

Beispiel: Flankenwertung und Toggle FF

Flankenwertung

Eine positive Flanke am Eingang 32.0 soll abgefragt werden. Der Ausgang 32.0 soll für einen Zyklus auf 1 gesetzt werden, falls die positive Flanke kommt.

Toggle FF

Mit diesem Ausgang 32.0 soll nun ein Toggle FF angesteuert werden, dessen Ausgang auf 32.7 liegt

ORGANIZATION BLOCK OB1

TITLE = "Zyklisches Hauptprogramm"

AUTHOR: Graf

FAMILY: Flankenwertung und RS FF

NAME: Vorlesung

VERSION: 1.0

VAR_TEMP

```
OB1_EV_CLASS:BYTE           //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
OB1_SCAN_1:BYTE             //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
OB1_PRIORITY:BYTE          //1 (Priority of 1 is lowest)
OB1_OB_NUMBR:BYTE          //1 (Organization block 1, OB1)
OB1_RESERVED_1:BYTE        //Reserved for system
OB1_RESERVED_2:BYTE        //Reserved for system
OB1_PREV_CYCLE:INT         //Cycle time of previous OB1 scan (milliseconds)
OB1_MIN_CYCLE:INT          //Minimum cycle time of OB1 (milliseconds)
OB1_MAX_CYCLE:INT          //Maximum cycle time of OB1 (milliseconds)
OB1_DATE_TIME:DATE_AND_TIME //Date and time OB1 started
```

END_VAR

BEGIN

NETWORK //Nr.:1

TITLE = Flankenwertung S5

```
U   E   32.0
UN  M   32.0
=   A   32.0
UN  E   32.0
UN  M   32.1
=   A   32.1
```

```
U   E   32.0
=   M   32.0
UN  E   32.0
=   M   32.1
```

```
U   A   32.0
S   A   32.2
R   A   32.3
```

```
U   A   32.1
S   A   32.3
R   A   32.2
```

NETWORK //Nr.:2

TITLE= Toggle FF S5

```
U   A   32.0
U   A   32.7
R   M   32.7
```

```
U   A   32.0
UN  A   32.7
S   M   32.7
```

```
U   M   32.7
=   A   32.7
```

NETWORK //Nr.:3

TITLE = Flankenauswertung S7

U	E	33.0
FP	M	0.0
=	A	33.0
U	E	33.0
FN	M	0.1
=	A	33.1
U	A	33.0
S	A	33.2
R	A	33.3
U	A	33.1
S	A	33.3
R	A	33.2

NETWORK //Nr.:4

TITLE= Toggle FF S7

U	A	33.0
U	A	33.7
R	M	33.7
U	A	33.0
UN	A	33.7
S	M	33.7
U	M	33.7
=	A	33.7

END_ORGANIZATION_BLOCK

Beispiel: RS Speicher für Pumpensteuerung mit Vorrang und Verriegelung

3 Pumpen sollen mit 6 Tastern ein und ausgeschaltet werden. Aus hat jeweils Vorrang. Pumpe 2 und 3 sollen gegeneinander verriegelt sein.

Pumpe 1 (A0.0)	Ein (E0.0)
	Aus (E0.1)
Pumpe 1 (A0.2)	Ein (E0.2)
	Aus (E0.3)
Pumpe 1 (A0.4)	Ein (E0.4)
	Aus (E0.5)

ORGANIZATION BLOCK OB1

```
TITLE = "Zyklisches Hauptprogramm"
AUTHOR:  MHJTW
FAMILY:  nb
NAME:    nb
VERSION: 1.0
```

```
VAR_TEMP
  OB1_EV_CLASS:BYTE           //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
  OB1_SCAN_1:BYTE            //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
  OB1_PRIORITY:BYTE          //1 (Priority of 1 is lowest)
  OB1_OB_NUMBR:BYTE           //1 (Organization block 1, OB1)
  OB1_RESERVED_1:BYTE        //Reserved for system
  OB1_RESERVED_2:BYTE        //Reserved for system
  OB1_PREV_CYCLE:INT          //Cycle time of previous OB1 scan (milliseconds)
  OB1_MIN_CYCLE:INT           //Minimum cycle time of OB1 (milliseconds)
  OB1_MAX_CYCLE:INT           //Maximum cycle time of OB1 (milliseconds)
  OB1_DATE_TIME:DATE_AND_TIME //Date and time OB1 started
END_VAR
BEGIN
NETWORK //Nr.:1
TITLE =
  U   E           0.0           //Pumpe 1
  S   A           0.0
  U   E           0.1
  R   A           0.0

  U   E           0.2           //Pumpe 2
  UN  A           0.4
  S   A           0.2
  U   E           0.3
  R   A           0.2

  U   E           0.4           //Pumpe 3
  UN  A           0.2
  S   A           0.4
  U   E           0.5
  R   A           0.4
```

```
END_ORGANIZATION_BLOCK
```

Beispiel: kombinierte Ein- Ausschaltverzögerung

Eine kombinierte Ein- Ausschaltverzögerung soll programmiert werden. Ausgang 0.0 wird mit Eingang 0.0 ein und ausgeschaltet, und zwar beim Einschalten um 2s und beim Ausschalten ebenfalls um 2s verzögert.

ORGANIZATION BLOCK OB1

TITLE = "Ein und Ausschaltverzögerung kombiniert"

AUTHOR: Graf
FAMILY: SS 2000
NAME: einaus
VERSION: 1.0

```
VAR_TEMP
  OB1_EV_CLASS:BYTE           //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
  OB1_SCAN_1:BYTE            //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
  OB1_PRIORITY:BYTE          //1 (Priority of 1 is lowest)
  OB1_OB_NUMBR:BYTE           //1 (Organization block 1, OB1)
  OB1_RESERVED_1:BYTE         //Reserved for system
  OB1_RESERVED_2:BYTE         //Reserved for system
  OB1_PREV_CYCLE:INT          //Cycle time of previous OB1 scan (milliseconds)
  OB1_MIN_CYCLE:INT           //Minimum cycle time of OB1 (milliseconds)
  OB1_MAX_CYCLE:INT           //Maximum cycle time of OB1 (milliseconds)
  OB1_DATE_TIME:DATE_AND_TIME //Date and time OB1 started
END_VAR
BEGIN
NETWORK //Nr.:1
TITLE = Zeitwert laden
  L   S5T#2S
NETWORK //Nr.:2
TITLE= Einschaltverzögerung
  U   E           0.0
  SE  T           0
  U   T           0
  =   M           0.0
NETWORK //Nr.:3
TITLE= Ausschaltverzögerung der Einschaltverzögerung
  U   M           0.0
  SA  T           1
  U   T           1
  =   A           0.0
END_ORGANIZATION_BLOCK
```

Beispiel: Taktgenerator mit 2 Timern

Ein Taktgenerator soll programmiert werden, der den Ausgang 0.0 mit einer Frequenz $f = 1\text{Hz}$ ansteuert. Der Takt soll mit dem Eingang 0.0 ein- und ausgeschaltet werden können. Programmieren Sie den Taktgenerator mit 2 Timern.

ORGANIZATION BLOCK OB1

```
TITLE = "Taktgenerator mit 2 Timern"
AUTHOR:  Graf
FAMILY:  SS 2000
NAME:    takt_2t
VERSION: 1.0

VAR_TEMP
  OB1_EV_CLASS:BYTE           //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
  OB1_SCAN_1:BYTE             //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
  OB1_PRIORITY:BYTE           //1 (Priority of 1 is lowest)
  OB1_OB_NUMBR:BYTE           //1 (Organization block 1, OB1)
  OB1_RESERVED_1:BYTE         //Reserved for system
  OB1_RESERVED_2:BYTE         //Reserved for system
  OB1_PREV_CYCLE:INT          //Cycle time of previous OB1 scan (milliseconds)
  OB1_MIN_CYCLE:INT           //Minimum cycle time of OB1 (milliseconds)
  OB1_MAX_CYCLE:INT           //Maximum cycle time of OB1 (milliseconds)
  OB1_DATE_TIME:DATE_AND_TIME //Date and time OB1 started
END_VAR

BEGIN
NETWORK //Nr.:1
TITLE = Zeitwert laden
  L      S5T#500MS
NETWORK //Nr.:2
TITLE= Timer 0 starten wenn Timer 1 =0 und Ein
  U      E      0.0
  UN     T      1
  SI     T      0
NETWORK //Nr.:3
TITLE= Timer 1 starten wenn Timer 0 = 0
  UN     A      0.0
  SI     T      1
NETWORK //Nr.:4
TITLE= Timer 0 ausgeben
  U      T      0
  =      A      0.0
END_ORGANIZATION_BLOCK
```

Beispiel: Taktgenerator mit einem Timer

Ein Taktgenerator soll programmiert werden, der den Ausgang 0.0 mit einer Frequenz $f = 1\text{Hz}$ ansteuert. Der Takt soll mit dem Eingang 0.0 ein- und ausgeschaltet werden können. Programmieren Sie den Taktgenerator mit 1 Timer.

ORGANIZATION BLOCK OB1

```
TITLE = "Taktgenerator mit einem Timer"
AUTHOR:  Graf
FAMILY:  SS 2000
NAME:    takt_1t
VERSION: 1.0

VAR_TEMP
  OB1_EV_CLASS:BYTE           //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
  OB1_SCAN_1:BYTE             //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
  OB1_PRIORITY:BYTE           //1 (Priority of 1 is lowest)
  OB1_OB_NUMBR:BYTE           //1 (Organization block 1, OB1)
  OB1_RESERVED_1:BYTE         //Reserved for system
  OB1_RESERVED_2:BYTE         //Reserved for system
  OB1_PREV_CYCLE:INT          //Cycle time of previous OB1 scan (milliseconds)
  OB1_MIN_CYCLE:INT           //Minimum cycle time of OB1 (milliseconds)
```

```

OBI_MAX_CYCLE:INT           //Maximum cycle time of OBI (milliseconds)
OBI_DATE_TIME:DATE_AND_TIME //Date and time OBI started
T_Merker:BOOL               //Timer Merker ist alle S = 0
Toggle_Merker:BOOL         //Toggle Merker
END_VAR
BEGIN
NETWORK //Nr.:1
TITLE = Zeitwert setzen
L S5T#1S
NETWORK //Nr.:2
TITLE= T_Merker alle s fuer einen Zyklus auf 0
U E 0.0
UN #T_Merker
SV T 0

U T 0
= #T_Merker
NETWORK //Nr.:3
TITLE= Ruecksetzen falls Ausgang 1
U A 0.0
UN #T_Merker
U E 0.0
R #Toggle_Merker
NETWORK //Nr.:4
TITLE= Setzen falls Ausgang 0
UN A 0.0
UN #T_Merker
U E 0.0
S #Toggle_Merker
NETWORK //Nr.:5
TITLE= Ausgang je nach Togglemerker
U #Toggle_Merker
= A 0.0
END_ORGANIZATION_BLOCK

```

Beispiel: Ampel als Zustandsmaschine mit RS FF

Eine vereinfachte Ampelanlage soll gesteuert werden. Die Zustände sind Rot (3s), Gelb (1s) und Grün (2s). Ein Reset ist vorzusehen, bei dem alle Zustände auf Aus gehen. Mit einem Ein-Taster geht die Ampel in den Zustand Rot. Die einfache Ampel ist als Zustandsmaschine mit RS FF auszuführen. Es soll symbolisch programmiert werden.

Symbolic Datei

Einschalter	E	0.0	BOOL	Ein
Restschalter	E	0.1	BOOL	Reset
Rot Merker	M	0.0	BOOL	Merker Rot
Gelb Merker	M	0.1	BOOL	Merker Gelb
Gruen Merker	M	0.2	BOOL	Merker Gelb
Rot	A	0.0	BOOL	Rot
Gelb	A	0.1	BOOL	Gelb
Gruen	A	0.2	BOOL	Grün
Timer Rot	T	0	TIMER	Timer Rot
Timer Gelb	T	1	TIMER	Timer Gelb
Timer Grün	T	2	TIMER	Timer Grün

ORGANIZATION BLOCK OB 100

```
TITLE = "Complete Restart"
AUTHOR:  Graf
FAMILY:  Ampel einfach
NAME:    Vorlesungsbeispiel
VERSION: 1.0

VAR_TEMP
  OB100_EV_CLASS:BYTE           //16#13, Event class 1, Entering event state, Event
logged in diagnostic buffer
  OB100_STARTUP:BYTE           //16#81/82/83/84 Method of startup
  OB100_PRIORITY:BYTE          //27 (Priority of 1 is lowest)
  OB100_OB_NUMBR:BYTE          //100 (Organization block 100, OB100)
  OB100_RESERVED_1:BYTE        //Reserved for system
  OB100_RESERVED_2:BYTE        //Reserved for system
  OB100_STOP:WORD              //Event that caused CPU to stop (16#4xxx)
  OB100_STRT_INFO:DWORD        //Information on how system started
  OB100_DATE_TIME:DATE_AND_TIME //Date and time OB100 started
END_VAR
BEGIN
NETWORK //Nr.:1
TITLE =
  SET
  R    "Rot Merker"           //Merker Rot
  R    "Gruen Merker"        //Merker Gelb
  R    "Gruen Merker"        //Merker Gelb
  BE
END_ORGANIZATION_BLOCK
```

ORGANIZATION BLOCK OB1

```
TITLE = "Zyklisches Hauptprogramm"
AUTHOR:  Graf
FAMILY:  Ampel einfach
NAME:    Vorlesungsbeispiel
VERSION: 1.0

VAR_TEMP
  OB1_EV_CLASS:BYTE           //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
  OB1_SCAN_1:BYTE             //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
  OB1_PRIORITY:BYTE           //1 (Priority of 1 is lowest)
  OB1_OB_NUMBR:BYTE           //1 (Organization block 1, OB1)
  OB1_RESERVED_1:BYTE         //Reserved for system
  OB1_RESERVED_2:BYTE         //Reserved for system
  OB1_PREV_CYCLE:INT           //Cycle time of previous OB1 scan (milliseconds)
  OB1_MIN_CYCLE:INT           //Minimum cycle time of OB1 (milliseconds)
```

```

OBI_MAX_CYCLE:INT //Maximum cycle time of OBI (milliseconds)
OBI_DATE_TIME:DATE_AND_TIME //Date and time OBI started
Ein_Flankenmerker:BOOL
END_VAR
BEGIN
NETWORK //Nr.:1
TITLE = Flankenerkennung
U "Einschalter" //Ein
FP M 1.0
= #Ein_Flankenmerker
NETWORK //Nr.:2
TITLE= Rot
L S5T#3S
O "Restschalter" //Reset
O "Gelb Merker" //Merker Gelb
R "Rot Merker" //Merker Rot
O #Ein_Flankenmerker
O(
U "Gruen Merker" //Merker Gelb
UN "Timer Grün" //Timer Grün
)
S "Rot Merker" //Merker Rot
SV "Timer Rot" //Timer Rot
NETWORK //Nr.:3
TITLE= Gelb
L S5T#1S
O "Restschalter" //Reset
O "Gruen Merker" //Merker Gelb
R "Gelb Merker" //Merker Gelb
U "Rot Merker" //Merker Rot
UN "Timer Rot" //Timer Rot
S "Gelb Merker" //Merker Gelb
SV "Timer Gelb" //Timer Gelb
NETWORK //Nr.:4
TITLE= Grün
L S5T#2S
O "Restschalter" //Reset
O "Rot Merker" //Merker Rot
R "Gruen Merker" //Merker Gelb
U "Gelb Merker" //Merker Gelb
UN "Timer Gelb" //Timer Gelb
S "Gruen Merker" //Merker Gelb
SV "Timer Grün" //Timer Grün
NETWORK //Nr.:5
TITLE= Anzeige
U "Rot Merker" //Merker Rot
S "Rot" //Rot
R "Gelb" //Gelb
R "Gruen" //Grün

U "Gelb Merker" //Merker Gelb
R "Rot" //Rot
S "Gelb" //Gelb
R "Gruen" //Grün

U "Gruen Merker" //Merker Gelb
R "Rot" //Rot
R "Gelb" //Gelb
S "Gruen" //Grün

UN "Rot Merker" //Merker Rot
UN "Gelb Merker" //Merker Gelb
UN "Gruen Merker" //Merker Gelb
R "Rot" //Rot
R "Gelb" //Gelb
R "Gruen" //Grün
END_ORGANIZATION_BLOCK

```

Beispiel: Ampel als Zustandsmaschine mit Zählern

Eine vereinfachte Ampelanlage soll gesteuert werden. Die einfache Ampel ist als Zustandsmaschine mit Zähler auszuführen. Die Zustände sind Rot (6s, A0.0), Gelb (1s, A0.1) und Grün (3s, A0.2). Ein Reset (E0.1) ist vorzusehen, bei dem die Ampel in den Zustand Rot geht und dort verharret. Mit einem Ein Taster (E0.0) geht die Ampel ebenfalls in den Zustand Rot, jedoch wird zusätzlich der Zähler gestartet. Es soll symbolisch programmiert werden. Die Zählergrenzen sind als Variable auszuführen, damit eine Änderung leicht erfolgen kann.

Smbolic Datei

Start	E	0.0	BOOL	Starttaste
Stop	E	0.1	BOOL	Stoptaste
Timer0	T	0	TIMER	Timer 0
Timer1	T	1	TIMER	Timer 1
Zähler0	Z	0	COUNTER	Zähler 0
rot	A	0.0	BOOL	rot
gelb	A	0.1	BOOL	gelb
gruen	A	0.2	BOOL	gruen
Zaehlerwert	AB	1	BYTE	Zaehlerwert

ORGANIZATION BLOCK OB1

```
TITLE = "einfache Ampel mit Zähler"
AUTHOR: Graf
FAMILY: nb
NAME: nb
VERSION: 1.0

VAR_TEMP
  OB1_EV_CLASS:BYTE //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
  OB1_SCAN_1:BYTE //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
  OB1_PRIORITY:BYTE //1 (Priority of 1 is lowest)
  OB1_OB_NUMBR:BYTE //1 (Organization block 1, OB1)
  OB1_RESERVED_1:BYTE //Reserved for system
  OB1_RESERVED_2:BYTE //Reserved for system
  OB1_PREV_CYCLE:INT //Cycle time of previous OB1 scan (milliseconds)
  OB1_MIN_CYCLE:INT //Minimum cycle time of OB1 (milliseconds)
  OB1_MAX_CYCLE:INT //Maximum cycle time of OB1 (milliseconds)
  OB1_DATE_TIME:DATE_AND_TIME //Date and time OB1 started
  Takt:BOOL //Taktvariable
  rot_grenze:INT //bis hier rot
  gruen_grenze:INT //ab hier grün
  reset_wert:INT //ab hier Zähler rücksetzen

END_VAR
BEGIN
NETWORK //Nr.:1
TITLE= Grenzen setzen
  L 5
  T #rot_grenze
  L 7
  T #gruen_grenze
  L 10
  T #reset_wert
NETWORK //Nr.:2
TITLE = Takt
  L S5T#500MS

  U "Start" //Starttaste
  UN "Timer1" //Timer 1
  SI "Timer0" //Timer 0

  UN #Takt
  SI "Timer1" //Timer 1
```

```

        U   "Timer0"           //Timer 0
        =   #Takt
NETWORK //Nr.:3
TITLE=   zählen
        U   #Takt
        ZV  "Zähler0"         //Zähler 0
NETWORK //Nr.:4
TITLE=   rot

        L   "Zähler0"         //Zähler 0
        L   #rot_grenze
        <=I
        =   "rot"             //rot
NETWORK //Nr.:5
TITLE=   gelb

        L   "Zähler0"         //Zähler 0
        L   #rot_grenze
        >I
        U(
        L   "Zähler0"         //Zähler 0
        L   #gruen_grenze
        <I
        )
        =   "gelb"            //gelb
NETWORK //Nr.:6
TITLE=   grün
        L   "Zähler0"         //Zähler 0
        L   #gruen_grenze
        >=I
        =   "gruen"           //gruen
NETWORK //Nr.:7
TITLE=   Zähler rücksetzen
        L   "Zähler0"         //Zähler 0
        L   #reset_wert
        ==I
        R   "Zähler0"         //Zähler 0
NETWORK //Nr.:8
TITLE=   Anzeige Zählerwert
        L   "Zähler0"         //Zähler 0
        T   "Zaehlerwert"     //Zaehlerwert
END_ORGANIZATION_BLOCK

```

Beispiel: Pulsgenerator mit OB33 und Netzausfallanzeige

Ein Pulsgenerator soll über den OB33 programmiert werden. Ausgang A0.0 soll im Takt von 1Hz ein und ausgeschalten werden. Ausgang A0.7 soll anzeigen fals die Spannung ausgefallen ist.

OB101

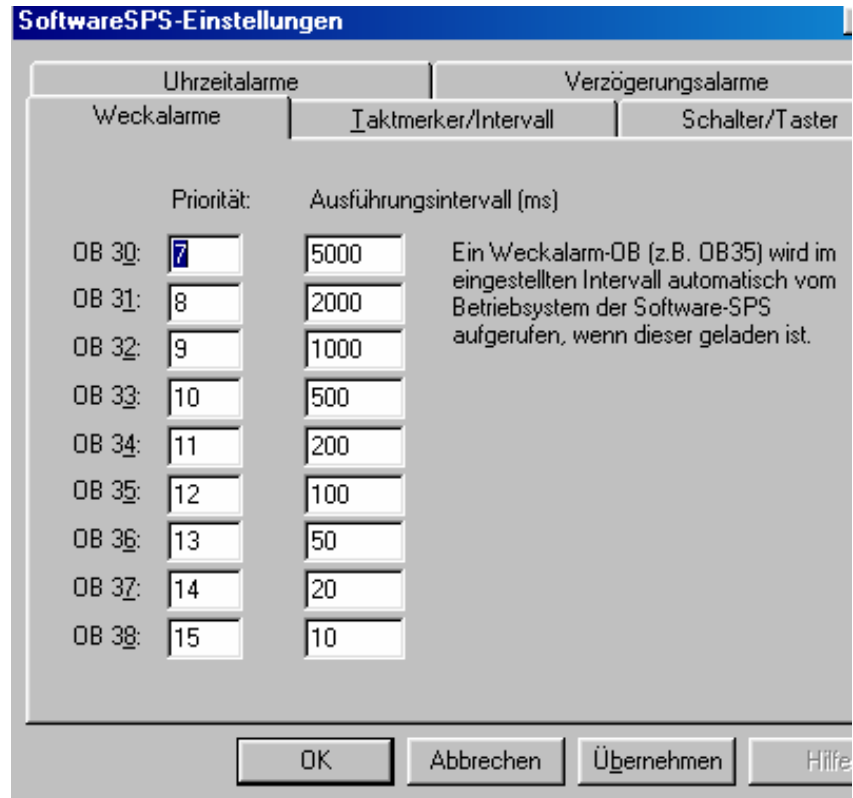
```
ORGANIZATION_BLOCK OB 101
TITLE = "Restart"
.....
BEGIN
NETWORK //Nr.:1
TITLE =
    SET
    S A 0.7
END_ORGANIZATION_BLOCK
```

OB100

```
ORGANIZATION_BLOCK OB 100
TITLE = "Complete Restart"
.....
BEGIN
NETWORK //Nr.:1
TITLE =
    SET
    R A 0.7
END_ORGANIZATION_BLOCK
```

OB33

```
ORGANIZATION_BLOCK OB 33
TITLE = "Cyclic Interrupt"
.....
BEGIN
NETWORK //Nr.:1
TITLE =
    UN A 0.0
    = A 0.0
END_ORGANIZATION_BLOCK
```



Beispiel: Analogeingabe, Wandlung und Anzeige der Spannung als BCD Wert

Auf dem ersten Steckplatz soll ein Analogeingangsmodul gesteckt werden. Der erste Kanal soll eingelesen werden. Die Konfiguration ist ± 10 V. Auf Ausgang AW0 soll der Analogwert als 4 Digit BCD Wert dargestellt werden. Negative Werte sollen durch Ausgang A2.0 dargestellt werden

```

ORGANIZATION_BLOCK OB1
TITLE = "Zyklisches Hauptprogramm"
AUTHOR:  MHJTW
FAMILY:  nb
NAME:    nb
VERSION: 1.0

```

```
VAR_TEMP
```

```

.....
//
negativ:BOOL           lokale Variable
                        //Bit ob Analogwert positiv oder negativ
END_VAR
BEGIN
NETWORK //Nr.:1
TITLE =
    U  #negativ        //falls vorher negativ
    R  #negativ        //vorbelegt positiv
    L  PEW      256    //Analogwert einlesen
    L  W#16#9000      //kleister negativer Wert
    <D                    //positiv
    POP                //Analogwert zurück in Akku1
    SPB  pos          //weitere Bearbeitung
    INVI                //1 Komplement Akku1_L
    S  #negativ        //negativ merken
pos :NOP 1
    DTR                //double to real
    L  2.764800e+04    //6C00 = 10V = 27648 //93FF = -10V

    /R                    //normieren auf 1
    L  1.000000e+01
    *R                    //Real Darstellung 10V = 10
    T  MD      0
    L  1.000000e+02    //Real Darstellung 10V = 1000
    *R
    RND                //Real nach Double
    DTB                //Double nach BCD
    T  AW      0

    U  #negativ        //positiv oder negativ
    =  A      2.0

```

```
END_ORGANIZATION_BLOCK
```

