```
1   FUNCTION_BLOCK FB_FruitTemperatureControl
2   VAR_INPUT
3       fTfruitRef : LREAL;
4       fTthermostat : LREAL; //AI Adres Thermostat
5       fTfruit : LREAL; //AI Adress fruit temperature sensor
6
7       fKp : LREAL;
8       fKi : LREAL;
9       fKt : LREAL;
10
11      eControlMode : eModeTemperatureControl; //0 = Normal, 1 = Measure, 2 =
    Control
12      bReset : BOOL; //Rising edge needed
13  END_VAR
14  VAR_OUTPUT
15      fTthermostatOut : LREAL; //AQ Adres Thermostat out
16      sStateText : STRING(255);
17      bRelay : BOOL; //DQ relay
18
19      fTestUvoorSat : LREAL; //Temporary output variables for graph
20      fTestUerrorSat : LREAL; //Temporary output variables for graph
21      fTestUantiWind : LREAL; //Temporary output variables for graph
22      fTestTsetOut : LREAL; //Temporary output variables for graph
23
24  END_VAR
25  VAR
26      tTi : TIME;
27      tTt : TIME;
28      //PI
29      FB_PI : FB_CTRL_PI;
30      ParamsPi : ST_CTRL_PI_PARAMS;
31      eErrorIdPI : E_CTRL_ERRORCODES;
32      bErrorPI : BOOL;
33      //I
34      FB_I : FB_CTRL_I;
35      ParamsI : ST_CTRL_I_PARAMS;
36      eErrorIdI : E_CTRL_ERRORCODES;
37      bErrorI : BOOL;
38      //Limiter
39      FB_LIMIT : FB_CTRL_LIMITER;
40      ParamsLimiter : ST_CTRL_LIMITER_PARAMS;
41      eErrorIdLimiter : E_CTRL_ERRORCODES;
42      bErrorLimiter : BOOL;
43      fMinTset : LREAL;
44      fMaxTset : LREAL;
45      tCycleTime : TIME := T#1000MS; //Sampling time = 1s
46
47      fTfruitZero : LREAL;
48      fTsetZero : LREAL;
49      bRunOnce : BOOL := TRUE;
50      bSafety : BOOL := TRUE;
51      rTrigger : R_TRIG;
52
53      //controller output variables
54      fUafterSat : LREAL;
55      fUerrorSat : LREAL;
56      fUvoorSat : LREAL;
```

```
57          fUantiWind : LREAL := 0 ;
58          fUpi : LREAL ;
59          fTsetOut : LREAL ;
60          //Timer Tset =< -2°C
61          bTimerAtMinus2 : BOOL := FALSE ;
62          tMaxTimeMinus2 : TIME := T#56000S ; //15h33min20s
63          bTimerAtMinus2Out : BOOL ;
64          TimerAtMinus2 : TON ;
65          tElapsedTimeAtMinus2 : TIME ;
66
67          //Timer Tset > -2°C
68          bTimerAboveMinus2 : BOOL := FALSE ;
69          tMinTimeAboveMinus2 : TIME := T#1800S ; //30min
70          bTimerAboveMinus2Out : BOOL ;
71          TimerAboveMinus2 : TON ;
72          tElapsedTimeAboveMinus2 : TIME ;
73          bRtrigOut : BOOL ;
74      END_VAR
75
```

```
1       //Integral gain to integral time
2       tTi := LREAL_TO_TIME ( ( fKp / fKi ) * 1000 ) ;
3       tTt := LREAL_TO_TIME ( ( 1 / fKt ) * 1000 ) ;
4
        //*************************************************************************
5       IF ( eControlMode = 2 AND bSafety = TRUE ) THEN //Control mode
6           IF bRunOnce THEN
7               fTfruitZero := fTfruit ; //Important that the cold store is running
        for a few days with Tset = TfruitRef
8               fTsetZero := fTfruitRef ;
9               bRunOnce := FALSE ;
10          END_IF
11
12          //PI blok
13          ParamsPi . fKp := fKp ;
14          ParamsPi . tTn := tTi ;
15          ParamsPi . fOutMaxLimit := 1E38 ;
16          ParamsPi . fOutMinLimit := - 1E38 ;
17          ParamsPi . tCtrlCycleTime := tCycleTime ;
18          ParamsPi . tTaskCycleTime := tCycleTime ;
19          ParamsPi . bARWOnIPartOnly := TRUE ;
20
21          FB_PI ( fSetpointValue      := fTfruitRef - fTfruitZero ,
22                  fActualValue        := fTfruit - fTfruitZero  ,
23                  fManSyncValue   := 0 ,
24                  bSync               := FALSE ,
25                  eMode               := E_CTRL_MODE . eCTRL_MODE_ACTIVE ,
26                  bHold               := FALSE ,
27                  stParams            := ParamsPi ,
28                  fOut                => fUpi ,
29                  bARWactive      => ,
30                  eErrorId            => eErrorIdPI ,
31                  bError              => bErrorPI
32                      ) ;
```

```
33          fUvoorSat  :=  fUpi + fUantiWind ;
34              fTestUvoorSat := fUvoorSat ;      //Test variabels
35          //Limiter
36          fMinTset  :=  - 1 ;
37          fMaxTset  :=  3 + fTfruitRef ;
38
39
40          ParamsLimiter . fMaxOutput  :=  fMaxTset ;
41          ParamsLimiter . fMinOutput  :=  fMinTset ;
42          ParamsLimiter . tCtrlCycleTime  :=  tCycleTime ;
43          ParamsLimiter . tTaskCycleTime  :=  tCycleTime ;
44
45          FB_LIMIT  ( fIn                  :=  fUvoorSat ,
46                      stParams             :=  ParamsLimiter ,
47                      fOut                 =>  fUafterSat ,
48                      eErrorId             =>  eErrorIdLimiter ,
49                      bError               =>  bErrorLimiter ) ;
50
51          //I blok
52          ParamsI . fOutMaxLimit  :=  1E38 ;
53          ParamsI . fOutMinLimit  :=  - 1E38 ;
54          ParamsI . tCtrlCycleTime  :=  tCycleTime ;
55          ParamsI . tTaskCycleTime  :=  tCycleTime ;
56          ParamsI . tTi  :=  tTt ;
57
58              fTestUerrorSat  :=  fUerrorSat ;      //Test variabels
59          fUerrorSat  :=  fUafterSat - fUvoorSat ;
60          FB_I (   fIn                 :=  fUerrorSat ,
61                   eMode               :=  E_CTRL_MODE . eCTRL_MODE_ACTIVE ,
62                   stParams            :=  ParamsI ,
63                   fOut                =>  fUantiWind ,
64                   eErrorId            =>  eErrorIdI ,
65                   bError              =>  bErrorI ) ;
66          fTestUantiWind  :=  fUantiWind ;      //Test variabels
67          //TsetOut to TthermostatOut
68          fTsetOut  :=  fUafterSat  +  fTsetZero ;
69          fTthermostatOut  :=  fTthermostat  +  ( fTsetOut - fTsetZero ) ;
70
71              fTestTsetOut  :=  fTsetOut ;      //Test variabels
72              GVL . fTestTthermostatOut  :=  fTthermostatOut ;   //Test variabels
73          //-2°C timer
74          IF ( fTsetOut  >  - 2 )  THEN
75              bTimerAboveMinus2  :=  TRUE ;
76          ELSE
77              bTimerAboveMinus2  :=  FALSE ;
78          END_IF
79
        //*****************************************************************//
80          TimerAboveMinus2 ( IN  :=  bTimerAboveMinus2 ,
81                             PT  :=  tMinTimeAboveMinus2 ,
82                             Q  =>  bTimerAboveMinus2Out ,
83                             ET  =>  tElapsedTimeAboveMinus2 ) ;
84          IF ( bTimerAboveMinus2Out  =  TRUE )  THEN  //We are safe
85              bTimerAboveMinus2  :=  FALSE ;
86              TimerAboveMinus2 ( IN  :=  bTimerAboveMinus2 ,
87                  PT  :=  tMinTimeAboveMinus2 ,
```

```
88                  Q  =>  bTimerAboveMinus2Out ,
89                  ET  =>  tElapsedTimeAboveMinus2 ) ;
90
91              bTimerAtMinus2  :=  FALSE ;
92              TimerAtMinus2 ( IN  :=  bTimerAtMinus2 ,
93                  PT  :=  tMaxTimeMinus2 ,
94                  Q  =>  bTimerAtMinus2Out ,
95                  ET  =>  tElapsedTimeAtMinus2 ) ;
96
97          END_IF
98
99          IF  ( fTsetOut  <=  - 2  AND  NOT ( bTimerAboveMinus2Out ) ) THEN
100             bTimerAtMinus2  :=  TRUE ;
101         END_IF
102         TimerAtMinus2 ( IN  :=  bTimerAtMinus2 ,
103             PT  :=  tMaxTimeMinus2 ,
104             Q  =>  bTimerAtMinus2Out ,
105             ET  =>  tElapsedTimeAtMinus2 ) ;
106         IF ( bTimerAtMinus2Out )  THEN  //To long at -2
107             bSafety  :=  FALSE ;
108         END_IF
109
110         sStateText  :=  'PI temperature control is running' ;
111         bRelay  :=  TRUE ;
112
        //*****************************************************************************
113     ELSE
114         IF ( eControlMode  =  1  AND  bSafety  =  TRUE ) THEN  //Measurement mode
115             bRunOnce  :=  TRUE ;
116
117             //Resetting PI, I and TON
118             bTimerAtMinus2  :=  FALSE ;
119             TimerAtMinus2 ( IN  :=  bTimerAtMinus2 ,
120                 PT  :=  tMaxTimeMinus2 ,
121                 Q  =>  bTimerAtMinus2Out ,
122                 ET  =>  tElapsedTimeAtMinus2 ) ;
123
124             bTimerAboveMinus2  :=  FALSE ;
125             TimerAboveMinus2 ( IN  :=  bTimerAboveMinus2 ,
126                 PT  :=  tMinTimeAboveMinus2 ,
127                 Q  =>  bTimerAboveMinus2Out ,
128                 ET  =>  tElapsedTimeAboveMinus2 ) ;
129
130             FB_PI ( fSetpointValue        :=  fTfruitRef - fTfruitZero ,
131                 fActualValue         :=  fTfruit - fTfruitZero  ,
132                 fManSyncValue    :=  0 ,
133                 bSync                :=  FALSE ,
134                 eMode                :=  E_CTRL_MODE . eCTRL_MODE_RESET ,
135                 bHold                :=  FALSE ,
136                 stParams             :=  ParamsPi ,
137                 fOut                 =>  fUpi ,
138                 bARWactive       =>  ,
139                 eErrorId             =>  eErrorIdPI ,
140                 bError               =>  bErrorPI ) ;
141             FB_I ( fIn               :=  fUerrorSat ,
142                 eMode                :=  E_CTRL_MODE . eCTRL_MODE_RESET ,
```

```
143                 stParams            :=  ParamsI ,
144                 fOut                =>  fUantiWind ,
145                 eErrorId            =>  eErrorIdI ,
146                 bError              =>  bErrorI ) ;
147             fTthermostatOut  :=  fTthermostat ;
148             sStateText  :=  'Normal control and collecting data' ;
149             bRelay  :=  TRUE ;
150 
151             //Data measurement algoritm;
152 
153         END_IF
154 
        //***************************************************************************

155         IF  ( eControlMode  =  0  AND  bSafety  =  TRUE )  THEN  //Normal mode
156             bRunOnce  :=  TRUE ;
157 
158             //Resetting PI, I and TON
159             bTimerAtMinus2  :=  FALSE ;
160             TimerAtMinus2 ( IN  :=  bTimerAtMinus2 ,
161                 PT  :=  tMaxTimeMinus2 ,
162                 Q  =>  bTimerAtMinus2Out ,
163                 ET  =>  tElapsedTimeAtMinus2 ) ;
164 
165             bTimerAboveMinus2  :=  FALSE ;
166             TimerAboveMinus2 ( IN  :=  bTimerAboveMinus2 ,
167                 PT  :=  tMinTimeAboveMinus2 ,
168                 Q  =>  bTimerAboveMinus2Out ,
169                 ET  =>  tElapsedTimeAboveMinus2 ) ;
170 
171             FB_PI ( fSetpointValue        :=  fTfruitRef - fTfruitZero ,
172                 fActualValue         :=  fTfruit - fTfruitZero  ,
173                 fManSyncValue    :=  0 ,
174                 bSync                :=  FALSE ,
175                 eMode                :=  E_CTRL_MODE . eCTRL_MODE_RESET ,
176                 bHold                :=  FALSE ,
177                 stParams             :=  ParamsPi ,
178                 fOut                 =>  fUpi ,
179                 bARWactive       =>  ,
180                 eErrorId             =>  eErrorIdPI ,
181                 bError               =>  bErrorPI ) ;
182             FB_I ( fIn               :=  fUerrorSat ,
183                 eMode                :=  E_CTRL_MODE . eCTRL_MODE_RESET ,
184                 stParams             :=  ParamsI ,
185                 fOut                 =>  fUantiWind ,
186                 eErrorId             =>  eErrorIdI ,
187                 bError               =>  bErrorI ) ;
188 
189             fTthermostatOut  :=  fTthermostat ;
190             sStateText  :=  'Normal control and NOT collecting data' ;
191             bRelay  :=  FALSE ;
192 
193         END_IF
194 
        //***************************************************************************
```

```
195         IF NOT bSafety THEN //Error state
196
197             //Resetting PI, I and TON
198             bRunOnce := TRUE;
199             bTimerAtMinus2 := FALSE;
200             TimerAtMinus2(IN := bTimerAtMinus2,
201                 PT := tMaxTimeMinus2,
202                 Q => bTimerAtMinus2Out,
203                 ET => tElapsedTimeAtMinus2);
204
205             bTimerAboveMinus2 := FALSE;
206             TimerAboveMinus2(IN := bTimerAboveMinus2,
207                 PT := tMinTimeAboveMinus2,
208                 Q => bTimerAboveMinus2Out,
209                 ET => tElapsedTimeAboveMinus2);
210
211
212             FB_PI(fSetpointValue       := fTfruitRef - fTfruitZero,
213                 fActualValue        := fTfruit - fTfruitZero ,
214                 fManSyncValue   := 0,
215                 bSync               := FALSE,
216                 eMode               := E_CTRL_MODE.eCTRL_MODE_RESET,
217                 bHold               := FALSE,
218                 stParams            := ParamsPi,
219                 fOut                => fUpi,
220                 bARWactive      => ,
221                 eErrorId            => eErrorIdPI,
222                 bError              => bErrorPI);
223             FB_I(fIn                := fUerrorSat,
224                 eMode               := E_CTRL_MODE.eCTRL_MODE_RESET,
225                 stParams            := ParamsI,
226                 fOut                => fUantiWind,
227                 eErrorId            => eErrorIdI,
228                 bError              => bErrorI);
229
230             //Reset button with rising edge
231             rTrigger(CLK:=bReset,
232                         Q => bRtrigOut);
233             IF(bRtrigOut = TRUE) THEN
234                 bSafety := TRUE;
235             END_IF
236
237             sStateText := 'Error mode, settemperature remained to long at -2°C.
    Press reset in order to acces other control modes.
238                                         Normal operation without data
    collecting currently active';
239             bRelay := FALSE;
240             fTthermostatOut := fTthermostat;
241         END_IF
242     END_IF
243
```