Beschreibung aller Unterprogramme der S7-200 Bibliothek PID-EASY

Welche Unterprogramme sind im Softwarepaket PID-Easy für S7-200 enthalten ?

Unterprogrammname	Unterprogrammfunktion
INIT_TIMER_ACK	Permanenttimer, Zeittakte, Quittiertakt + Quittierpuls
DELTA_t	Zeitdifferenz zwischen 2-Aufrufen des 1ms-TIMERS + Speicherung ISTWERT
TIMER_tD_ON_OFF	TIMER tv_EIN, tvAUS, tmin_AUS, tmin_EIN; tv: {0,...,32767*0.1min + 0,...,32767*0.1s}
MUL_MINUS1_INT	Multiplikation mit *(-1) ohne Überlauf
DATA_SWITCH_INT	Datenumschalter für 2 Integerwerte
AH_SWITCH_INT	AUTO-HAND-Umschaltung für Integerwert (Stellgröße AUTO-HAND!)
MIN_MAX_LIMIT	Begrenzung eines Integerwertes auf eine Ober- + Untergrenze
MIN_MAX_LIMIT_TOTBAND	Begrenzung eines Integerwertes auf eine Ober- + Untergrenze mit Totband
RAMP_INT	Abbildung einer Eingangsgröße mit UP- + DOWN-Rampe auf eine Ausgangsgröße
DAMP_INT	Dämpfung eines Signales über Zeittakt + FIFO + Mittelwertbildung
SWG_INT	Sollwertführung über max 10 Geradenstützpunkte: Pi(xi,yi)
SWG_TIME	Zeitplanführung für Sollwerte mit max 10 Stützpunkten: Pi(wi,ti)
NORMSIGNAL_ANALOGOUTPUT	Skalierung eines Normsignales {-1000,…,0,…,+1000} in Analogausgangssignal
SCALE_LINEAR_LIMIT_INT	Lineare Skalierung eines beliebigen Signales mit Ober- + Untergrenze
AVERAGE_2_INT	Mittelwert von 2-Integerwerten
AVERAGE_2_8_INT	Mittelwert von 2 bis 8 Integerwerten
MIN_MAX_SELECT_2_INT	Auswahl von Minimum + Maximum aus 2 Integerwerten
MIN_MAX_SELECT_2_4_INT	Auswahl von Minimum + Maximum aus 2 bis 8 Integerwerten
MIN_MAX_ALARM	Alarm, wenn ein Integerwert ein Minimum bzw. Maximum erreicht
HY_SWITCH_hw	Hystereseschalter mit : Hysterese + Sollwert, direkt- + inverswirkend
HY_SWITCH_xON_OFF	Hystereseschalter mit : Einschalt- + Ausschaltpunkt, direkt- + inverswirkend
P_CONT	Proportionalregler mit Kaskadeneingang
PID_CONT	PID-Regler mit Kaskadeneingang
ZUSATZSEQUENZ	Zusatzsequenz
THREE_STEP_CONT	Dreipunktregler
AD_TAKTGEBER	Analog- Digitalwandler mit puls- / pausenmodulierten Digitalausgängen
AD_SM	Analog- Digitalwandler mit Stellgliedfunktion 'AUF / ZU'

In der Bibliothek benutzte MERKER + TIMER und ihre SYMBOLE

SYMBOL	ADRESSE	KOMMENTAR
PLS_QUITT	M20.0	BASIC: Quittiertaste als CPU-interner PULS
PLS_500ms	M20.1	BASIC: PULS aller 500ms
PLS1_1s	M20.2	BASIC: PULS1 aller 1s: Immer 500ms vor PLS2
PLS2_1s	M20.3	BASIC: PULS2 aller 1s: Immer 500ms nach PLS1
tD_RUN	M20.4	BASIC: NACH FIRST_SCAN UND TIMER tD_RUN HIGH:=1
dt_TIMER_10ms	MW22	BASIC: TIMER 10ms : Zeitdifferenz zwischen 2 CPU-Zyklen
dt_TIMER_100ms	MW24	BASIC: TIMER 100ms: Zeitdifferenz zwischen 2 CPU-Zyklen
BASIC_PRV0_1_BYTE	MB21	Für UP 'INIT_TIMER_ACK': 1 BYTE PRIVATE SPEICHER
BASIC_PRV1_2_BYTE	MW26	Für UP 'INIT_TIMER_ACK': 2 BYTE SPEICHER = t_ALT, 10ms TIMER
BASIC_PRV2_2_BYTE	MW28	Für UP 'INIT_TIMER_ACK': 2 BYTE SPEICHER = t_ALT, 100ms TIMER
BASIC_PRV3_1_BYTE	MB30	Für UP 'INIT_TIMER_ACK': 1 BYTE SPEICHER=TIMER Q_TAKT
BASIC_PRV4_1_BYTE	MB31	Für UP 'INIT_TIMER_ACK': 1 BYTE SPEICHER=TIMER tD_RUN
PERMANENT_TIMER_1ms	T64	BASIC: Timer 1ms: Im UP 'INIT_TIMER_ACK' permanent neu gestartet
PERMANENT_TIMER_10ms	T100	BASIC: Timer 10ms: Im UP 'INIT_TIMER_ACK' permanent neu gestartet
PERMANENT_TIMER_100ms	T255	BASIC: Timer 100ms: Im UP 'INIT_TIMER_ACK' permanent neu gestartet

UP = UNTERPROGRAMM

SUBROUTINE_BLOCK INIT_TIMER_ACK:SBR37
TITLE=UNTERPROGRAMM 'INIT_TIMER_ACK':
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken 200,201,202,203 benutzt!
!!ES IST IN JEDEM CPU-ZYKLUS IM NETZWERK 1, ALS ERSTES PROGRAMM ZU BEARBEITEN.

```
################################################################################
Kurzbeschreibung:
Im Unterprogramm 'INI_TIMER_ACK' werden die Permanent-Timer aktiviert, aus
Hardware- oder OP-Quittierungen ein SPS-Interner Quittierpuls und ein Quittier-
takt zur Ansteuerung eines Hardware-Ausganges gebildet. Nach FIRST_SCAN=SM0.1,
wird eine Verzögerungszeit gestartet und nach Ablauf dieser Zeit eine Speicher-
stelle auf TRUE gesetzt.
Mit dieser Wartezeit können Peripheriezugriffsfehler verhindert werden, da nach
Netz 'AUS-EIN', die dezentrale Peripherie oder Erweiterungseinheiten zeitver-
zögert an das Netz gehen.
################################################################################
!!IN DIESEM UNTERPROGRAMM WERDEN FOLGENDE SPEICHERSTELLEN UP-INTERN BESCHRIEBEN:
!!UND WIE FOLGT BENUTZT:
   SYMBOL                ADRESSE              KOMMENTAR
+)PERMANENT_TIMER_1ms     T64     Timer  1ms: Wird im SBR40 permanent neu gestartet
  PERMANENT_TIMER_10ms    T100    Timer 10ms: Wird im SBR40 permanent neu gestartet
  PERMANENT_TIMER_100ms   T255    Timer100ms: Wird im SBR40 permanent neu gestartet
*)PLS_QUITT               M20.0   Quittiertaste als CPU-interner PULS
*)PLS_500ms               M20.1   PULS aller 500ms
*)PLS1_1s                 M20.2   PULS1 aller 1s: Immer 500ms vor   PLS2
*)PLS2_1s                 M20.3   PULS2 aller 1s: Immer 500ms nach  PLS1
*)tD_RUN                  M20.4   NACH FIRST_SCAN UND TIMER tD_RUN HIGH  :=1
  SBR40_PRV0_1_BYTE       MB21    SBR40: 1 BYTE PRIVATE SPEICHER
*)dt_TIMER_10m            MW22    TIMER 10ms :Zeitdifferenz  zwischen 2 CPU-Zyklen
*)dt_TIMER_100ms          MW24    TIMER 100ms:Zeitdifferenz  zwischen 2 CPU-Zyklen
  SBR40_PRV1_2_BYTE       MW26    SBR40: 2 BYTE SPEICHER = t_ALT, 10ms TIMER
  SBR40_PRV2_2_BYTE       MW28    SBR40: 2 BYTE SPEICHER = t_ALT,100ms TIMER
  SBR40_PRV3_1_BYTE       MB30    SBR40: 1 BYTE SPEICHER = TIMER ACK_CLOCK
  SBR40_PRV4_1_BYTE       MB31    SBR40: 1 BYTE SPEICHER = TIMER tD_RUN
!!Der mit '+)' gekennzeichnete TIMER wird über das Unterprogramm SBR41, 'DELA_t'
  ausgewertet.
!!Die mit '*)' gekennzeichneten Speicherstellen werden im Programm gelesen

in:
ACK_HW   [BOOL]: QUITTIERTASTE = INPUT VON HARDWARE
ACK_OP   [BOOL]: QUITTIERTASTE = INPUT VOM OP
                 !!Der 1. CPU-Zyklus = FIRST-SCAN, gilt auch als Quiittierung
                 !!Jede QUITTIERTASTE WIRD IM UP mit separatem PULS(+) ausgewertet!
tD_RUN    [INT]: Verzögerungszeit nach FIRST_SCAN    {0,1,...,255}*0,1s
                 Diese Zeit wird im ERSTEN CPU-Zyklus stets neu gestartet.
                 Ist der TIMER=HIGH, dann 'M20.4:=1'!
                 !!Fehleingabe: tD_RUN > 255 -> Korrektur: tD_RUN:=255*0,1s
                                tD_RUN <   0 -> Korrektur: tD_RUN:=  0*0,1s

OUT
ACK_CLOCK[BOOL]: QUITTIERTAKT: NACH QUITTIERUNG + FIRST_SCAN FÜR 2s = HIGH


!!Die og. PERMANENT-TIMER werden nach folgenden Kriterien permanent gestartet:
  Im ERSTEN CPU-Zyklus - der SM0.1 = FIRST_SCAN ist HIGH  - werden alle Perma-
  nent-TIMER mit dem Anfangswert '=0' neu gestartet. Der Aktualwert der TIMER,
  t_AKT, wird im UP SBR40 in jedem Zyklus verglichen 't_AKT > 16000? Ist der
  Vergleich wahr, so wird der TIMER ebenfalls neu gestartet, aber mit der Dif-
  ferenz (t_AKT-16000) vorbesetzt. Da die 10ms- und 100ms-TIMER nur zum Zyklus-
```

beginn aktualisiert werden, wird deren Zeitdifferenz zwischen 2-CPU-Zyklen in Merkerworte geschrieben. Diese Merkerworte können in Speicherstellen summiert werden und es lassen sich damit beliebig viele TIMER bilden.
Die 1ms-TIMER werden im laufenden Zyklus aktualisiert. Jede Anwendung benötigt einen Altwertspeicher und über den Aufruf des UP 'SBR41' kann auch die Zeitdifferenz welche zwischen 2-Aufrufen des 'SBR41' vergangen ist für den 1ms-TIMER ermittelt werden. Die durch Aufsummieren gebildeten TIMER haben gegenüber einen Vergleichs-TIMER bei einer Zeitmessung von 32000s.einen Fehler von ca.0,001%,
Da nur die 1ms-TIMER im Interrupt bearbeitet werden, haben diese an verschiedenen Stellen des CPU-Zyklusses verschiedene Werte. Ist'FIRST-SCAN=1', werden im UP 'SBR41' alle Altwertspeicher mit dem Istwert des 1ms-TIMERS beschrieben und DELTA_t:=0. In jedem anderen CPU-Zyklus wird vom UP 'SBR41' die Zeitdifferenz 'DELTA_t' zurückgeliefert, die an der 'MEßSTELLE' zwischen 2-CPU-Zyklen vergangen ist. Wird das 'DELTA_t' aufsummiert, dann kann man mit einer Genauigkeit von ca 0,001% beliebig viele Zeiten im RASTER von '1ms' bilden.

++
Beispiele:

```
//### TIMER 10ms/100ms: Nachstehend 100ms-TIMER
//*** TIMER-START mit PULS(+) - nur bei hoher Timergenauigkeit notwendig, da das
//    'dt_TIMER_100ms' korrekterweise erst im nächsten CPU-Zyklus addiert werden
//    darf
  LD    TIMER_START                  //=1->TIMERSTART / =0->TIMER_SPEICHER:=0
  EU
  =      #TEMP_HSP_PULS
//+++ TIMER: TIMER_SPEICHER inkrementieren ODER zurücksetzen
  LDN    TIMER_START                   //=1->TIMERSTART / =0->TIMER_SPEICHER:=0
  O      #TEMP_HSP_PULS
  O      Erst_Zykl_ein                 //Alternativ SM0.1 -> Keine Remanenz !
  MOVW   +0, TIMER_SPEICHER            //KEIN TIMER-START -> TIMER_SPEICHER:=0
  NOT
  +I     dt_TIMER_100ms, TIMER_SPEICHER //TIMER_SPEICHER:= TIMER_SPEICHER+dt_TIMER
  LD     Überlauf_unzulässig           //Überlauf ?
  OW>=   TIMER_SPEICHER, TIMER_SOLLWERT //TIMER_SPEICHER >= TIMER_SOLLWERT ?
  U      TIMER_START                   //=1->TIMERSTART / =0->TIMER_SPEICHER:=0
  =      TIMER_HIGH                    //=1 -> TIMER_SPEICHER >= TIMER_SOLLWERT !
  MOVW   TIMER_SOLLWERT, TIMER_SPEICHER //TIMER_SPEICHER:=TIMER_SOLLWERT

//### Ist ein Fehler in der Größenordnung einer CPU-Zykluszeit akzeptabel,
//    dann kann das vorstehende Programm vereinfacht werden
//+++ TIMER: TIMER_SPEICHER inkrementieren ODER zurücksetzen
  LDN    TIMER_START                   //=1->TIMERSTART / =0->TIMER_SPEICHER:=0
  O      Erst_Zykl_ein                 //Alternativ SM0.1 -> Keine Remanenz !
  MOVW   +0, TIMER_SPEICHER            //KEIN TIMER-START -> TIMER_SPEICHER:=0
  NOT
  +I     dt_TIMER_100ms, TIMER_SPEICHER //TIMER_SPEICHER:= TIMER_SPEICHER+dt_TIMER
  LD     Überlauf_unzulässig           //Überlauf ?
  OW>=   TIMER_SPEICHER, TIMER_SOLLWERT //TIMER_SPEICHER >= TIMER_SOLLWERT ?
  U      TIMER_START                   //=1->TIMERSTART / =0->TIMER_SPEICHER:=0
  =      TIMER_HIGH                    //=1 -> TIMER_SPEICHER >= TIMER_SOLLWERT !
  MOVW   TIMER_SOLLWERT, TIMER_SPEICHER //TIMER_SPEICHER:=TIMER_SOLLWERT
```

```
//### 1ms-TIMER mit TIMER-SPEICHER-Doppelwort + Altwertspeicher VW2402
//*** Permanente Aktualisierung des TIMER-Altwertspeichers
  LD     Immer_ein
  MOVD   &VB2402, AC3               //Adresse des Altwertspeichers '&VB2200
  CALL   DELTA_t                    //UP SBR38
//Das UP 'DELTA_t' liefert im 'AC0' die Zeitdifferenz DELTA_t=[T(n+1)-T(n)]
//im DINT-Format zurück.
//+++ TIMER_SPEICHER inkrementieren ODER zurücksetzen
  LDN    TIMER_START                   //=1->TIMERSTART / =0->TIMER_SPEICHER:=0
  O      Erst_Zykl_ein                 //Alternativ SM0.1 -> Keine Remanenz !
  MOVD   +0, TIMER_SPEICHER            //KEIN TIMER-START -> TIMER_SPEICHER:=0
  NOT
  +D     AC0, TIMER_SPEICHER           //TIMER_SPEICHER:=TIMER_SPEICHER+dt_TIMER
  LD     Überlauf_unzulässig           //Überlauf ?
  OD>=   TIMER_SPEICHER, TIMER_SOLLWERT //TIMER_SPEICHER >= TIMER_SOLLWERT ?
  U      TIMER_START                   //=1->TIMERSTART / =0->TIMER_SPEICHER:=0
  =      TIMER_HIGH                    //=1 -> TIMER_SPEICHER >= TIMER_SOLLWERT !
  MOVD   TIMER_SOLLWERT, TIMER_SPEICHER //TIMER_SPEICHER:=TIMER_SOLLWERT
!!Die Programmbeispiele gelten nur für CPU-Zykluszeiten tZYK<=16000*1ms=16s.

PLS500ms   [BOOL]: PULS:=1, aller  500ms
PLS1_1s    [BOOL]: PULS:=1, aller 1s, um 500ms vor  PLS2_1s
PLS2_1s    [BOOL]: PULS:=1, aller 1s, um 500ms nach PLS1_1s
!!Diese Pulse sind nur brauchbar, wenn die CPU-Zykluszeit 't_ZYK' kleiner ist,
  als der Zeitabstand zwischen 2 Pulsen. Die maximale Zykluszeit muß jedoch
  im Bereich 't_ZYK<500ms' liegen, wenn der 500ms-PULS und die 1s-PULSE
  benutzt werden sollen.
!!PLS1_1s UND PLS2_1s sind um 500ms zueinander versetzt:

                         PLS1_1s='|'
    ^
    |
   1+     |           |           |           |           |
    |     |           |           |           |           |
    |     |           |           |           |           |
    +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
    0    500   1000  1500  2000  2500  3000  3500  4000  4500  5000  t[ms]

                         PLS2_1s='!'
    ^
    |
   1+           !           !           !           !           !
    |           !           !           !           !           !
    |           !           !           !           !           !
    +-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
    0    500   1000  1500  2000  2500  3000  3500  4000  4500  5000  t[ms]
```

Diese versetzten Pulse können in KOP-Anwendungen dann benutzt werden, wenn garantiert werden muß, daß 2 Schaltvorgänge einen zeitlichen Abstand voneinander haben müssen. Z.B. wird mit 'E_AZ' eine Klappe 'AUF' und 'ZU' gefahren. Am Klappenstellmotor darf das Signal 'AUF' und 'ZU' nie gleichzeitig anliegen. Da dieser Vorgang nicht zeitkritisch ist, kann folgende KOP-Schaltung benutzt werden:

```
NETZWERK: Motorklappe 'AUF'
   |      E_AZ       PLS1_1s                ZU           AUF
   +--------| |---------| |--------+--------|/|----------( )-|
   |      AUF        PLS1_1s       |
   +--------| |---------|/|--------+
   |
NETZWERK: Motorklappe 'ZU'
   |      E_AZ       PLS2_1s                AUF          ZU
   +--------|/|---------| |--------+--------|/|----------( )-|
   |      ZU         PLS2_1s       |
   +--------| |---------|/|--------+
   |
```

Diese Schaltung garantiert, daß die CPU-Ausgänge 'AUF' UND 'ZU' bei jedem Umschaltvorgang mindestens 500ms lang gleichzeitig 'AUS' sind und dann erst der angeforderte Ausgang "EINSCHALTEN" kann.

SUBROUTINE_BLOCK DELTA_t:SBR38
TITLE=UNTERPROGRAMM: 'DELTA_t'
!!ACHTUNG, IN DIESEM UP werden KEINE Sprungmarken benutzt!
!!Dieses Unterprogramm ist nur gemeinsam mit dem UP 'INIT_TIMER_ACK' lauffähig.

Im UP 'DELTA_t' wird für den im UP 'INIT_TIMER_ACK' permanent neu gestarteten 1ms-TIMER - 'T64' - die ZEITDIFFERENZ DELTA_t=[T(n+1)-T(n)] berechnet, die zwischen 2 Unterprogrammaufrufen 'n' und 'n+1'vergangen ist. 'DELTA_t' ist unterschiedlich, da die 1ms-TIMER im laufenden Zyklus aktualisiert werden. Gleichzeitig wird vom Unterprogramm der TIMER-Neuwert=T(n+1) in einem INT-Speicher, dessen Adresse an das Unterprogramm im 'AC3' übergeben wird, als TIMER-Altwert=T(n) abgespeichert.

```
!!An das Unterprogramm sind folgende Parameter zu übergeben
  Mit 'MOVD &VBxxxx,AC3' wird an das Unterprogramm die Adresse
                         eines Speicherbereiches übergeben.
!!Vom Unterprogramm werden folgende Werte zurückgeliefert:
  Im AC0 gibt das Unterprogramm die Zeitdifferenz im 'DINT-FORMAT' zurück
  ('DELTA_t' liegt immer im Bereich von  0 <= DELTA_t <= max ZYKLUSZEIT.)
                         und
  schreibt in das als Adresse übergebene Speicherwort den aktuellen TIMERSTAND.
  (Dieser TIMERSTAND ist dann für den nächsten Aufruf des UP der TIMER-ALTWERT)
!!Das Unterprogramm verändert nur 'AC0'!!
!!Ist SM0.1=1 -> FIRST_SCAN=HIGH: DELTA_t:=0!!
```

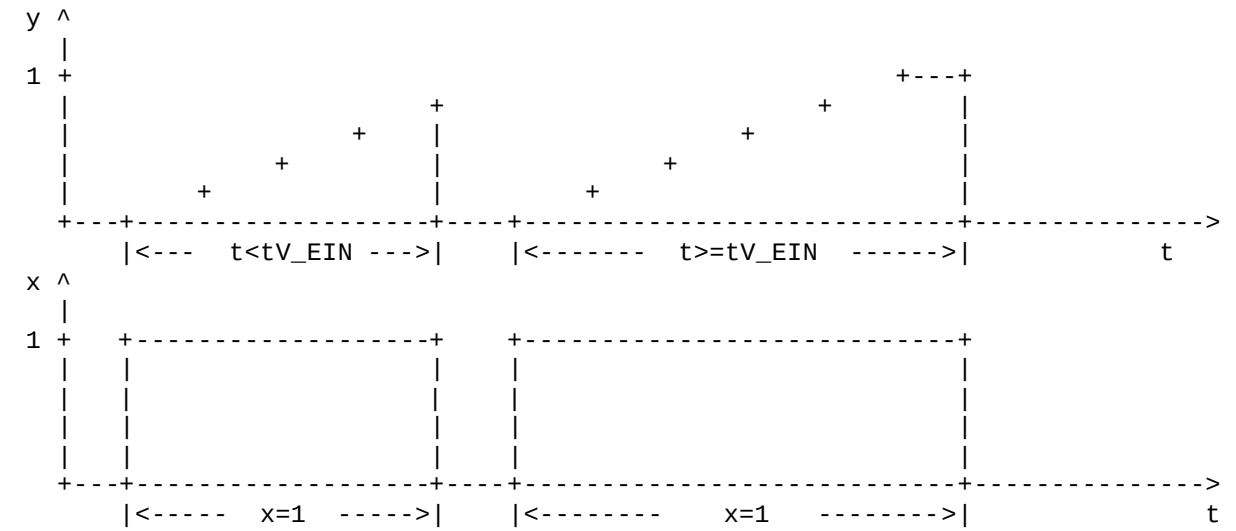
Beispiel: Die Speicherstelle für den TIMERISTWERT ist VW810. Mit

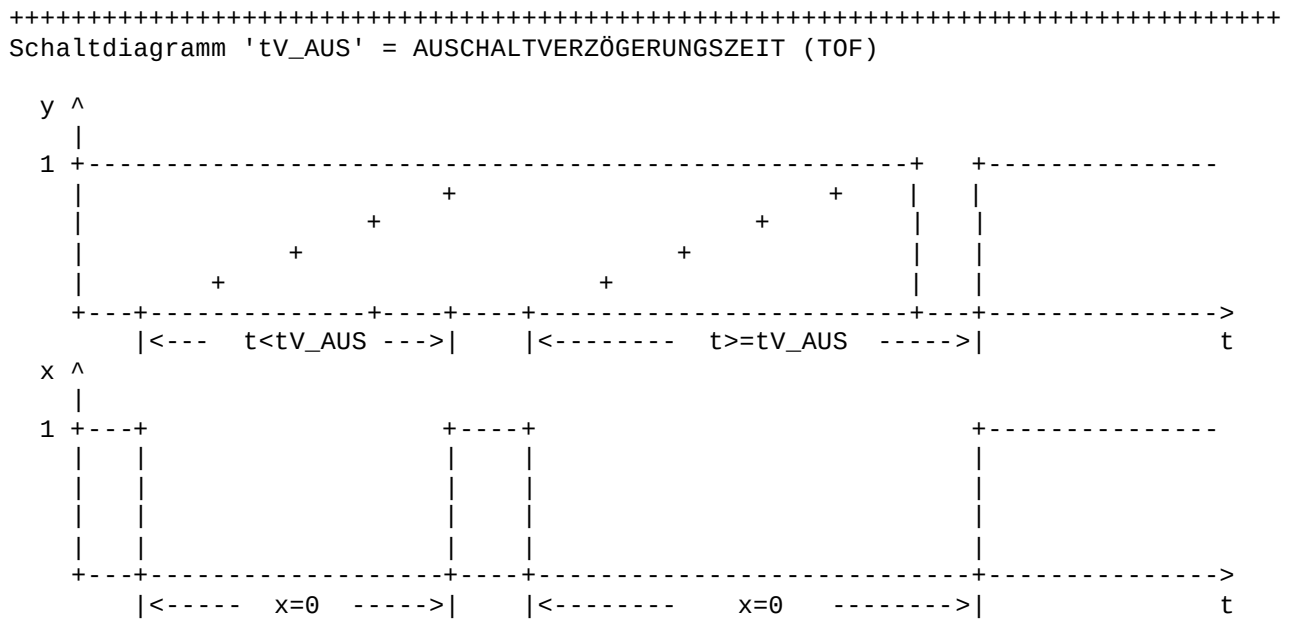
```
          LD SM 0.0
          MOVD &VW810,AC3
          CALL DELTA_t
```

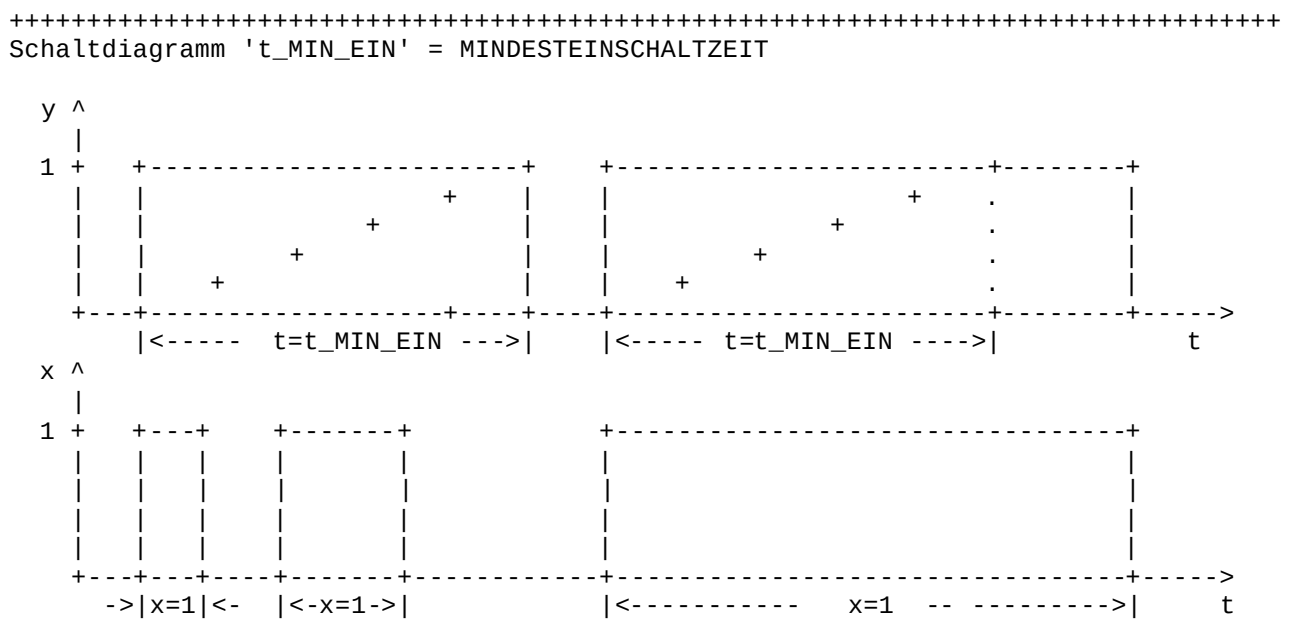
wird in VW810 der aktuelle TIMERSTAND T(n+1) des 'T64' gespeichert. Im 'AC0' wird Zeitdifferenz DELTA_t=[T(n+1)-T(n)] im DINT-Format zurückgeliefert.

SUBROUTINE_BLOCK TIMER_tD_ON_OFF:SBR39
TITLE=UNTERPROGRAMM: 'TIMER_tD_ON_OFF'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!
!!Dieses Unterprogramm ist nur gemeinsam mit dem UP 'INIT_TIMER_ACK'
!!und dem UP 'DELTA_t' lauffähig.

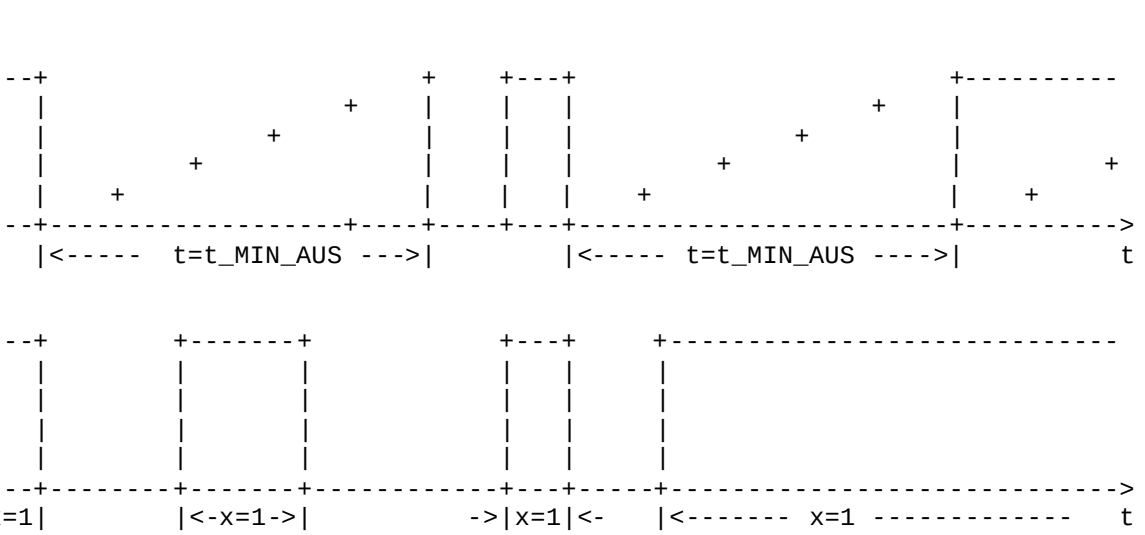
```
################################################################################
Kurzbeschreibung:
Das UP 'IMER_tD_ON_OFF' stellt einen TIMER dar, der als EINSCHALTVERZÖGERUNGSZEIT
ODER MINDESTAUSSCHALTZEIT UND AUSCHALTVERZÖGERUNGSZEIT ODER MINDESTEINSCHALTZEIT
parametriert werden kann. Die Zeiten können in MINUTEN UND SEKUNDEN als Fest-
kommazahl mit dem Faktor '*0.1' eingegeben werden.
################################################################################
in:
x           [BOOL]: =0/1  -> AKTIVIERUNG 'y' ANALOG MODE_tV
RST         [BOOL]: =1    -> y:=0 UND ALLE TIMER werden wie folgt zurückgesetzt:
                     tV_EIN:                          Der Zeitablauf beginnt neu
                     tV_AUS + tMIN_EIN + t_MIN_AUS:   Die Zeitabläufe werden sofort
                                                      beendet und gelten als abge-
                                                      schlossen
MODE_tV_EIN [BOOL]: Der TIMER hat 2 unterschiedliche Betriebsarten:
                     =0          -> tV_EIN (analog TON)
                     =1 ODER <>0 -> t_MIN_AUS = MINDESTAUSSCHALTZEIT
tV_EIN_s     [INT]: Einschalt-Verzögerung in SEKUNDEN   {0,1,...,32767}*0,1s
tV_EIN_m     [INT]: Einschalt-Verzögerung in MINUTEN    {0,1,...,32767}*0,1min
                     !!Fehleingabe tV<0 -> tV:=0
MODE_tV_AUS [BOOL]: Der TIMER hat 2 unterschiedliche Betriebsarten:
                     =0          -> tV_AUS (analog TOF)
                     =1 ODER <>0 -> t_MIN_EIN = MINDESTEINSCHALTZEIT
tV_AUS_s     [INT]: Ausschalt-Verzögerung in SEKUNDEN   {0,1,...,32767}*0,1s
tV_AUS_m     [INT]: Ausschalt-Verzögerung in MINUTEN    {0,1,...,32767}*0,1min
                     !!Fehleingabe tV<0 -> tV:=0
PRV_4BYTE   [DINT]: ADRESSE EINES 4-BYTE langen Speicherbereiches '&VBxxxx'
                     Diese Speicher werden UP-Intern benötigt und dürfen von
                     keinem anderen Programm nenutzt werden.

out:
y           [BOOL]: =0/1 als Funktion von 'x', der TIMER und 'RST'
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Schaltdiagramm 'tV_EIN' = EINSCHALTVERZÖGERUNGSZEIT (TON)
```




Schaltdiagramm 'tV_AUS' = AUSCHALTVERZÖGERUNGSZEIT (TOF)
y ^
1
t<tV_AUS
t>=tV_AUS
t
x ^
1
x=0
x=0
t


Schaltdiagramm 't_MIN_EIN' = MINDESTEINSCHALTZEIT
y ^
1
t=t_MIN_EIN
t=t_MIN_EIN
t
x ^
1
x=1
x=1
t

++
Schaltdiagramm 't_MIN_AUS' MINDESTAUSSCHALTZEIT



SUBROUTINE_BLOCK MUL_MINUS1_INT:SBR40
TITLE=UNTERPROGRAMM: 'MUL_MINUS1_INT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!
##
Kurzbeschreibung:
Das UP 'MUL_MINUS1_INT' multipliziert einen INTEGERWERT mit "-1". Dabei wird
das Produkt (-1)*(-32768) auf den WERT =+32767 gesetzt. Diese Multiplikation
wird zur Invertierung von Signalen benötigt.
##

in:
x [INT]: Eingangssignal {-32768,...,+32767}

out:
y [INT]: Ausgangssignal = x*(-1) {+32767,...,-32767}

!! Da die INTEGER-Multiplikation von (-32768)*(-1)= +32768 ist und damit einen
Überlauf des INT-Zahlenbereiches verursacht, wird UP-intern der negative Wert
'x=-32768' auf 'x:=-32767' gesetzt und dann mit '-1' multipliziert.

SUBROUTINE_BLOCK DATA_SWITCH_INT:SBR41
TITLE=UNTERPROGRAMM: 'DATA_SWITCH_INT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!
##
Kurzbeschreibung:
Mit dem UP 'DATA_SWITCH_INT' ist es möglich, abhängig vom Zustand einer binären
Speicherstelle einen von 2-Integerwerten auszuwählen.
##

in:
SW [BOOL]: =0 -> y:=x0 ODER =1 -> y:=x1
x0 [INT]: Integerwert {-32768,...,+32768}
x1 [INT]: Integerwert {-32768,...,+32768}
out:
y [INT]: x0 ODER x1

SUBROUTINE_BLOCK AH_SWITCH_INT:SBR42
TITLE=UNTERPROGRAMM: 'AH_SWITCH_INT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!

```
################################################################################
Kurzbeschreibung:
Mit dem UP 'AH_SWITCH_INT' kann vom HMI-System ein beliebiges INTEGER-Signal von
AUTO auf HAND über eine Tastfunktion und einem UP-internen FLIP-FLOP umgeschaltet
werden. Im Moment der Umschaltung ist das HAND- mit dem AUTOSIGNAL identisch und
kann dann frei gewählt werden. Die Begrenzung des HANDSIGNALES auf einen zulässi-
gen Wertebereich muß im HMI-System erfolgen.
In der Symboltabelle sind für alle AUTO-HAND-Umschalter 8BYTE lange Variablen-
Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der
IN- und OUTPUTS des AUTO-HAND-Umschalters benutzt werden müssen. Die Adresse der
Anfangsvariablen dieser Strukturen ist über den INPUT 'PRV8YTE' an das Unter-
programm zu übergeben. Alle mit '[W]' gekennzeichneten Variablen können wahlweise
im OB1, im Datenbaustein oder das HMI-System parametriert werden.
!!SIMATIC HMI = Human Machine Interfaces!!
################################################################################

in:
x         [INT]: DATEN = AUTO                        {-32768,...,+32767}
PRV8YTE  [DINT]: Anfangsadresse eines 8 BYTE langen Variablenspeicherberei-
                 ches, dessen Struktur in der Symboltabelle festgelegt ist.
                 Diese Struktur muß für jeden AUTO-HAND-Umschalter in der Sym-
                 boltabelle vorhanden sein. Weiter unten sind die einzelnen
                 INPUTS in diese Struktur näher erläutert.
                 Ist z.B. das erste Element dieser Struktur 'VW1400', so
                 ist der Input 'PRV8BYTE' mit '&VB1400' zu parametrieren.
out:
y         [INT]: DATEN AUTO ODER HAND              {-32768,...,+32767}
                 AH =0 -> y:=x
                 AH =1 -> y:=xAH

!!Im Automatikbetrieb, AH=0, gilt  'xAH:=x'.
!!Bei Umschaltung von AUTO nach HAND, AH=0->1, hat 'xAH' den Anfangswert 'x'.
!!Nach der Umschaltung, AH=1, ist 'xAH' beliebig vom HMI veränderbar.

################################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[RW]'      gekennzeichneten Parameter können im AUTOMATIKBETRIEB nur
                     gelesen und müssen im HANDBETRIEB über das HMI-System mit
                     Werten versorgt werden.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                     nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten para-
                     metriert werden und die Outputs dürfen Lokalvariablen sein.
                     Lokalvariable sind in den Netzwerken nur in aufsteigender
                     Richtung lesbar!
```

```
#############################################################################
Aufbau der Struktur für einen AUTO-HAND-Umschalter vom TYP 'AH_SWITCH_INT',
dessen Anfangsadresse 'VW1400' ist:
AH_1_1_x          VW1400  [PI]: xAUTO
AH_1_2_y          VW1402  [PO]: DATEN = AUTO/HAND, ANALOG FC-OUTPUT "y"
AH_1_3_xAH        VW1404  [RW]: AH=0 -> xAH:=xAUTO; AH=1-> xAH:=xHAND vom HMI
AH_1_4_TAST       V1406.0  [W]: UMSCHALTTASTER:  AUTO/HAND
AH_1_5_AH         V1406.1  [R]: FLIP-FLOP =0-> AUTO; =1-> HAND
AH_1_6_PRV_1BOOL  V1406.2  [R]: 1 BIT privater Speicher
#############################################################################
Für jeden AUTO-HAND-Umschalter sind das folgende Parameter vom Typ '[W]'/'[RW]':
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
xAH     [INT]: Je nach Schaltzustand des FLIP-FLOPs 'SG_AH' nimmt 'xAH' folgende
               Werte an:
                 AH=0 -> xAH:=x, vom HMI-System sind keine Eingaben möglich
                 AH=1 -> xAH hat den Anfangswert 'x' und kann vom HMI-System
                         verändert werden
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
TAST   [BOOL]: Softkey-Taste =0/1 vom HMI-System für FLIP-FLOP 'AH'
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

SUBROUTINE_BLOCK MIN_MAX_LIMIT:SBR43
TITLE=UNTERPROGRAMM: 'MIN_MAX_LIMIT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!

```
#############################################################################
Kurzbeschreibung:
Das UP 'MIN_MAX_LIMIT' begrenzt ein INTEGERSIGNAL auf eine frei parametrierbare
UNTER- UND OBERGRENZE.
#############################################################################

in:
x       [INT]: x                                {-32768,...,+32767}
yMIN    [INT]: MIN-Limit für y                  {-32768,...,+32767}
yMAX    [INT]: MAX-Limit für y                  {-32768,...,+32767}

out:
y       [INT]: Wenn  yMIN < x < yMAX,             dann y:=x
               Wenn  x <= yMIN, UND yMIN < yMAX,  dann y:=yMIN
               Wenn  x >= yMAX, UND yMIN < yMAX,  dann y:=yMAX
               Wenn  x beliebig UND yMIN = yMAX,  dann y:=yMIN=yMAX
```

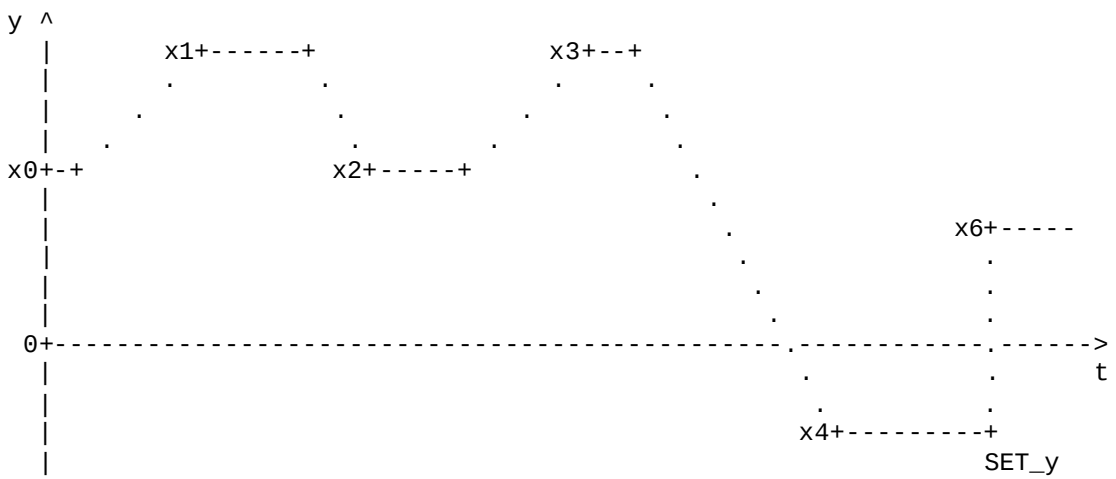
SUBROUTINE_BLOCK MIN_MAX_LIMIT_TOTBAND:SBR44
TITLE=UNTERPROGRAMM: 'MIN_MAX_LIMIT_TOTBAND'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!

```
###############################################################################
Kurzbeschreibung:
Das UP 'MIN_MAX_LIMIT_TOTBAND' begrenzt ein INTEGERSIGNAL=EINGANGSSIGNAL auf eine
frei parametrierbare UNTER- UND OBERGRENZE. Durch ein TOTBAND kann das AUSGANGS-
SIGNAL stabilisiert werden. Das Ausgangssignal folgt den Änderungen des Eingangs-
signales nur dann, wenn sich das Eingangssignal um einen größeren Wert ändert,
als der Betrag des TOTBANDES ist ODER das Eingangssignal die Unter-/Obergrenze
erreicht hat.
###############################################################################

in:
PRV_2BYTE   [DINT]: ADRESSE EINES 2-BYTE langen Speicherbereiches '&VBxxxx'
                    Diese Speicher werden UP-Intern benötigt und dürfen von
                    keinem anderen Programm benutzt werden.
x            [INT]: x                                  {-32768,...,+32767}
d            [INT]: d = Totband                             {0,...,+32767}
                    !! Ist d<0, dann UP-intern d:=0
yMIN         [INT]: MIN-Limit für y                    {-32768,...,+32767}
yMAX         [INT]: MAX-Limit für y                    {-32768,...,+32767}

out:
y    [INT]: Wenn [x>(y+d) ODER x<(y-d)] UND [yMIN < x < yMAX], dann y:=x
            Wenn  x <= yMIN,            UND     [yMIN < yMAX], dann y:=yMIN
            Wenn  x >= yMAX,            UND     [yMIN < yMAX], dann y:=yMAX
```

SUBROUTINE_BLOCK RAMP_INT:SBR45
TITLE=UNTERPROGRAMM: 'RAMP_INT'
!!ACHTUNG, IN DIESEM UP wird die Sprungmarke 240 benutzt!
!!Dieses Unterprogramm ist nur gemeinsam mit dem UP 'INIT_TIMER_ACK'
!!und dem UP 'DELTA_t' lauffähig.

```
###############################################################################
Kurzbeschreibung:
Das UP 'RAMP_INT' führt permanent den momentanen SOLLWERT eines INTEGER-SIGNALES
mit einer vorgebbaren RAMPE an den berechneten ISTWERT dieses Signales heran.
Z.B. wird ein berechnetes Reglerausgangssignal mit einer Rampe auf den Analog-
ausgang für einen Frequenzumformer eines Antriebes abgebildet.Die Steilheit
der Rampen für das Inkrementieren und Dekrementieren ist unterschiedlich para-
metrierbar.
###############################################################################

in:
SET_y     [BOOL]: =1 -> y:=x ohne Rampenfunktion!
                  !!UP-intern wird der SM0.1 = FIRST_SCAN ausgewertet.
                    Für den 1.OB1-Zyklus gilt: 'y:=x' ohne Rampenfunktion!
RMP_UP     [INT]: Inkremente pro Sekunde, wenn 'x>y'   {1,2,...,32767}*0.1*INC/s
                  !!RMP_UP   <1 wird UP-intern korrigiert: RMP_UP  :=1
RMP_DOWN   [INT]: Dekremente pro Sekunde, wenn 'x<y'   {1,2,...,32767}*0.1*DEC/s
                  !!RMP_DOWN <1 wird UP-intern korrigiert: RMP_DOWN:=1
x          [INT]: SOLLWERT der INT-Variablen       {-32768,...,+32767}
PRV6BYTE  [DINT]: ADRESSE EINES 6-BYTE langen Speicherbereiches '&VBxxxx'
                  Diese Speicher werden UP-Intern benötigt und dürfen von
                  keinem anderen Programm nenutzt werden.
out:
y          [INT]: ISTWERT der INT-Variablen        {-32767,...,+32767}
                  'y' wird mit einer RAMPE an den Wert 'x' angeglichen.
```

!!Alle Eingabewerte werden im UP 'RAMP_INT' auf die angegebenen Grenzen geprüft.
  Sind Eingaben fehlerhaft, so werden sie auf die zulässige Untergrenze=UG bzw.
  Obergrenze=OG UP-intern korrigiert.
##
Beispiel:



SUBROUTINE_BLOCK DAMP_INT:SBR46
TITLE=UNTERPROGRAMM: 'DAMP_INT'
!!ACHTUNG, IN DIESEM UP wird keine Sprungmarke benutzt!
!!Dieses Unterprogramm ist nur gemeinsam mit dem UP 'INIT_TIMER_ACK'
!!und dem UP 'DELTA_t' lauffähig.
##
Kurzbeschreibung:
Das UP 'DAMP_INT' dämpft ein INTEGER-SIGNAL, z.B. ein Analogeingangssignal,
durch Mittelwertbildung über 'n'{1,2,...20} Werte. Für die Mittelwertbildung
wird jedesmal nach Ablauf eines Abtastintervalles der momentane Wert des Sig-
nales einem FIFO gespeichert. Dabei wird vor der Berechnung des Mittelwertes
stets das älteste Signal aus dem FIFO heraus- und das neue Signal in das FIFO
hineingeschoben. Die ZEIT für das ABTASTINTERVALL ist frei parametrierbar.
##
in:
n   [INT]: Anzahl der INTEGERSIGNALE   {1,2,...,20}
                  die zur Berechnung des Mittelwertes im FIFO gespeichert werden.
                  Für n=1 -> y=x!
                  Fehleingabe: n<1 -> n:=1 ODER n>20 -> n:=20
tABTAST   [INT]: Abtastintervall-Zeit   {0,1,...,+32767}*1ms
                  nach deren Ablauf ein neues Signal im FIFO abgelegt wird.
                  Fehleingabe: tABTAST<0 -> tABTAST:=0
                  !!Wird der FC 'DAMP_INT' in einem Interrupt aufgerufen und soll
                    dessen Zeitintervall als ABTASTZEIT wirksam sein, so ist
                    tABTAST=0 zu wählen.
x   [INT]: zu dämpfendes INTEGERSIGNAL   {-32768,...,+32767}

PRV46BYTE [DINT]: ADRESSE EINES 46-BYTE langen Speicherbereiches '&VBxxxx'
                  Diese Speicher werden UP-Intern benötigt und dürfen von
                  keinem anderen Programm nenutzt werden. Die Struktur dieser
                  Speicher ist nachstehend beschrieben.
out:
y   [INT]: MITTELWERT aus den FIFO-Inputs   {-32767,...,+32767}

!!Alle Eingabewerte werden im UP 'RAMP_INT' auf die angegebenen Grenzen geprüft.
  Sind Eingaben fehlerhaft, so werden sie auf die zulässige Untergrenze=UG bzw.
  Obergrenze=OG UP-intern korrigiert.
##
Aufbau der Struktur für einen Sollwertgeber, dessen Anfangsadresse 'VW1600' ist:
Ist die Anfangsadresse dieser Struktur VW2000, su muß PRV46BYTE mit '&VB2000'
parametriert werden:
!!Der Aufbau dieser Struktur dient nur zur Information, denn die Daten sind aus-
  nahmslos vom Typ '[R]', d.h. sie können nur gelesen werden.

```
DAMP_01_05_t_ALT         VW2000  [R]: Altwert TIMER
DAMP_01_06_TIMER         VD2002  [R]: TIMER
DAMP_01_07_FIFO_SP01     VW2006  [R]: FIFO SPEICHER 01
DAMP_01_08_FIFO_SP02     VW2008  [R]: FIFO SPEICHER 02
DAMP_01_09_FIFO_SP03     VW2010  [R]: FIFO SPEICHER 03
DAMP_01_10_FIFO_SP04     VW2012  [R]: FIFO SPEICHER 04
DAMP_01_11_FIFO_SP05     VW2014  [R]: FIFO SPEICHER 05
DAMP_01_12_FIFO_SP06     VW2016  [R]: FIFO SPEICHER 06
DAMP_01_13_FIFO_SP07     VW2018  [R]: FIFO SPEICHER 07
DAMP_01_14_FIFO_SP08     VW2020  [R]: FIFO SPEICHER 08
DAMP_01_15_FIFO_SP09     VW2022  [R]: FIFO SPEICHER 09
DAMP_01_16_FIFO_SP10     VW2024  [R]: FIFO SPEICHER 10
DAMP_01_17_FIFO_SP11     VW2026  [R]: FIFO SPEICHER 11
DAMP_01_18_FIFO_SP12     VW2028  [R]: FIFO SPEICHER 12
DAMP_01_19_FIFO_SP13     VW2030  [R]: FIFO SPEICHER 13
DAMP_01_20_FIFO_SP14     VW2032  [R]: FIFO SPEICHER 14
DAMP_01_21_FIFO_SP15     VW2034  [R]: FIFO SPEICHER 15
DAMP_01_22_FIFO_SP16     VW2036  [R]: FIFO SPEICHER 16
DAMP_01_23_FIFO_SP17     VW2038  [R]: FIFO SPEICHER 17
DAMP_01_24_FIFO_SP18     VW2040  [R]: FIFO SPEICHER 18
DAMP_01_25_FIFO_SP19     VW2042  [R]: FIFO SPEICHER 19
DAMP_01_26_FIFO_SP20     VW2044  [R]: FIFO SPEICHER 20
```
##

Berechnungsgrunlagen:
Die INTEGERSIGNALE werden in einem FIFO so gespeichert, daß immer der letzte
Wert heraus- und der NEUWERT des INTEGERSIGNALES als Erstwert in das FIFO hi-
neingeschoben wird. Dann wird aus den 'n'-Werten der Mittelwert 'y' berechnet.
Ein neues INTEGERSIGNAL wird jedoch immer nur dann im im FIFO abgelegt, wenn
die ABTASTINTERVALLZEIT abgelaufen ist.
Ist 'FIRST_SCAN'=TRUE, so werden 'n' Elemente des FIFO's mit dem momentan
anliegenden INTEGERSIGNAL 'x' aufgefüllt. Für den Mittelwert gilt dann 'y=x'!

Der Mittelwert wird wie folgt berechnet

$$y = \frac{\sum_{k=1}^{n} x(k)}{n} \quad ; \qquad n\{1,2,...,20\}$$

SUBROUTINE_BLOCK SWG_INT:SBR47
TITLE=UNTERPROGRAMM: 'SWG_INT'
!!ACHTUNG, IN DIESEM UP wird die Sprungmarke 241 benutzt!

```
################################################################################
Kurzbeschreibung:
Das UP 'SWG_INT' stellt eine lineare Zuordnung zwischen der Führungsgröße 'x'
und dem Sollwert 'y' in Form von Geradengleichungen 'y=a*x+b' her. UP-intern
wird für jeden Wert 'x' ein Punktpaar Pi UND P[i+1] aus den n{1,2,...,10} vorge-
gebenen Stützpunkten bestimmt, für das gilt 'xi <= x <= x(i+1)'.Aus diesen beiden
Punkten werden die Konstanten 'a' und 'b' der Geradengleichung berechnet und
dann kann für jeden Wert 'x' der zugeordnete Wert 'y' ermittelt werden, wenn 'x'
im Bereich dieser Stützpunkte liegt.
In der Symboltabelle sind für alle Sollwertgeber 38BYTE lange Variablen-
Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der
IN- und OUTPUTS benutzt werden müssen. Die Adresse der Anfangsvariablen dieser
Strukturen ist über den INPUT 'PRV48BYTE' an das Unterprogramm zu übergeben. Al-
le mit '[W]' gekennzeichneten Variablen können wahlweise im OB1, im Datenbaustein
oder das HMI-System parametriert werden.
Es kann eine Sollwertführung durch eine aus bis aus 10 Geradenabschnitten be-
stehende Kurve (Ein typischer Einsatzfall ist die in Abhängigkeit von der Außen-
temperatur geführte Vorlauftemperatur einer Warmwasserheizung = Heizkurve. Die
Heizkurve wird durch Aufteilung in Geradenabschnitte linearisiert!) realisiert
werden. Der Anfangs- und Endpunkt der Geradenstützpunkte stellt gleichzeitig
einen Grenzwert für den Sollwert dar.
!!SIMATIC HMI = Human Machine Interfaces!!
################################################################################

in:
x          [INT]: Führungsgröße      = Abszisse Punkt P(x,y)  {-32768,...,+32767}

PRV48BYTE [DINT]: Anfangsadresse eines maximal 48 BYTE langen Variablenspeicher-
                  bereiches, dessen Struktur in der Symboltabelle festgelegt
                  ist. Diese Struktur muß für jeden Sollwertgeber-INT in der
                  Symboltabelle vorhanden sein. Weiter unten sind die einzel-
                  nen INPUTS in diese Struktur näher erläutert. Ist z.B. das
                  erste Element dieser Struktur 'VW1600', so muß der Input
                  'PRV48BYTE' mit '&VB1600' zu parametriert werden.
                  Die Anzahl der vom 'UP' benutzten Speicher ist von 'n' abhängig.
                  Sie können wie folgt berechnet werden:
                       ANZAHL PRIVATE SPEICHER = (n*4 + 8)!

out:
OK        [BOOL]: OK:=1; PARAMETRIERFEHLER: n<2 ODER n>10 ODER xi>=x(i+1) -> OK:=0
y          [INT]: Geführter Sollwert = Ordinate Punkt P(x,y)  {-32768,...,+32767}

!!Alle Eingabewerte der Parameter des UP 'SWG_INT' werden auf die angegebenen
  Grenzen geprüft. Sind Eingaben fehlerhaft, so werden OK:=0 !
################################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                     nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten pa-
                     rametriert werden und die Outputs dürfen Lokalvariablen sein.
```

Lokalvariable sind in den Netzwerken nur in aufsteigender Richtung lesbar!

##

Aufbau der Struktur für einen Sollwertgeber, dessen Anfangsadresse 'VW1600' ist:

SWG_INT_1_01_x	VW1600	[PI]:Für BuB, identisch mit Input 'x'
SWG_INT_1_02_ok	V1602.0	[PO]:Für BuB, identisch mit Output 'y'
SWG_INT_1_03_y	VW1604	[PO]:Für BuB, identisch mit Output 'y'
SWG_INT_1_04_n	VW1606	[W]: Anzahl der benutzten Stützpunkte {2,3,...,10}
SWG_INT_1_05_x1	VW1608	[W]: Abszisse P1 {-32768,...,+32767}
SWG_INT_1_06_y1	VW1610	[W]: Ordinate P1 {-32768,...,+32767}
SWG_INT_1_07_x2	VW1612	[W]: Abszisse P2 {-32768,...,+32767}
SWG_INT_1_08_y2	VW1614	[W]: Ordinate P2 {-32768,...,+32767}
SWG_INT_1_09_x3	VW1616	[W]: Abszisse P3 {-32768,...,+32767}
SWG_INT_1_10_y3	VW1618	[W]: Ordinate P3 {-32768,...,+32767}
SWG_INT_1_11_x4	VW1620	[W]: Abszisse P4 {-32768,...,+32767}
SWG_INT_1_12_y4	VW1622	[W]: Ordinate P4 {-32768,...,+32767}
SWG_INT_1_13_x5	VW1624	[W]: Abszisse P5 {-32768,...,+32767}
SWG_INT_1_14_y5	VW1626	[W]: Ordinate P5 {-32768,...,+32767}
SWG_INT_1_15_x6	VW1628	[W]: Abszisse P6 {-32768,...,+32767}
SWG_INT_1_16_y6	VW1630	[W]: Ordinate P6 {-32768,...,+32767}
SWG_INT_1_17_x7	VW1632	[W]: Abszisse P7 {-32768,...,+32767}
SWG_INT_1_18_y7	VW1634	[W]: Ordinate P7 {-32768,...,+32767}
SWG_INT_1_19_x8	VW1636	[W]: Abszisse P8 {-32768,...,+32767}
SWG_INT_1_20_y8	VW1638	[W]: Ordinate P8 {-32768,...,+32767}
SWG_INT_1_21_x9	VW1640	[W]: Abszisse P9 {-32768,...,+32767}
SWG_INT_1_22_y9	VW1642	[W]: Ordinate P9 {-32768,...,+32767}
SWG_INT_1_23_x10	VW1644	[W]: Abszisse P10 {-32768,...,+32767}
SWG_INT_1_24_y10	VW1646	[W]: Ordinate P10 {-32768,...,+32767}

##

Für jeden Sollwertgeber sind folgende Parameter vom Typ '[W]':

++

n [INT]: Anzahl der benutzten Stützpunkte {2,...,10}
n<2 ODER n>10 -> ERROR: y:=0 UND OK-BIT:=0

++

Für jeden Stützpunkt Pi i {1,2,...,10}
Die Koordinaten (xi; yi):

x1	[INT]:	Punkt P1 = Abszisse	{-32768,...,+32767}
y1	[INT]:	Punkt P1 = Ordinate	{-32768,...,+32767}
x2	[INT]:	Punkt P2 = Abszisse	{-32768,...,+32767}
y2	[INT]:	Punkt P2 = Ordinate	{-32768,...,+32767}
:		:	:
:		:	:
x10	[INT]:	Punkt P10 = Abszisse	{-32768,...,+32767}
y10	[INT]:	Punkt P10 = Ordinate	{-32768,...,+32767}

Um jedem Wert 'x' einen eindeutigen Wert 'y' zuordnen zu können, muß für alle verwendeten Stützpunkte folgende Bedingungen eingehalten werden:
x1 < x2 < ...< xi; i{1,2,...,n}
Wird diese Bedingung nicht eingehalten -> ERROR: y:=0 UND OK-BIT:=0
(Die 'yi' können beliebige Werte annehmen)

++

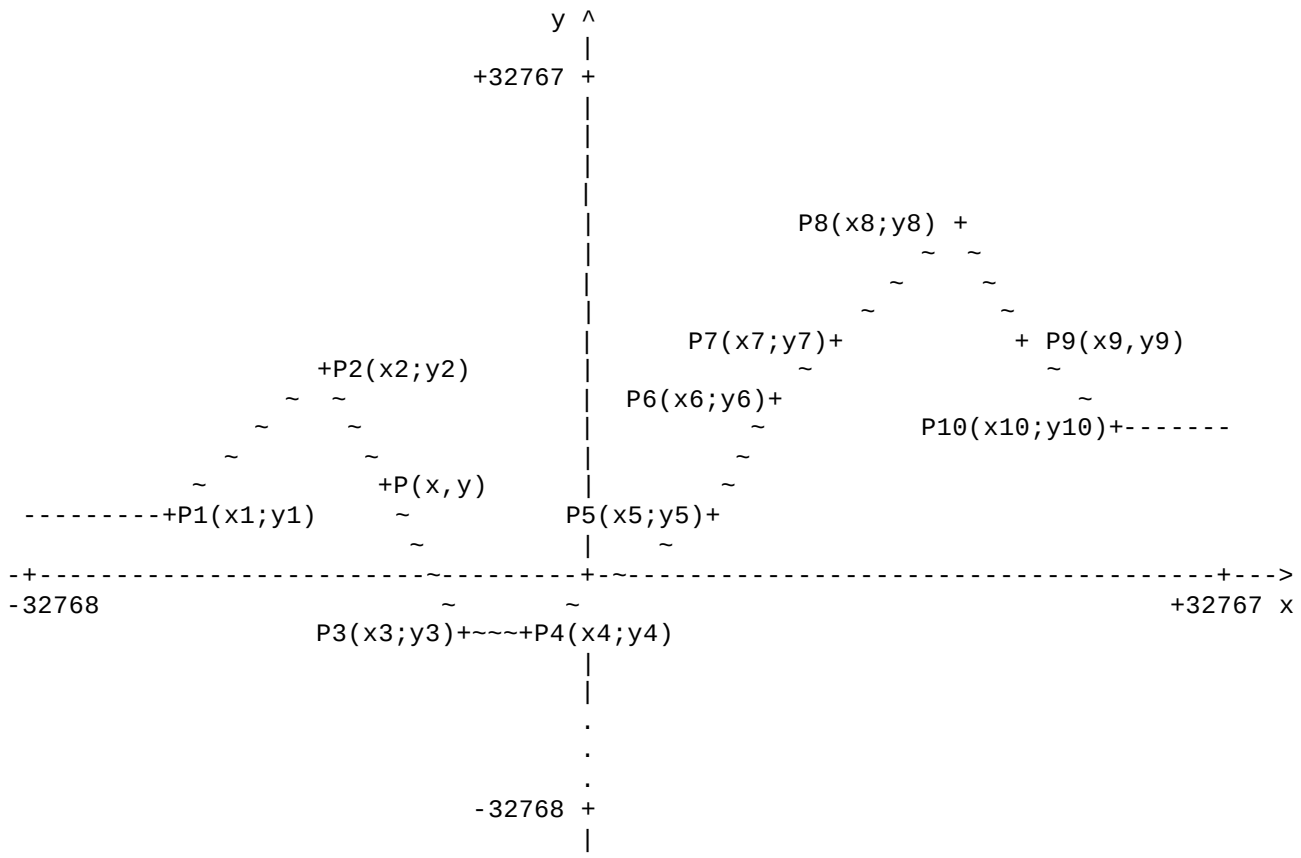
Berechnungsgrundlagen:

In diesem UP wird eine lineare Zuordnung zwischen der Führungsgröße 'x' und dem Sollwert 'y' in Form der Geradengleichung 'y=a*x+b' hergestellt. UP-intern wird für jeden Wert 'x' ein Punktpaar Pi UND P[i+1] aus den 'n' Stützpunkten, bestimmt, für das gilt 'xi <= x <= x(i+1)'. Aus diesen beiden Punkten werden die Konstanten 'a' und 'b' der Geradengleichung berechnet und dann kann für jeden Wert 'x' der zugeordnete Wert 'y' ermittelt werden, wenn 'x' im o.g. Bereich liegt.

'y' wird für den Wert 'x' im Bereich 'xi <= x < x(i+1)' wie folgt berechnet:

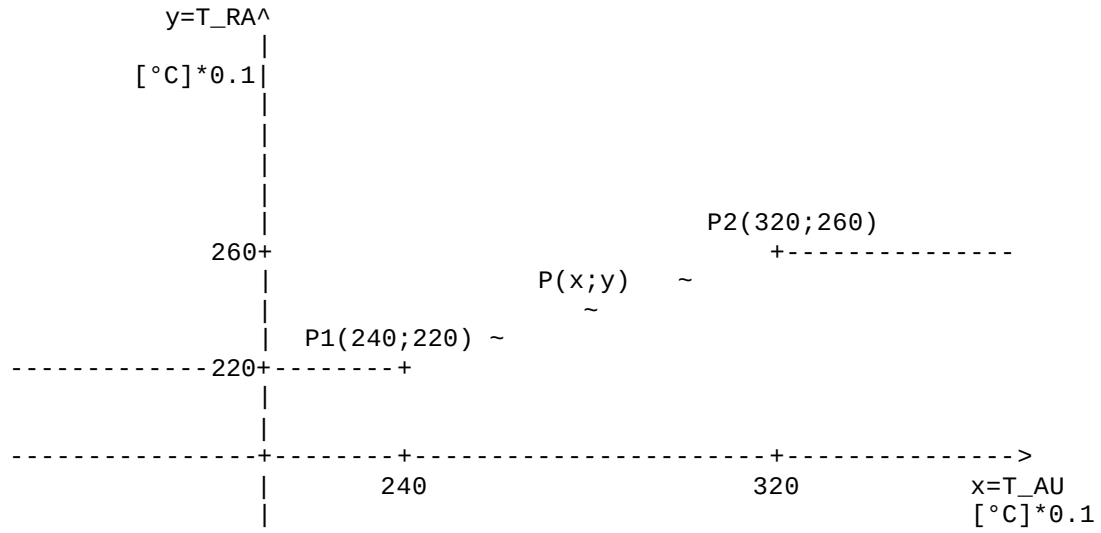
$$y = \frac{[y(i+1)-yi]}{[x(i+1)-xi]} * (x-xi) + yi$$

Für alle x<x1 -> y:=y1
Für alle x>xn -> y:=yn; n{2,3,...,10}
Gilt yi = y(i+1) -> y:=yi für alle x im Bereich 'xi <= x < x(i+1)'
Wird die Bedingung 'x1<x2<...<xn' verletzt -> ERROR, y:=0 UND BIE-BIT:=0

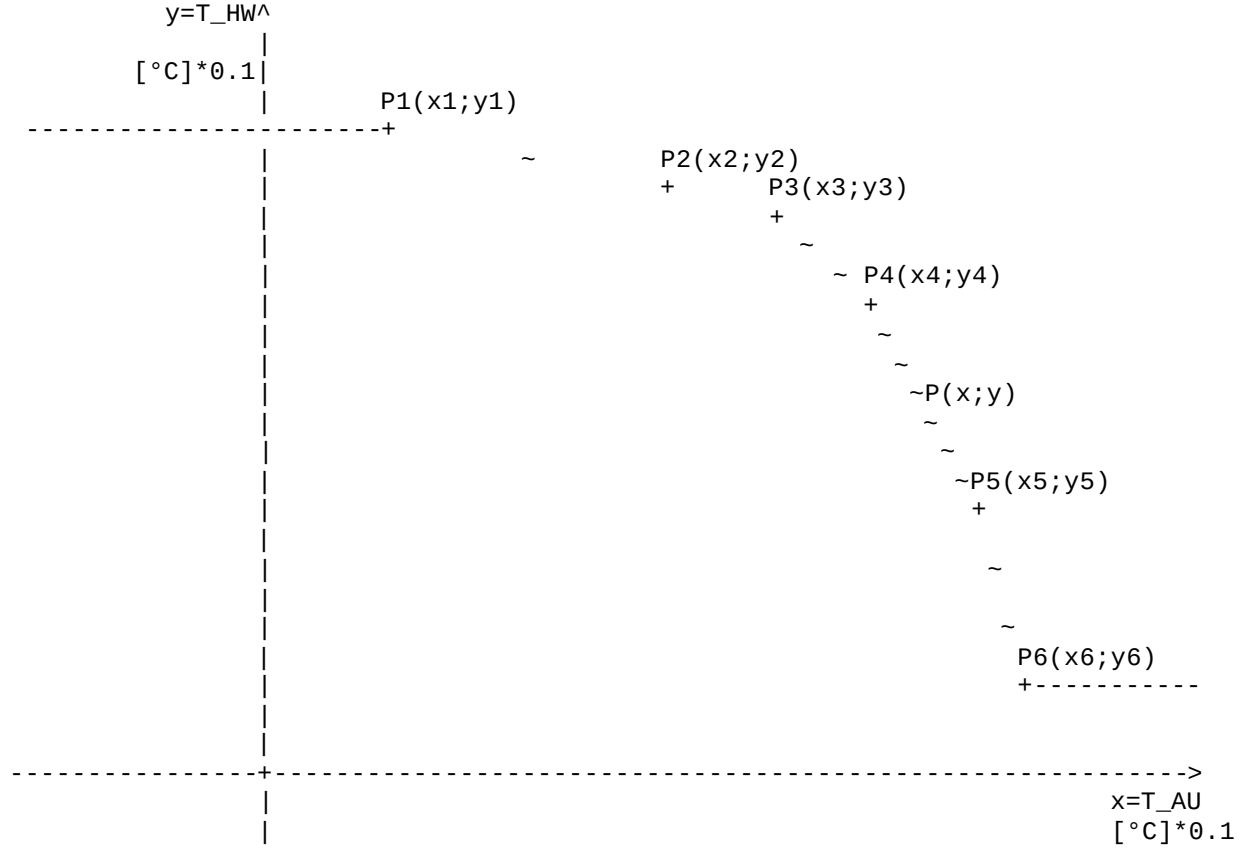


BEISPIELE:

1.Anhebung der Raumtemperatur in Abhängigkeit von der Außentemperatur: n=2
(Bei hohen Außentemperaturen wird in klimatisierten Arbeitsräumen die Raumtemperatur in abhängigkeit von der Außentemperatur angehoben!)



2.Außentemperaturabhängige Führung der Heißwassertemperatur (Heizkurve): n=6
(Die Heißwassertemperatur für Raumheizungen wird mit sinkender Außentemperatur nach einer Heizkurve angehoben. Diese Heizkurve kann mit ausreichender Genauigkeit durch einen Polygonzug linearisiert werden !)



SUBROUTINE_BLOCK SWG_TIME:SBR48
TITLE=UNTERPROGRAMM: 'SWG_TIME'
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken 242 bis 248 benutzt!
!!Dieses Unterprogramm ist nur gemeinsam mit dem UP 'INIT_TIMER_ACK'
!!und dem UP 'DELTA_t' lauffähig.

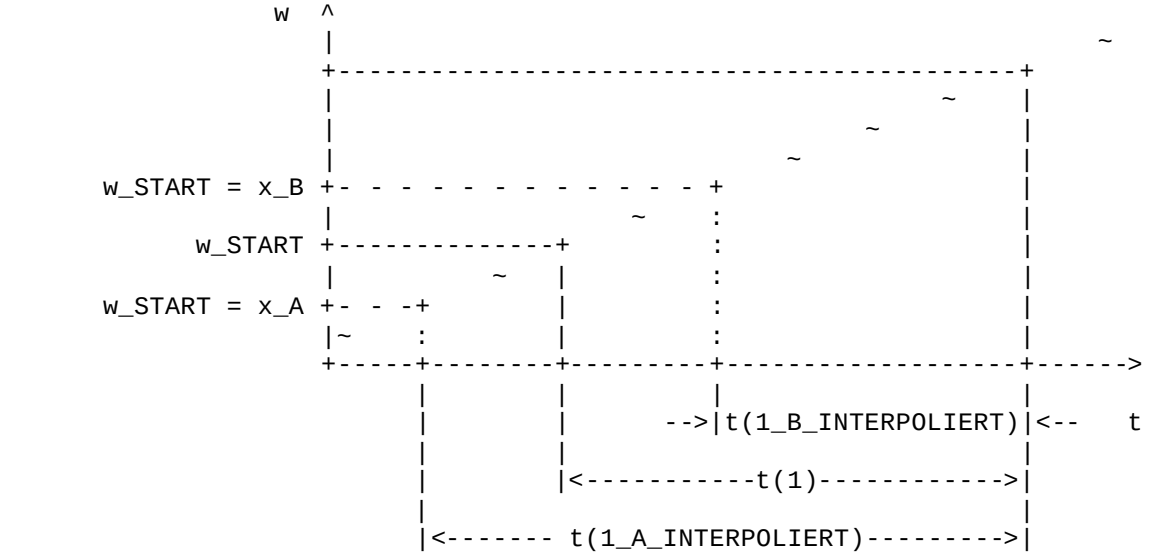
```
###############################################################################
Kurzbeschreibung:
Das UP 'SWG_TIME' stellt eine frei parametrierbare Sollwert-Zeitplanführung
= 'ZPF' dar. In der Symboltabelle sind für alle 'ZPF' 200BYTE lange Variablen-
Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der
IN- und OUTPUTS benutzt werden müssen. Die Adresse der Anfangsvariablen dieser
Strukturen ist über den INPUT 'PRV200BYTE' an das Unterprogramm zu übergeben.
Alle mit '[W]' gekennzeichneten Variablen können wahlweise im OB1, im Datenbau-
stein oder das HMI-System parametriert werden.
Die 'ZPF' besteht aus maximal 10 Sollwertstützpunkten = 'SSP'. Zwischen 2 'SSP'
wird der Sollwert zeitabhängig nach einer Geradengleichung 'w = a*t + b' berech-
net. Die 'Fahrkurve' des Sollwertes besteht also aus 10 Geradenabschnitten. Beim
Umschalten von einem 'SSP' auf den nächsten oder am Ende der 'ZPF' werden für sta-
tistische Auswertungen, separat für jeden 'SSP', der Istwert der Regelgröße, die
benötigte Zeit und ein Zeitstempel im Format 'TTMMhhmm' abgespeichert. Alle für
die 'ZPF' benötigten Parameter werden im INTEGER-FORMAT eingegeben. Die Zeitein-
gaben sind im Festkommazahlen mit der Maßeinheit [0.01h] oder wahlweise [1.0s].
Damit ist die maximal eingebbare Zeit für jeden 'SSP' = 327.67[h] ODER 3276.7[s].
Die Zeitplanführung unterscheidet 2 Betriebsarten:
  1. Der Sollwert wird ohne Rücksicht auf den vorhandenen Istwert geführt.
  2. Ist der geführte Sollwert 'w' dem Istwert der Regelgröße 'x' um einen
     vorgebbaren Wert, 'MAX_DRIFT', vorausgeeilt, dann wird die Sollwertfüh-
     rung solange eingefroren, bis die Abweichung '|x-w|<=MAX_DRIFT' ist.
Mit der Funktion 'FREEZE' kann die Sollwertführung über eine Tastfunktion vom
HMI-System angehalten und wieder freigegeben werden. Ebenso sind die Funktionen
'START' und 'RESET' über Taster vom HMI-Sytem bedienbar. Das UP 'SWG_TIME' hat
ebenfalls die Inputs 'FREEZE', 'START' und 'RESET'. Damit kann auch vom Programm
 die Zeitplanführung gesteuert werden.
Die Zeitplanführung von Sollwerten ist für die Temperaturregelung von Brennöfen
und biologischen Prozessen von großer Bedeutung.
!!SIMATIC HMI = Human Machine Interfaces!!
###############################################################################

in:
START       [BOOL]: =1 mit UP-internem PLS(+) wird der Zeitplangeber gestartet.
                    Zum Zeitpunkt des Startes werden alle ISTWERTSPEICHER
                    gelöscht.
FREEZE      [BOOL]: =1 der momentane Zustand des Zeitplangebers wird eingefro-
                    ren, die Zeitabläufe werden gestoppt und der im Moment
                    berechnete Sollwert wird permanent ausgegeben.
                    !!FREEZE hat Vorrang vor START
RESET       [BOOL]: =1 der Zeitplangeber wird permanent zurückgesetzt und kann
                    mit 'START=1' nur dann neu gestartet werden, wenn der
                    Input 'RESET=0' ist. (Das alternative RESET vom HMI-Sy-
                    stem mit dem Parameter 'TAST_RESET' setzt den Zeitplan-
                    geber nur mit einem PULS(+) zurück!). Mit 'RESET=1' wird
                    'START:=0' UND 'n_AKT':=0', die ISTWERTE der vorangegan-
                    genen Zeitplanführung werden jedoch nicht gelöscht.
                    !!RESET  hat Vorrang vor FREEZE
!!Vom HMI-Sytem lassen sich über Eingaben in die V-Speicher ebenfalls die Steuer-
  kommandos 'START', 'FREEZE' und 'RESET' realisieren. Sie sind gleichrangig zu
  den gleichnamigen UP-Inputs und schließen sich nicht gegeneinander aus.
```

```
MOD_w_START  [BOOL]: =0 -> START der Zeitplanführung mit 'w:=w_START'
                     =1 -> START der Zeitplanführung mit 'w:=x'
MOD_ZPF      [BOOL]: MODUS DER    ZEITPLANFÜHRUNG=ZPF
                     =0 -> die Sollwertführung erfolgt unabhängig vom Istwert.
                           Nach Ablauf der Zeit 't' des aktuellen Stützpunktes
                           SSP[n] ist der Zielsollwert 'w' erreicht und es wird
                           auf den nächste Stützpunkt umgeschaltet.
                     =1 -> die Sollwertführung wird sofort angehalten, wenn der
                           Betrag der Abweichung zwischen Ist- und Sollwert grös-
                           ser oder gleich 'MAX_DRIFT' ist. Ebenso erfolgt die
                           Umschaltung auf den nächsten Stützpunkt nur dann, wenn
                           der Zielsollwert 'w' ererreicht ist UND die Abweichung
                           '|x-w|<MAX_DRIFT' ist.
MOD_t        [BOOL]: =0 -> alle Zeiteingaben haben die Maßeinheit          0.01*h
                     =1 -> alle Zeiteingaben haben die Maßeinheit           0.1*s
x             [INT]: Istwert der Regelgröße            {-32767,...,+32767}*10^m
                     !!Die Rückführung der Regelgröße 'x' ist für 'MOD_ZPF=1'
                       zwingend notwendig. Bei 'MOD_ZPF=0' UND 'MOD_w_START=0'
                       wird 'x' nur für statistische Auswertungen erfaßt.
PRV200BYTE   [DINT]: Anfangsadresse eines maximal 200 BYTE langen Variablenspei-
                     cherbereiches, dessen Struktur in der Symboltabelle festgelegt
                     ist. Diese Struktur muß für jeden 'ZPF' in der Symboltabelle
                     vorhanden sein. Weiter unten sind die einzelnen INPUTS in die-
                     se Struktur näher erläutert. Ist z.B. das erste Element dieser
                     Struktur 'VW1800', so muß der Input 'PRV200BYTE' mit '&VB1800'
                     parametriert werden. Die Anzahl der vom 'UP' benutzten Spei-
                     cher ist von 'n' abhängig.
                     Sie können wie folgt berechnet werden:
                        ANZAHL PRIVATE SPEICHER = (n*14 + 60)!
out:

OK           [BOOL]: OK:=1; PARAMETRIERFEHLER: n<2 ODER n>10 -> OK:=0
y             [INT]: AKTIVER SOLLWERT 'w(n)' DES 'SSP[n]' {-32768,...,+32767}*10^m

################################################################################
!!Alle Eingabewerte in das UP 'SWG_TIME' werden auf die angegebenen Grenzen über-
  prüft. Sind Eingaben fehlerhaft, so werden sie auf die zulässige Untergrenze=UG
  bzw. Obergrenze=OG UP-intern korrigiert. Fehleingaben von 'n' -> OK:=0 !
################################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                     nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten pa-
                     rametriert werden und die Outputs dürfen Lokalvariablen sein.
                     Lokalvariable sind in den Netzwerken nur in aufsteigender
                     Richtung lesbar!
################################################################################
```

Aufbau der Struktur für eine 'ZPF', deren Anfangsadresse 'VW1800' ist:

```
SWGT_1_01_START         V1800.0 [PI]:=1 -> START   'ZPF'
SWGT_1_02_FREEZE        V1800.1 [PI]:=1 -> FREEZE  'ZPF'
SWGT_1_03_RESET         V1800.2 [PI]:=1 -> RESET   'ZPF
SWGT_1_04_MOD_w_START   V1800.3 [PI]:MODUS DES STARTSOLLWERTES
SWGT_1_05_MOD_ZPF       V1800.4 [PI]:MODUS DER ZEITPLANFÜHRUNG=ZPF
SWGT_1_06_MOD_t         V1800.5 [PI]:MODUS DER ZEITEINGABEN: 0.01*h / 0.1*s
SWGT_1_07_x             VW1802  [PI]:ISTWERT DER REGELGRÖßE
SWGT_1_08_OK            V1804.0 [PO]:PARAMETRIERFEHLER  -> OK:=0
SWGT_1_09_y             VW1806  [PO]:y = SOLLWERT 'w(n)' DES  'SSP[n]'
SWGT_1_10_n             VW1808  [W]: ANZAHL BENUTZTER 'SSP'  {1,...,10}
SWGT_1_11_w_START       VW1810  [W]: STARTSOLLWERT
SWGT_1_12_MAX_DRIFT     VW1812  [W]: MAX |x-w| >0 FÜR MOD_ZPF=1
SWGT_1_13_TAST_START    V1814.0 [W]: HMI = TASTER-START   ZPF mit PLS(+)
SWGT_1_14_TAST_FREEZE   V1814.1 [W]: HMI = TASTER-FREEZE ZPF mit FLIP-FLOP
SWGT_1_15_TAST_RESET    V1814.2 [W]: HMI = TASTER-RESET  ZPF  mit PLS(+)
SWGT_1_16_INFO_START    V1814.3 [R]: =1 -> START der ZPF + ENDE:=0
SWGT_1_17_INFO_FREEZE   V1814.4 [R]: =1 -> DIE ZPF VON 'w' IST EINGEFROREN
SWGT_1_18_INFO_ENDE     V1814.5 [R]: =1 -> ENDE der ZPF  + START:=0
SWGT_1_19_n_AKT         VW1816  [R]: AKTUELLER SSP [n]  {1,2,...,10}
SWGT_1_20_x_START       VW1818  [R]: ISTWERT 'x' BEIM START
SWGT_1_21_w_AKT_END     VW1820  [R]: ENDWERT  'w(n)' DES AKTUELLEN 'SSP[n]'
SWGT_1_22_t_AKT_SOLL    VW1822  [R]: SOLLWERT 't(n)' DES AKTUELLEN 'SSP[n]'
SWGT_1_23_t_AKT_IST     VD1824  [R]: ISTWERT  't(n)' DES AKTUELLEN 'SSP[n]'
SWGT_1_24_t_GES_SOLL    VD1828  [R]: SOLLWERT DER GESAMTZEIT FÜR ZPF
SWGT_1_25_t_GES_IST     VD1832  [R]: ISTWERT  DER GESAMTZEIT FÜR ZPF
SWGT_1_26_t_GES_REST    VD1836  [R]: RESTZEIT DER GESAMTZEIT FÜR ZPF
SWGT_1_27_DT_START      VD1840  [R]: ZEITSTEMPEL = START ZPF
SWGT_1_28_t_ALT         VW1844  [R]: ALTWERTSPEICHER 1ms-TIMER
SWGT_1_29_t_SSP_ZPF     VD1846  [R]: TIMER für ZPF SSP(n):  1ms
SWGT_1_30_t_VRT_AKT     VD1850  [R]: VORTEILER t_AKT_IST: 0.01h=36000ms/0.1s=100ms
SWGT_1_31_t_VRT_GES     VD1854  [R]: VORTEILER t_GES_IST: 0.01h=36000ms/0.1s=100ms
SWGT_1_32_PRV_2BYTE     VW1858  [R]: PRIVATE SPEICHER: 2xBYTE
SWGT_1_33_t_01          VW1860  [W]: ZEIT FÜR w(0) -> w(1)  SSP[1]
SWGT_1_34_w_01          VW1862  [W]: SOLLWERT w(1) AM ENDE  SSP[1]
SWGT_1_35_x_END_01      VW1864  [R]: ISTWERT  x(1) AM ENDE  SSP[1]
SWGT_1_36_t_GES_01      VD1866  [R]: GESAMTZEIT FÜR         SSP[1]
SWGT_1_37_DT_ST_01      VD1870  [R]: ZEITSTEMPEL = AM ENDE  SSP[1]
SWGT_1_38_t_02          VW1874  [W]: ZEIT FÜR w(1) -> w(2)  SSP[2]
SWGT_1_39_w_02          VW1876  [W]: SOLLWERT w(2) AM ENDE  SSP[2]
SWGT_1_40_x_END_02      VW1878  [R]: ISTWERT  x(2) AM ENDE  SSP[2]
SWGT_1_41_t_GES_02      VD1880  [R]: GESAMTZEIT FÜR         SSP[2]
SWGT_1_42_DT_ST_02      VD1884  [R]: ZEITSTEMPEL = AM ENDE  SSP[2]
SWGT_1_43_t_03          VW1888  [W]: ZEIT FÜR w(2) -> w(3)  SSP[3]
SWGT_1_44_w_03          VW1890  [W]: SOLLWERT w(3) AM ENDE  SSP[3]
SWGT_1_45_x_END_03      VW1892  [R]: ISTWERT  x(3) AM ENDE  SSP[3]
SWGT_1_46_t_GES_03      VD1894  [R]: GESAMTZEIT FÜR         SSP[3]
SWGT_1_47_DT_ST_03      VD1898  [R]: ZEITSTEMPEL = AM ENDE  SSP[3]
SWGT_1_48_t_04          VW1902  [W]: ZEIT FÜR w(3) -> w(4)  SSP[4]
SWGT_1_49_w_04          VW1904  [W]: SOLLWERT w(4) AM ENDE  SSP[4]
SWGT_1_50_x_END_04      VW1906  [R]: ISTWERT  x(4) AM ENDE  SSP[4]
SWGT_1_51_t_GES_04      VD1908  [R]: GESAMTZEIT FÜR         SSP[4]
SWGT_1_52_DT_ST_04      VD1912  [R]: ZEITSTEMPEL = AM ENDE  SSP[4]
SWGT_1_53_t_05          VW1916  [W]: ZEIT FÜR w(4) -> w(5)  SSP[5]
SWGT_1_54_w_05          VW1918  [W]: SOLLWERT w(5) AM ENDE  SSP[5]
SWGT_1_55_x_END_05      VW1920  [R]: ISTWERT  x(5) AM ENDE  SSP[5]
SWGT_1_56_t_GES_05      VD1922  [R]: GESAMTZEIT FÜR         SSP[5]
```

```
SWGT_1_57_DT_ST_05     VD1926  [R]: ZEITSTEMPEL = AM ENDE  SSP[5]
SWGT_1_58_t_06         VW1930  [W]: ZEIT FÜR w(5) -> w(6)  SSP[6]
SWGT_1_59_w_06         VW1932  [W]: SOLLWERT w(5) AM ENDE  SSP[6]
SWGT_1_60_x_END_06     VW1934  [R]: ISTWERT  x(5) AM ENDE  SSP[6]
SWGT_1_61_t_GES_06     VD1936  [R]: GESAMTZEIT FÜR         SSP[6]
SWGT_1_62_DT_ST_06     VD1940  [R]: ZEITSTEMPEL = AM ENDE  SSP[6]
SWGT_1_63_t_07         VW1944  [W]: ZEIT FÜR w(6) -> w(7)  SSP[7]
SWGT_1_64_w_07         VW1946  [W]: SOLLWERT w(7) AM ENDE  SSP[7]
SWGT_1_65_x_END_07     VW1948  [R]: ISTWERT  x(7) AM ENDE  SSP[7]
SWGT_1_66_t_GES_07     VD1950  [R]: GESAMTZEIT FÜR         SSP[7]
SWGT_1_67_DT_ST_07     VD1954  [R]: ZEITSTEMPEL = AM ENDE  SSP[7]
SWGT_1_68_t_08         VW1958  [W]: ZEIT FÜR w(7) -> w(8)  SSP[8]
SWGT_1_69_w_08         VW1960  [W]: SOLLWERT w(8) AM ENDE  SSP[8]
SWGT_1_70_x_END_08     VW1962  [R]: ISTWERT  x(8) AM ENDE  SSP[8]
SWGT_1_71_t_GES_08     VD1964  [R]: GESAMTZEIT FÜR         SSP[8]
SWGT_1_72_DT_ST_08     VD1968  [R]: ZEITSTEMPEL = AM ENDE  SSP[8]
SWGT_1_73_t_09         VW1972  [W]: ZEIT FÜR w(8) -> w(9)  SSP[9]
SWGT_1_74_w_09         VW1974  [W]: SOLLWERT w(9) AM ENDE  SSP[9]
SWGT_1_75_x_END_09     VW1976  [R]: ISTWERT  x(9) AM ENDE  SSP[9]
SWGT_1_76_t_GES_09     VD1978  [R]: GESAMTZEIT FÜR         SSP[9]
SWGT_1_77_DT_ST_09     VD1982  [R]: ZEITSTEMPEL = AM ENDE  SSP[9]
SWGT_1_78_t_10         VW1986  [W]: ZEIT FÜR w(9) -> w(10) SSP[10]
SWGT_1_79_w_10         VW1988  [W]: SOLLWERT w(10) AM ENDE SSP[10]
SWGT_1_80_x_END_10     VW1990  [R]: ISTWERT  x(10) AM ENDE SSP[10]
SWGT_1_81_t_GES_10     VD1992  [R]: GESAMTZEIT FÜR         SSP[10]
SWGT_1_82_DT_ST_10     VD1996  [R]: ZEITSTEMPEL = AM ENDE  SSP[10]
################################################################################

Für jede  Zeitplanführung sind folgende Parameter vom Typ '[W]':

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
n            [INT]: Nummer des letzten benötigten SSP          {1,2,...,10}
                    n<1 ODER n>10 -> OK:=0 UND y:=x!
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
w_START      [INT]: Startsollwert der Zeitplanführung   {-32767,...,+32767}*10^m
                    !!Die Zeit t(1) ist auf die Führung von w_START nach w(1)
                      bezogen. Für 'MOD_w_START=1' wird die Zeit t(1) für den
                      vorhandenen Istwert 'x' wie folgt interpoliert:
                                              |t(1)*[w(1)-x_START]|
                          t(1_INTERPOLIERT)= |-------------------|
                                              |  [w(1)-w_START]   |
                      Ist '[w(1)-w_START]=0', dann kann diese Berechnung nicht
                      ausgeführt werden und es gilt dann für alle 'x_START':
                          t(1_INTERPOLIERT):=t(1)
                      Falls 'w(1)=x_START' ist, dann ist t(1)=0 und es wird so-
                      fort auf SSP(2) umgeschaltet.
                    !!Durch diese Interpolation wird erreicht, daß mit der durch
                      die Werte 'w_START, w(1), t(1)' vorgegebenen Rampe, vom Ist-
                      wert 'x' als 'x=w_START' ausgehend die 'ZPF' gestartet wird.
                    !!'t(1_INTERPOLIERT)' wird auf '+32767*0,01h/1s' begrenzt.
```




```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
MAX_DRIFT    [INT]: MAX|x-w|>0 FÜR MOD_ZPF=1                 {0,...,+32767}*10^m
                    Ist der geführte Sollwert 'w' dem Istwert der Regelgröße 'x'
                    um einen vorgebbaren Wert, 'MAX_DRIFT', vorausgeeilt, dann
                    wird die Sollwertführung solange eingefroren, bis die Abwei-
                    chung '|x-w|<=MAX_DRIFT' ist.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
TAST_START  [BOOL]: HMI = TASTER-START  ZEITPLANFÜHRUNG mit PLS(+)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
TAST_FREEZE [BOOL]: HMI = TASTER-FREEZE ZEITPLANFÜHRUNG mit FLIP-FLOP
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
TAST_RESET  [BOOL]: HMI = TASTER-RESET  ZEITPLANFÜHRUNG mit PLS(+)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Für jeden Stützpunkt SSP[n]                              n {1,2,...,10}
müssen folgend Werte eingegeben werden:
t_01       [INT]: ZEIT FÜR w(START)-> w(1)   SSP[1] {0,1,...,32767}*0.01h/0.1s
w_01       [INT]: SOLLWERT w(1)    AM ENDE   SSP[1] {-32767,...,+32767}*10^m

t_02       [INT]: ZEIT FÜR w(1)    -> w(2)   SSP[2] {0,1,...,32767}*0.01h/0.1s
w_02       [INT]: SOLLWERT w(1)    AM ENDE   SSP[2] {-32767,...,+32767}*10^m

  :          :           :             :        :                :
  :          :           :             :        :                :

t_10       [INT]: ZEIT FÜR w(9)   -> w(10)  SSP[10]  {0,1,...,32767}*0.01h/0.1s
w_10       [INT]: SOLLWERT w(10)  AM ENDE   SSP[10]  {-32767,...,+32767}*10^m

!!Abhängig von 'MOD_ZPF' hat die Eingabe von 't(n)=0' unterschiedliche Wirkung:
  MOD_ZPF=0 -> 'w' wird sprunghaft von w(n-1) auf w(n) geändert, n_AKT:=n+1, und
               die ZPF wird mit den Daten des SSP[n+1] fortgesetzt.
  MOD_ZPF=1 -> 'w' wird sprunghaft von w(n-1) auf w(n) geändert. Danach wird so-
               lange mit der Umschaltung auf die Daten des SSP[n+1] gewartet,
               bis '|[x-w(n)]|<= MAX_DRIFT' ist.
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Folgende Werte der 'ZPF' können gelesen und ausgewertet werden:

INFO_START [BOOL]: =1 -> START der ZPF + ENDE:=0
INFO_FREEZE[BOOL]: =1 -> DIE ZPF VON 'w' IST EINGEFROREN
INFO_ENDE  [BOOL]: =1 -> ENDE der ZPF  + START:=0
n_AKT       [INT]: NUMMER DES AKTIVEN         'SSP[n]' {1,2,...,10}
x_START     [INT]: ISTWERT 'x' BEIM START               {-32767,...,+32767}*10^m
w_AKT_END   [INT]: ENDWERT 'w(n)' DES AKTIVEN 'SSP[n]' {-32767,...,+32767}*10^m
t_AKT_SOLL  [INT]: SOLLWERT't(n)' DES AKTIVEN 'SSP[n]' {0,1,...,32767}*0.01h/0.1s
AKT_IST     [INT]: ISTWERT 't(n)' DES AKTIVEN 'SSP[n]' {0,1,...,32767}*0.01h/0.1s
t_GES_SOLL  [INT]: SOLLWERT DER GESAMTZEIT FÜR ZPF      {0,1,...,32767}*0.01h/0.1s
t_GES_IST   [INT]: ISTWERT  DER GESAMTZEIT FÜR ZPF      {0,1,...,32767}*0.01h/0.1s
GES_REST    [INT]: RESTZEIT DER GESAMTZEIT FÜR ZPF      {0,1,...,32767}*0.01h/0.1s
                   !!DIESE RESTZEIT IST EIN THEORETISCHER WERT, WELCHER WEDER
                     DIE DURCH FREEZE ODER BEI MOD_ZPF=1 VERBRAUCHTE ZEIT BERÜCK-
                     SICHTIGT
DT_START  [DWORD]: ZEITSTEMPEL = START ZPF              {TT,MM,hh,mm}

Für jeden Stützpunkt SSP[n] der 'ZPF' kann zur Auswertung gelesen werden:

t_GES_01    [INT]: GESAMTZEIT FÜR            'SSP[1]'  {0,1,...,32767}*0.01h/0.1s
DT_ST_01  [DWORD]: ZEITSTEMPEL = AM ENDE     'SSP[1]'  {TT,MM,hh,mm}

t_GES_02    [INT]: GESAMTZEIT FÜR            'SSP[2]'  {0,1,...,32767}*0.01h/0.1s
DT_ST_02  [DWORD]: ZEITSTEMPEL = AM ENDE     'SSP[2]'  {TT,MM,hh,mm}

   :          :                   :                           :
   :          :                   :                           :

t_GES_10    [INT]: GESAMTZEIT FÜR            'SSP[10]' {0,1,...,32767}*0.01h/0.1s
DT_ST_10  [DWORD]: ZEITSTEMPEL = AM ENDE     'SSP[10]' {TT,MM,hh,mm}

!!ALLE ZEITSTEMPEL WURDEN SPEICHERBEDINGT AUF MINUTEN-GENAUIGKEIT BEGRENZT. SIE
  DIENEN NUR ZUR GROBEN KONTROLLE DES ZEITABLAUFES DER 'ZPF'.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Begriffe:
HMI      = SIMATIC Human Machine Interfaces
x        = Istwert  der Regelgröße            x        {-32768,...,+32767}*10^m
w        = Sollwert der Regelgröße            w        {-32768,...,+32767}*10^m
y        = geführter Sollwert                 y        {-32768,...,+32767}*10^m
SSP(n)   = Stützpunkt der Zeitplanführung     n        {1,2,...10}
t(n)     = Zeit für Sollwertführung t(n-1) -> t(n)     {0,1,...,32767}*0.01h/0.1s
w(n)     = Sollwert am ENDE von SSP(n)        w(n)     {-32768,...,+32767}*10^m
t(n)_IST = Im SSP(n) abgelaufene Zeit         t(n)_IST {0,1,...,32767}*0.01h/0.1s
DT       = Zeitstempel [TT,MM,hh,mm] als HEX-/=BCD-ZAHL
                                    TAG = TT           {1,2,...,28,29,30,31}
                                  MONAT = MM           {1,2,...,12}
                                STUNDEN = hh           {0,1,...,23}
                                MINUTEN = mm           {0,1,...,59}
10^m     = gemeinsamer Koeffizient aller damit gekennzeichneter Größen, der sich
           bei der Berechnung herauskürzt. Der korrekte Wert ist nur als fiktive
           Kommastelle oder angehängte Nullen beim HMI-System wichtig.
m        = Exponent zur Basis 10          m         {-t,...,-2,-1,0,1,2,...,+t}
```

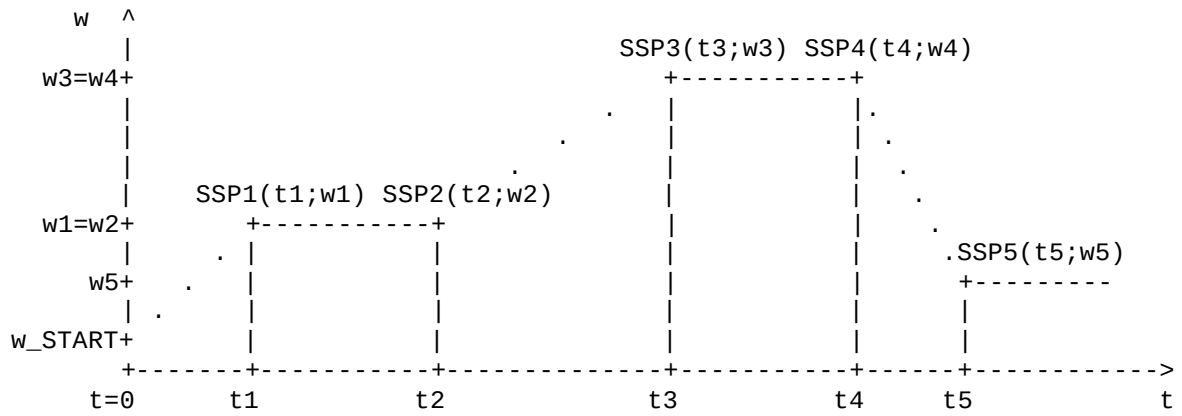
++
Berechnungsgrundlagen:
'y' wird im aktuellen Sollwertstützpunkt 'SSP[n] wie folgt berechnet:

$$y = \frac{[w(n)-w(n-1)]*t\_IST(n)}{t(n)} + w(n-1)$$

Für n=1 UND MOD_w_START=1, wird t(1), wie unter 'w_START' beschrieben, interpoliert.

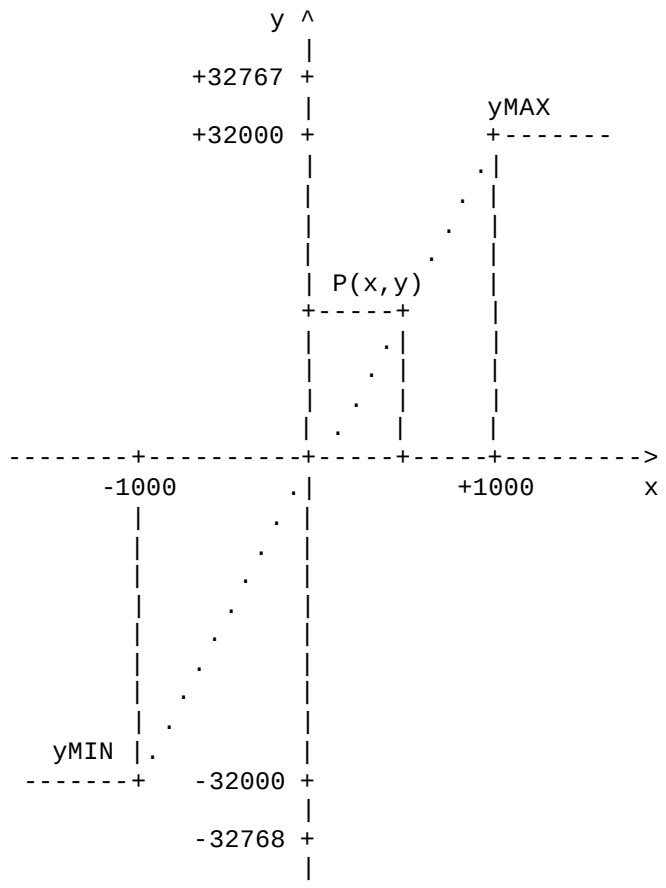
!!t(n)=0 UND/ODER [w(n)-w(n-1)]-> w(n)=w(n-1)
  Abhängig von 'MOD_ZPF' haben diese Konstellationen unterschiedliche Wirkung:
  MOD_ZPF=0 -> 'w' wird sprunghaft von w(n-1) auf w(n) geändert, n_AKT:=n+1, und
               die ZPF wird mit den Daten des SSP[n+1] fortgesetzt.
  MOD_ZPF=1 -> 'w' wird sprunghaft von w(n-1) auf w(n) geändert. Danach wird solange mit der Umschaltung auf die Daten des SSP[n+1] gewartet,
               bis '|[x-w(n)]|<= MAX_DRIFT' ist.
!![w(n)-w(n-1)] ->
++

Beispiel:
Sollwertzeitplangeber mit 5 Stützpunkten {SSP1,...,SSP5}
MOD_w_START=0 -> 'w' beginnt mit dem Wert 'w_START'
MOD_ZPF   =0 -> nach Ablauf von 't(n)' gilt stets 'w=w(n)'



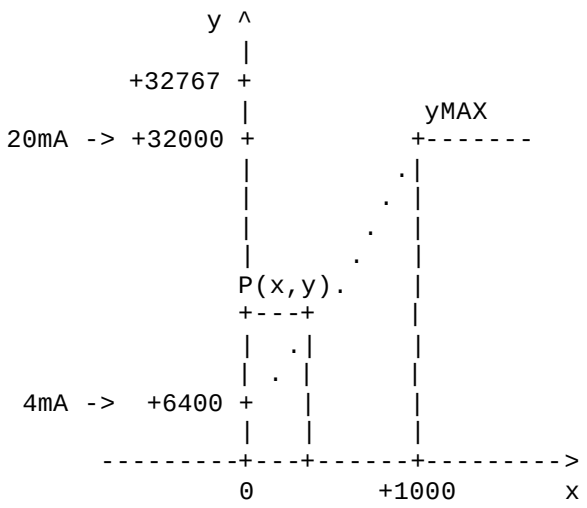
SUBROUTINE_BLOCK NORMSIGNAL_ANALOGOUTPUT:SBR49
TITLE=UNTERPROGRAMM: 'NORMSIGNAL_ANALOGOUTPUT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!
##
Kurzbeschreibung:
Mit dem UP 'NORMSIGNAL_ANALOG_OUT' wird der Bereich {-1000,...,0,...,+1000}, in dem sich der INTEGERWERT eines NORMSIGNALES bewegt, in das SPS-interne Analogausgangssignal {-32000,...,0,...,+32000} umgewandelt. Liegt das Normsignal nicht im o.g. Bereich, so wird es UP-intern begrenzt. Es besteht die Möglichkeit, eine 0-20mA Analogausgangsbaugruppe für ein 4-20mA Ausgangssignal zu parametrieren.
##

```
in:
MOD [BOOL]: =0 -> Unipolar + Bipolar
            =1 -> 4-20mA
x    [INT]: NORMSIGNAL (STELLGRÖßE)                      {-1000,...,0,...,+1000}
            x>1000 ODER x<-1000 wird 'UP-intern wie folgt begrenzt:
              unipolar + bipolar -> x {-1000,...,+1000}
              4-20mA             -> x     {0,...,+1000}

out:
y    [INT]: SPS-internes ANALOGAUSGANGSSIGNAL          {-32000,...,0,...,+32000}
               unipolar + bipolar -> y     {-32000,...,+32000}
               4-20mA             -> x      {+6400,...,+32000}
               (Dazu wird eine 0-20mA Baugruppe benutzt, die bei dem Anfangswert
                von +6400 = 4mA Ausgangsstrom hat!)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Berechnungsgrundlagen:

Für unipolare und bipolare Ausgangssignale wird 'y'wie folgt berechnet:

       32000
  y = -------- * x
       1000
```



Für 4-20mA Ausgangssignale ist 'y' stets unipolar und berechnet sich:

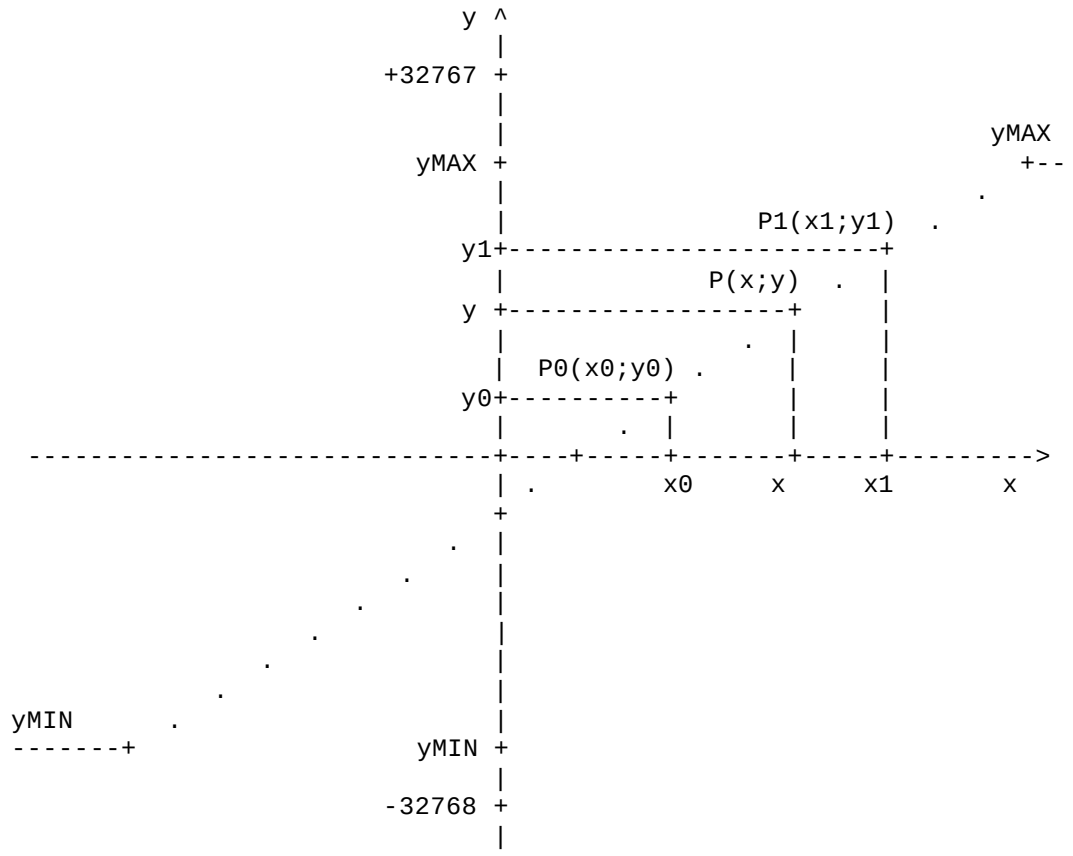
$$y = \frac{(32000-6400)}{1000} * x + 6400 = \frac{25600}{1000} * x + 6400$$



SUBROUTINE_BLOCK SCALE_LINEAR_LIMIT_INT:SBR50
TITLE=UNTERPROGRAMM: 'SCALE_LINEAR_LIMIT_INT', 'y=a*x+b'
!!ACHTUNG, IN DIESEM UP wird die Sprungmarke 204 benutzt!

##
Kurzbeschreibung:
Mit dem UP 'SCALE_LINEAR_LIMIT_INT' kann ein beliebiger Bereich einer im INTEGER-FORMAT vorliegenden Punktmenge {X} linear auf die Punktmenge {Y} abgebildet werden. Die lineare Zuordnung zwischen den Punktmengen wird in Form der Geradengleichung 'y=a*x+b' hergestellt. Die Punktmenge {Y} wird im INTEGERFORMAT berechnet und kann auf einen MIN-/MAX-Wert begrenzt werden.
Das UP 'SCALE_LINEAR_LIMIT_INT' ist für die Skalierung beliebiger Analogeingangswerte in physikalische Größen vorgesehen. Es können auch beliebige Bereiche eines Integerwertes in ein Analogausgangssignal gewandelt werden.
##

```
in:
x      [INT]:  Abszisse Punkt P(x;y)                        {-32768,...,+32767}
yMIN   [INT]:  MIN-Limit für y                               {-32768,...,<yMAX}
yMAX   [INT]:  MAX-Limit für y                               {yMIN<,...,+32767}
x0     [INT]:  Abszisse Punkt P0(x0;y0)                      {-32768,...,+32767}
y0     [INT]:  Ordinate Punkt P0(x0;y0)                      {-32768,...,+32767}
x1     [INT]:  Abszisse Punkt P1(x1;y1)                      {-32768,...,+32767}
y1     [INT]:  Ordinate Punkt P1(x1;y1)                      {-32768,...,+32767}

out:
OK    [BOOL]:  OK:=1;   WENN ÜBERLAUF INT-BEREICH -> OK   :=0
y      [INT]:  Ordinate Punkt P(x;y)                             {yMIN,...,yMAX}

!!!Für 'y' gilt:
       Wenn  y <= yMIN,                  dann y:=yMIN
       Wenn  y >= yMAX,                  dann y:=yMAX
       Wenn  yMIN >= yMAX,               dann y unbegrenzt
       Wenn  x0=x1,                      dann y:=0 + OK:=0
       Wenn  y0=y1,                      dann y:=y0 + Begrenzung auf yMIN/MAX

!!!Überlauf INTEGER-Bereich -> Korrektur 'y' auf MIN/MAX-INTEGER UND BIE-BIT:=0

In diesem FC wird eine lineare Zuordnung zwischen den Punktmengen
{X} und {Y} in Form der Geradengleichung 'y=a*x+b' hergestellt.
FC-intern werden anhand von 2 beliebig vorgebbaren Punkten P0(x0;y0)
und P1(x1;y1) die Konstanten 'a' und 'b' der Gleichung bestimmt.
Dann kann für jeden Wert 'x' der zugeordnete Wert 'y' berechnet wer-
den.

Durch die Eingabe von yMIN-/yMAX kann der berechnete Wert von 'y'
auf einen unteren und oberen BEREICH begrenzt werden.
Damit werden ÜBER- UND UNTERsteuerungsbereiche der Analog-EIN- und
AUSgangskarten ausgeblendet.(Z.B. wird erreicht, daß Analogausgangs-
werte immer im Bereich {0,1,...,27648} liegen.)

Die Menge, {X} ODER {Y}, welche die PHYSIKALISCHE GRÖßE darstellt, muß im gesam-
ten Wertebereich mit dem Koeffizienten Z=10^m; m{-t,...,-1,0,+1,...,+t} multi-
plizierbar sein, damit für das BuB-System eine genormte Dezimalzahldarstellung
der 'INTEGERWERTE' als 'FESTKOMMAZAHL' möglich ist.



 'y' wird wie folgt berechnet:

      (y1-y0)
  y = --------*(x-x0) + y0
      (x1-x0)

 y0=y1       ->  y:=y0 für alle x
 x0=x1       ->  y:=0 UND ERROR, das BIE-BIT:=0
 yMIN>=yMAX  ->  Die Begrenzung ist unwirksam und es gilt:
                 yMIN:=-32768; yMAX:=+32767
```



Beispiele:

1.)
Analogeingangskarte:
Der Meßwertgeber liefert ein Signal {4,...,20}mA. Damit wird ein Meßbereich
für eine Füllstandsmessung von {0,...,1700}mm abgedeckt.Die Analogeingangskarte
hat für ein Signal {0,...,20}mA den SPS-internen Wertebereich {0,...,32000}.
Der Eingangswert 4mA hat den SPS-internen Wert von [(32000:20mA)*4mA]=6400.
Eine Drahtbruchüberwachung soll möglich sein.
Aus dem o.g. ergibt sich folgede Parametrierung:

P0(6400;0)	x0=6400 ->	y0= 0	*[0,001m]
P1(32000;1700)	x1= +32000 ->	y1= 1700	*[0,001m]
Folgende Begrenzungen sind sinnvoll:	yMIN = 0		
	yMAX = + 1700		

2.)
Analogausgangskarte:
Das SPS-interne Reglerausgangssignal liegt im Bereich {-1000,...,+1000} vor.
Dieses soll in das SPS-interne Analogausgangs-Signal {-32000,...,0,...,+32000}
so gewandelt werden, daß der gesamte Bereich des Reglerausgangssignales, +/-10V,
wirksam wird.
Daraus ergibt sich folgende Parametrierung:

P0(-1000; -32000)	x0= -1000	und	y0= -32000
P1(+1000; +32000)	x1= +1000	und	y1= +32000
Folgende Begrenzungen sind sinnvoll:	yMIN = -32000		

SUBROUTINE_BLOCK AVERAGE_2_INT:SBR51
TITLE=UNTERPROGRAMM: 'AVERAGE_2_INT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!

```
################################################################################
Kurzbeschreibung:
Das Unterprogramm 'AVERAGE_2' berechnet aus 2 beliebigen INTEGERVARIABLEN einen
MITTELWERT im INTEGERFORMAT.
################################################################################

in:
x1       [INT]:  INT-Variable 1                      {-32768,...,+32767}
x2       [INT]:  INT-Variable 2                      {-32768,...,+32767}

out:
y        [INT]:  Mittelwert                          {-32768,...,+32767}


Berechnungsgrundlagen:

         x1 + x2
     y = -------
            2

!!Ist der REST der DIVISION 'REST<>0' dann wird 'y' auf die nächste, größere
  bzw. kleinere Zahl gerundet.
```

SUBROUTINE_BLOCK AVERAGE_2_8_INT:SBR52
TITLE=UNTERPROGRAMM: 'AVERAGE_2_8_INT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!

```
################################################################################
Kurzbeschreibung:
Das UP 'AVERAGE_2_8_INT' berechnet aus 2 bis 8 beliebigen INTEGERVARIABLEN einen
MITTELWERT im INTEGERFORMAT. Die Anzahl der in die Berechnung des Mittelwertes
eingehenden Variablen ist im Bereich {2,3,...,8} frei wählbar.
################################################################################
in:
n   [INT]:  Anzahl der Variablen                                  {2,3,...,8}
            Durch den Wert von 'n' wird die Anzahl und Reihenfolge der Variablen
            bestimmt, die in die Durchschnittsberechnung eingehen.
            !!Fehleingabe: n<2 -> n:=2 ODER n>8 -> n:=8   ->   OK:=0,
x1   [INT]: INT-Variable 1                                  {-32768,...,+32767}
x2   [INT]: INT-Variable 2                                  {-32768,...,+32767}
x3   [INT]: INT-Variable 3                                  {-32768,...,+32767}
x4   [INT]: INT-Variable 4                                  {-32768,...,+32767}
x5   [INT]: INT-Variable 5                                  {-32768,...,+32767}
x6   [INT]: INT-Variable 6                                  {-32768,...,+32767}
x7   [INT]: INT-Variable 7                                  {-32768,...,+32767}
x8   [INT]: INT-Variable 8                                  {-32768,...,+32767}

out:
OK  [BOOL]: OK:=1;   Fehleingabe von 'n' -> OK   :=0
y    [INT]: Mittelwert                                      {-32768,...,+32767}
```

Berechnungsgrundlagen:

$$y = \frac{x1 + x2 \ldots + x(n)}{n} \; ; \quad n\{2,3,\ldots,8\}$$

Beispiel: n=5

$$y = \frac{x1 + x2 + x3 + x4 + x5}{5}$$

!!Ist der REST der DIVISION 'REST<>=0' dann wird 'y' auf die nächste, größere
bzw. kleinere Zahl gerundet.

!!Die Inputs x6, x7 UND x8 sind bei n=5 OHNE BEDEUTUNG.

SUBROUTINE_BLOCK MIN_MAX_SELECT_2_INT:SBR53
TITLE=UNTERPROGRAMM: 'MIN_MAX_SELECT_2_INT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!

```
################################################################################
Kurzbeschreibung:
Das UP 'MIN_MAX_SELECT_2_INT' ermittelt aus 2 beliebigen INTEGERVARIABLEN das
MINIMUM und MAXIMUM .
################################################################################
in:
x1       [INT]:  INT-Variable 1                              {-32768,...,+32767}
x2       [INT]:  INT-Variable 2                              {-32768,...,+32767}

out:
yMIN     [INT]:  MINIMUM von                                              {x1,x2}
yMAX     [INT]:  MAXIMUM von                                              {x1,x2}
```

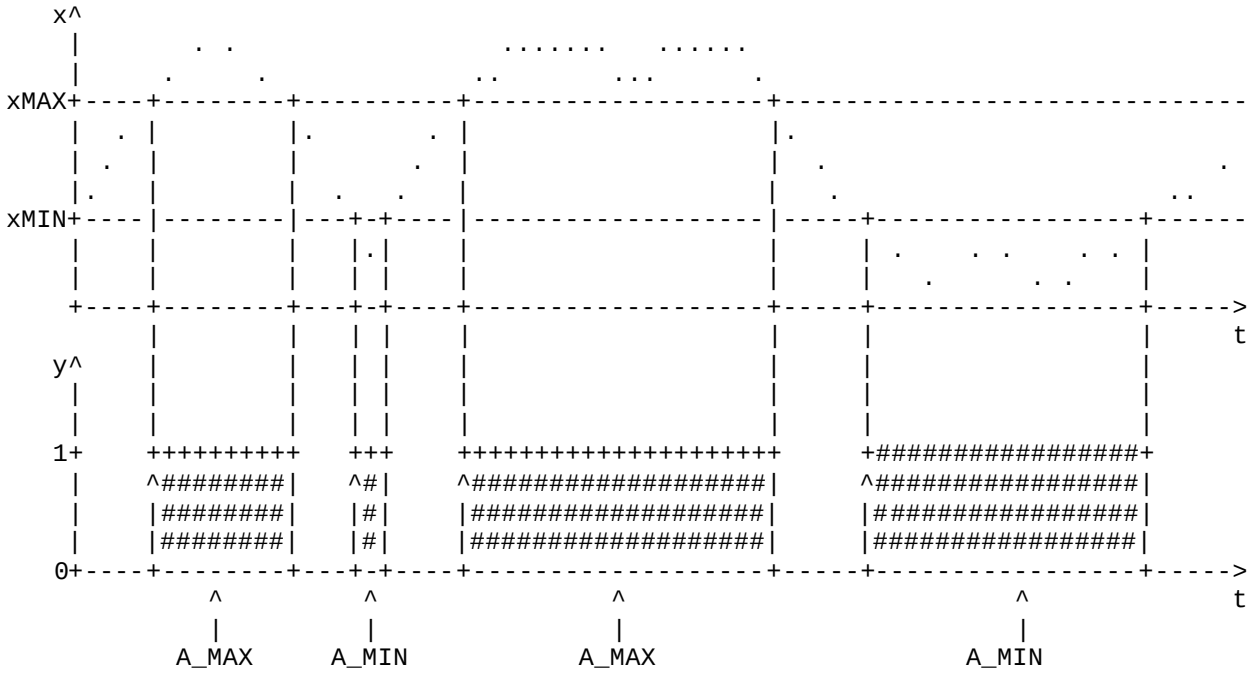
SUBROUTINE_BLOCK MIN_MAX_SELECT_2_4_INT:SBR54
TITLE=UNTERPROGRAMM: 'MIN_MAX_SELECT_2_4_INT'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!

```
################################################################################
Kurzbeschreibung:
Das UP 'MIN_MAX_SELECT_2_4_INT' ermittelt aus 4 beliebigen INTEGERVARIABLEN das
MINIMUM und MAXIMUM . Die Anzahl der in die Ermittelung des MINIMUMS und MAXIMUMS
eingehenden Variablen ist im Bereich {2,3,...,4} frei wählbar.
################################################################################

in:
n   [INT]:  Anzahl der Variablen                                    {2,3,...,4}
            Durch den Wert von 'n' wird die Anzahl und Reihenfolge der Variablen
            bestimmt, die in die Durchschnittsberechnung eingehen.
            !!Fehleingabe: n<2 -> n:=2 ODER n>4 -> n:=4   ->   OK:=0,
x1    [INT]: INT-Variable 1                                  {-32768,...,+32767}
x2    [INT]: INT-Variable 2                                  {-32768,...,+32767}
x3    [INT]: INT-Variable 3                                  {-32768,...,+32767}
x4    [INT]: INT-Variable 4                                  {-32768,...,+32767}

out:
OK   [BOOL]: OK:=1;   Fehleingabe von 'n' -> OK   :=0
yMIN  [INT]:  MINIMUM von                                               {x1,x2}
yMAX  [INT]:  MAXIMUM von                                               {x1,x2}
```

SUBROUTINE_BLOCK MIN_MAX_ALARM:SBR55
TITLE=UNTERPROGRAMM: 'MIN_MAX_ALARM'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!

```
################################################################################
Kurzbeschreibung:
Das UP 'MIN_MAX_ALARM' aktiviert jeweils einen MIN- ODER MAX-Alarm, wenn eine
INTEGER-Variable frei parametrierbare MIN-/MAX-Grenzwerte erreicht bzw. unter-
oder überschreitet.
################################################################################

in:
x       [INT]: Zu überwachende Integervariable      {-32768,...,+32767}
                     x<= xMIN -> MINIMUM-ALARM
                     x>= xMAX -> MAXIMUM-ALARM
xMIN    [INT]: GRENZWERT FÜR MINIMUM-ALARM          {-32768,...,+32767}
xMAX    [INT]: GRENZWERT FÜR MAXIMUM-ALARM          {-32768,...,+32767}

out:
A_MIN  [BOOL]: =1 -> MINIMUM-ALARM
A_MAX  [BOOL]: =1 -> MAXIMUM-ALARM
```

Grafische Darstellung:



SUBROUTINE_BLOCK HY_SWITCH_hw:SBR56
TITLE=UNTERPROGRAMM: 'HY_SWITCH_hw'
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken 207 + 208 benutzt!

```
################################################################################
Kurzbeschreibung:
Das UP 'HYST_SWITCH_hw' stellt einen Hystereseschalter dar, dessen Output 'y'
sich aus der Regelgröße 'x', dem Sollwert 'w' und der Hysterese 'h' berechnet.
Mit den Inputs 'ENABLE' und 'FREEZE' können Sequenzverriegelungen programmiert
werden. Die Wirkungsrichtung des Hysteresestereseschalters, direkt- oder umgekehrt
wirkend, ist konfigurierbar.
In der Symboltabelle sind für alle Hystereseschalter 9BYTE lange Variablen-
Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der
IN- und OUTPUTS des Reglers benutzt werden müssen. Die Adresse der Anfangsvariab-
len dieser Strukturen ist über den INPUT 'PRV9BYTE' an das Unterprogramm zu über-
geben. Alle mit '[W]' gekennzeichneten Variablen können wahlweise im OB1, im Da-
tenbaustein oder das HMI-System parametriert werden.
!!SIMATIC HMI = Human Machine Interfaces!!
################################################################################

in:
ENABLE      [BOOL]: =0 -> DISABLE: y:=yUW:=yDW:=0;  =1 -> ENABLE: Berechnung 'y'
                    !!'ENABLE' hat Vorrang vor 'FREEZE'
FREEZE      [BOOL]: Sequenzverriegelung:
                    =0 -> der Schaltzustand von 'y' ist wie beschrieben von
                          den Parametern 'ENABLE', 'x', w' und 'h' abhängig
                    =1 -> der momentane Schaltzustand von 'y' wird eingefroren
                          und ist von 'x' unabhängig. 'ENABLE' ist wirksam!
IN_DI        [BOOL]: =0 -> umgekehrt wirkend; =1 -> direkt wirkend
x            [INT]: Istwert  der Regelgröße              {-32768,...,+32767}
w            [INT]: Sollwert der Regelgröße              {-32768,...,+32767}
h            [INT]: Hysterese                                {1,2,...,32767}
             !!Fehleingabe h<=0 -> UP-intern h:=1
PRV9BYTE   [DINT]: Anfangsadresse eines 9 BYTE langen Variablenspeicherberei-
                    ches, dessen Struktur in der Symboltabelle festgelegt ist.
                    Diese Struktur muß für jeden Hystereseschalter in der Sym-
                    boltabelle vorhanden sein. Weiter unten sind die einzelnen
                    INPUTS in diese Struktur näher erläutert.
                    Ist z.B. das erste Element dieser Struktur 'V200.0', so
                    ist der Input 'PRV8BYTE' mit '&VB200' zu parametrieren.

out:
y           [BOOL]: Stellgröße IN  x <= w-h UND  ENABLE=1 -> y:=1
                                   x >= w   ODER ENABLE=0 -> y:=0
                    Stellgröße DI  x >= w+h UND  ENABLE=1 -> y:=1
                                   x <= w   ODER ENABLE=0 -> y:=0

!!UP-intern werden alle Werte im 'DINT-Format' verglichen. Bei ungünstigen
  Wertekombinationen  kann es vorkommen, daß 'w-h' oder 'w+h' außerhalb
  des Wertebereiches von 'x' liegen. In solchen Fällen ist 'y' permanent :=0.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!!Alle Eingabewerte der Parameter des UP 'HYST_SWITCH_hw' werden auf die angege-
  benen Grenzen geprüft. Sind Eingaben fehlerhaft, so werden sie auf die zuläs-
  sige Untergrenze=UG bzw. Obergrenze=OG UP-intern korrigiert.
################################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
```

Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der identisch bezeichneten In- und Outputs des zugeordneten UP's reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS können im OB1 und vom HMI-System nur gelesen werden. Statt der der Variablen können alle Inputs auch mit Konstanten parametriert werden und die Outputs dürfen Lokalvariablen sein. Lokalvariable sind in den Netzwerken nur in aufsteigender Richtung lesbar!

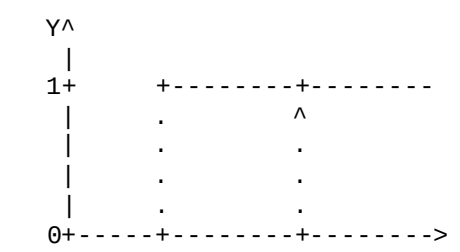
```
################################################################################
Aufbau der Struktur für einen Hystereseschalter vom TYP 'HYST_SWITCH_hw, dessen
Anfangsadresse 'V200.0' ist:
HY_wh_1_1_ENABLE         V200.0 [PI]: I=0->DISABLE: y:=0; =1->ENABLE: BERECHNUNG y
HY_wh_1_2_FREEZE         V200.1 [PI]: SEQUENZVERRIEGELUNG
HY_wh_1_3_IN_DI          V200.2 [PI]: =0 -> INVERS  /  =1 -> DIRECT WORKING
HY_wh_1_4_x              VW202  [PI]: REGELGRÖßE=ISTWERT    {-32768,...,+32767}
HY_wh_1_5_w              VW204  [P]I: REGELGRÖßE=SOLLWERT   {-32768,...,+32767}
HY_wh_1_6_h              VW206  [PI]: HYSTERESE  (h=0 -> h:=1) {1,2,...,+32767}
HY_wh_1_7_y              V208.0 [PO]: y: IN: x<=w-h  ODER  DI:x>=w+h  ->y=1
HY_wh_1_8_PRV_1BOOL      V208.1 [R]:  1 BIT privater Speicher
################################################################################
Begriffe:
HMI      = SIMATIC 'Human Machine Interfaces'
IN_DI    = Wirkrichtung des Hystereseschalters IN_DI  {0,1}
x        = Istwert  der Regelgröße                x  {-32768,...,+32767}
w        = Sollwert der Regelgröße                w  {-32768,...,+32767}
h        = Hysterese  (h<=0 -> h:=1)              h {1,2,...,+32767}
y        = Stellgröße umgekehrt ODER direkt wirkend y {0,1}

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Beispiele:

Umgekehrt wirkend   'IN'

   Y^
    |
   1+     +--------+--
    |     ^        .
    |     .        .
    |     .        .
    |     .        .
    |     .        .
   0+-----+--------+--------->
         w-h       w         x
          |<-- h --|

Direkt wirkend      'DI'

   Y^
    |
   1+     +--------+--
    |     .        ^
    |     .        .
    |     .        .
    |     .        .
   0+-----+--------+-------->
         w       w+h       x
```

SUBROUTINE_BLOCK HY_SWITCH_xON_OFF:SBR57
TITLE=UNTERPROGRAMM: 'HY_SWITCH_xON_xOFF'
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken 209 + 210 benutzt!

```
################################################################################
Kurzbeschreibung:
Das UP 'HY_SWITCH_xON_xOFF' stellt einen Hystereseschalter dar, dessen Out-
put 'y' sich aus der Regelgröße 'x', dem Einschaltpunkt 'xON' und dem Auschalt-
punkt 'xOFF' berechnet. Dabei wird die parametrierbare Wirkrichtung des Hyste-
reseschalters, direkt- oder umgekehrt wirkend, beachtet.
Mit den Inputs 'ENABLE' und 'FREEZE' können Sequenzverriegelungen programmiert
werden.
In der Symboltabelle sind für alle Hystereseschalter 9BYTE lange Variablen-
Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der
IN- und OUTPUTS des Hystereseschalters benutzt werden müssen. Die Adresse der
Anfangsvariablen dieser Strukturen ist über den INPUT 'PRV9BYTE' an das Unter-
programm zu übergeben. Alle mit '[W]' gekennzeichneten Variablen können wahlweise
im OB1, im Datenbaustein oder das HMI-System parametriert werden.
!!SIMATIC HMI = Human Machine Interfaces!!
################################################################################

in:
ENABLE   [BOOL]: =0 -> DISABLE: y:==0;  =1 -> ENABLE: Berechnung 'y'
                    !!'ENABLE' hat Vorrang vor 'FREEZE'
FREEZE   [BOOL]: Sequenzverriegelung:
                 =0 -> der Schaltzustand von 'y' ist wie beschrieben von den
                       Parametern 'ENABLE', 'x', 'xON' und 'xOFF' abhängig
                 =1 -> der momentane Schaltzustand von 'y' wird eingefroren
                       und ist von 'x' unabhängig. 'ENABLE' ist wirksam!
IN_DI     BOOL]: =0 -> umgekehrt wirkend; =1 -> direkt wirkend
x         [INT]: Istwert  der Regelgröße                [-32768,...,+32767]
xON       [INT]: Einschaltpunkt                         [-32768,...,+32767]
xOFF      [INT]: Ausschaltpunkt                         [-32768,...,+32767]
PRV9BYTE [DINT]: Anfangsadresse eines 9 BYTE langen Variablenspeicherberei-
                 ches, dessen Struktur in der Symboltabelle festgelegt ist.
                 Diese Struktur muß für jeden Hystereseschalter in der Sym-
                 boltabelle vorhanden sein. Weiter unten sind die einzelnen
                 INPUTS in diese Struktur näher erläutert.
                 Ist z.B. das erste Element dieser Struktur 'V250.0', so
                 ist der Input 'PRV9BYTE' mit '&VB250' zu parametrieren
out:
OK       [BOOL]: =1->OK; =0->FEHLER  xON=xOFF
                                     xON>xOFF -> DI UND  IN_DI=0 -> IN
                                     xON<xOFF -> IN UND  IN_DI=1 -> DI
y        [BOOL]: Stellgröße, die je nach Wahl von 'IN_DI' und den Relationen
                 zwischen x, xON und xOFF folgende Zustände hat:
                 UMGEKEHRT WIRKEND = 'IN':
                 xON <  xOFF:  x >= xOFF                        -> y:=0
                               x <= xON                         -> y:=1
                 xON >= xOFF:  x =  beliebig, FREEZE = beliebig -> y:=0

                 DIREKT WIRKEND    ='DI':
                 xON > xOFF:   x <= xOFF                        -> y:=0
                               x >= xON                         -> y:=1
                 xON <= xOFF:  x =  beliebig, FREEZE = beliebig -> y:=0
```

```
##############################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                    nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten para-
                     metriert werden und die Outputs dürfen Lokalvariablen sein.
                     Lokalvariable sind in den Netzwerken nur in aufsteigender
                     Richtung lesbar!
##############################################################################
Aufbau der Struktur für einen Hystereseschalter vom TYP 'HYST_SWITCH_xON_xOFF',
dessen Anfangsadresse 'V250.0' ist:
HY_xON_OFF_1_1_ENABLE     V250.0 [PI]: =0->DISABLE: y:=0; =1->ENABLE: BERECHNUNG y
HY_xON_OFF_1_2_FREEZE     V250.1 [PI]: SEQUENZVERRIEGELUNG
HY_xON_OFF_1_3_IN_DI      V250.2 [PI]: =0 -> INVERS  /  =1 -> DIRECT WORKING
HY_xON_OFF_1_4_x          VW252  [PI]: REGELGRÖßE=ISTWERT  {-32768,...,+32767}
HY_xON_OFF_1_5_xON        VW254  [PI]: EINSCHALTPUNKT   {-32768,...,+32767}
HY_xON_OFF_1_6_xOFF       VW256  [PI]: AUSSCHALTPUNKT  {-32768,...,+32767}
HY_xON_OFF_1_7_OK         V258.0 [PO]: =1->OK; =0->FEHLER xON, xOFF, IN_DI
HY_xON_OFF_1_8_y          V258.1 [PO]: y: IN: x<=w-h  ODER  DI:x>=w+h  ->y=1
HY_xON_OFF_1_9_PRV_1BOOL  V258.2 [R]:  1 BIT privater Speicher
##############################################################################



Begriffe:
HMI      = SIMATIC 'Human Machine Interfaces'
IN_DI    = Wirkrichtung des Hystereseschalters  IN_DI              {0,1}
x        = Istwert  der Regelgröße                    {-32768,...,+32767}
xON      = Einschaltpunkt                             [-32768,...,+32767]
xOFF     = Ausschaltpunkt                             [-32768,...,+32767]
y        = Stellgröße umgekehrt ODER direkt wirkend y {0,1}

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Beispiele:

Umgekehrt wirkend   'IN'
xON <  xOFF:  x >= xOFF -> y:=0 bzw.  x <= xON -> y:=1

   Y^
    |
   1+     +--------+--------
    |     ^        .
    |     .        .
    |     .        .
    |     .        .
   0+-----+--------+-------->
         xON     xOFF      x
```

```
Direkt wirkend      'DI'
xON > xOFF:   x <= xOFF -> y:=0 bzw.  x >= xON -> y:=1
```

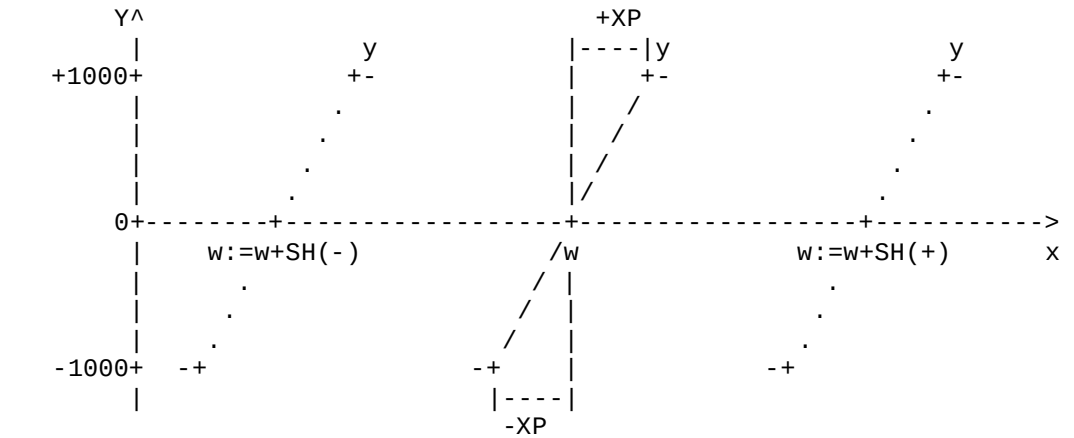


SUBROUTINE_BLOCK P_CONT:SBR58
TITLE=UNTERPROGRAMM: 'P_CONT'
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken 211 + 212 benutzt!

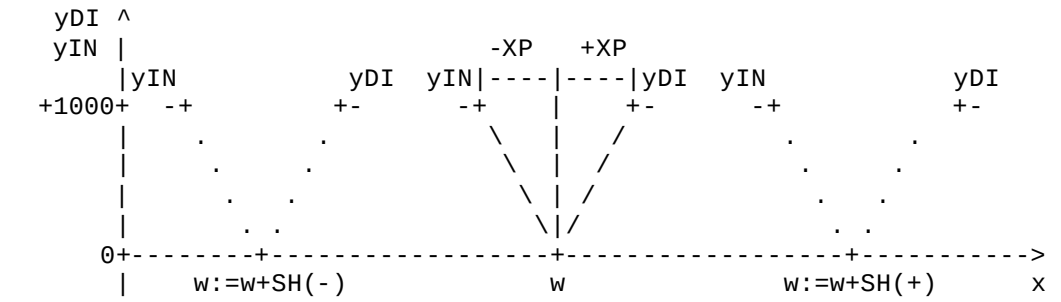
```
################################################################################
Kurzbeschreibung:
Das UP 'P_CONT' stellt einen Proportionalregler dar, dessen Parameter im INTEGER-
FORMAT eingegeben werden. Der UP-Input'SH' ermöglicht Sollwertschiebungen für
Kaskadenregelungen. Ober- und Untergrenze der Sollwertschiebung sind parame-
trierbar.
Der Regler besitzt folgende 3 Norm-Ausgangssignale im Bereich {-1000,...,+1000}:
  'y'   {-1000,...,0,...,+1000}, als direktwirkendes Signal, dessen Vorzeichen
        durch das Vorzeichen der Regelabweichung '(x-w)' bestimmt wird.
  'yIN' {+1000,...,0} als umgekehrtwirkendes Signal für '(x-w)<=0 :=|-y| ODER :=0'
  'yDI' {0,...,+1000} als direktwirkendes    Signal für '(x-w)>=0 := +y  ODER :=0'
Alle Parameter des Reglers sind Festkommazahlen. Aus diesem Grund wird statt der
Proportionalverstärkung 'KP' das Proportionalband 'XP' als Parameter des
Reglers verwendet. (XP=1/KP!)
In der Symboltabelle sind für alle Proportionalregler 22BYTE lange Variablen-
Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der
IN- und OUTPUTS des Reglers benutzt werden müssen. Die Adresse Anfangsvariablen
dieser Strukturen ist über den INPUT 'PRV22BYTE' an das Unterprogramm zu über-
geben. Alle mit '[W]' gekennzeichneten Variablen können wahlweise im OB1, im Da-
tenbaustein oder das HMI-System parametriert werden.
!!SIMATIC HMI = Human Machine Interfaces!!
################################################################################

in:
ENABLE    [BOOL]: =0 -> DISABLE: y:=yIN:=yDI:=0;  =1 -> ENABLE -> Berechnung
x          [INT]: Istwert der Regelgröße               {-32768,...,+32767}*10^m
SH         [INT]: Sollwertschiebung/Kaskadeninput      {-32768,...,+32767}*10^m
                  Die Schiebeweite wird durch die Eingabewerte 'w_SH_MIN' und
                  und 'w_SH_MAX' wie folgt begrenzt:
                  SH>0:= SH(+) -> [w+SH(+)]<=wSH_MAX;   w > wSH_MAX -> SH:=0
                  SH<0:= SH(-) -> [w+SH(-)]>=wSH_MIN;   w < wSH_MIN -> SH:=0
PRV22BYTE [DINT]: Anfangsadresse eines 22 BYTE langen Variablen-
                  speicherbereiches, dessen Struktur in der Symboltabelle
                  festgelegt ist. Diese Struktur muß für jeden P-Regler in
                  der Symboltabelle vorhanden sein. Weiter unten sind die
                  einzelnen INPUTS in diese Struktur näher erläutert.
                  Ist z.B. das erste Element dieser Struktur 'V300.0', so
                  ist der Input 'PRV22BYTE' mit '&VB300' zu parametrieren.
out:
y           [INT]: Stellgröße  = Normsignal            {-1000,...,0,...,+1000}
yIN         [INT]: Stellgröße umgekehrt wirkend               {+1000,...,+1,0}
```

```
                  y>0 -> yIN:=0 UND y<=0 -> yIN:=-y
yDI         [INT]: Stellgröße direkt wirkend                  {0,+1,...,+1000}
                  y<0 -> yDI:=0 UND y>=0 -> yDI:=y
!!Alle Eingabewerte der Parameter des UP 'P_CONT' werden auf die angegebenen
  Grenzen geprüft. Sind Eingaben fehlerhaft, so werden sie auf die zuläs-
  sige Untergrenze=UG bzw. Obergrenze=OG UP-intern korrigiert.
################################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                    nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten pa-
                     rametriert werden und die Outputs dürfen Lokalvariablen sein.
                     Lokalvariable sind in den Netzwerken nur in aufsteigender
                     Richtung lesbar!
################################################################################
Aufbau der Struktur für einen P-Regler, dessen Anfangsadresse 'V300.0' ist:
P_1_01_ENABLE     V300.0  [PI]: Input ENABLE
P_1_02_x          VW302   [PI]: Input x            {-32768,...,+32767}*10^m
P_1_03_SH         VW304   [PI]: Input SH           {-32768,...,+32767}*10^m
P_1_04_y          VW306   [PO]: Output y       {-1000,...,0,...,+1000}
P_1_05_yIN        VW308   [PO]: Output yIN              {1000,...,1,0}
P_1_06_yDI        VW310   [PO]: Output yDI              {0,1,...,1000}
P_1_07_w          VW312   [W]:  w                   {-32768,...,+32767}*10^m
P_1_08_w_SH_MIN   VW314   [W]:  w_SH_MIN           {-32768,...,+32767}*10^m
P_1_09_w_SH_MAX   VW316   [W]:  w_SH_MAX           {-32768,...,+32767}*10^m
P_1_10_XP     VW318   [W]:  XP<=0 -> UP-intern :=1 {1,2,...,32767}*10^m
P_1_11_w_eff        VW320   [R]:  w_eff=(w+SH) -> wirkender Sollwert
################################################################################
Für jeden P-Regler sind folgende Parameter vom Typ '[W]':
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
w        [INT]: Sollwert, Führungsgröße                {-32768,...,+32767}*10^m
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
w_SH_MIN [INT]: MINIMUM für 'w:=[w+SH(-)]'             {-32768,...,+32767}*10^m
                Gilt bereits 'w<w_SH_MIN', dann kann 'w' durch SH(-) nicht noch
                zu kleineren Werten geschoben werden.
                !! w_SH_MIN ist keine Begrenzung für die Sollwerteingabe 'w' !!
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
w_SH_MAX [INT]: MAXIMUM für 'w:=[w+SH(+)]'            {-32768,...,+32767}*10^m
                Gilt bereits 'w>w_SH_MAX', dann kann 'w' durch SH(+) nicht noch
                zu größeren Werten geschoben werden.
                !! w_SH_MAX ist keine Begrenzung für die Sollwerteingabe 'w' !!
!! Relationen, wie w_SH_MIN >= w_SH_MAX verursachen lediglich, und das ist ab-
   hängig von der Lage von 'w', daß nur SH(-) ODER nur SH(+) ODER keine Soll-
   wertschiebung möglich ist.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
XP      [INT]: Proportionalband                          {1,2,...,32767}*10^m
               Ist 'Kp' die Proportionalverstärkung, so gilt: 'XP = 1/Kp'
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Begriffe:
HMI      = SIMATIC Human Machine Interfaces
x        = Istwert  der Regelgröße        x         {-32768,...,+32767}*10^m
w        = Sollwert der Regelgröße        w         {-32768,...,+32767}*10^m
w_eff    = wirkender Sollwert   w_eff=(w+SH)        {-32768,...,+32767}*10^m
```

```
SH       = Sollwertschiebung              SH        {-32768,...,+32767}*10^m
SH(-)    = negativer Bereich von SH       SH(-)       {-32768,...,-1,0}*10^m
SH(+)    = positiver Bereich von SH       SH(+)       {0,+1,...,+32767}*10^m
w_SH_MIN = MINIMUM für w:=[w+SH(-)]  w_SH_MIN     {-32768,...,+32767}*10^m
           Gilt bereits 'w < w_SH_MIN', dann kann 'w' durch SH(-) nicht noch
           zu kleineren Werten geschoben werden.
           !! w_SH_MIN ist keine Begrenzung für die Sollwerteingabe 'w' !!
w_SH_MAX = MAXIMUM für w:=[w+SH(+)]  w_SH_MAX     {-32768,...,+32767}*10^m
           Gilt bereits 'w > w_SH_MAX', dann kann 'w' durch SH(+) nicht noch
           zu größeren Werten geschoben werden.
           !! w_SH_MAX ist keine Begrenzung für die Sollwerteingabe 'w' !!
           !!Relationen, wie w_SH_MIN >= w_SH_MAX verursachen lediglich, und
           das ist abhängig von der Lage von 'w', daß nur SH(-) ODER nur SH(+)
           ODER keine Sollwertschiebung möglich ist.
XP       = Proportionalband               XP            {1,2,...,32767}*10^m
xW       = (x-w) = Regelabweichung        xW        {-65536,...,+65534}*(10^m)
y        = Stellgröße                      y          {-1000,...,+1000}*0.1%
yIN      = Stellgröße umgekehrt wirkend  yIN             {1000,...,1,0}*0.1%
yDI      = Stellgröße direkt    wirkend  yDI             {0,1,...,1000}*0.1%
10^m     = gemeinsamer Koeffizient aller damit gekennzeichneter Größen, der sich
           bei der Berechnung herauskürzt. Der korrekte Wert ist nur als fiktive
           Kommastelle oder angehängte Nullen beim BuB wichtig.
m        = Exponent zur Basis 10          m {-t,...,-3,-2,-1,0,1,2,3,...,+t}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Die Stellgröße wird wie folgt berechnet:

       xW*1000
   y = -------
         Xp

und begrenzt:
  -1000<=y<=+1000  (Das gilt damit auch für yDI, yIN!)

Wobei gilt:
   xW=(x-w); w:=w_eff=(w+SH); (Siehe aber auch wSH_MIN und wSH_MAX !)
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Darstellung der Zusammenhänge 'w+SH' mit 'y', 'yIN' und 'yDI'
```



++
yIN: y<=0 -> yIN:=|y| UND y>0 -> yIN:=0; yDI: y<0 -> yDI:0 UND y=>0 -> yDI:=y



SUBROUTINE_BLOCK PID_CONT:SBR59
TITLE=UNTERPROGRAMM: 'PID_CONT'
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken 213 bis 217 benutzt!
!!Dieses Unterprogramm ist nur gemeinsam mit dem UP 'INIT_TIMER_ACK'
!!und dem UP 'DELTA_t' lauffähig.

##
Kurzbeschreibung:
Das UP 'PID_CONT' stellt einen Proportional-, Integral- und Differentialregler dar,dessen Parameter im INTEGER-FORMAT eingegeben werden. Der UP-Input'SH' ermöglicht Sollwertschiebungen für Kaskadenregelungen. Ober- und Untergrenze der Sollwertschiebung sind parametrierbar.
Der Regler besitzt folgende 3 Norm-Ausgangssignale im Bereich {-1000,...,+1000}:
'y' {-1000,...,0,...,+1000}, als direktwirkendes Signal, dessen Vorzeichen durch das Vorzeichen der Regelabweichung '(x-w)' bestimmt wird.
'yIN' {+1000,...,0} als umgekehrtwirkendes Signal für '(x-w)<=0 :=|-y| ODER :=0'
'yDI' {0,...,+1000} als direktwirkendes Signal für '(x-w)>=0 := +y ODER :=0'
Alle Parameter des Reglers sind Festkommazahlen. Aus diesem Grund wird statt der Proportionalverstärkung 'K' das Proportionalband 'XP' als Parameter des Reglers verwendet - (XP=1/K!). Die Nachstellzeit 'TN' und Vorhaltezeit 'TV' sind ebenfalls Integerzahlen mit der Maßeinheit [0.1s]. Die einzelnen Komponenten der Stellgröße - der Proportional-, der Integral- und der Differentialanteil - sind separat ein- und ausschaltbar. Der Integralanteil wird begrenzt und damit das 'HOCHINTEGRIEREN' des Reglers verhindert.
Für den D-Anteil ist eine Haltezeit 'tH_yD' konfigurierbar. Der D-Anteil wirkt als Trenderkennung und zeigt kein Sprungverhalten. Die Ansteuerung von Stellgliedern wird dadurch optimal mit PID-Verhalten möglich.
In der Symboltabelle sind für alle Proportionalregler 52BYTE lange Variablen-Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der IN- und OUTPUTS des Reglers benutzt werden müssen. Die Adresse der Anfangsvariablen dieser Strukturen ist über den INPUT 'PRV52BYTE' an das Unterprogramm zu übergeben. Alle mit '[W]' gekennzeichneten Variablen können wahlweise im OB1, im Datenbaustein oder das HMI-System parametriert werden.
!SIMATIC HMI = Human Machine Interfaces!!
##
in:
ENABLE [BOOL]: =0 -> DISABLE: y:=yIN:=yDI:=0; =1 -> ENABLE -> Berechnung
x [INT]: Istwert der Regelgröße {-32768,...,+32767}*10^m
SH [INT]: Sollwertschiebung/Kaskadeninput {-32768,...,+32767}*10^m
Die Schiebeweite wird durch die Eingabewerte 'w_SH_MIN' und und 'w_SH_MAX' wie folgt begrenzt:
SH>0:= SH(+) -> [w+SH(+)]<=wSH_MAX; w > wSH_MAX -> SH:=0
SH<0:= SH(-) -> [w+SH(-)]>=wSH_MIN; w < wSH_MIN -> SH:=0

```
PRV52BYTE [DINT]: Anfangsadresse eines 52 BYTE langen Variablen-
                  speicherbereiches, dessen Struktur in der Symboltabelle
                  festgelegt ist. Diese Struktur muß für jeden PID-Regler in
                  der Symboltabelle vorhanden sein. Weiter unten sind die
                  einzelnen INPUTS in diese Struktur näher erläutert.
                  Ist z.B. das erste Element dieser Struktur 'V600.0', so
                  ist der Input 'PRV52BYTE' mit '&VB600' zu parametrieren.
out:
y           [INT]: Stellgröße  = Normsignal            {-1000,...,0,...,+1000}
yIN         [INT]: Stellgröße umgekehrt wirkend               {+1000,...,+1,0}
                   y>0 -> yIN:=0 UND y<=0 -> yIN:=-y
yDI         [INT]: Stellgröße direkt wirkend                  {0,+1,...,+1000}
                   y<0 -> yDI:=0 UND y>=0 -> yDI:=y
!!Alle Eingabewerte der Parameter des UP 'PID_CONT' werden auf die angegebenen
  Grenzen geprüft. Sind Eingaben fehlerhaft, so werden sie auf die zuläs-
  sige Untergrenze=UG bzw. Obergrenze=OG UP-intern korrigiert.
################################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                     nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten para-
                     metriert werden und die Outputs dürfen Lokalvariablen sein.
                     Lokalvariable sind in den Netzwerken nur in aufsteigender
                     Richtung lesbar!
################################################################################
Aufbau der Struktur für einen PID-Regler, dessen Anfangsadresse 'V600.0' ist:
PID_1_01_ENABLE       V600.0  [PI]: Wie FC-Input ENABLE
PID_1_02_x            VW602   [PI]: Input x               {-32768,...,+32767}*10^m
PID_1_03_SH           VW604   [PI]: Input SH              {-32768,...,+32767}*10^m
PID_1_04_y            VW606   [PO]: Output y                {-1000,...,+1000}
PID_1_05_yIW          VW608   [PO]: Output yIN                 {1000,...,1,0}
PID_1_06_yDI          VW610   [PO]: Output yDI                 {0,1,...,1000}
PID_1_07_w            VW612   [W]:  w                       {-32768,...,+32767}*10^m
PID_1_08_w_SH_MIN     VW614   [W]:  w_SH_MIN               {-32768,...,+32767}*10^m
PID_1_09_w_SH_MAX     VW616   [W]:  w_SH_MAX               {-32768,...,+32767}*10^m
PID_1_10_XP           VW618   [W]:  XP<=0 -> UP-intern :=1     {1,2,...,32767}*10^m
PID_1_11_TN           VW620   [W]:  TN<=0 -> kein I-Verhalten  {0,1,...,32767}*0.1s
PID_1_12_TV           VW622   [W]:  TV<=0 -> kein D-Verhalten  {0,1,...,32767}*0.1s
PID_1_13_tH_yD        VW624   [W]:  tH_yD<=0 -> FC-intern :=1  {1,2,...,32767}*1ms
PID_1_14_NSW          V626.0  [W]:  =1 -> bei Änderung von w_eff, XP oder TN:
                                    Neuberechnung der Stellgröße
PID_1_15_yP_aus       V626.1  [W]:  =1 -> yP = ausgeschaltet
PID_1_16_yI_aus       V626.2  [W]:  =1 -> yI = ausgeschaltet
PID_1_17_yD_aus       V626.3  [W]:  =1 -> yD = ausgeschaltet
PID_1_18_w_eff        VW628   [R]:  w_eff=(w+SH) -> wirkender Sollwert
PID_1_19_yP_LIM       VW630   [R]:  yP begrenzt              {-1000,...,+1000}
PID_1_20_yI_LIM       VW632   [R]:  yI begrenzt              {-1000,...,+1000}
PID_1_21_yD_LIM       VW634   [R]:  yD begrenzt              {-1000,...,+1000}
PID_1_22_t_alt        VW636   [R]:  Altwert TIMER              {0,1,...,32767}
PID_1_23_TIMER_tH_yD  VW638   [R]:  TIMER für Haltezeit yD     {0,1,...,32767}
PID_1_24_yI_LIM_REAL  VD640   [R]:  yI begrenzt REAL     {-1000.0,...,+1000.0}
PID_1_25_xWn_alt      VD644   [R]:  Altwert xW(n-k)         {-65536,...,+65534}
PID_1_26_XP_alt       VW648   [R]:  Altwert XP zur Erfassung der Änderung von XP
```

```
PID_1_27_TN_alt       VW650   [R]:  Altwert TN zur Erfassung der Änderung von TN
##############################################################################
Für jeden PID-Regler sind folgende Parameter vom Typ '[W]':
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
w        [INT]: Sollwert, Führungsgröße                 {-32768,...,+32767}*10^m
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
w_SH_MIN [INT]: MINIMUM für 'w:=[w+SH(-)]'              {-32768,...,+32767}*10^m
                Gilt bereits 'w<w_SH_MIN', dann kann 'w' durch SH(-) nicht noch
                zu kleineren Werten geschoben werden.
                !! w_SH_MIN ist keine Begrenzung für die Sollwerteingabe 'w' !!
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
w_SH_MAX [INT]: MAXIMUM für 'w:=[w+SH(+)]'             {-32768,...,+32767}*10^m
                Gilt bereits 'w>w_SH_MAX', dann kann 'w' durch SH(+) nicht noch
                zu größeren Werten geschoben werden.
                !! w_SH_MAX ist keine Begrenzung für die Sollwerteingabe 'w' !!
!! Relationen, wie w_SH_MIN >= w_SH_MAX verursachen lediglich, und das ist ab-
   hängig von der Lage von 'w', daß nur SH(-) ODER nur SH(+) ODER keine Soll-
   wertschiebung möglich ist.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
XP       [INT]: Proportionalband                          {1,2,...,32767}*10^m
               Ist 'Kp' die Proportionalverstärkung, so gilt: 'XP = 1/Kp'
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
TN       [INT]: Nachstellzeit                             {0,1,...,32767}*0.1s
             TN=0 -> der Integral-Anteil 'yI' wird nicht berechnet.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
TV       [INT]: Vorhaltezeit                              {0,1,...,32767}*0.1s
             TV=0 -> der Differential-Anteil 'yD' wird nicht berechnet.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
tH_yD    [INT]: Haltezeit                                 {1,2,...,10000}*0.01s
             Für diese Zeit wird ein berechneter 'yD-Anteil' konstant gehalten
             und erst nach Ablauf der Zeit neu berechnet. Dadurch wirkt 'yD'
             nicht als sprunghafte Änderung. Es geht als 'Trenderkennung' in
             die berechnete Stellgröße ein und berücksichtigt das Totzeit-
             verhalten von Sensoren und Wandlungszeiten von Analogeingangs-
             baugruppen.
             Empfohlene Einstellung: '200*0.01s = 2s'.
             Siehe auch Anmerkung zu den Berechnungsgrundlagen!
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
NSW     [BOOL]: =1 -> Wenn die Werte für 'w_eff, XP und TN' durch Eingaben ver-
             ändert werden, dann wird 'yI':=0 und 'yD':=0 und mit den geän-
             derten Werten neu berechnet. Auch alle zur Berechnung gespeicher-
             ten Altwerte :=0!
             Wird lediglich 'TN' UND/ODER 'XP' verändert, dann wird nur der
             Integralanteil 'yI':=0 neu berechnet.
             =0 -> Werden neue Sollwerte eingegeben, dann werden diese Werte
             übernommen, aber eine neue Stellgröße auf der Basis der bishe-
             rigen Altwerte berechnet.
             !!Für Folgeregler bei Kaskadenregelungen muß 'NSW:=0'.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
yP_aus [BOOL]: =0 -> yP ist eingeschaltet
             =1 -> yP ist ausgeschaltet
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
yI_aus [BOOL]: =0 -> yI ist eingeschaltet, wenn TN>0 ist !
               =1 -> yI ist ausgeschaltet, unabhängig von 'TN'!
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
yD_aus [BOOL]: =0 -> yD ist eingeschaltet, wenn TV>0 ist !
             =1 -> yD ist ausgeschaltet, unabhängig von 'TV'!
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Begriffe:

HMI	= SIMATIC Human Machine Interfaces		
x	= Istwert der Regelgröße	x	{-32768,...,+32767}*10^m
w	= Sollwert der Regelgröße	w	{-32768,...,+32767}*10^m
w_eff	= wirkender Sollwert w_eff=(w+SH)		{-32768,...,+32767}*10^m
SH	= Sollwertschiebung	SH	{-32768,...,+32767}*10^m
SH(-)	= negativer Bereich von SH	SH(-)	{-32768,...,-1,0}*10^m
SH(+)	= positiver Bereich von SH	SH(+)	{0,+1,...,+32767}*10^m
w_SH_MIN	= MINIMUM für w:=[w+SH(-)]	w_SH_MIN	{-32768,...,+32767}*10^m

Gilt bereits 'w < w_SH_MIN', dann kann 'w' durch SH(-) nicht noch zu kleineren Werten geschoben werden.
!! w_SH_MIN ist keine Begrenzung für die Sollwerteingabe 'w' !!

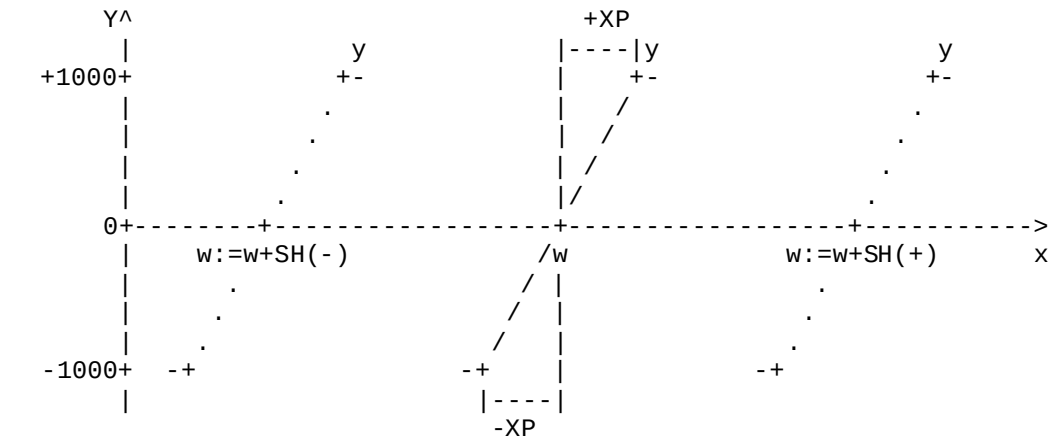
w_SH_MAX	= MAXIMUM für w:=[w+SH(+)]	w_SH_MAX	{-32768,...,+32767}*10^m

Gilt bereits 'w > w_SH_MAX', dann kann 'w' durch SH(+) nicht noch zu größeren Werten geschoben werden.
!! w_SH_MAX ist keine Begrenzung für die Sollwerteingabe 'w' !!
!!Relationen, wie w_SH_MIN >= w_SH_MAX verursachen lediglich, und das ist abhängig von der Lage von 'w', daß nur SH(-) ODER nur SH(+) ODER keine Sollwertschiebung möglich ist.

XP	= Proportionalband	XP	{1,2,...,32767}*10^m
TN	= Nachstellzeit	TN	{0,1,...,32767}*0.1s; TN=0 -> yI:=0
TV	= Vorhaltezeit	TV	{0,1,...,32767}*0.1s; TV=0 -> yD:=0

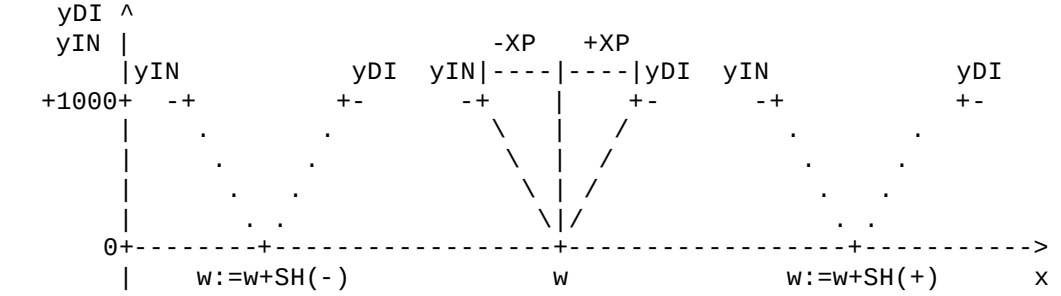
xW	= (x-w) = Regelabweichung	xW	{-65536,...,+65534}*(10^m)
y	= Stellgröße	y	{-1000,...,+1000}*0.1%
yIN	= Stellgröße umgekehrt wirkend	yIN	{1000,...,1,0}*0.1%
yDI	= Stellgröße direkt wirkend	yDI	{0,1,...,1000}*0.1%
yP	= Proportionalanteil in y	yP	{-1000,...,+1000}
yI	= Integralanteil in y	yI	{-1000,...,+1000}
yD	= Differentialanteil in y	yI	{-1000,...,+1000}
n	= CPU-Zyklusnummer		
dTZ	= Die an derselben Programmstelle gemessene, zwischen 2 CPU-Zyklen vergangene Zeit.	dTZ_max=16000ms	
dTI	= Zeit 'dT' für das Integrationsintervall	dTI=dTZ	
dTD	= Zeit 'dT' für das Differentiationsintervall.	dTD=tH_yD	
T(n)	= Timer-Neuwert im CPU-Zyklus (n)		
T(n-1)	= Timer-Altwert im CPU-Zyklus (n-1)		
T(n-k)	= Timer-Altwert im CPU-Zyklus (n-k)		
10^m	= gemeinsamer Koeffizient aller damit gekennzeichneter Größen, der sich bei der Berechnung herauskürzt. Der korrekte Wert ist nur als fiktive Kommastelle oder angehängte Nullen beim HMI-System wichtig.		
m	= Exponent zur Basis 10	m	{-t,...,-3,-2,-1,0,1,2,3,...,+t}
tH_yD	= Haltezeit yD	tH_yD	{1,2,...,32767}*0.001s

++
Darstellung der Zusammenhänge 'w+SH' mit 'y', 'yIN' und 'yDI'



++

yIN: y<=0 -> yIN:=|y| UND y>0 -> yIN:=0; yDI: y<0 -> yDI:0 UND y=>0 -> yDI:=y



++
Die Stellgröße 'y' berechnet sich aus 3 Anteilen:
 y=yP+yI+yD;

Begrenzungen der Anteile:
 yP: -1000<=yP<=+1000 (!!Nur in y, yDW, yUW)
 yI: yP>=0: -1000<=yI<=(1000-yP); yP<0: (-1000-yP)<=yI<=+1000
 yD: -1000<=yD<=+1000

Unbegrenzt:
 yP_UNLIM: -65536000<=yP<=+65534000

Der Wert 'yP' ist repräsentativ für die tatsächlich vorhandene Regelabweichung.
Deshalb wird das unbegrenzte 'yP' wird zu dem begrenzten 'yD' und dem begrenzten
'yI' addiert und anschließend zur Stellgröße 'y' begrenzt!

Stellgröße
 y=yP_UNLIM+yI+yD; nach Addition begrenzt auf: -1000<=y<=+1000

Berechnungsgrundlagen:
Proportionalanteil:
     1
 yP=---*xW(n)*1000;
     Xp

Integralanteil:

$$yI = \frac{1}{TI} \sum_{i=0}^{i=n} xW(n)*[T(n)-T(n-1)]*1000;$$

Differentialanteil:

$$yD = TD * \frac{[xW(n)-xW(n-k)]}{[T(n)-T(n-k)]} * 1000$$

!! Der berechnete Wert 'yD' wird bis zur nächsten, durch die Zeit 'tH_yD' bestimmten, neuen Berechnung, als konstant betrachtet und geht in 'y' ein. Dadurch wirkt 'yD' wie eine Trenderkennung. Der Wert für 'tH_yD' ist hinreichend groß einzugeben, daß überhaupt eine Änderung des Wertes '[xW(n)-xW(n-k)]' über Sensor und AI-Baugruppe erfaßt werden kann. Damit wird ein ständiges Springen von 'yD' zwischen '0' und dem berechneten Wert verhindert.

Mit den Substitutionen:

$$TN := \frac{TI}{XP} \quad \text{UND} \quad TV := TD*XP$$

Berechnet sich die Stellgröße y:

$$y = \frac{1000}{XP} * xW(n) + \frac{1000}{XP*TN} * \sum_{i=0}^{i=n} xW(n)*dTI + \frac{1000*[xW(n)-xW(n-k)]*TV}{XP*dTD}$$

Wobei gilt:

$dTI = dTZ = [T(n)-T(n-1)]; \quad dTD = [T(n)-T(n-k)] = tH\_yD;$

SUBROUTINE_BLOCK ZUSATZSEQUENZ:SBR60
TITLE=UNTERPROGRAMM: 'ZUSATZSEQUENZ'
!!ACHTUNG, IN DIESEM UP werden keine Sprungmarken benutzt!!

##
Kurzbeschreibung:
Mit dem UP 'ZUSATZSEQUENZ' kann ein beliebiger Bereich einer im Integerformat vorliegenden Variablen in ein Normstellsignal {0,...,+1000} gewandelt werden. Zusatzsequenzen werden häufig in der Klimatechnik benötigt, wo es z.B möglich ist, abhängig vom Außenluftzustand, eine energetisch günstigere Form der Luftaufbereitung zu wählen.
##
in:

x	[INT]: Abszisse des beliebigen Punktes P	{-32768,...,+32767}
x0	[INT]: Abszisse Punkt P0 mit der Ordinate=0	{-32768,...,+32767}
x1000	[INT]: Abszisse Punkt P1 mit der Ordinate=1000	{-32768,...,+32767}

out:

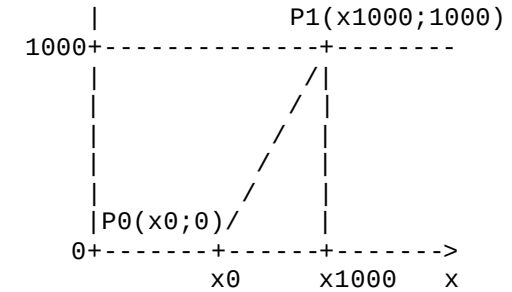
OK	[BOOL]: OK:=1; x0=x1000->OK:=0	
y	[INT]: Ordinate des beliebigen Punktes P	{0,...,+1000}

$$y = \frac{1000}{(x1000-x0)} * (x-x0); \quad y<0 \rightarrow y:=0; \quad y>1000 \rightarrow y:=1000;$$

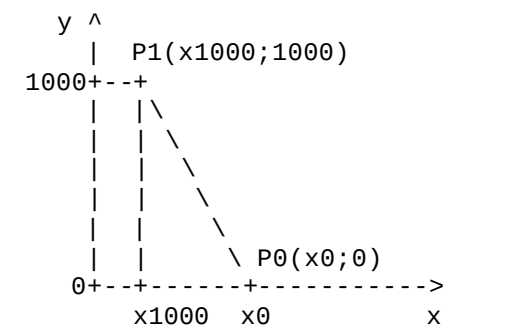
!!x1000 > x0 -> Zusatzsequenz direkt   wirkend =DI
!!x1000 < x0 -> Zusatzsequenz umgekehrt wirkend =IN
!!x1000 = x0 -> ERROR: y:=0 UND OK:=0

Beispiele:

 x0 < x1000 -> Zusatzsequenz ist direkt wirkend =DI



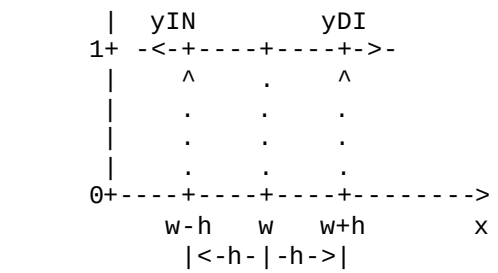
 x0 > x1000 -> Zusatzsequenz ist umgekehrt wirkend =IN



SUBROUTINE_BLOCK THREE_STEP_CONT:SBR61
TITLE=UNTERPROGRAMM: 'TREE_STEP_CONT'
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken 218 + 219 benutzt!
##
Kurzbeschreibung:
Das UP 'TREE_STEP_CONT' stellt einen klassischen 3-PUNKT-REGLER dar.
In der Symboltabelle sind für alle 3-PUNKT-REGLER 9BYTE lange Variablen-
Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der
IN- und OUTPUTS des Reglers benutzt werden müssen. Die Adresse der Anfangsvariab-
len dieser Strukturen ist über den INPUT 'PRV9BYTE' an das Unterprogramm zu über-
geben. Alle mit '[W]' gekennzeichneten Variablen können wahlweise im OB1, im Da-
tenbaustein oder das HMI-System parametriert werden.
!!SIMATIC HMI = Human Machine Interfaces!!
##
in:
ENABLE   [BOOL]: =0 -> DISABLE: y:=yIN:=yDI:=0; =1 -> ENABLE: Berechnung 'y'
x   [INT]: Istwert der Regelgröße   {-32768,...,+32767}
w   [INT]: Sollwert der Regelgröße   {-32768,...,+32767}
h   [INT]: Hysterese   {1,2,...,32767}
            !!Fehleingabe h<=0 -> UP-intern h:=1
PRV9BYTE   [DINT]: Anfangsadresse eines 9 BYTE langen Variablenspeicherberei-
                   ches, dessen Struktur in der Symboltabelle festgelegt ist.
                   Diese Struktur muß für jeden 3-PUNKT-REGLER in der Sym-
                   boltabelle vorhanden sein. Weiter unten sind die einzelnen
                   INPUTS in diese Struktur näher erläutert.

```
                Ist z.B. das erste Element dieser Struktur 'V450.0', so
                ist der Input 'PRV9BYTE' mit '&VB450' zu parametrieren.

out:
yIN       [BOOL]: Stellgröße INVERS:
                         x <= (w-h)  UND  ENABLE=1 -> yIN:=1
                (w-h)<   x <  w      UND  ENABLE=1 -> yIN=0/1 (KEINE ÄNDERUNG!)
                         x >= w      ODER ENABLE=0 -> yIN:=0
                (w-h)<   -32768                    -> yIN:=0

yDI       [BOOL]: Stellgröße DIREKT:
                (w+h)<= x           UND  ENABLE=1 -> yDI:=1
                 w <    x < (w+h)   UND  ENABLE=1 -> yDI=0/1 (KEINE ÄNDERUNG!)
                 w >=   x           ODER ENABLE=0 -> yDI:=0
                (w+h) >+32767                      -> yDI:=0
!!UP-intern werden alle Werte im 'DINT-Format' verglichen. Bei ungünstigen Werte-
  kombinationen  kann es vorkommen, daß 'w-h' oder 'w+h' außerhalb des Wertebe-
  reiches von 'x' liegen. In solchen Fällen ist 'yIN' ODER 'yDI' permanent :=0.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!!Alle Eingabewerte der Parameter des UP 'TREE_STEP_CONT' werden auf die angege-
  benen Grenzen geprüft. Sind Eingaben fehlerhaft, so werden sie auf die zuläs-
  sige Untergrenze=UG bzw. Obergrenze=OG UP-intern korrigiert.
################################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                     nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten para-
                     metriert werden und die Outputs dürfen Lokalvariablen sein.
                     Lokalvariable sind in den Netzwerken nur in aufsteigender
                     Richtung lesbar!
################################################################################
Aufbau der Struktur für einen Dreipunktregler, dessen Anfangsadresse 'V450.0' ist:
TSC_1_1_ENABLE     V450.0  [PI]: =0->DISABLE: y:=0; =1->ENABLE: BERECHNUNG yDI/yIN
TSC_1_2_x          VW452   [PI]: REGELGRÖßE=ISTWERT       {-32768,...,+32767}
TSC_1_3_w          VW454   [P]I: REGELGRÖßE=SOLLWERT      {-32768,...,+32767}
TSC_1_4_h          VW456   [PI]: HYSTERESE  (h=0 -> h:=1)    {1,2,...,+32767}
TSC_1_5_yDI        V458.0  [PO]: x<=w->yDI:=0;   w<x<(w+h)->yDI=0/1; x>=w+h->yDI=1
TSC_1_6_yIN        V458.1  [PO]: x>=w->yIN:=0;(w-h)<x<w   ->yIN=0/1; x<=w+h->yIN=1
TSC_1_7_PRV_2BOOL  V458.2+3 [R]: 2 BIT private Speicher
################################################################################
Begriffe:
HMI      = SIMATIC 'Human Machine Interfaces'
IN_DI    = Wirkrichtung des Hystereseschalters IN_DI  {0,1}
x        = Istwert  der Regelgröße                x  {-32768,...,+32767}
w        = Sollwert der Regelgröße                w  {-32768,...,+32767}
h        = Hysterese  (h<=0 -> h:=1)              h {1,2,...,+32767}
yIN      = Stellgröße INVERS wirkend             yIN {0,1}
yDI      = Stellgröße DIREKT wirkend             yDI {0,1}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Beispiel:

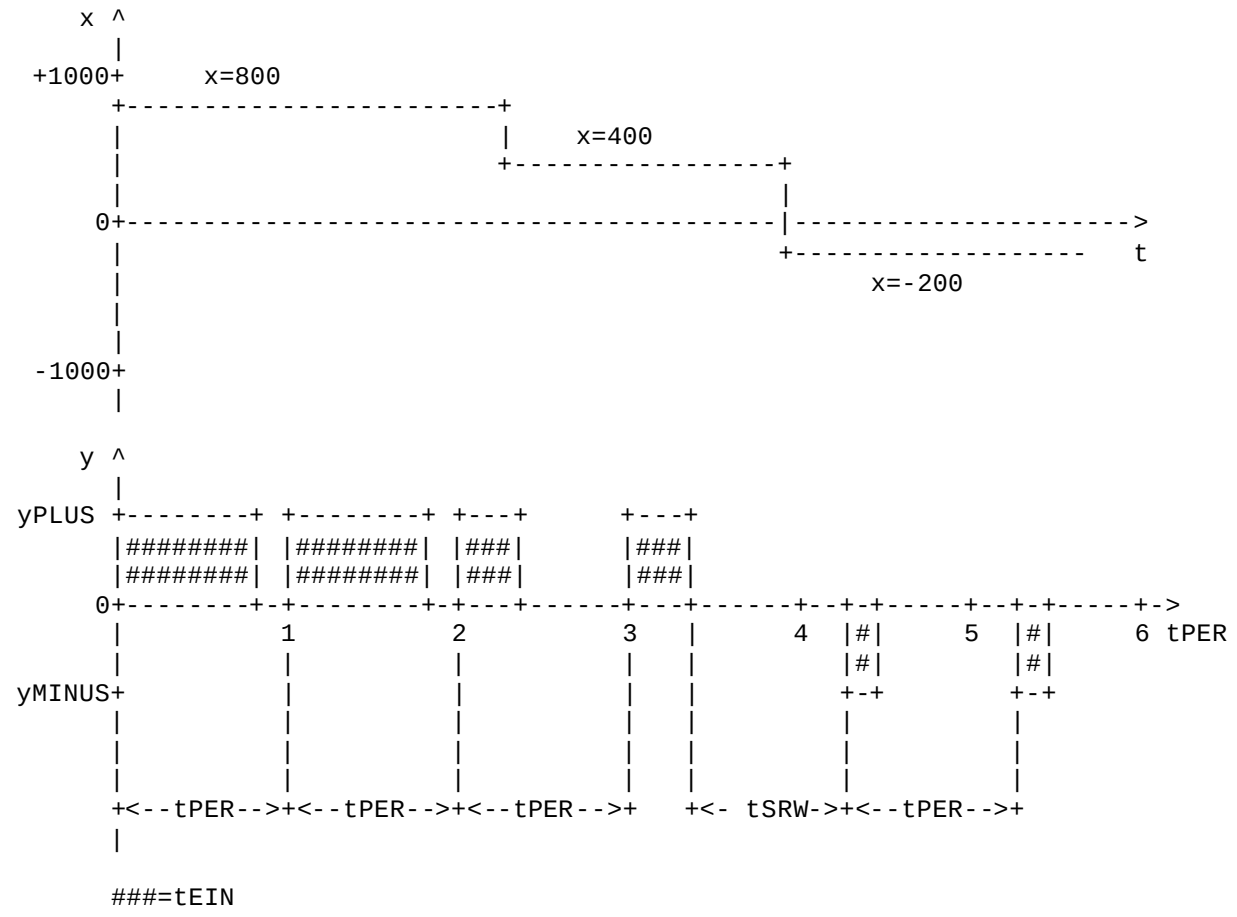


SUBROUTINE_BLOCK AD_TAKTGEBER:SBR62
TITLE=UNTERPROGRAMM: 'AD_TAKTGEBER'
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken 220 bis 226 benutzt!
!!Dieses Unterprogramm ist nur gemeinsam mit dem UP 'INIT_TIMER_ACK'
!!und dem UP 'DELTA_t' lauffähig.

##
Kurzbeschreibung:
Mit dem UP 'AD_TAKTGEBER' kann ein analoges Normsignal {-1000,...,0,...+1000}
in die periodisch taktenden, impulsdauermodulierten digitalen Stellsignale
'yPLUS' und 'yMINUS' gewandelt werden.
Die Periodendauer, eine minimale Ein-/Ausschaltzeit und eine Totzeit bei Schalt-
richtungswechsel muß parametriert werden.
In der Symboltabelle sind für alle 'AD_TAKTGEBER' 18BYTE lange Variablen-
Strukturen festgelegt, die, wie nachstehend beschrieben, zum parametrieren der
IN- und OUTPUTS des 'AD_TAKTGEBER' benutzt werden müssen.
Die Adresse der Anfangsvariablen dieser Strukturen ist über den INPUT 'PRV18BYTE'
an das Unterprogramm zu übergeben. Alle mit '[W]' gekennzeichneten Variablen können
wahlweise im OB1, im Datenbaustein oder das HMI-System parametriert werden.
Mit dem 3-Punkt-Analog-Digitalwandler können impulsdauermoduliert Stellglieder
vom Typ 'AUF-ZU' oder 'AUS-EIN' über analoge Reglerausgangssignale angesteuert
werden.
Z.B. lassen sich damit Magnetventile, 3-Punkt-Stellantriebe, Dosierpumpen, Thyri-
storschalter für Elektroheizungen usw. als Stellglieder in Regelkreisen einsetzen.
!!SIMATIC HMI = Human Machine Interfaces!!
##
in:
x [INT]: Stellgröße {-1000,...,+1000}
UP-intern wird 'x' wie folgt begrenzt:
x < -1000 -> x:=-1000 ODER x > +1000 -> x:=+1000
PRV18BYTE [DINT]: Anfangsadresse eines 18 BYTE langen Variablenspeicherberei-
ches, dessen Struktur in der Symboltabelle festgelegt ist.
Diese Struktur muß für jeden 'AD_TAKTGEBER' in der Sym-
boltabelle vorhanden sein. Weiter unten sind die einzelnen
INPUTS in diese Struktur näher erläutert.
Ist z.B. das erste Element dieser Struktur 'VW1000', so
ist der Input 'PRV18BYTE' mit '&VB1000' zu parametrieren.

out:
yPLUS [BOOL]: =1, wenn x > 0 UND 'tEIN > t_min_E_A' UND EINSCHALTZEIT aktiv
=0, wenn x > 0 UND 'tAUS > t_min_E_A' UND AUSSCHALTZEIT aktiv
ODER wenn x = 0 UND/ODER wenn SRW = HIGH ist.
yMINUS [BOOL]: =1, wenn x < 0 UND 'tEIN > t_min_E_A' UND EINSCHALTZEIT aktiv
=0, wenn x < 0 UND 'tAUS > t_min_E_A' UND AUSSCHALTZEIT aktiv
ODER wenn x = 0 UND/ODER wenn SRW = HIGH ist.
++

```
!!Alle Eingabewerte der Parameter des UP 'AD_TAKTGEBER' werden auf die angege-
  benen Grenzen geprüft. Sind Eingaben fehlerhaft, so werden sie auf die zuläs-
  sige Untergrenze=UG bzw. Obergrenze=OG UP-intern korrigiert.
###############################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                    nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten para-
                     metriert werden und die Outputs dürfen Lokalvariablen sein.
                     Lokalvariable sind in den Netzwerken nur in aufsteigender
                     Richtung lesbar!
###############################################################################
Aufbau der Struktur für einen 'AD_TAKTGEBER', dessen Anfangsadresse 'VW1000' ist:
AD_TKT_1_1_x          VW1000 [PI]: Stellgröße         {-1000,...,0,...,1000}
AD_TKT_1_2_yPLUS      V1002.0[PO]: =1, wenn x > 0 UND "tEIN > t_min_EIN" !
AD_TKT_1_3_yMINUS     V1002.1[PO]: =1, wenn x < 0 UND "tEIN > t_min_EIN" !
AD_TKT_1_4_tPER       VW1004  [W]: Periodendauer              {0,1,...,32767}*0,1s
AD_TKT_1_5_tMIN_E_A   VW1006  [W]: Mindest EIN-/AUS-Schaltzeit{0,1,...,32767}*0,1s
AD_TKT_1_6_tAUS_SRW   VW1008  [W]: Ausschaltzeit bei SRW      {0,1,...,32767}*0,1s
AD_TKT_1_7_PRV_1_DINT VD1010  [R]: Private Speicher 1: 1xDINT
AD_TKT_1_8_PRV_2_DINT VD1014  [R]: Private Speicher 2: 1xDINT
###############################################################################
Begriffe:
HMI       = SIMATIC 'Human Machine Interfaces'
x         = Normierte Stellgröße                          {-1000,...,+1000}
yPLUS     = Digitales Stellsignal, impulsdauermoduliert, wenn x>0 ist
yMINUS    = Digitales Stellsignal, impulsdauermoduliert, wenn x<0 ist
tPER      = Dauer einer Periode                             {0,1,...,32767}*0,1ms
tMIN_E_A  = Minimale Ein- / Ausschaltzeit                   {0,1,...,32767}*0,1ms
tAUS_SRW  = Ausschaltzeit bei SRW                           {0,1,...,32767}*0,1ms
SRW       = Signalrichtungswechsel, d.h. von Signal 'yPLUS=1'
            soll direkt auf Signal 'yMINUS=1' geschaltet werden.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Berechnungsgrundlagen:
!!Periodendauer
  Die Periodendauer besteht aus einer EIN- und AUS-SCHALTZEIT, die sich wie
  folgt berechnen:
            |x| * tPER
   tEIN = ------------    UND   tAUS = (tPER - tEIN)
              1000
!!Ist 'x > 0' , dann wird 'yPLUS' oder  wenn 'x < 0', dann wird 'yMINUS'
  eingeschaltet.
!!Gilt 'tPER=0' ODER 'x=0' ODER 'tEIN <= tMIN_E_A, dann wird unabhängig vom
  anliegenden Schaltbefehl - 'x>0' ODER 'x<0' - 'yPLUS UND yMINUS :=0'!
!!Ist  'tAUS <= tMIN_E_A, dann 'yPLUS ODER yMINUS permanent :=1', wenn 'tPER>0'
  UND |x|>0 ist!
!!Jede Periode wird mit der Einschaltzeit gestartet.
!!Bei Signalrichtungswechsel wird die Ausschaltzeit 'tAUS_SRW' gestartet. Läuft
  gerade die Auszeit 'tAUS', dann ist die aufgelaufene Ausschaltzeit automatisch
  Bestandteil der Ausschaltzeit 'tAUS_SRW'.

 Grafische Darstellung des Zusammenhanges zwischen der analogen Stellgröße 'x'
 und der Dauer der taktenden Stellgrößen 'yPLUS' / 'yMINUS':
```



SUBROUTINE_BLOCK AD_SM:SBR63
TITLE=UNTERPROGRAMM: 'AD_SM'
!!ACHTUNG, IN DIESEM UP werden die Sprungmarken (227 bis 239)benutzt!
!!Dieses Unterprogramm ist nur gemeinsam mit dem UP 'INIT_TIMER_ACK'
!!und dem UP 'DELTA_t' lauffähig.

##
Kurzbeschreibung:
Mit dem UP 'AD_SM' können digitale Stellantriebe von einem analogen Normsignal
mit dem Wertebereich {0,...,1000} angesteuert werden. Aus dem absoluten Wert
dieses Signales und dem Stellungs-Istwert {0,...,1000}*0.1% des Stellantriebes
wird die Stellrichtung und die Zeitdauer des Stellbefehles 'AUF'/'ZU' berech-
net. Ist die Zeitdauer kleiner als das parametrierte Totband, so erfolgt keine
Verstellung des Stellantriebes.
Die Basis für die Berechnung der Stellzeit und des Stellungsistwertes ist die
Laufzeit des Stellantriebes gemäß technischer Dokumentation.

In der Symboltabelle sind für alle ANALOG-DIGITAL-WANDLER FÜR STELLMOTOREN 'AUF/ZU'
38BYTE lange Variablenstrukturen festgelegt, die, wie nachstehend beschrieben,
zum parametrieren der IN- und OUTPUTS des UP's 'AD_SM' benutzt werden müssen.
Die Adresse der Anfangsvariablen dieser Strukturen ist über den INPUT 'PRV38BYTE'

```
an  das Unterprogramm zu übergeben. Alle mit '[W]' gekennzeichneten Variablen
können wahlweise im OB1, im Datenbaustein oder das HMI-System parametriert werden.
Vom HMI-System kann der Stellantrieb von der Betriebsart 'AUTO' nach 'HAND'
umgeschaltet und im Bereich {0,...,1000}*0.1% angesteuert werden. Über den
FC-Input 'FREEZE' ist eine Sequenzverriegelung mehrerer Stellantriebe möglich.
Das UP 'AD/SM' läßt eine Laufzeit- = Endlagenüberwachung nach 3 MODIS zu. Neben
dem direkten Lesezugriff auf Speicher des 'DB_SM' kann aus dem 'INFO-Byte' der
Zustand des Stellantriebes 'AUF', 'ZU', 'STÖRUNG' (=Störung Laufzeitüberwa-
chung!), 'NEUWERT STÖRUNG' und 'HAND' jederzeit ermittelt werden.
!!SIMATIC HMI = Human Machine Interfaces!!
################################################################################

in:
QUITT       [BOOL]: =0->1 die 'STÖRUNG' wird mit internem PULS quittiert,
                     liegt aber solange an, bis die Ursache beseitigt ist. STÖ-
                     RUNGSURSACHEN (Siehe auch MOD !) gelten als beseitigt, wenn
                     bei 'ST_MS' keine Störung mehr anliegt. Die Störung 'ST_LZ'
                     ist vom Parameter 'MOD' wie folgt abhängig:
                     MOD 0: Es wird keine Störung generiert
                     MOD 1: Die geforderte Endlage muß anliegen ODER der SM fährt
                            aus den Endlagenstellungen heraus -> EA=0 UND EZ=0.
                            Ist EA=1 und EZ=1, dann ist die Störung quittierbar,
                            wird aber erst =0, wenn die Ursachen beseitigt sind.
                     MOD 2: Die Endlage EZ (=EZ ODER =EA) muß anliegen ODER der
                            SM fährt aus den Endlagenstellungen heraus UND der
                            SIGNALVERLAUF von 'EZ' wird '1-0-1'.
FREEZE      [BOOL]: Sequenzverriegelung:
                     =0 -> der Schaltzustand wird wie beschrieben berechnet und
                           von 'yZ/yA' gesteuert.
                     =1 -> der momentane Stellungs-Iswert 'y_PRZ'/'y_TIME' wird
                           eingefroren. Die Ausgänge 'yZ/yA' nehmen dann folgende
                           Schaltzustände an:
                           0 < y_PRZ < 1000  ->  yZ:=yA:=0
                           y_PRZ=0           ->  yZ:=1 + Endlagenüberwachung
                           y_PRZ=1000        ->  yA:=1 + Endlagenüberwachung



EZ          [BOOL]: Rückmeldung Endlage 'ZU'
                     MOD=0: Rückmeldung Endlage =0/1-> keine Bedeutung
                     MOD=1: Rückmeldung Endlage =1  -> Endlage   =   ZU
                                                =0  -> Endlage nicht ZU
                     MOD=2: Rückmeldung Endlage =1  -> Endlage   =   ZU ODER AUF
                                                =0  -> Endlage weder ZU noch AUF
EA          [BOOL]: Rückmeldung Endlage 'AUF'
                     MOD=0: Rückmeldung Endlage =0/1-> keine Bedeutung
                     MOD=1: Rückmeldung Endlage =1  -> Endlage   =   AUF
                                                =0  -> Endlage nicht AUF
                     MOD=2: Rückmeldung Endlage =0/1-> keine Bedeutung
MS           [BOOL]: =0 -> STÖRUNG: Die Motorschutzeinrichtung hat angesprochen

MOD          [INT]: MOD=0 -> Keine Rückmeldung der Endlagen, keine Überwachung
                    MOD=1 -> Rückmeldung der Endlagen UND Überwachung
                    MOD=2 -> Rückmeldung 'EZ'('EA' ODER 'EZ') UND Überwachung
                    Fehleingaben: MOD<0-> MOD:=0 ODER MOD>2 -> MOD:=2!
x            [INT]: Analoge  Stellgröße                 {0,1,...,1000}*0.1%
```

```
PRV38BYTE   [DINT]: Anfangsadresse eines 38 BYTE langen Variablenspeicherberei-
                    ches, dessen Struktur in der Symboltabelle festgelegt ist.
                    Diese Struktur muß für jeden 'AD_SM'-Stellantrieb in der Sym-
                    boltabelle vorhanden sein. Weiter unten sind die einzelnen
                    INPUTS in diese Struktur näher erläutert.
                    Ist z.B. das erste Element dieser Struktur 'VW1400', so
                    ist der Input 'PRV38BYTE' mit '&VB1440' zu parametrieren.
out:
yZ          [BOOL]: Digitale Stellgröße         =1  -> Stellmotor = ZU
yA          [BOOL]: Digitale Stellgröße         =1  -> Stellmotor = AUF
ST_LZ       [BOOL]: =1 -> Störung Laufzeit-/Endlagenüberwachung
                    !!AUS DER STÖRUNG 'ST_LZ' kann diagnostiziert werden, ob
                      der Stellantrieb einen mechanischen oder elektrischen
                      Defekt hat, die Endlagenschalter nicht mehr funktionieren
                      oder die elektrischen Verbindungen unterbrochen sind.
ST_MS      [BOOL]: =1 -> Störung Motorschutzeinrichtung
SST        [BOOL]: =1 -> Sammelstörung
                    !!Sammelstörung 'SST=1' bewirkt die sofortige Ausschaltung
                      des Stellantriebes -> 'yZ=yA:=0'!
INFO       [BYTE]: Informationsbyte über den Zustand des SM, mit dem BIT-Muster
                   'hsqAZaz0, wobei die BITS folgende Bedeutung haben:
                     BIT 0   = 0 : ohne Bedeutung
                     BIT 1   = z : =1  -> der SM = ZU
                     BIT 2   = a : =1  -> der SM = AUF
                     BIT 3   = Z : =1  -> der SM fährt ZU  / schließt
                     BIT 4   = A : =1  -> der SM fährt AUF / öffnet
                     BIT 5   = q : =1  -> Störung = Neuwert,nach Quittierung :=0
                     BIT 6   = s : =1  -> Störung Laufzeitüberwachung/Motorschutz
                     BIT 7   = h : =1  -> Die Stellgröße kommt vom HMI = 'xH'
!!Die Rückmeldungen der Endlagen beeinflussen NICHT den Zustand der Ausgänge .
!!Den Schaltzustand der Ausgänge beeinflussen nur 'x', 'FREEZE' UND 'SST'.
!!Bei Erstinbetriebnahme einer Anlage ODER Remanenzverlust ODER nach Austausch
  eines defekten Stellmotors ist es möglich, daß der BERECHNETE STELLUNGSISTWERT
  und die für MOD 0+2 BERECHNETEN ENDLAGENSTELLUNGEN mit den tatsächlichen HARD-
  WAREPOSITIONEN nicht übereinstimmen. Es kann zu einmaligen Störmeldungen kom-
  men, deren Ursache in undefinierten Speicherwerten des zu suchen ist.
  Nach Quittieren der Störung ist diese Störungsursache dauerhaft beseitigt.
  Die berechneten Positionen korrigieren sich automatisch, weil der SM in den
  ENDLAGENSTELLUNGEN permanent das Signal 'AUF=1' ODER 'ZU=1' erhält.
!!Der Input 'x' und alle Eingabewerte werden UP intern auf zulässige Werte
  überprüft und auf die angegebenen Grenzen korrigiert.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
!!Alle Eingabewerte der Parameter des UP 'AD_SM' werden auf die angegebenen
  Grenzen geprüft. Sind Eingaben fehlerhaft, so werden sie auf die zulässige
  Untergrenze=UG bzw. Obergrenze=OG UP-intern korrigiert.
################################################################################
In die Struktur müssen folgende Parameter eingegeben oder können gelesen werden:
Alle mit '[W]'       gekennzeichneten Parameter sind zwingend im OB1, im Daten-
                     baustein oder im HMI-System mit Werten zu versorgen.
Alle mit '[RW]'      gekennzeichneten Parameter können im AUTOMATIKBETRIEB nur
                     gelesen und müssen im HANDBETRIEB über das HMI-System mit
                     Werten versorgt werden.
Alle mit '[R]'       gekennzeichneten Parameter können nur gelesen werden
Alle mit '[PI]/[PO]' gekennzeichneten Parameter sind für die Parametrierung der
                     identisch bezeichneten In- und Outputs des zugeordneten UP's
                     reserviert.INPUTS müssen im OB1 beschrieben und OUTPUTS kön-
                     nen im OB1 und vom HMI-System nur gelesen werden. Statt der
                     der Variablen können alle Inputs auch mit Konstanten para-
```

metriert werden und die Outputs dürfen Lokalvariablen sein.
Lokalvariable sind in den Netzwerken nur in aufsteigender
Richtung lesbar!

```
###############################################################################
Aufbau der Struktur für einen ANALOG-DIGITAL-WANDLER FÜR STELLMOTOREN 'AUF/ZU',
dessen Anfangsadresse '&VB1200' ist:
AD_SM_1_01_QUITT      V1200.0[PI]: =1->Quittierung Störung mit internem Puls (+)
AD_SM_1_02_FREEZE     V1200.1[PI]: =1->der momentane Zustand wird eingefroren
AD_SM_1_03_EZ         V1200.2[PI]: =1->Endlagenschalter='ZU' (Siehe auch MODUS 2!)
AD_SM_1_04_EA         V1200.3[PI]: =1->Endlagenschalter = 'AUF'
AD_SM_1_05_MS         V1200.4[PI]: =0->die Motorschutzeinrichtung hat angesprochen
AD_SM_1_06_MOD        VW1202 [PI]: Modus der Endlagenerfassung        {0,1,2}
AD_SM_1_07_x          VW1204 [PI]: identisch mit Input x:      {0,1,...,1000}*0.1%
AD_SM_1_08_yZ         V1206.0[PO]: =1->Stellrichtung = ZU
AD_SM_1_09_yA         V1206.1[PO]: =1->Stellrichtung = AUF
AD_SM_1_10_ST_LZ      V1206.2[PO]: =1->Störmeldung Laufzeit / Endlagen
AD_SM_1_11_ST_MS      V1206.3[PO]: =1->Störmeldung Motorschutz
AD_SM_1_12_SST        V1206.4[PO]: =1->Sammelstörmeldung
AD_SM_1_13_INFO       VB1207 [PO]: Informationsbyte:2#hsqAZaz0
AD_SM_1_14_tLZ        VW1208  [W]: Laufzeit                   {1,2,...,32767}*0.1s
AD_SM_1_15_tUE        VW1210  [W]: Endlagen-Überwachungszeit  {0,1,...,32767}*0.1s
AD_SM_1_16_tTB        VW1212  [W]: Totband für Laufzeit       {0,1,...,32767}*0.01s
AD_SM_1_17_tMIN_AUS   VW1214  [W]: MINIMALE Auszeit           {0,1,...,32767}*0.01s
AD_SM_1_18_xAH          VW1216 [RW]: AH=0-> xAH:=x; AH=1->xAH:=xHAND vom HMI
AD_SM_1_19_TAST_AH    V1218.0 [W]: Umschalttaste AUTO/HAND
AD_SM_1_20_AH         V1218.1 [R]: FLIP-FLOP =0->xAUTO; =1->xHAND
AD_SM_1_21_E_ZU_INFO  V1218.2 [R]: =1 -> ZU  = Istwert Endlage 'ZU',  analog INFO
AD_SM_1_22_E_AUF_INFO V1218.3 [R]: =1 -> AUF = Istwert Endlage 'AUF', analog INFO
AD_SM_1_23_y_PRZ      VW1220  [R]: Stellungs-Istwert          {0,1,...,1000}*0.1%
AD_SM_1_24_PRV_1_DINT VD1222  [R]: Private Speicher 1: 1xDINT
AD_SM_1_25_PRV_2_DINT VD1226  [R]: Private Speicher 2: 1xDINT
AD_SM_1_26_PRV_3_DINT VD1230  [R]: Private Speicher 3: 1xDINT
AD_SM_1_27_PRV_4_DINT VD1234  [R]: Private Speicher 4: 1xDINT
###############################################################################
Erläuterungen zu den [W] / [RW] Operanden:
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
MOD     [INT]: MOD=0 -> Keine Rückmeldung der Endlagen, keine Überwachung
               MOD=1 -> Rückmeldung der Endlagen UND Überwachung
               MOD=2 -> Rückmeldung 'EZ'('EA' ODER 'EZ') UND Überwachung
               Fehleingaben: MOD<0-> MOD:=0 ODER MOD>2 -> MOD:=2!
!!Auch bei MOD=0 muß eine Stellzeit, das ist die Zeit in welcher der Stellan-
  trieb von einer Endlage zur anderen fährt, eingegeben werden. Nach Ablauf
  dieser Stellzeit werden im UP die Endlagen aktualisiert, die entspre-
  chend Stellrichtung erreicht würden:

  Die erreichte ENDLAGE wird bei MOD=0 im DB_SM in den Speicherstellen
 'E_ZU_INFO' und 'E_AUF_INFO' nach folgenden Kriterien hinterlegt und
  analog dazu in das INFOBYTE geschrieben:
  E_ZU_INFO [BOOL]: =1, wenn STELLBEFEHL=ZU  UND STELLUNGSISTWERT yPRZ=0
  E_AUF_INFO[BOOL]: =1, wenn STELLBEFEHL=AUF UND STELLUNGSISTWERT yPRZ=1000

!!Bei MOD=1 muß beim Stellbefehl 'y_ZU' die Rückmeldung der Endlage 'EZ' bzw.
  beim SB 'y_AUF' die Rückmeldung der Endlage 'EA' erfolgen. Liegen beide End-
  lagensignale gleichzeitig an ODER liegt nach dem Erreichen derEndlagenposi-
  tion UND nach Ablauf der Überwachungszeit 'tUE' nicht das der Endlagenposition
  entsprechende Endlagensignal an, dann wird 'STÖRUNG:=1' gesetzt.
  Die 'STÖRUNG' wird zurückgesetzt, WENN QUITTIERT WURDE UND KEINE STÖRUNGS-
  SITUATION ANLIEGT.
```

Die erreichte ENDLAGE wird bei MOD=1 im DB_SM in den Speicherstellen 'E_ZU_INFO' und 'E_AUF_INFO' hinterlegt und analog dazu in das INFOBYTE geschrieben:
E_ZU_INFO [BOOL]: =EZ, ist also identisch mit INPUT ENDLAGE 'EZ'
E_AUF_INFO[BOOL]: =EA, ist also identisch mit INPUT ENDLAGE 'EA'

!!Bei MOD=2 sind die Endlagenschalter hardwareseitig parallel zu schalten.
Es tritt dann am Eingang 'EZ' folgender Signalverlauf auf:

Stellbefehl=SB UND Ausgangstellung=AS	Signalverlauf an 'EZ'
SB x=100% ->'AUF' + AS 'ZU'	1 -> 0 -> 1
SB x=100% ->'AUF' + AS 'ZU<y<AUF'	0 -> 1
SB 0%<(x>y_PRZ)<100% ->'AUF' + AS 'ZU<y<AUF'	0 -> 0
SB 0%<(x<y_PRZ)<100% ->'ZU' + AS 'ZU<y<AUF'	0 -> 0
SB x=0% ->'ZU' + AS 'ZU<y<AUF'	0 -> 1
SB x=0% ->'ZU' + AS 'AUF'	1 -> 0 -> 1

Tritt der Signalverlauf '1-0-1' nicht ein ODER bleibt nach Ablauf der Überwachungszeit die Rückmeldung 'EZ'=0, dann wird der Antrieb als gestört gemeldet.
Die 'STÖRUNG' wird zurückgesetzt, WENN QUITTIERT WURDE UND KEINE STÖRUNGSSITUATION ANLIEGT.

Die erreichte ENDLAGE wird bei MOD=2 in den Speicherstellen 'E_ZU_INFO' und 'E_AUF_INFO' nach folgenden Kriterien hinterlegt und analog dazu in das INFOBYTE geschrieben:
E_ZU_INFO [BOOL]: :=1, wenn STELLBEFEHL=ZU UND STELLUNGSISTWERT yPRZ=0
UND EZ=1 MIT DEM SIGNALVERLAUF '?-0-1'!
E_AUF_INFO[BOOL]: :=1, wenn STELLBEFEHL=ZU UND STELLUNGSISTWERT yPRZ=1000
UND EZ=1 MIT DEM SIGNALVERLAUF '?-0-1' !
!!'E_ZU_INFO' UND/ODER 'E_AUF_INFO' werden immer dann zurückgesetzt, wenn 'EZ=0' geworden IST.
!!Für den FALL, daß 'EZ=1' permanent anliegt, bleibt die Stellungsrückmeldung unverändert.

++

tLZ [INT]: Laufzeit=Stellzeit 'ZU <-->'AUF' {1,2,...,32767}*0.1s

++

tUE [INT]: Überwachungszeit {1,2,...,32767}*0.1s

Bei den SB 'xZ' und 'xA' wird nach Ablauf der Stellzeit die Überwachungszeit 'tUE' gestartet. Wenn nach Ablauf von 'tUE' nicht die oben beschriebene Rückmeldung der Endlagen 'EZ' bzw. 'EA' anliegt, wird das Störungsbit 'STÖRUNG' gesetzt. Die Endlagenüberwachung ist bei MOD=0 unwirksam.

!!Liegt die Störung Laufzeit an, so wird nach Betätigung der Quittiertaste die Störung zurückgesetzt und die Überwachungszeit neu gestartet.

++

tTB [INT]: Totband {1,...,32767}*0.01s

Der analoge Stellbefehl 'x' wird in eine Stellzeit

$$x\_TIME = \frac{x * tLZ * 100}{1000\%}\ [ms]$$

umgerechnet und der Stellungs-Istwert in 'y_TIME' gespeichert.

Verändert sich 'x_TIME' in den Grenzen

(y_TIME-tTB)<= x_TIME <= (y_TIME+tTB)

wobei alle Werte in [ms] umgerechnet sind, dann bleiben die digitalen Stellungssignale 'yZ' UND 'yA' ausgeschaltet!

!!Das Totband ist unwirksam für die Endstellungen:
x=0% -> SB 'ZU' :=1
x=100% -> SB 'AUF':=1

++

tMIN_AUS [INT]: Minimale Auszeit für den SB ZU/AUF {1,...,32767}*0.01s
Bei jedem Wechsel des Stellbefehles von AUF->ZU oder ZU->AUF wird die Auszeit aktiviert.

++

xAH [INT]: AUTO-Istwert / Handstellbefehl {0,1,...,1000}*0.1%
Von der HMI-Ebene kann der Stellmotor von 'HAND' angesteuert werden. Dabei ist eine HAND-Stellungsvorgabe in [%] möglich.

!!Das Handstellsignal 'xAH' nimmt folgende Werte an:

AH = 0	->	x_HAND:=0
HAND = Flanke(0-1) ODER HAND=ZU ODER HAND=AUF	->	x_HAND:=Istwert 'y_PRZ'
HAND = 1 UND NICHT HAND=ZU/AUF	->	x_HAND:=Eingabewert

++

TAST_AH [BOOL]: Taste vom HMI: FLIP-FLOP-Umschaltung AUTO/HAND

++

Begriffe:

HMI = SIMATIC 'Human Machine Interfaces'

x_TIME = Der analoge Stellbefehl 'x' wird in einen Stellzeit-Sollwert

$$x\_TIME = \frac{x*tLZ*100}{1000\%}\ [ms]$$ umgerechnet {0,1,...,3276700}*0.001s

y_TIME = Istwert der Stellzeit {0,1,...,3276700}*0.001s
Der Istwert der Stellzeit nähert sich immer dem Wert x_TIME, wenn sich x bzw. x_TIME nicht ändert.

y_PRZ = Die Umrechnung des Stellungsistwertes in [%] {0,1,...,1000}*0.1%

$$y\_PRZ = \frac{y\_TIME*1000\%}{tLZ*100}$$

Der Istwert der Stellzeit in [%], also 'y_PRZ', nähert sich immer dem Wert 'x', wenn sich 'x' nicht ändert.

tTB = Totband {1,...,32767}*0.01s
(y_TIME-tTB)<= x_TIME <= (y_TIME+tTB)
dann bleiben die digitalen Stellungssignale 'yZ'+'yA' ausgeschaltet!

!!Die Formulierung 'nähert sich' resultiert daraus, daß die kleinste Stellzeit von der Zykluszeit der CPU abhängt.Es ist Zufall, wenn der Zustand 'y_PRZ=x' bzw. 'y_TIME=x_TIME' bei einer Stellungsänderung sofort eintritt. Das Programm ist so aufgebaut, daß es für den Fall Totband 'tTB=0', unbedingt den Ausgleich 'y_PRZ=x' herstellen will, wenn ein Unterschied zwischen beiden Werten vorhanden ist. Das Totband 'tTB>0' verhindert ein eventuelles Pendeln des digitalen Stellsignales.