beschreibung

Der Funktionsbaustein SQRTD bildet die Quadratwurzel des Wertes am Eingang E. Das Ergebnis steht am Ausgang A zur Verfügung und wird grundsätzlich auf eine ganze Zahl abgerundet. Der Wert am Eingang E muß eine positive Zahl sein. Ist der Wert am Eingang negativ, so wird am Ausgang A der Wert "Null" und am Ausgang ERR der Wert TRUE ausgegeben.

E DINT

Aus dem Wert am Eingang E wird die Quadratwurzel gebildet und steht als Wert des Ausgangsoperanden A zur Verfügung.

A DINT

Am Ausgang A steht der Wert der Quadratwurzel zur Verfügung.

ERR BOOL

Der Ausgang ERR zeigt an, ob der Wert des Eingangsoperanden E positiv (>= 0) oder negativ (< 0) ist.

Eingang E >= 0 -> ERR = FALSE und A = Quadratwurzel

Eingang E < 0 -> ERR = TRUE und A = 0

Beispiel

Deklaration:

```
SQRTD_1 : SQRTD;
SQRTD_E AT %MD2000.0 : DINT;
SQRTD_A AT %MD2000.1 : DINT;
SQRTD_ERR AT %MX0.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
SQRT
K0,1
E
A
ERR
```

FBD:

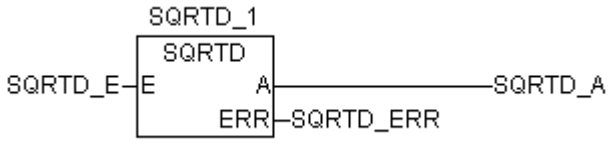


ABB AWL des Beispiels:

```
!BA 0
SQRT
K0,1
MD0,0
MD0,1
M0,0
```

Funktionsaufruf in AWL

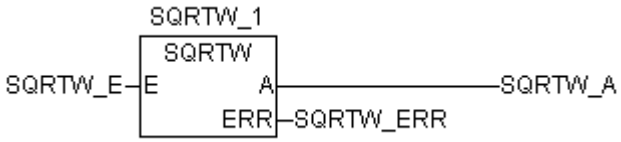
```
CAL   SQRTD_1(E := SQRTD_E)

LD    SQRTD_1.A
ST    SQRTD_A
LD    SQRTD_1.ERR
ST    SQRTD_ERR
```

Funktionsaufruf in ST

```
SQRTD_1(E := SQRTD_E);

SQRTD_A := SQRTD_1.A;
SQRTD_ERR := SQRTD_1.ERR;
```

Quadratwurzel, Wort SQRTW S40

Der Funktionsbaustein SQRTW bildet die Quadratwurzel des Wertes am Eingang E. Das Ergebnis steht am Ausgang A zur Verfügung und wird grundsätzlich auf eine ganze Zahl abgerundet. Der Wert am Eingang E muß eine positive Zahl sein. Ist der Wert am Eingang negativ, so wird am Ausgang A der Wert "Null" und am Ausgang ERR der Wert TRUE ausgegeben.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	SQRTW	Instanzname
E	INT	Eingang
A	INT	Quadratwurzel des Eingangswertes
ERR	BOOL	Fehler bei negativem Eingangswert, Operanden M, A (nicht E, S)

Beschreibung

Der Funktionsbaustein SQRTW bildet die Quadratwurzel des Wertes am Eingang E. Das Ergebnis steht am Ausgang A zur Verfügung und wird grundsätzlich auf eine ganze Zahl abgerundet. Der Wert am Eingang E muß eine positive Zahl sein. Ist der Wert am Eingang negativ, so wird am Ausgang A der Wert "Null" und am Ausgang ERR der Wert TRUE ausgegeben.

E INT

Aus dem Wert am Eingang E wird die Quadratwurzel gebildet und steht als Wert des Ausgangsoperanden A zur Verfügung.

A INT

Am Ausgang A steht der Wert der Quadratwurzel zur Verfügung.

ERR BOOL

Der Ausgang ERR zeigt an, ob der Wert des Eingangsoperanden E positiv (>= 0) oder negativ (< 0) ist.

Eingang E >= 0 -> ERR = FALSE und A = Quadratwurzel

Eingang E < 0 -> ERR = TRUE und A = 0

Beispiel

Deklaration:

```
SQRTW_1 : SQRTD;
SQRTW_E AT %MW1000.0 : DINT;
SQRTW_A AT %MW1000.1 : DINT;
SQRTW_ERR AT %MX0.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
SQRT
K0,0
E
A
ERR
```

FBD:

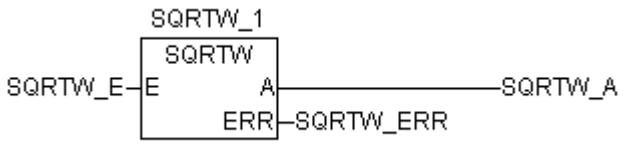


ABB AWL des Beispiels:

```
!BA 0
SQRT
K0,0
MW0,0
MW0,1
M0,0
```

Funktionsaufruf in AWL

```
CAL   SQRTW_1(E := SQRTD_E)

LD    SQRTW_1.A
ST    SQRTW_A
LD    SQRTW_1.ERR
ST    SQRTW_ERR
```

Funktionsaufruf in ST

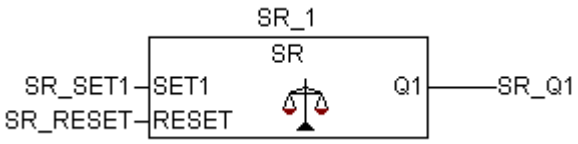
```
SQRTW_1(E := SQRTW_E);

SQRTW_A := SQRTW_1.A;
SQRTW_ERR := SQRTW_1.ERR;
```

Speicher dominierend setzen SR S40

Zustand TRUE am Eingang SET1 setzt den Operanden Q1 auf Zustand TRUE.

Zustand TRUE am Eingang RESET setzt den Operanden Q1 zurück auf Zustand FALSE.

=> dominierend SET



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	SR	Instanzname
SET1	BOOL	Setz-Eingang
RESET	BOOL	Rücksetz-Eingang
Q1	BOOL	Flip-Flop-Ausgang

Beschreibung

Zustand TRUE am Eingang SET1 setzt den Operanden Q1 auf Zustand TRUE.

Zustand TRUE am Eingang RESET setzt den Operanden Q1 zurück auf Zustand FALSE.

Gleichzeitiger TRUE-Zustand am Eingang SET1 und RESET setzt den Operanden Q1 auf Zustand TRUE (dominierend Setzen).

Zustand FALSE am Eingang SET1 oder RESET hat keinen Einfluß auf den Operanden Q1.

Beispiel

Deklaration:

```
SR_1 : SR;
SR_SET1 AT %MX0.0 : BOOL;
SR_RESEET AT %MX0.1 : BOOL;
SR_Q1 AT %MX0.2 : BOOL;
```

Übersetzung in ABB AWL:

```
! RESET
=R Q1
! SET1
=S Q1
```

FBD:

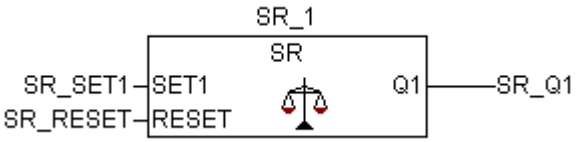


ABB AWL des Beispiels:

```
! M0,1
=R M0,2
! M0,0
=S M0,2
```

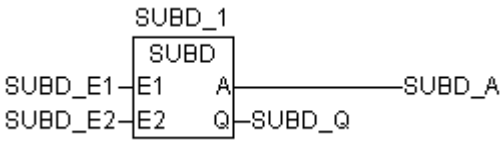
Funktionsaufruf in AWL

```
CAL   SR_1(SET1 := SR_SET,
           RESET = SR_RESET)

LD    SR_1.Q1
ST    SR_Q
```

Funktionsaufruf in ST

```
SR_1  (SET1 := SR_SET,
       RESET := SR_RESET);

SR_Q  := SR_1.Q1;
```

Subtraktion Doppelwort SUBD S40



Der Wert des Operanden am Eingang E2 wird vom Wert des Operanden am Eingang E1 subtrahiert und das Ergebnis dem Operanden am Ausgang A zugewiesen.

Das Ergebnis wird auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt (Zahlenbereich: -2147483647 … 2147483647). Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein TRUE-Signal zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein FALSE-Signal zugewiesen.

Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	SUBD	Instanzname
E1	DINT	Minuend
E2	DINT	Subtrahend
A	DINT	Ergebnis (Differenz)
Q	BOOL	Ergebnis begrenzt

Beschreibung

Der Wert des Operanden am Eingang E2 wird vom Wert des Operanden am Eingang E1 subtrahiert und das Ergebnis dem Operanden am Ausgang A zugewiesen.

Das Ergebnis wird auf den maximalen bzw. minimalen Wert des Zahlenbereichs begrenzt (Zahlenbereich: -2147483647 … 2147483647). Hat eine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein TRUE-Signal zugewiesen. Hat keine Begrenzung stattgefunden, wird dem binären Operanden am Ausgang Q ein FALSE-Signal zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
SUBD_1 : SUBD;
SUBD_E1 AT %MD2000.0 : DINT;
SUBD_E2 AT %MD2000.1 : DINT;
SUBD_A AT %MD2000.2 : DINT;
SUBD_Q AT %MX0.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
SUBD
E1
E2
A
Q
```

FBD:

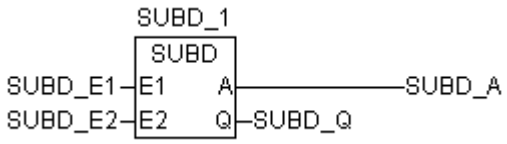


ABB AWL des Beispiels:

```
!BA 0
SUBD
MD0,0
MD0,1
MD0,2
M0,0
```

Funktionsaufruf in AWL

```
CAL   SUBD_1(E1 := SUBD_E1, E2 := SUBD_E2)

LD    SUBD1.A
ST    SUBD_A
LD    SUBD1.Q
ST    SUBD_Q
```

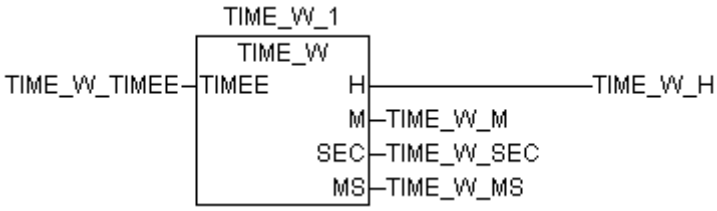
Funktionsaufruf in ST

```
SUBD_1(E1 := SUBD_E1, E2 := SUBD_E2);

SUBD_A:=SUBD1.A;
SUBD_Q:=SUBD1.Q;
```

Zeit nach Wort-Wandlung TIME_W S40

Diese Funktion dient zur Ausgabe von Zeitwerten mit Hilfe von Wort-Variablen.

Die Zeit TIMEE in [ms] wird in Stunden, Minuten, Sekunden und Millisekunden gewandelt.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	TIME_W	Instanzname
TIMEE	DINT	Zeitwert
H	INT	Stundenwert
M	INT	Minutenwert
SEC	INT	Sekundenwert
MS	INT	Millisekundenwert

Beschreibung

Diese Funktion dient zur Ausgabe von Zeitwerten mit Hilfe von Wort-Variablen.

Der Zeitwert TIMEE wird in Stunden, Minuten, Sekunden und Millisekunden gewandelt.

TIMEE **DINT**

Die Eingabe erfolgt in Millisekunden.

Wertebereich:
$0 \leq TIMEE \leq 986399999$

Ist TIMEE größer als 986399999, werden H, M, SEC und MS auf den maximalen Wert gesetzt.

Ist TIMEE negativ, werden H, M, SEC und MS auf den Wert Null gesetzt.

H **INT**

Ausgabe der Stunden.

Wertebereich:
$0 \leq H \leq 273$

M **INT**

Ausgabe der Minuten.

Wertebereich:
$0 \leq M \leq 59$

SEC **INT**

Ausgabe der Sekunden.

Wertebereich:
$0 \leq SEC \leq 59$

MS **INT**

Ausgabe der Millisekunden.

Wertebereich:
$0 \leq MS \leq 999$

Beispiel

Deklaration:

```
TIME_W_1 : TIME_W;
TIME_W_TIMEE AT %MD2000.0 : DINT;
TIME_W_H AT %MW1000.0 : INT;
TIME_W_M AT %MW1000.1 : INT;
TIME_W_SEC AT %MW1000.2 : INT;
TIME_W_MS AT %MW1000.3 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
TIME_W
H
M
SEC
MS
```

FBD:

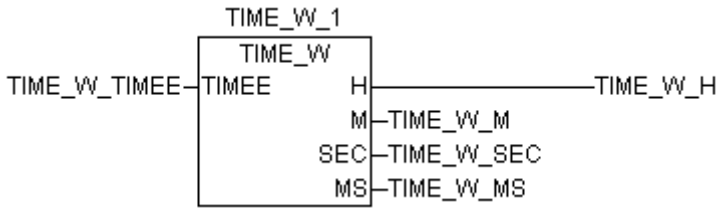


ABB AWL des Beispiels:

```
!BA 0
TIME_W
MD0,0
MW0,0
MW0,1
MW0,2
MW0,3
```

Funktionsaufruf in AWL

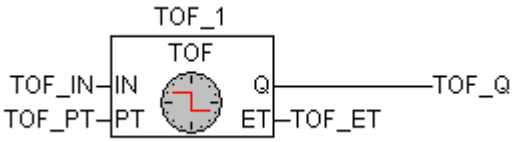
```
CAL   TIME_W_1(TIMEE := TIME_W_TIMEE);

LD    TIME_W_1.H
ST    TIME_W_H
LD    TIME_W_1.M
ST    TIME_W_M
LD    TIME_W_1.SEC
ST    TIME_W_SEC
LD    TIME_W_1.MS
ST    TIME_W_MS
```

Funktionsaufruf in ST

```
TIME_W_1(TIMEE := TIME_W_TIMEE);

TIME_W_H := TIME_W_1.H;
TIME_W_M := TIME_W_1.M;
TIME_W_SEC := TIME_W_1.SEC;
TIME_W_MS := TIME_W_1.MS;
```

Ausschaltverzögerung TOF S40

Die TRUE/FALSE-Flanke des Eingangs E wird um die Zeitdauer T verzögert am Ausgang A ausgegeben. Geht der Eingang E vor Ablauf der Zeit T wieder auf TRUE-Pegel, so bleibt der Ausgang A auf TRUE-Pegel.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	TOF	Instanzname
E	BOOL	Eingangssignal
T	TIME	Verzögerungszeit
A	BOOL	Verzögertes Signal, Operanden M, A (nicht E, S)
ET	TIME	Zeitanzeige

Beschreibung

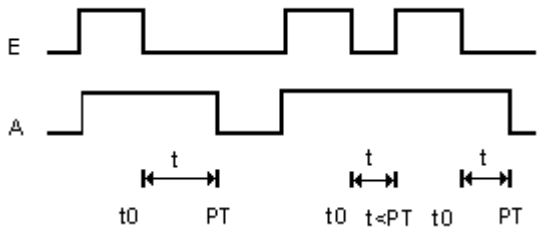
Die TRUE/FALSE-Flanke des Eingangs E wird um die Zeitdauer T verzögert am Ausgang A ausgegeben. Geht der Eingang E vor Ablauf der Zeit T wieder auf TRUE-Pegel, so bleibt der Ausgang A auf TRUE-Pegel.

Die abgelaufene Zeit kann am Ausgang ET abgerufen werden und der Wert der Verzögerungszeit am Eingang T kann verändert werden, während das Zeitwerk läuft. Die Verzögerungszeit wird in Millisekunden angegeben.

Gültiger Zeitbereich: 1 ms ... 24,8 Tage.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Sinnvoller Bereich für T: > 1 Zykluszeit.



Allgemeines Verhalten

• Gestartete Zeitwerke werden vom Betriebssystem der SPS bearbeitet und sind deshalb vollkommen unabhängig von der Bearbeitung des SPS-Programms. Erst nach Ablauf des Zeitwerks erfolgt eine entsprechende Meldung des Betriebssystems an den zugehörigen Zeitbaustein im SPS-Programm.

• Die Bearbeitung eines Zeitwerks im Betriebssystem der SPS wird durch folgende Befehle beeinflußt: Alle laufenden Zeitwerke werden gestoppt und initialisiert, wenn einer der folgenden Fehler auftritt:

- SPS-Programm abbrechen
- RUN/STOP-Schalter von RUN -> STOP
- Warm- oder Kaltstart

Beispiel

Deklaration:

```
TOF_1 : TOF;
TOF_IN AT %MX0.0 : BOOL;
TOF_PT AT %MD40001.0 : TIME := t#10s; (* 10000 ms *)
TOF_Q AT %MX0.1 : BOOL;
TOF_ET AT %MD2000.0 : TIME;
```

Übersetzung in ABB AWL:

```
!BA 0
TOF
IN
PT
Q
ET
```

FBD:

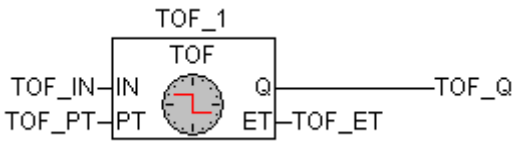


ABB AWL des Beispiels:

```
!BA 0
TOF
M0,0
KD1,0                    ; 10000
M0,1
MD0,0
```

Funktionsaufruf in AWL

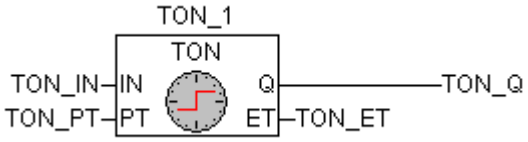
```
CAL   TOF_1(E := TOF_E, T := TOF_T)

LD    TOF_1.A
ST    TOF_A
LD    TOF_1.ET
ST    TOF_ET
```

Funktionsaufruf in ST

```
TOF_1(E := TOF_E, T := TOF_T);

TOF_A := TOF_1.A;
TOF_ET := TOF_1.ET;
```

Einschaltverzögerung TON S40

Die FALSE/TRUE-Flanke des Eingangs E wird um die Zeitdauer T verzögert am Ausgang A ausgegeben. Geht der Eingang E vor Ablauf der Zeit T wieder auf FALSE-Pegel, so bleibt der Ausgang A auf FALSE-Pegel.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	TON	Instanzname
E	BOOL	Eingangssignal
T	TIME	Verzögerungszeit
A	BOOL	Verzögertes Signal
ET	TIME	Zeitanzeige

Beschreibung

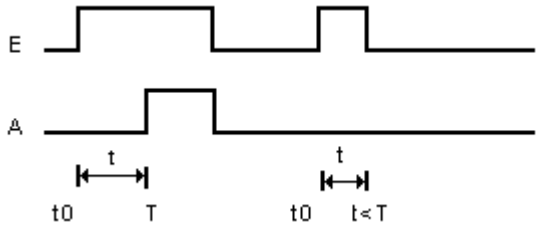
Die FALSE/TRUE-Flanke des Eingangs E wird um die Zeitdauer T verzögert am Ausgang A ausgegeben. Geht der Eingang E vor Ablauf der Zeit T wieder auf 0-Pegel, so bleibt der Ausgang A auf 0-Pegel.

Die abgelaufene Zeit kann am Ausgang ET abgerufen werden und der Wert der Verzögerungszeit am Eingang T kann verändert werden, während das Zeitwerk läuft. Die Verzögerungszeit wird in Millisekunden angegeben.

Gültiger Zeitbereich: 1 ms ... 24,8 Tage.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Sinnvoller Bereich für T: > 1 Zykluszeit



Allgemeines Verhalten

• Gestartete Zeitwerke werden vom Betriebssystem der SPS bearbeitet und sind deshalb vollkommen unabhängig von der Bearbeitung des SPS-Programms. Erst nach Ablauf des Zeitwerks erfolgt eine entsprechende Meldung des Betriebssystems an den zugehörigen Zeitbaustein im SPS-Programm.

• Die Bearbeitung eines Zeitwerks im Betriebssystem der SPS wird durch folgende Befehle beeinflußt: Alle laufenden Zeitwerke werden gestoppt und initialisiert, wenn einer der folgenden Fehler auftritt:

- SPS-Programm abbrechen
- RUN/STOP-Schalter von RUN -> STOP
- Warm- oder Kaltstart

Beispiel

Deklaration:

```
TON_1 : TON;
TON_IN AT %MX0.0 : BOOL;
TON_PT AT %MD40001.0 : TIME := t#10s; (* 10000 ms *)
TON_Q AT %MX0.1 : BOOL;
TON_ET AT %MD2000.0 : TIME;
```

Übersetzung in ABB AWL:

```
!BA 0
TON
IN
PT
Q
ET
```

FBD:

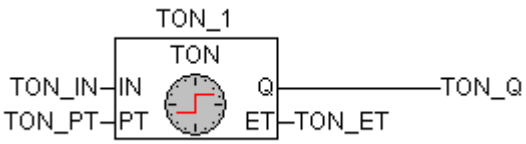


ABB AWL des Beispiels:

```
!BA 0
TON
M0,0
KD1,0                    ; 10000
M0,1
MD0,0
```

Funktionsaufruf in AWL

```
CAL   TON_1(E := TON_E, T := TON_T)

LD    TON_1.A
ST    TON_A
LD    TON_1.ET
ST    TON_ET
```

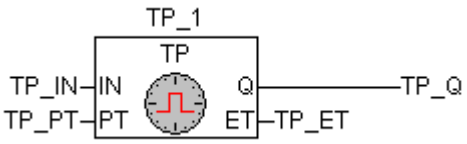
Funktionsaufruf in ST

```
TON_1(E := TON_E, T := TON_T);

TON_A := TON_1.A;
TON_ET := TON_1.ET;
```

Monostabiles Kippglied «konstant» TP S40

Die FALSE/TRUE-Flanke am Eingang E bewirkt eine FALSE/TRUE-Flanke am Ausgang A. Nach Ablauf der Zeitdauer T wird der Ausgang A wieder auf FALSE-Pegel zurückgesetzt.

Eine zweite FALSE/TRUE-Flanke des Eingangs E vor Ablauf der Zeit T wird ignoriert.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	ASV	Instanzname
E	BOOL	Eingangssignal
T	TIMER	Verzögerungszeit
A	BOOL	Verzögertes Signal
ET	DINT	Zeitanzeige

Beschreibung

Die FALSE/TRUE-Flanke am Eingang E bewirkt eine FALSE/TRUE-Flanke am Ausgang A. Nach Ablauf der Zeitdauer T wird der Ausgang A wieder auf FALSE-Pegel zurückgesetzt.

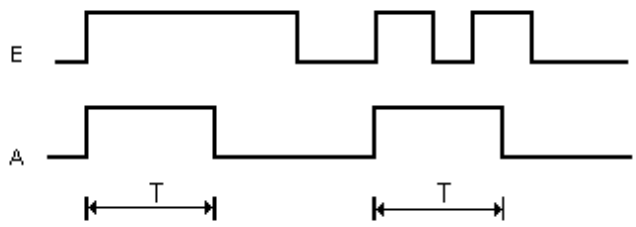
Eine zweite FALSE/TRUE-Flanke des Eingangs E vor Ablauf der Zeit T wird ignoriert.

Die abgelaufene Zeit kann am Ausgang ET abgerufen werden und der Wert der Verzögerungszeit am Eingang E kann verändert werden, während das Zeitwerk läuft.

Die Verzögerungszeit wird in Millisekunden angegeben. Gültiger Zeitbereich: 1 ms ... 24,8 Tage.

Maximaler Zeitversatz am Ausgang: < 1 Zykluszeit

Sinnvoller Bereich für PT: > 1 Zykluszeit.



Allgemeines Verhalten

• Gestartete Zeitwerke werden vom Betriebssystem der SPS bearbeitet und sind deshalb vollkommen unabhängig von der Bearbeitung des SPS-Programms. Erst nach Ablauf des Zeitwerks erfolgt eine entsprechende Meldung des Betriebssystems an den zugehörigen Zeitbaustein im SPS-Programm.

• Die Bearbeitung eines Zeitwerks im Betriebssystem der SPS wird durch folgende Befehle beeinflußt: Alle laufenden Zeitwerke werden gestoppt und initialisiert, wenn einer der folgenden Fehler auftritt:

- SPS-Programm abbrechen
- RUN/STOP-Schalter von RUN -> STOP
- Warm- oder Kaltstart

Beispiel

Deklaration:

```
TP_1 : TOF;
TP_IN AT %MX0.0 : BOOL;
TP_PT AT %MD40001.0 : TIME := t#10s; (* 10000 ms *)
TP_Q AT %MX0.1 : BOOL;
TP_ET AT %MD2000.0 : TIME;
```

Übersetzung in ABB AWL:

```
!BA 0
TP
IN
PT
Q
ET
```

FBD:

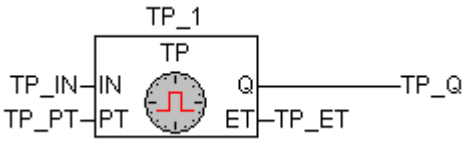


ABB AWL des Beispiels:

```
!BA 0
TP
M0,0
KD1,0                    ; 10000
M0,1
MD0,0
```

Funktionsaufruf in AWL

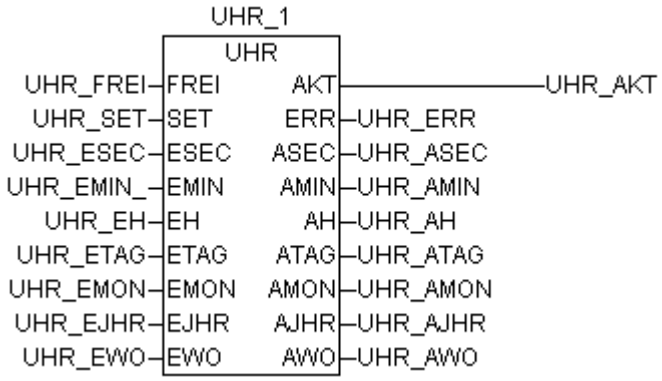
```
CAL   TP_1(E := TP_E, T := TP_T)

LD    TP_1.A
ST    TP_A
LD    TP_1.ET
ST    TP_ET
```

Funktionsaufruf in ST

```
TP_1(E := TP_E, T := TP_T);

TP_A := TP_1.A;
TP_ET := TP_1.ET;
```

Uhr anzeigen und stellen UHR S40

Der Baustein ermöglicht das Stellen und Anzeigen der aktuellen Uhrzeit und des aktuellen Datums.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	UHR	Instanzname
FREI	BOOL	Freigabe für die Bearbeitung des Bausteins
SET	BOOL	FALSE/TRUE-Flanke bewirkt das Stellen von Uhrzeit und Datum
ESEC	INT	Stelleingang für die Sekunden
EMIN	INT	Stelleingang für die Minuten
EH	INT	Stelleingang für die Stunden
ETAG	INT	Stelleingang für die Tage
EMON	INT	Stelleingang für die Monate
EJHR	INT	Stelleingang für die Jahre
EWO	INT	Stelleingang für die Wochentage
AKT	BOOL	Aktualität (Brauchbarkeit) der Daten an den Ausgängen
ERR	INT	Fehlerkennung
ASEC	INT	Ausgang Sekunden
AMIN	INT	Ausgang Minuten
AH	INT	Ausgang Stunden
ATAG	INT	Ausgang Tage
AMON	INT	Ausgang Monate
AJHR	INT	Ausgang Jahre
AWO	INT	Ausgang Nr. des Wochentages

Beschreibung

Der Baustein ermöglicht das Stellen und Anzeigen der aktuellen Uhrzeit und des aktuellen Datums.

Die Eingänge und Ausgänge sind weder doppelbar noch invertierbar noch negierbar.

Das Stellen der Uhr erfolgt über die Stelleingänge für die Uhrzeit und das Datum. Die an den Stelleingängen anliegenden Werte werden durch eine FALSE/TRUE-Flanke am Eingang SET übernommen. Solange am Eingang FREI ein TRUE-Signal anliegt, werden an den Bausteinausgängen die aktuelle Uhrzeit und das Datum angezeigt.

FREI **BOOL**

Freigabe des Bausteins

FREI = FALSE:
Der Baustein wird nicht bearbeitet. Die Ausgänge AKT und ERR werden auf FALSE bzw. 0 gesetzt. Die Zeit- und Datumsausgänge werden vom Baustein nicht mehr verändert.

FREI = TRUE: Baustein wird bearbeitet

SET **BOOL**

FALSE/TRUE-Flanke:
Die Uhr wird auf die an den Zeit- und Datumseingängen anstehenden Werte gestellt.

Während des Stellvorgangs sind die Uhrzeit und das Datum an den Bausteinausgängen ungültig (Ausgang AKT = FALSE).

Stelleingänge für Datum und Uhrzeit

Bei einer FALSE/TRUE-Flanke am Eingang SET wird die Uhr auf die an den Stelleingängen anliegenden Werte gesetzt. Sind die angegebenen Stellwerte unzulässig, wird der Ausgang AKT auf FALSE gesetzt und am Ausgang ERR erfolgt eine Fehlermeldung. Die an den Zeit- und Datumsausgängen anliegenden Werte sind in diesem Fall ungültig. Die Uhr muß noch einmal gestellt werden.

ESEC **INT**

Stelleingang für die Sekunden.
Wertebereich: 0 … 59.

EMIN **INT**

Stelleingang für die Minuten.
Wertebereich: 0 … 59.

EH **INT**

Stelleingang für die Stunden.

Die Uhr arbeitet im 24-h-Mode, d. h. sie springt von 23:59:59 Uhr auf 0:0:0 Uhr.

Wertebereich: 0 … 23.

ETAG **INT**

Stelleingang für die Tage (wievielter Tag des Monats).

Die Uhr kennt die Anzahl der Tage in Abhängigkeit von den Monaten und Schaltjahren. Ein Schaltjahr besteht für die Uhr dann, wenn die Jahreszahl ein ganzzahliges Vielfaches von 4 ist. Der maximale Wert für die Tage (28, 29, 30, 31) ist abhängig vom Monat.

Wertebereich: 1…28, 29, 30, 31.

EMON **INT**

Stelleingang für die Monate.
Wertebereich: 1 … 12.

EJHR **INT**

Stelleingang für die Jahre.
Die Uhr zeigt nur die Jahre und Jahrzehnte an.
Wertebereich: 0 … 99.

EWO **INT**

Stelleingang für die Nummer des Wochentags.

An diesem Eingang wird angegeben, der wievielte Wochentag der Tag ist, an dem die Eingabe erfolgt. D. h. man kann festlegen, welcher Wochentag (z. B. Sonntag oder Montag) der Tag mit der Nummer 1 sein soll.

Wertebereich: 1 … 7.

Beispiel:
Die Uhr wird am Freitag, den 01.07.88 gestellt. Wird für die Wochentagsnummer der Wert 6 eingegeben, so ist nun der Freitag als 6. Tag der Woche und damit auch der Sonntag als 1. Tag der Woche definiert.

AKT **BOOL**

Anzeige der Aktualität (Brauchbarkeit) der Ausgänge.

AKT ist TRUE, wenn:

- die Datums- und Zeitausgänge im aktuellen Zyklus aktualisiert wurden.
- die Werte an den Ausgängen konsistent sind, d. h. keiner der Werte an den Datums- bzw. Zeitausgängen hat sich während des Aktualisierungsvorgangs geändert. Sie stammen alle vom selben Uhrentakt ab.
- die Uhr richtig gestellt wurde.

AKT = TRUE → ERR = 0:
Datum/Uhrzeit sind gültig

AKT = FALSE → ERR > 0:
Datum/Uhrzeit sind ungültig.
Der Grund wird als Fehlerkennung am Ausgang ERR angezeigt.

ERR **INT**

Am Ausgang ERR steht im Fehlerfall die relevante Fehlerkennung zur Verfügung.

Bedeutung der Fehlerkennungen:

- **Kein Fehler aufgetreten:**

ERR=0: Kein Fehler aufgetreten oder FREI = FALSE d. h. Baustein gesperrt.

- **Fehler beim Stellen der Uhr:**

ERR=1: 0 < SEC < 59 ist nicht eingehalten
ERR=2: 0 < MIN < 59 ist nicht eingehalten
ERR=3: 0 < H < 23 ist nicht eingehalten
ERR=4: 1 < TAG < 28, 29, 30, 31 (je nach Monat) ist nicht eingehalten
ERR=5: 1 < MON < 12 ist nicht eingehalten
ERR=6: 0 < JHR < 99 ist nicht eingehalten
ERR=7: 1 < WTG < 7 ist nicht eingehalten
ERR=8: Die Sendemailbox ist momentan durch einen anderen Auftraggeber belegt. Der Baustein wartet bis die Mailbox frei wird und führt dann den Stellvorgang für Datum/Uhrzeit aus.
ERR=9: Datum/Uhrzeit an den Ausgängen sind ungültig.
ERR=10: Der Stellvorgang von Datum/Uhrzeit läuft gerade; er kann mehrere SPS-Zyklen dauern.
ERR=11: Stellvorgang war nicht erfolgreich, bitte wiederholen (Requestcode unbekannt).
ERR=12: Stellvorgang war nicht erfolgreich, bitte wiederholen (Mail-Parameter ungültig)
ERR=13: Stellvorgang war nicht erfolgreich, bitte wiederholen (Requestcode nicht ausführbar).

- **Fehler beim Anzeigen des Datums und der Uhrzeit:**

ERR=9: Datum/Uhrzeit an den Ausgängen sind ungültig. **Ausgänge für Datum und Uhrzeit**

Die Aktualisierung der Ausgänge erfolgt immer dann, wenn am Eingang FREI ein TRUE-Signal anliegt und die Uhr einmal gestellt worden ist. Während des Stellvorgangs sind die Ausgänge für das Datum und die Uhrzeit ungültig.

Wenn der Ausgang AKT gleich TRUE ist, sind die Ausgänge für Datum und Uhrzeit gültig. Im Fehlerfall wird am Ausgang ERR eine Fehlerkennung ausgegeben.

ASEC **INT**

Ausgang Sekunden.
Wertebereich: 0 … 59.

AMIN **INT**

Ausgang Minuten.
Wertebereich: 0 … 59.

AH **INT**

Ausgang Stunden.
Wertebereich: 0 … 23.

ATAG **INT**

Ausgang Tage.
Wertebereich: 1 … 28, 29, 30, 31.

AMON **INT**

Ausgang Monate.
Wertebereich: 1 … 12.

AJHR **INT**

Ausgang Jahre.
Wertebereich: 0 … 99.

AWO **INT**

Ausgang Nr. des Wochentages
Wertebereich: 1 … 7.

Beispiel

Deklaration:

```
UHR_1 : UHR;
UHR_FREI AT %MX0.0 : BOOL;
UHR_SET AT %MX0.1 : BOOL;
UHR_ESEC AT %MW1000.0 : INT;
UHR_EMIN AT %MW1000.1 : INT;
UHR_EH AT %MW1000.2 : INT;
UHR_ETAG AT %MW1000.3 : INT;
UHR_EMON AT %MW1000.4 : INT;
UHR_EJHR AT %MW1000.5 : INT;
UHR_EWO AT %MW1000.6 : INT;
UHR_AKT AT %MX0.2 : BOOL;
UHR_ERR AT MW1000.7 : INT;
UHR_ASEC AT MW1000.8 : INT;
UHR_AMIN AT MW1000.9 : INT;
UHR_AH AT MW1000.10 : INT;
UHR_ATAG AT MW1000.11 : INT;
UHR_AMON AT MW1000.12 : INT;
UHR_AJHR AT MW1000.13 : INT;
UHR_AWO AT MW1000.14 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
UHR
SET
ESEC
EMIN
EH
ETAG
EMON
EJHR
EWO
AKT
ERR
ASEC
AMIN
AH
ATAG
AMON
AJHR
AWO
```

FBD:

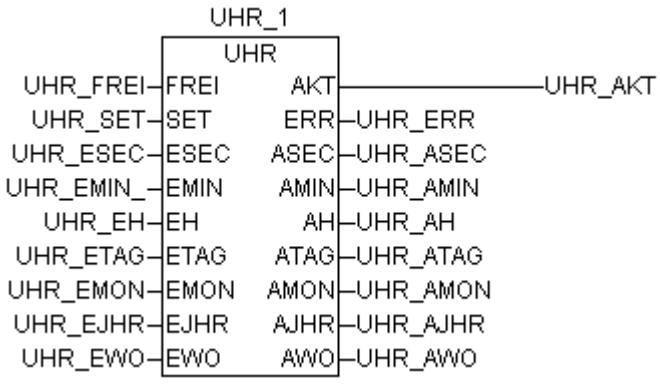


ABB AWL des Beispiels:

```
!BA 0
UHR
M0,0
M0,1
MW1,0
MW1,1
MW1,2
MW1,3
MW1,4
MW1,5
MW1,6
M0,2
MW1,7
MW1,8
MW1,9
MW1,10
MW1,11
MW1,12
MW1,13
MW1,14
```

Funktionsaufruf in AWL

```
CAL   UHR_1( FREI := UHR_FREI,
             SET := UHR_SET,
             ESEC := UHR_ESEC,
             EMIN := UHR_EMIN,
             EH := UHR_EH,
             ETAG := UHR_ETAG,
             EMON := UHR_EMON,
             EJHR := UHR_EJHR,
             EWO := UHR_EWO)

LD    UHR_1.ERR
ST    UHR_ERR
LD    UHR_1.ASEC
ST    UHR_ASEC
LD    UHR_1.AMIN
ST    UHR_AMIN
LD    UHR_1.AH
ST    UHR_AH
LD    UHR_1.ATAG
ST    UHR_ATAG
LD    UHR_1.AMON
ST    UHR_AMON
LD    UHR_1.AJHR
ST    UHR_AJHR
LD    UHR_1.AWO
ST    UHR_AWO
LD    UHR_1.AKT
ST    UHR_AKT
```

Hinweis: Der Funktionsaufruf in AWL muß einzeilig erfolgen.

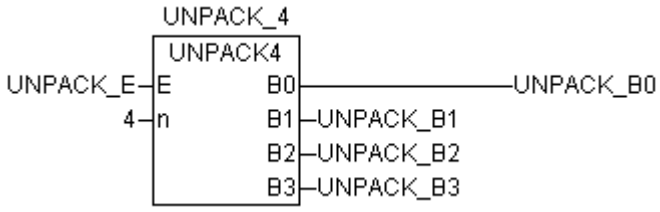
Funktionsaufruf in ST

```
UHR_1      FREI := UHR_FREI,
           SET := UHR_SET,
           ESEC := UHR_ESEC,
           EMIN := UHR_EMIN,
           EH := UHR_EH,
           ETAG := UHR_ETAG,
           EMON := UHR_EMON,
           EJHR := UHR_EJHR,
           EWO := UHR_EWO);

UHR_ERR    := UHR_1.ERR;
UHR_ASEC   := UHR_1.ASEC;
UHR_AMIN   := UHR_1.AMIN;
UHR_AH     := UHR_1.AH;
UHR_ATAG   := UHR_1.ATAG;
UHR_AMON   := UHR_1.AMON;
UHR_AJHR   := UHR_1.AJHR;
UHR_AWO    := UHR_1.AWO;
UHR_AKT    := UHR_1.AKT;
```

Entpacken eines Wortes in binäre Variablen UNPACK(4,..) S40

Der Baustein entpackt die Wortvariable am Eingang E. Jedes Bit der Eingangsvariablen wird jeweils einer Binärvariablen (B0 … B15) am Ausgang zugewiesen.

UNPACK4 – max. 4 Ausgangsvariablen
UNPACK8 – max. 8 Ausgangsvariablen
UNPACK16 – max. 16 Ausgangsvariablen



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	UNPACK	Instanzname
E	INT	zu entpackende Wortvariable
N	INT	Anzahl der Bits
B0..B15	BOOL	binäre Ausgangsvariable

Beschreibung

Der Baustein entpackt die Wortvariable am Eingang E. Jedes Bit der Eingangsvariablen wird jeweils einer Binärvariablen (B0 … Bn-1) am Ausgang zugewiesen.

E INT

Am Eingang E wird die zu entpackende Variable angegeben. Jedes Bit (Bit0 … Bit15) dieser Eingangsvariablen wird der zugeordneten Ausgangsvariablen (B0 … Bn-1) zugewiesen.

B0…B15 BOOL

Den binären Ausgängen B0 … B15 werden die zugeordneten Bits der Variablen am Eingang E zugewiesen.

Zuordnung:

Eingangsvariable Bit0 → B0
Eingangsvariable Bit1 → B1
. . .
. . .
Eingangsvariable Bit15 → B15

Beispiel

Deklaration:

```
UNPACK_4: UNPACK4;
UNPACK_E AT %MW1001.0: INT;
UNPACK_B0 AT %MX1.0: BOOL;
UNPACK_B1 AT %MX1.1: BOOL;
UNPACK_B2 AT %MX1.2: BOOL;
UNPACK_B3 AT %MX1.3: BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
UNPACK
E
#n
B0
B1
B2
B3
```

FBD:

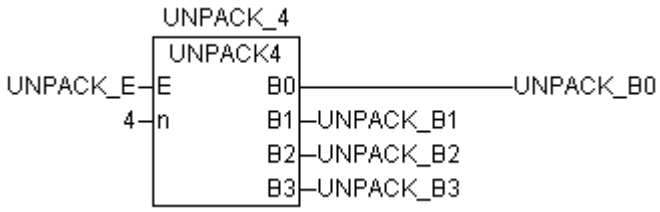


ABB AWL des Beispiels:

```
!BA 0
UNPACK
M0,0
#4
M1,0
M1,1
M1,2
M1,3
```

Funktionsaufruf in AWL

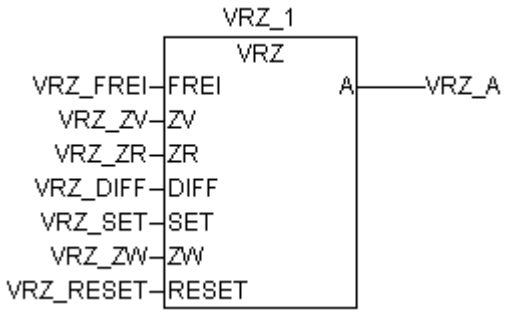
```
CAL   UNPACK_4(E := UNPACK_E,
      n := 4)

LD    UNPACK_4.B0
ST    UNPACK_B0
LD    UNPACK_4.B1
ST    UNPACK_B1
LD    UNPACK_4.B2
ST    UNPACK_B2
LD    UNPACK_4.B3
ST    UNPACK_B3
```

Funktionsaufruf in ST

```
UNPACK_4   (E := UNPACK_E,
           n := 4);

UNPACK_B0  :=UNPACK_4.B0
UNPACK_B1  :=UNPACK_4.B1
UNPACK_B2  :=UNPACK_4.B2
UNPACK_B3  :=UNPACK_4.B3;
```

Vorwärts-, Rückwärts-Zähler VRZ S40

Der Baustein dient zum Zählen von Impulsen. Beim Zählvorgang wird jeweils die positive Flanke des Impulses ausgewertet. Der Zähler kann sowohl vorwärts als auch rückwärts zählen, wobei die Schrittweite pro Zählvorgang vorgegeben werden kann. Die Voreinstellung des Zählerstandes auf einen Zwischenwert ist möglich.



Bausteintyp

Funktionsblock mit Vergangenheitswerten

Parameter

Instanz	VRZ	Instanzname
FREI	BOOL	Freigabe der Bausteinbearbeitung
ZV	BOOL	Impulseingang vorwärtszählen
ZR	BOOL	Impulseingang rückwärtszählen
DIFF	INT	Änderung Zählerstand pro positiver Flanke (Schrittweite)
SET	BOOL	Setzen des Zählers auf einen Zwischenwert
ZW	INT	Zwischenwert
RESET	BOOL	Rücksetzen des Zählers
A	INT	Ausgang für Zählerstand

Beschreibung

Der Baustein dient zum Zählen von Impulsen. Beim Zählvorgang wird jeweils die positive Flanke des Impulses ausgewertet. Der Zähler kann sowohl vorwärts als auch rückwärts zählen, wobei die Schrittweite pro Zählvorgang vorgegeben werden kann. Die Voreinstellung des Zählerstandes auf einen Zwischenwert ist möglich.

FREI BOOL

Mit dem Eingang FREI wird der Zählvorgang freigegeben oder gesperrt.

Es gilt:

FREI = FALSE → Zählvorgang gesperrt
FREI = TRUE → Zählvorgang freigegeben

ZV BOOL

Jede positive Flanke (FALSE → TRUE-Flanke) am Eingang ZV erhöht den momentanen Zählerstand um die am Eingang DIFF angegebene Schrittweite.

ZR BOOL

Jede positive Flanke (FALSE → TRUE-Flanke) am Eingang ZR erniedrigt den momentanen Zählerstand um die am Eingang DIFF angegebene Schrittweite.

DIFF INT

Am Eingang DIFF wird die Schrittweite für den Zählervorgang angegeben. Die Schrittweite ist der Wert, um den der Zähler bei jeder positiven Flanke am Eingang ZV bzw. ZR verändert wird.

SET BOOL

Mit einem TRUE-Signal am Eingang SET wird der Zählerstand auf den am Eingang ZW angegebenen Wert gesetzt. Solange am Eingang SET ein TRUE-Signal anliegt, ist der Zählvorgang blockiert. Der Setzvorgang ist auch wirksam, wenn am Eingang FREI ein TRUE-Signal anliegt.

ZW INT

Am Eingang ZW wird der Wert angegeben, auf den der Zählerstand durch ein TRUE-Signal am Eingang SET gesetzt wird.

RESET **BOOL**

Ein TRUE-Signal am Eingang RESET setzt den Zählerstand auf den Wert 0. Der Reset-Eingang RES hat die höchste Priorität aller Eingänge.

A **INT**

Am Ausgang A steht der aktuelle Zählerstand zur Verfügung.

Erreicht der Zählerstand die positive bzw. negative Grenze des Zahlenbereichs, so wird der Zählerstand auf diesen Wert begrenzt.

Die Eingänge und der Ausgang sind weder doppelbar noch negierbar/invertierbar.

Beispiel

Deklaration:

```
VRZ_1 : VRZ;
VRZ_FREI AT %MX0.0 : BOOL;
VRZ_ZV AT %MX0.1 : BOOL;
VRZ_ZR AT %MX0.2 : BOOL;
VRZ_DIFF AT %MW1000.0 : INT;
VRZ_SET AT %MX0.3 : BOOL;
VRZ_ZW AT %MW1000.1 : INT;
VRZ_RESET AT %MX0.4 : BOOL;
VRZ_A AT %MW1000.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
VRZ
RESET
DIFF
SET
ZW
FREI
ZV
ZR
A
```

FBD:

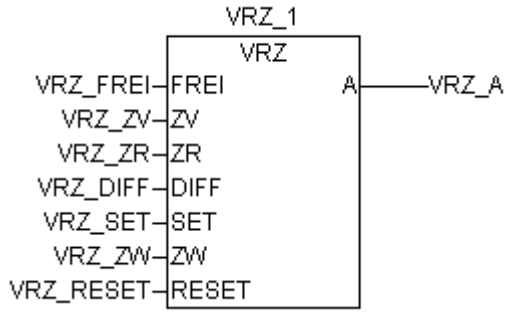


ABB AWL des Beispiels:

```
!BA 0
VRZ
M0,4
MW0,0
M0,3
MW0,1
M0,0
M0,1
MW0,2
```

Funktionsaufruf in AWL

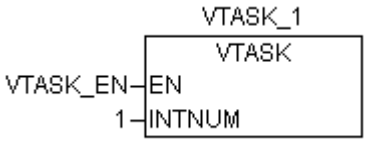
```
CAL   VRZ_1(FREI := VRZ_FREI,
        ZV := VRZ_ZV,
        ZR := VRZ_ZR,
        DIFF := VRZ_DIFF,
        SET := VRZ_SET,
        ZW := VRZ_ZW,
        RESET := VRZ_RESET)

LD    VRZ_1.A
ST    VRZ_A
```

Funktionsaufruf in ST

```
VRZ_1(FREI := VRZ_FREI, ZV := VRZ_ZV,
        ZR := VRZ_ZR, DIFF := VRZ_DIFF,
        SET := VRZ_SET, ZW := VRZ_ZW,
        RESET := VRZ_RESET);

VRZ_A:=VRZ_1.A;
```

Bestätigung der Task-Unterbrechung VTASK S40

Freigeben oder Sperren (Unterbrechen) der Taskroutine.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	VTASK	Instanzname
Ene	BOOL	Bestätigung der Unterbrechung
INTNUM	INT	Nummer der Unterbrechung (Interrupt-Nummer)

Beschreibung

Die Funktion VTASK bestätigt die Ausführung einer Unterbrechung, deren Name in der Variable NAME angegeben ist.

Im Funktionsplan hat der Baustein den Namen der Unterbrechung.

In der Anweisungsliste wird die Funktion VTASK mit den Parametern NAME und EN verwendet.

Beispiel

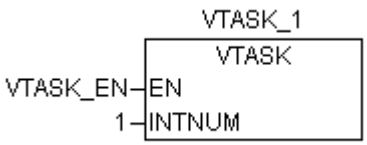
Deklaration:

```
VTASK_1 : VTASK;
VTASK_EN AT %MX0.0 : BOOL;
```

Übersetzung in ABB AWL:

```
!BA 0
VTASK
EN
#INTNUM
```

FBD:



Hinweis:
Die Task für INT 1 muß den Namen "INT1" haben,
für INT 2 "INT2" und für die Zyklustask 10 "INT10_Zykluszeit [ms]"

ABB AWL des Beispiels:

```
!BA 0
VTASK
M0,0
#1
...
!PE             (* Programmende *)
!BA 0
TASK
#1
#1
...             (* Code der Interrupttask *)
RET
NOP
```

Funktionsaufruf in AWL

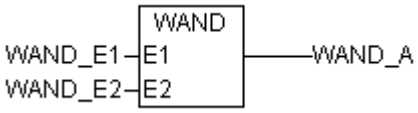
```
CAL   VTASK_1(enable := VTASK_enable,
      N := VTASK_N )
```

Funktionsaufruf in ST

```
VTASK_1   (enable := VTASK_enable,
          N := VTASK_N );
```

UND-Verknüpfung, Wort WAND S40

Der Funktionsbaustein bildet bitweise die UND-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.



Bausteintyp

Funktion

Parameter

E1	INT	Operand 1
E2	INT	Operand 2
A	INT	Ergebnis der UND-Verknüpfung

Beschreibung

Der Funktionsbaustein bildet bitweise die UND-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
WAND_E1 AT %MW1000.0 : INT;
WAND_E2 AT %MW1000.1 : INT;
WAND_A AT %MW1000.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
WAND
E1
E2
A
```

FBD:

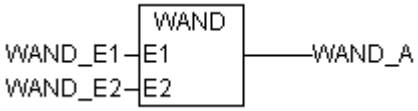


ABB AWL des Beispiels:

```
!BA 0
WAND
MW0,0
MW0,1
MW0,2
```

Funktionsaufruf in AWL

```
LD    WAND_E1
WAND  WAND_E2
ST    WAND_A
```

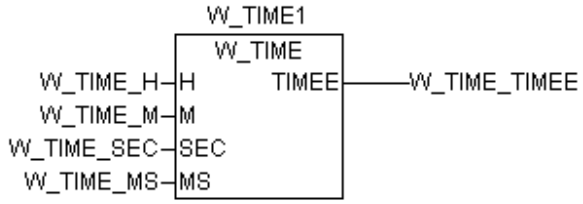
Funktionsaufruf in ST

```
WAND_A    := WAND(E1 := WAND_E1,
             E2 := WAND_E2);
```

Wort nach Zeit-Wandlung W_TIME S40

Diese Funktion dient zur Einstellung eines Zeitwertes mit Hilfe von Worten.

Die Wort-Werte werden zur Nutzung von Zeitgeber-Funktionen in ein Doppelwort gewandelt.



Bausteintyp

Funktionsblock ohne Vergangenheitswerte

Parameter

Instanz	W_TIME	Instanzname
H	INT	Stundenwert
M	INT	Minutenwert
SEC	INT	Sekundenwert
MS	INT	Millisekundenwert
TIMEE	DINT	Zeitwert

Beschreibung

Diese Funktion dient zur Einstellung eines Zeitwertes mit Hilfe von Worten.

Die Wort-Werte werden zur Nutzung von Zeitgeber-Funktionen in ein Doppelwort gewandelt.

Die Eingabe erfolgt in Millisekunden.

H **INT**

Eingabe der Stunden.

Wertebereich:
$0 \leq H \leq 273$

M **INT**

Eingabe der Minuten.

Wertebereich:
$0 \leq M \leq 32767$

SEC **INT**

Eingabe der Sekunden.

Wertebereich:
$0 \leq SEC \leq 32767$

MS **INT**

Eingabe der Millisekunden.

Wertebereich:
$0 \leq MS \leq 32767$

TIMEE **DINT**

Die Ausgabe des gewandelten Zeitwertes erfolgt in Millisekunden.

Wertebereich:
$0 \leq TIMEE \leq 986399999$

Wird für einen Parameter ein negativer Wert eingestellt, wird zur internen Berechnung der Wert 0 verwendet.

Beispiel

Deklaration:

```
W_TIME_H AT %MW1000.0 : INT;
W_TIME_M AT %MW1000.1 : INT;
W_TIME_SEC AT %MW1000.2 : INT;
W_TIME_MS AT %MW1000.3 : INT;
W_TIME_TIMEE AT %MD2000.0 : DINT;
```

Übersetzung in ABB AWL:

```
!BA 0
W_TIME
H
M
SEC
MS
TIMEE
```

FBD:

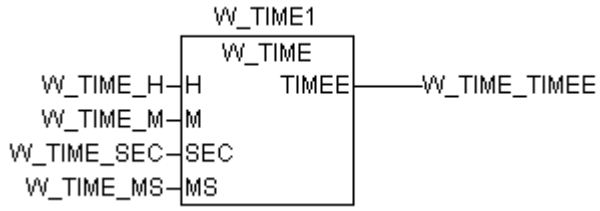


ABB AWL des Beispiels:

```
!BA 0
W_TIME
MW0,0
MW0,1
MW0,2
MW0,3
MD0,0
```

Funktionsaufruf in AWL

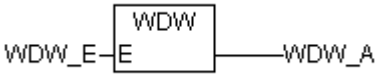
```
CAL   W_TIME1( H := W_TIME_H,
               M := W_TIME_M,
               SEC := W_TIME_SEC,
               MS := W_TIME_MS)
LD    W_TIME1.TIMEE
ST    W_TIME_TIMEE
```

Funktionsaufruf in ST

```
W_TIME1     ( H := W_TIME_H,
              M := W_TIME_M,
              SEC := W_TIME_SEC,
              MS := W_TIME_MS);
W_TIME_TIMEE := W_TIME1.TIMEE;
```

Wort nach Doppelwort-Wandlung WDW S40

Der Wert des Wortoperanden am Eingang E wird in eine Doppelwortgröße gewandelt, und das Ergebnis dem Doppelwort-Operanden am Ausgang A zugewiesen.



Bausteintyp

Funktion

Parameter

E	INT	zu wandelnde Wortgröße
A	DINT	Ergebnis der Wandlung, Doppelwortgröße

Beschreibung

Der Wert des Wortoperanden am Eingang E wird in eine Doppelwortgröße gewandelt, und das Ergebnis dem Doppelwort-Operanden am Ausgang A zugewiesen.

Wertebereich für E1:
8000H ≤ E1 ≤ 7FFFH
-32768 ≤ E1 ≤ 32767

Ein- und Ausgang sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
WDW_E AT %MW1000.0 : INT;
WDW_A AT %MD2000.0 : DINT;
```

Übersetzung in ABB AWL:

```
!BA 0
WDW
E
A
```

FBD:

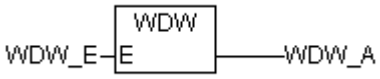


ABB AWL des Beispiels:

```
!BA 0
WDW
MW0,0
MD0,0
```

Funktionsaufruf in AWL

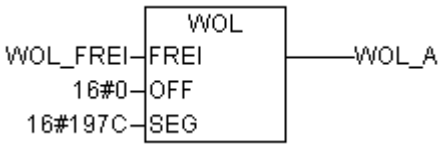
```
LD    WDW_E
WDW
ST    WDW_DW
```

Funktionsaufruf in ST

```
WDW_DW := WDW(WDW_E);
```

Wort lesen mit Freigabe WOL S40

TRUE-Signal am Eingang FREI bewirkt, daß der Wert der angegebenen physikalischen Adresse gelesen und dem Operanden am Ausgang A zugewiesen wird.



Bausteintyp

Funktion

Parameter

FREI	BOOL	Freigabe des Bausteins
OFF	INT	Offset-Adresse des Speicherplatzes
SEG	INT	Segment-Adresse des Speicherplatzes
A	INT	Ausgang, dem der gelesene Wert zugewiesen wird.

Beschreibung

TRUE-Signal am Eingang FREI bewirkt, daß der Wert der angegebenen physikalischen Adresse gelesen und dem Operanden am Ausgang A zugewiesen wird.

Bei FALSE-Signal am Eingang FREI erfolgt kein Lesen und keine Zuweisung.

Die Eingänge und der Ausgang sind weder doppelbar noch negierbar.

FREI BOOL

Mit dem Operand am Eingang FREI wird die Bearbeitung des Bausteins freigegeben oder gesperrt.

Es gilt:

FREI = FALSE → Bearbeitung gesperrt
FREI = TRUE → Bearbeitung freigegeben

OFF INT

Am Eingang OFF wird die zu lesenden Offset-Adresse angegeben. Die Angabe erfolgt als 16-Bit-Adresse.

SEG INT

Am Eingang ADRESSE wird die zu lesenden Segment-Adresse angegeben. Die Angabe erfolgt als 16-Bit-Adresse.

A INT

Dem Operanden am Ausgang A wird der gelesene Wert zugewiesen.

Beispiel

Deklaration:

```
WOL_FREI AT %MX0.0 : BOOL;
WOL_A AT %MW1000.0 : INT
```

Übersetzung in ABB AWL:

```
!BA 0
WOL
FREI
OFF
SEG
```

FBD:

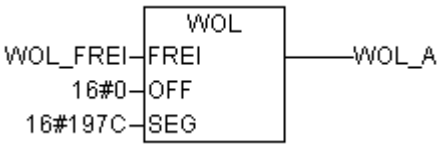


ABB AWL des Beispiels:

```
!BA 0
WOL
M0,0
#0
#H197C
```

Funktionsaufruf in AWL

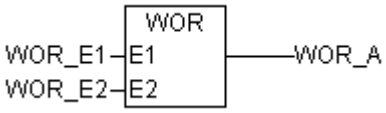
```
LD      WOL_FREI
WOL     WOL_OFF, WOL_ADR
ST      WOL_A
```

Funktionsaufruf in ST

```
WOL_A := WOL(WOL_FREI,
             WOL_OFF,
             WOL_SEG);
```

ODER-Verknüpfung, Wort WOR S40

Der Funktionsbaustein bildet bitweise die ODER-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.



Bausteintyp

Funktion

Parameter

E1	INT	Operand 1
E2	INT	Operand 2
A	INT	Ergebnis der ODER-Verknüpfung

Beschreibung

Der Funktionsbaustein bildet bitweise die ODER-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
WOR_E1 AT %MW1000.0 : INT;
WOR_E2 AT %MW1000.1 : INT;
WOR_A AT %MW1000.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
WOR
E1
E2
A
```

FBD:

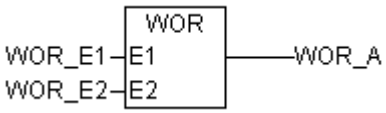


ABB AWL des Beispiels:

```
!BA 0
WOR
MW0,0
MW0,1
MW0,2
```

Funktionsaufruf in AWL

```
LD    WOR_E1
WOR   WOR_E2
ST    WOR_A
```

Funktionsaufruf in ST

```
WOR_A    := WOR(E1 := WOR_E1,
            E2 := WOR_E2);
```

Exklusiv-ODER-Verknüpfung, Wort WXOR S40

Der Funktionsbaustein bildet bitweise die XOR-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.

Bausteintyp

Funktion

Parameter

E1	INT	Operand 1
E2	INT	Operand 2
A	INT	Ergebnis der XOR-Verknüpfung

Beschreibung

Der Funktionsbaustein bildet bitweise die XOR-Verknüpfung der an den Eingängen E1 und E2 anliegenden Operanden. Das Ergebnis wird dem Operanden am Ausgang zugewiesen.

Die Ein- und Ausgänge sind weder doppelbar noch negierbar.

Beispiel

Deklaration:

```
WXOR_E1 AT %MW1000.0 : INT;
WXOR_E2 AT %MW1000.1 : INT;
WXOR_A AT %MW1000.2 : INT;
```

Übersetzung in ABB AWL:

```
!BA 0
WXOR
E1
E2
A
```

FUP:

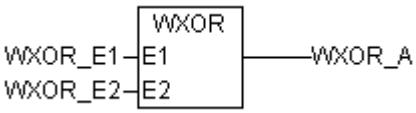


ABB AWL des Beispiels:

```
!BA 0
WXOR
MW0,0
MW0,1
MW0,2
```

Funktionsaufruf in AWL

```
LD    WXOR_E1
WXOR  WXOR_E2
ST    WXOR_A
```

Funktionsaufruf in ST

```
WXOR_A    := WXOR(E1 := WXOR_E1,
             E2 := WXOR_E2);
```

Glossar

BOOL

Variablen vom Typ BOOL können die Wahrheitswerte TRUE und FALSE annehmen. Es werden 8 Bit Speicherplatz reserviert.

DINT

DINT gehört zu den ganzzahligen Datentypen.

Die unterschiedlichen Zahlentypen decken einen unterschiedlichen Zahlenbereich ab. Für die ganzzahligen Datentypen gelten die folgenden Bereichsgrenzen:

Typ	Untere Grenze	Obere Grenze	Speicherplatz
DINT:	-2147483648	2147483647	32 Bit

Dadurch kann es vorkommen, daß bei der Typkonvertierung von größeren auf kleinere Typen Informationen verlorengehen.

DWORD

DWORD gehört zu den ganzzahligen Datentypen.

Die unterschiedlichen Zahlentypen decken einen unterschiedlichen Zahlenbereich ab. Für die ganzzahligen Datentypen gelten die folgenden Bereichsgrenzen:

Typ	Untere Grenze	Obere Grenze	Speicherplatz
DWORD	0	4294967295	32 Bit

Dadurch kann es vorkommen, daß bei der Typkonvertierung von größeren auf kleinere Typen Informationen verlorengehen.

INT

INT gehört zu den ganzzahligen Datentypen.

Die unterschiedlichen Zahlentypen decken einen unterschiedlichen Zahlenbereich ab. Für die ganzzahligen Datentypen gelten die folgenden Bereichsgrenzen:

Typ	Untere Grenze	Obere Grenze	Speicherplatz
INT:	-32768	32767	16 Bit

Dadurch kann es vorkommen, daß bei der Typkonvertierung von größeren auf kleinere Typen Informationen verlorengehen.

WORD

WORD gehört zu den ganzzahligen Datentypen.

Die unterschiedlichen Zahlentypen decken einen unterschiedlichen Zahlenbereich ab. Für die ganzzahligen Datentypen gelten die folgenden Bereichsgrenzen:

Typ	Untere Grenze	Obere Grenze	Speicherplatz
WORD	0	65535	16 Bit

Dadurch kann es vorkommen, daß bei der Typkonvertierung von größeren auf kleinere Typen Informationen verlorengehen.

Funktionen

Funktionen sind Unterprogramme, die beliebig viele Eingangsparameter haben und genau ein Ergebnis-Element zurückliefern. Das zurückgelieferte Ergebnis kann von elementarem, aber auch von zusammengesetztem Datentyp sein.

Funktionen liefern bei gleicher Eingangsbeschaltung stets das gleiche Ergebnis (sie besitzen kein Gedächtnis).

Daraus abzuleiten sind folgende Regeln:

- Innerhalb von Funktionen dürfen globale Variablen weder gelesen noch geschrieben werden.
- Innerhalb von Funktionen dürfen Absolutoperanden weder gelesen noch geschrieben werden
- Innerhalb von Funktionen dürfen Funktionsblöcke nicht aufgerufen werden.

Funktionsblöcke

Funktionsblöcke sind Unterprogramme, die jeweils beliebig viele Eingangs-, Ausgangs- und interne Variablen haben. Sie werden von einem Programm aufgerufen oder von einem anderen Funktionsblock.

Da sie auch mehrfach genutzt werden können (mit jeweils anderen Datensätzen), können Funktionsblöcke (ihr Code und die Schnittstelle) als Typ betrachtet werden. Über die Zuordnung eines individuellen Datensatzes (Deklaration) wird eine Instanz des Funktionsblockes angelegt.

Anders als Funktionen können Funktionsblöcke statisch lokale Daten haben, die von einem Aufruf zum nächsten gerettet werden. Damit können z. B. Zähler realisiert werden, die ihren Zählerstand nicht vergessen dürfen. D. h. Funktionsblöcke können ein Gedächtnis haben.

Funktionen und Funktionsblöcke unterscheiden sich in zwei wesentlichen Punkten:

- Ein Funktionsblock hat beliebig viele Ausgangsparameter. Eine Funktion hat maximal einen Ausgangsparameter. Es ist zu beachten, daß sich die Ausgangsparameter von Funktionen und Funktionsblöcken syntaktisch unterscheiden.
- Ein Funktionsblock kann ein Gedächtnis haben, eine Funktion nicht.

Bei allen ABB-Funktionsblöcken ist zu beachten, daß Instanznamen nicht mehrfach vergeben werden dürfen, wenn unterschiedliche Datensätze aufgerufen werden sollen.

Beispiel:

Name	Typ
VRZ1	VRZ
VRZ2	VRZ
.	.

Index

U

V

W

Druckschrift-Nr.: 2CDC 120 069 M 0102

ABB STOTZ-KONTAKT GmbH
Eppelheimer Straße 82 Postfach 101680
69123 Heidelberg 69006 Heidelberg
Deutschland Deutschland

Telefon (06221) 701-0
Telefax (06221) 701-1111
E-Mail desst.helpline@de.abb.com
Internet http://www.abb.de/stotz-kontakt

ABB Control
10, rue Ampère Z.I. - B.P. 114
F-69685 Chassieu cedex, France

Telefon +33 (0) 4 7222 1722
Telefax +33 (0) 4 7222 1935
E-Mail
Internet http://www.abb.com/lowvoltage