Project Documentation

File: Vedlegg D -Software PLS.ecp

Date: 12/10/2021

Profile: e!COCKPIT

Table of contents

1 DUT: PowerSources

```
1   TYPE PowerSources :
2   (
3       Battery , Shore , Generator
4       ) ;
5   END_TYPE
6
```

2 DUT: temp

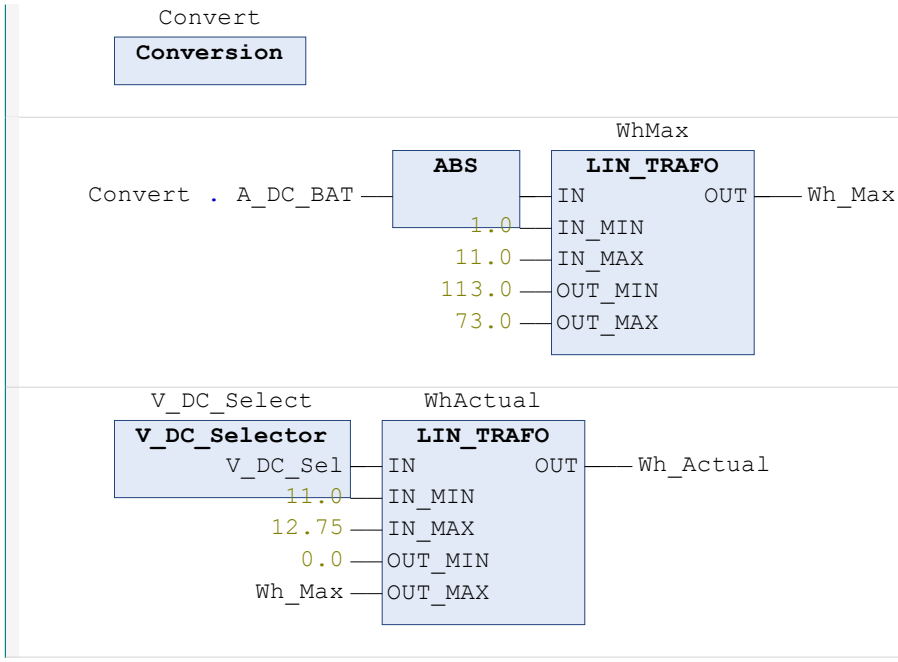
```
1   TYPE temp :
2   STRUCT
3       desimal : REAL ;
4       heltall : WORD ;
5   END_STRUCT
6   END_TYPE
7
```

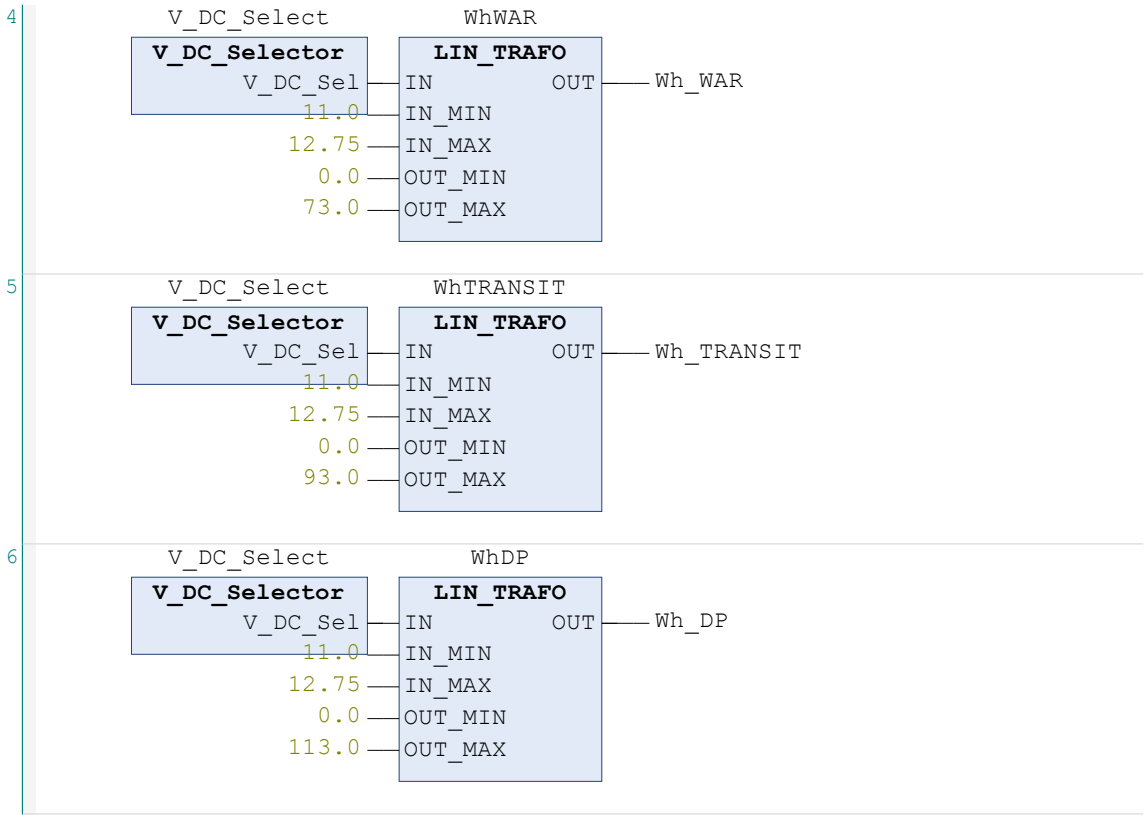
3 Global Variable List: Globals

```
1   //{attribute 'qualified_only'}
2   VAR_GLOBAL
3       V_DC_RAW AT %IW13 : INT ;
4       A_DC_BAT_RAW AT %IW1 : INT ;
5       A_DC_EMS_SUPPLY_RAW AT %IW2 : INT ;
6       A_AC_IN_RAW AT %IW3 : INT ;
7       A_AC_OUT_RAW AT %IW4 : INT ;
8       ShorePower AT %IW5 : INT ;
9       GenPower AT %IW6 : INT ;
10      Land AT %QX0.0 : BOOL ;
11      G1 AT %QX0.1 : BOOL ;
12      Servo AT %QX0.2 : BOOL ;
13      G1Start AT %QX1.7 : BOOL ;
14      G1Stopp AT %QX1.6 : BOOL ;
15      G1Temp : temp ;
16      BatTemp : temp ;
17      //A_DC_BAT : REAL ;
18  END_VAR
19
```

4 POU: Capacity_Estimator

```
1   FUNCTION_BLOCK Capacity_Estimator
2   VAR_INPUT
3   END_VAR
4   VAR_OUTPUT
5       Wh_Actual : REAL ;
6       Wh_WAR : REAL ;
7       Wh_TRANSIT : REAL ;
8       Wh_DP : REAL ;
9   END_VAR
10  VAR
11      V_DC_SELECT : V_DC_Selector ;
12      WhActual : LIN_TRAFO ;
13      WhMax : LIN_TRAFO ;
14      Wh_Max : REAL ;
15      WhWAR : LIN_TRAFO ;
16      WhTRANSIT : LIN_TRAFO ;
17      WhDP : LIN_TRAFO ;
18      Convert : Conversion ;
19  END_VAR
20
```

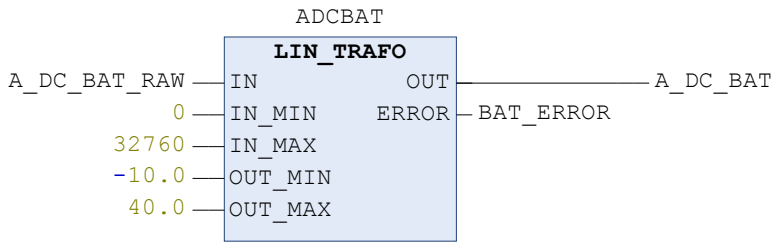



4
V_DC_Select
WhWAR
V_DC_Selector
V_DC_Sel
11.0
12.75
0.0
73.0
LIN_TRAFO
IN
OUT
IN_MIN
IN_MAX
OUT_MIN
OUT_MAX
Wh_WAR
5
V_DC_Select
WhTRANSIT
V_DC_Selector
V_DC_Sel
11.0
12.75
0.0
93.0
LIN_TRAFO
IN
OUT
IN_MIN
IN_MAX
OUT_MIN
OUT_MAX
Wh_TRANSIT
6
V_DC_Select
WhDP
V_DC_Selector
V_DC_Sel
11.0
12.75
0.0
113.0
LIN_TRAFO
IN
OUT
IN_MIN
IN_MAX
OUT_MIN
OUT_MAX
Wh_DP

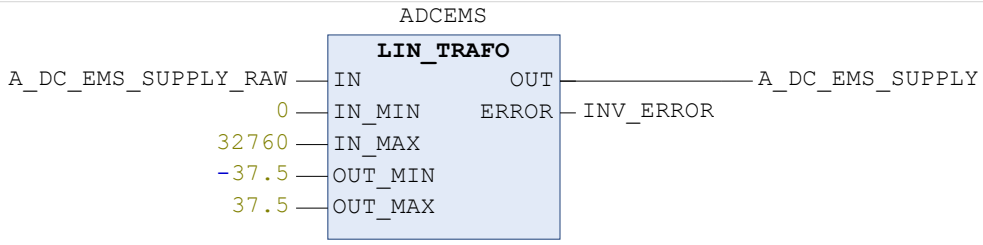
5 POU: Conversion

```
1   FUNCTION_BLOCK Conversion
2   VAR_INPUT
3   END_VAR
4   VAR_OUTPUT
5       A_DC_BAT : REAL ;
6       BAT_ERROR : BOOL ;
7       A_DC_EMS_SUPPLY : REAL ;
8       INV_ERROR : BOOL ;
9       A_AC_OUT : REAL ;
10      AC_OUT_ERROR : BOOL ;
11      A_AC_IN : REAL ;
12      AC_IN_ERROR : BOOL ;
13      V_DC : REAL ;
14      V_DC_ERROR : BOOL ;
15  END_VAR
16  VAR
17      ADCBAT : LIN_TRAFO ;
18      AACOUT : LIN_TRAFO ;
19      AACIN : LIN_TRAFO ;
20      VDC : LIN_TRAFO ;
21      ADCEMS : LIN_TRAFO ;
22  END_VAR
23
```

1

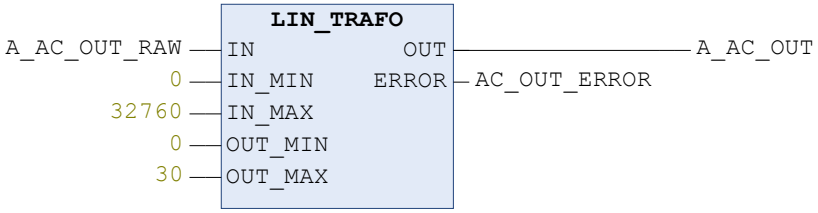


2



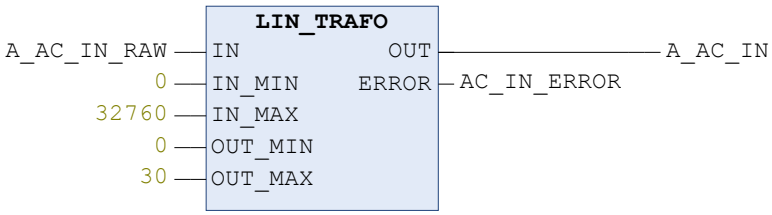
3

AACOUT



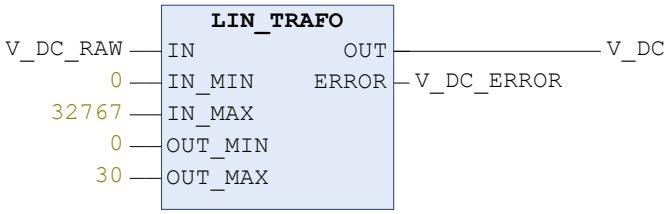
4

AACIN



5

VDC



6 POU: PLC_PRG

```
PROGRAM PLC_PRG
VAR PERSISTENT
    V_DC_Min : REAL := 11 ;
    Wh_Bat_Min : REAL := 50 ;
    Wh_Bat_Preferred : REAL := 110 ;
END_VAR
VAR
    V_DC_Select : V_DC_Selector ;
    States : PowerSources := Battery ;
    Convert : Conversion ;
    Wh_Count : Wh_In_Out_Bat ;
    GenStart : BOOL ;
    PowerAndTime : Power_And_Time ;
    Wh_Total_Bat : REAL ;
    Battery_Voltage_Delay : TON ;
    Mesured_Level : REAL ;
    Bat_Cap_Estimator : Capacity_Estimator ;
    Wh_Count_Timer : TON ;
    Temp_Time : TIME ;
    Sum_Wh_Test : REAL ;
    test_temp : TIME ;
    V_DC_AVR : Volts_Average ;
    G1StartTP : TP := ( PT := T#2S ) ;
    G1StartBreak : TON := ( PT := T#1S ) ;
    HG1Temp : BOOL ;
    HBTemp : BOOL ;
    ShoreSwitch : BOOL ;
    GenSwitch : BOOL ;
    BatSwitch : BOOL ;
    IngenStrom : TON := ( PT := T#500MS ) ;
    Vent : TON := ( PT := T#500ms ) ;
END_VAR
```

```
G1Temp . heltall := %IW9 ;
G1Temp . desimal := G1Temp . heltall / 10.0 ;
//Gjør om teperaturverdi til riktig grad.
BatTemp . heltall := %IW10 ;
BatTemp . desimal := BatTemp . heltall / 10.0 ;
                              //Gjør om temperaturverdi til riktig grad.
Wh_Count ( ) ;
//Her oppdateres alle funksjonsblokker som skal.
Convert ( ) ;
//Vise verdier til enhver tid.
PowerAndTime ( ) ;
Bat_Cap_Estimator ( ) ;
V_DC_AVR ( ) ;
V_DC_Select ( ) ;

CASE States OF
    Battery :
```

```
        //Batteridriftstilstand.
14              Land := FALSE ;
        //Kontaktorer for landstrøm og aggregat legges ut.
15              G1 := FALSE ;
16              G1Stopp := TRUE ;
        //Gir stoppsignal til aggregat.
17              BatSwitch := FALSE ;
        //Nullstiller batteriuknappen på brukergrensesnitt.
18              Bat_Cap_Estimator (Wh_Actual => Wh_Total_Bat) ;
        //FB for energinivåestimering oppdateres.
19              IF ShoreSwitch THEN
        //Går til landstrømstilstand hvis landstrøm knapp

20                  States := Shore ;
        //trykkes.
21              END_IF
22              IF ( ( Convert . V_DC <= V_DC_Min ) OR GenSwitch )
23              AND NOT ( ShorePower > 10000 ) THEN
        //Går til generatordrift hvis spenning går for lavt eller
24                  GenStart := TRUE ;
        //man selv ønsker å gå over til generator drift.
25                  States := Generator ;

26              END_IF
27
28          Shore :
        //Landstrømstilstand
29              G1 := FALSE ;
        //Kontaktor for generator legges ut.
30              Land := TRUE ;
        //Kontaktor for landstrøm legges inn.
31              G1Stopp := TRUE ;
        //Gir stoppsignal til aggrgeat.
32              ShoreSwitch := FALSE ;
        //Nullstiller knapp i brukergrensesnitt.
33              Wh_Count ( Start := TRUE ) ;                                    //FB
        for beregning av siste sekunds wattimer oppdateres.
34              Wh_Count_Timer ( IN := TRUE , PT := T#15S ) ;
        //Timer for å summere opp wattimer med forrige estimat oppdateres.
35              IF ( Wh_Count_timer . ET >= Temp_Time + T#1S ) THEN
        //Siste sekunds wattimer summeres med forrige estimerte nivå på
36                  Wh_Total_Bat := Wh_Total_Bat + Wh_Count . Wh_Bat ;
        //batteriet.
37                  Temp_Time := Wh_Count_Timer . ET ;
38              END_IF
39              IF IngenStrom . Q OR BatSwitch THEN
        //Går til batteridriftstilstand hvis landspenning bortfaller
40                  Wh_Count ( Start := FALSE , Time_var := T#0S ) ;
        //eller batteriknappen trykkes.
41                  Wh_Count_Timer ( IN := FALSE ) ;
        //Nullstiller timer.
42                  Temp_Time := T#0S ;
43                  Land := FALSE ;
44                  States := Battery ;
45              END_IF
46              IF GenSwitch THEN
        //Går til generator modus hvis knappen trykkes.
```

```
47                  GenStart := TRUE ;
48                  States := Generator ;
49              END_IF
50
51          Generator :
        //Generatordriftstilstand.
52              Land := FALSE ;
        //Kontaktor for landstrøm legges ut.
53              G1 := TRUE ;
        //Kontaktor for generatorstrøm leggges inn.
54              G1Stopp := FALSE ;
        //Sender ingen stoppsignal aggregat.
55              GenSwitch := FALSE ;
        //Nullstiller knapp på brukergrensesnitt.
56              Wh_Count ( Start := TRUE ) ;                                    //FB
        for beregning av siste sekunds wattimer oppdateres.
57              Wh_Count_Timer ( IN := TRUE , PT := T#15S ) ;
        //Timer for å summere opp wattimer med forrige estimat oppdateres.
58              IF ( Wh_Count_timer.ET >= Temp_Time + T#1S ) THEN
        //Siste sekunds wattimer summeres med forrige estimerte nivå på
59                  Wh_Total_Bat := Wh_Total_Bat + Wh_Count . Wh_Bat ;
        //batteriet.
60                  Temp_Time := Wh_Count_Timer.ET ;
61              END_IF
62              IF ( Wh_Total_Bat >= Wh_Bat_Preferred ) OR BatSwitch
        //Går til batteridriftstilstand hvis generatorspenning bortfaller,
63                  OR IngenStrom THEN
                                                        //eller batteriet har
        nådd ønsket nivå.
64                  Wh_Count ( Start := FALSE , Time_var := T#0S ) ;
65                  Wh_Count_Timer ( IN := FALSE ) ;
66                  G1Stopp := TRUE ;
67                  States := Battery ;
68              END_IF
69              IF ShoreSwitch THEN
        //Går til landstrømstilstand hvis landspenning finnes.
70                  Wh_Count ( Start := FALSE , Time_var := T#0S ) ;
71                  Wh_Count_timer ( IN := FALSE ) ;
72                  G1Stopp := TRUE ;
73                  States := Shore ;
74              END_IF
75              IF Vent.Q THEN
        //Venter med å sende stoppsignal.
76                  GenStart := FALSE ;
77              END_IF
78
79      END_CASE
80
81      Vent ( IN := A_AC_IN_RAW > 150 AND G1 ) ;
        //For overgang fra Landstrøm vil det være en reststrøm,
82
        //gjennom tester er det funnet ut å vente 0,5s fungerer.
83      IngenStrom (IN := A_AC_IN_RAW < 80 AND (Land OR G1 ) AND NOT
        //Overgangskriterier for overgang til batteri.
84      GenStart ) ;
85      G1StartTP(IN := GenStart);
```

```
        //GenStart går høy i 2 sekunder.
86      G1StartBreak(IN := GenStart AND NOT G1StartTP.Q);
        //Legger inn en pause på 1s etter funn i tester.
87      IF G1StartBreak.Q THEN
        //Nullstiller timer.
88          G1StartTP(IN := FALSE);
89      END_IF
90
91      IF GenStart = TRUE THEN
92          G1Stopp := FALSE;
93          G1 := TRUE;
        //Sikkerhet for at G1 kontaktor blir slått inn.
94          IF G1Temp.desimal <= 42.0 THEN
        //Gir signal til for å operere choke.
95              Servo := TRUE;
96          END_IF
97          G1Start := G1StartTP.Q;
        //Startsignal til aggregat.
98          ELSE
99              G1Start := FALSE;
100             Servo := FALSE;
101     END_IF
102
103     IF G1Temp.desimal > 250 THEN
        //Alarm til bruker hvis generator blir for varm.
104         HG1Temp := TRUE;
105     END_IF
106     IF BatTemp.desimal > 40 THEN
        //Alarm til bruker hvis batteri blir for varmt.
107         HBTemp := TRUE;
108     END_IF
109
```

7 POU: Power_And_Time

```
1       FUNCTION_BLOCK Power_And_Time
2       VAR_INPUT
3       END_VAR
4       VAR_OUTPUT
5           W_Battery : REAL;
6           W_EMS_Supply : REAL;
7           W_Inverter : REAL;
8           W_AC_IN : REAL;
9           W_AC_OUT : REAL;
10          EOT_WAR : TIME;
11          EOT_TRANSIT : TIME;
12          EOT_DP : TIME;
13          EOT_ACTUAL : TIME;
14          Wait_Estimates : BOOL;
15          ABS_Bat_Strom : REAL;
16      END_VAR
17      VAR
18          Wh_Count : Wh_In_Out_Bat;
19          Convert : Conversion;
20          Capacity : Capacity_Estimator;
21          V_DC_Select : V_DC_Selector;
```

```
22      END_VAR
23
```

```
1       Convert ( ) ;
2       Wh_Count ( ) ;
3       Capacity ( ) ;
4       V_DC_Select ( ) ;
5
6       Wait_Estimates := ( V_DC_Select . V_DC_Sel <= 12.75 ) ;
7
8       W_Battery := ABS ( Convert . A_DC_BAT * V_DC_Select . V_DC_Sel ) ;
9       IF LAND OR G1 THEN
10          W_Inverter := ( W_Battery + W_EMS_Supply ) ;
11          ELSE
12              W_Inverter := ( W_Battery - W_EMS_Supply ) ;
13      END_IF
14      W_EMS_Supply := ABS ( Convert . A_DC_EMS_SUPPLY * V_DC_Select .
        V_DC_Sel ) ;
15
16      W_AC_IN := Convert . A_AC_IN * 230 ;
17      W_AC_OUT := Convert . A_AC_OUT * 230 ;
18
19      EOT_WAR := REAL_TO_TIME ( 3600000 * Capacity . Wh_WAR / 122.4 ) ;
20      EOT_TRANSIT := REAL_TO_TIME ( 3600000 * Capacity . Wh_TRANSIT / 68.4 )
        ;
21      EOT_DP := REAL_TO_TIME ( 3600000 * Capacity . Wh_DP / 14.4 ) ;
22      EOT_ACTUAL := REAL_TO_TIME ( 3600000 * Capacity . Wh_Actual / W_Battery
        ) ;
23
```

8 POU: Volts_Average

```
1       FUNCTION_BLOCK Volts_Average
2       VAR_INPUT
3       END_VAR
4       VAR_OUTPUT
5           V_DC_AVERAGE : REAL ;
6       END_VAR
7       VAR
8           Average_Timer : TON ;
9           Temp_time : TIME ;
10          Values : ARRAY [ 1 .. 100 ] OF REAL ;
11          Convert : Conversion ;
12          n : INT ;
13          k : INT ;
14          Array_Sum : REAL ;
15          i : INT ;
16      END_VAR
17
```

```
1       Convert ( ) ;
        //Oppdaterer spenningsmåleren.
2       Average_Timer ( IN := TRUE , PT := T#15H ) ;
        //Starter timeren.
```

```
3       IF ( Average_Timer . ET >= Temp_Time + T#3S270MS ) THEN
        //Gjør at eksekveringen ikke gjøres oftere enn for hvert 3. sek og
4
        //270 ms.
5           FOR n := 1 TO 99 DO
        //Flytter alle verdier ett hakk videre i arrayen "Values", og frigjør
6               Values [ n ] := Values [ n + 1 ] ;
                                        //plass til en ny verdi i den første.
7           END_FOR
8           IF NOT ( Land OR G1 ) THEN
        //Mater inn aktuell spenning i den nye ledige plassen ved batteridrift,
9               Values [ 100 ] := Convert . V_DC ;
        //ved land- eller generatorfroft mates 0 inn.
10          ELSE
11              Values [ 100 ] := 0 ;
12          END_IF
13
14          FOR i := 1 TO 100 DO
        //Summerer opp og teller alle verdier over 0.
15              IF Values [ i ] > 0 THEN
16                  Array_Sum := Array_Sum + Values [ i ] ;
17                  k := k + 1 ;
18              END_IF
19          END_FOR
20          IF k > 0 THEN
        //Deler summen på antall verdier over 0 og deler på 1 hvis det ikke er
21              V_DC_AVERAGE := Array_Sum / k ;
        //noen verdier over null, for å ikke få 0/0 = uendelig.
22          ELSE
23              k := 1 ;
24              V_DC_AVERAGE := Array_Sum / k ;
25          END_IF
26          k := 0 ;
        //Nullstiller antall verdier for å telle på nytt ved neste eksekvering.
27          Array_Sum := 0 ;
        //Nullstiller summen for å telle på nytt ved neste eksekvering.
28          Temp_Time := Average_Timer . ET ;
        //Gjør at eksekveringen ikke gjøres igjen før om 3 sek og 270 ms.
29      END_IF
30
```

9 POU: V_DC_Selector

```
1       FUNCTION_BLOCK V_DC_Selector
2       VAR_INPUT
3       END_VAR
4       VAR_OUTPUT
5           V_DC_Sel : REAL ;
6       END_VAR
7       VAR
8           DC_Realtime : Conversion ;
9           DC_Average : Volts_Average ;
10      END_VAR
11
```

```
1       DC_Realtime ( ) ;
2       DC_Average ( ) ;
3
4       IF ( DC_Average . V_DC_AVERAGE > 12.75 ) OR ( DC_Average .
        V_DC_AVERAGE = 0.0 )
5       OR Land OR G1 THEN
6           V_DC_Sel := DC_Realtime . V_DC ;
7       ELSE
8           V_DC_Sel := DC_Average . V_DC_AVERAGE ;
9       END_IF
10
```

10 POU: Wh_In_Out_Bat

```
1       FUNCTION_BLOCK Wh_In_Out_Bat
2       VAR_INPUT
3           Start : BOOL ;
4           Time_var : TIME ;
5       END_VAR
6       VAR_OUTPUT
7           Wh_Bat : REAL ;
8       END_VAR
9       VAR
10          Wh_Timer : TON ;
11          Convert : Conversion ;
12      END_VAR
13
```

```
1       Wh_Timer ( IN := Start , PT := T#15H ) ;
2       IF ( Wh_Timer . ET >= Time_var + T#1S ) THEN
3           Convert ( ) ;
4           Wh_Bat := ABS ( ( ( TIME_TO_REAL ( Wh_Timer . ET - Time_var ) /
        3600000 ) * ( Convert . A_DC_BAT ) * Convert . V_DC ) ) ;
5           Time_var := Wh_Timer . ET ;
6       END_IF
7
```