```
---------------------------------------
-- Heating_Controller5A.adb
---------------------------------------
with Ada.Real_Time, Shared_Object1ImplPkg, System.Storage_Elements;
use Ada.Real_Time, Shared_Object1ImplPkg, System.Storage_Elements;

procedure Heating_Controller5A is
  shared_Object1ImplInst: Shared_Object1Impl;

  task Display_Update_Task1Impl is
    pragma Priority(5);
  end Display_Update_Task1Impl;



-- TASK Display_Update_Task1Impl

  task body Display_Update_Task1Impl is
    Max : constant := 95;
    Min : constant := 5;
    cttm1 : Integer := 0;
    ttm : Integer := 0;
    sinc_flag : Boolean := FALSE;
    sdec_flag : Boolean := FALSE;
    tm_tt : Integer;
    for tm_tt'Address use System'To_Address(16#ef00#);
    state_inc : Boolean;
    for state_inc'Address use System'To_Address(16#ef04#);
    state_dec : Boolean;
    for state_dec'Address use System'To_Address(16#ef08#);
    period: constant Time_Span := To_Time_Span(0.5);
    nextTime: Time := clock + period;

    procedure DIDisplay_Target_Temperature is
    begin
      tm_tt := ttm;
    end;

    procedure DISense_PressIncrease_Target_Temperature is
    begin
      sinc_flag := state_inc;
    end;

    procedure DISense_PressDecrease_Target_Temperature is
    begin
      sdec_flag := state_dec;
    end;

  begin
    loop
      delay until nextTime;
      shared_object1implInst.SOGet_Target_Temperature1(cttm1);
      DISense_PressIncrease_Target_Temperature;
      if(cttm1 < Max and sinc_flag = TRUE) then
         ttm := (cttm1 + 1);
         sinc_flag := FALSE;
      else
           sinc_flag := FALSE;
```

```
    end if;
    DISense_PressDecrease_Target_Temperature;
    if(sdec_flag = TRUE and cttm1 > Min) then
       ttm := (cttm1 - 1);
       sdec_flag := FALSE;
    else
         sdec_flag := FALSE;
    end if;
    shared_object1implInst.SOSet_Target_Temperature(ttm);
    DIDisplay_Target_Temperature;
    nextTime := nextTime + period;
  end loop;
end Display_Update_Task1Impl;

task Envir1Impl is
  pragma Priority(5);
end Envir1Impl;



-- TASK Envir1Impl

task body Envir1Impl is
  Max : constant := 95;
  Min : constant := 5;
  inc_flag : Boolean := FALSE;
  for inc_flag'Address use System'To_Address(16#ef04#);
  dec_flag : Boolean := FALSE;
  for dec_flag'Address use System'To_Address(16#ef08#);
  ts1 : Integer := 0;
  for ts1'Address use System'To_Address(16#ef14#);
  ts2 : Integer := 0;
  for ts2'Address use System'To_Address(16#ef18#);
  hss : Boolean := FALSE;
  for hss'Address use System'To_Address(16#ef0b#);
  anha : Boolean := FALSE;
  for anha'Address use System'To_Address(16#ef10#);
  ttd : Integer := 25;
  for ttd'Address use System'To_Address(16#ef00#);
  hsa : Boolean := FALSE;
  for hsa'Address use System'To_Address(16#ef20#);
  ctd : Integer := 20;
  for ctd'Address use System'To_Address(16#ef1b#);
  aota : Boolean := FALSE;
  for aota'Address use System'To_Address(16#ef24#);
  period: constant Time_Span := To_Time_Span(0.1);
  nextTime: Time := clock + period;

  procedure ENAlter_Temperature_Sensor1 is
  begin
    ts1 := (ts1 + 1);
  end;

  procedure ENAlter_Temperature_Sensor2 is
  begin
    ts2 := (ts2 + 1);
  end;
```

```
  procedure ENAlter_Heater_Status is
  begin
    hss := FALSE;
  end;

begin
  loop
    delay until nextTime;
    if(inc_flag = FALSE) then
       inc_flag := TRUE;
    else  if(inc_flag = TRUE) then
            inc_flag := FALSE;
          end if;
    end if;
    if(dec_flag = FALSE) then
       dec_flag := TRUE;
    else  if(dec_flag = TRUE) then
            dec_flag := FALSE;
          end if;
    end if;
    ENAlter_Heater_Status;
    ENAlter_Temperature_Sensor1;
    select
    else
      null;
    end select;
    nextTime := nextTime + period;
  end loop;
end Envir1Impl;

task Heater_Monitor_Task1Impl is
  pragma Priority(5);
end Heater_Monitor_Task1Impl;



-- TASK Heater_Monitor_Task1Impl

task body Heater_Monitor_Task1Impl is
  Max : constant := 95;
  Min : constant := 5;
  hs1 : Boolean := FALSE;
  nha : Boolean := FALSE;
  shs : Boolean := FALSE;
  state_hss : Boolean;
  for state_hss'Address use System'To_Address(16#ef0b#);
  state_nha : Boolean;
  for state_nha'Address use System'To_Address(16#ef10#);
  period: constant Time_Span := To_Time_Span(0.25);
  nextTime: Time := clock + period;

  procedure HMSense_Heater_Status is
  begin
    shs := state_hss;
  end;

  procedure HMActuate_NoHeat_Alarm is
  begin
```

```
    state_nha := nha;
  end;

begin
  loop
    delay until nextTime;
    HMSense_Heater_Status;
    shared_object1implInst.SOGet_Stored_HeatSource(hs1);
    if(hs1 /= shs) then
       nha := TRUE;
    else
         nha := FALSE;
    end if;
    HMActuate_NoHeat_Alarm;
    nextTime := nextTime + period;
  end loop;
end Heater_Monitor_Task1Impl;

task Temp_Ctrl_Task1Impl is
  pragma Priority(5);
end Temp_Ctrl_Task1Impl;



-- TASK Temp_Ctrl_Task1Impl

task body Temp_Ctrl_Task1Impl is
  Max : constant := 95;
  Min : constant := 5;
  stm1 : Integer := 0;
  stm2 : Integer := 0;
  avt : Integer := 0;
  cttm2 : Integer := 0;
  hsc : Boolean := FALSE;
  ota : Boolean := FALSE;
  t1 : Integer;
  for t1'Address use System'To_Address(16#ef14#);
  t2 : Integer;
  for t2'Address use System'To_Address(16#ef18#);
  tm_avt : Integer;
  for tm_avt'Address use System'To_Address(16#ef1b#);
  state_hsc : Boolean;
  for state_hsc'Address use System'To_Address(16#ef20#);
  state_ota : Boolean;
  for state_ota'Address use System'To_Address(16#ef24#);
  period: constant Time_Span := To_Time_Span(0.25);
  nextTime: Time := clock + period;

  procedure TCSense_Temperatures is
  begin
    stm1 := t1;
    stm2 := t2;
  end;

  procedure TCCalculate_Average_Temperature is
  begin
    avt := ((stm1 + stm2) / 2);
  end;
```

```
    procedure TCDisplay_Current_Temperature is
    begin
      tm_avt := avt;
    end;

    procedure TCActuate_Heat_Source is
    begin
      state_hsc := hsc;
    end;

    procedure TCActuate_OverHeat_Alram is
    begin
      state_ota := ota;
    end;

  begin
    loop
      delay until nextTime;
      TCSense_Temperatures;
      TCCalculate_Average_Temperature;
      TCDisplay_Current_Temperature;
      shared_object1implInst.SOGet_Target_Temperature2(cttm2);
      if(avt < cttm2) then
         hsc := TRUE;
      else
           hsc := FALSE;
      end if;
      shared_object1implInst.SOSet_Heat_Source_State(hsc);
      TCActuate_Heat_Source;
      if(avt > Max) then
         ota := TRUE;
      else
           ota := FALSE;
      end if;
      TCActuate_OverHeat_Alram;
      nextTime := nextTime + period;
    end loop;
  end Temp_Ctrl_Task1Impl;

begin
  null;
end Heating_Controller5A;



-------------------------------------
-- Shared_Object1Impl.ads
-------------------------------------
package Shared_Object1ImplPkg is

  protected type Shared_Object1Impl is
    procedure SOGet_Target_Temperature1(tm: out Integer);
    procedure SOSet_Target_Temperature(tm: in Integer);
    procedure SOGet_Target_Temperature2(tm: out Integer);
    procedure SOSet_Heat_Source_State(state: in Boolean);
    procedure SOGet_Stored_HeatSource(state: out Boolean);
  private
```

```
    ctm : Integer := 0;
    shss : Boolean := FALSE;
    cttm : Integer := 0;
  end Shared_Object1Impl;
end Shared_Object1ImplPkg;

----------------------------------------
-- Shared_Object1Impl.adb
----------------------------------------
package body Shared_Object1ImplPkg is
  protected body Shared_Object1Impl is
    procedure SOGet_Target_Temperature1(tm: out Integer) is
    begin
      tm := cttm;
    end;
    procedure SOSet_Target_Temperature(tm: in Integer) is
    begin
      cttm := tm;
    end;
    procedure SOGet_Target_Temperature2(tm: out Integer) is
    begin
      tm := cttm;
    end;
    procedure SOSet_Heat_Source_State(state: in Boolean) is
    begin
      shss := state;
    end;
    procedure SOGet_Stored_HeatSource(state: out Boolean) is
    begin
      state := shss;
    end;
  end Shared_Object1Impl;
end Shared_Object1ImplPkg;
```