```
MACHINE
  Machine3
REFINES
  Machine2
SEES
  Context3
VARIABLES
  users
  Pressure
  Heater
  Flag
  TimeStamp
  Delta
  DeltaTime
  SensorClock
  ControllerClock
  SensorAddress
  NextHeater
  roles
  CurrentMode
INVARIANTS
  inv1   :   users ⊆ USERS
  inv2   :   roles ∈ users ⟶ ROLES
  inv3   :   CurrentMode ∈ MODES
EVENTS
  INITIALISATION   ≙
  STATUS
   ordinary
  BEGIN
   act1   :   Pressure ≔ 55
   act2   :   Heater ≔ High
   act3   :   TimeStamp ≔ 0
   act5   :   Delta ≔ 0
   act4   :   Flag ≔ Cont
   act6   :   NextHeater ≔ High
   act9   :   SensorAddress ≔ 0
   act10   :   DeltaTime ≔ 0
   act7   :   SensorClock ≔ 0
   act8   :   ControllerClock ≔ 0
   act11   :   users ≔ ∅
   act12   :   roles ≔ ∅
   act13   :   CurrentMode ≔ AUTOMATED      //   current mode of the system
  END

  NewUser   ≙
  STATUS
   ordinary
  ANY
   us
   ro
  WHERE
   grd1   :   us ∈ USERS \ users
   grd2   :   ro ∈ ROLES
  THEN
   act1   :   roles(us) ≔ ro      //   Set the role of the new user
   act2   :   users ≔ users ∪ {us}      //   Add the new user to users. REQ 15
  END

  ChangeModeSupervised   ≙
  STATUS
   ordinary
  ANY
   us
  WHERE
   grd1   :   us ∈ users
   grd2   :   roles(us) = SUPERVISOR
   grd3   :   CurrentMode = AUTOMATED
  THEN
   act1   :   CurrentMode ≔ SUPERVISED      //   REQ 18
  END

  ChangeModeMonitored   ≙
  STATUS
   ordinary
```

```
ANY
 us
WHERE
 grd1   :   us ∈ users
 grd2   :   roles(us) ∈ {SUPERVISOR,OPERATOR}
 grd3   :   CurrentMode = AUTOMATED
THEN
 act1   :   CurrentMode ≔ MONITORED      //   REQ 16
END

ChangeModeAutomated   ≙
STATUS
 ordinary
ANY
 us
WHERE
 grd1   :   us ∈ users
 grd2   :   ¬(¬(roles(us) = SUPERVISOR ∧ CurrentMode = SUPERVISED) ∧ ¬((roles
            (us) ∈ {OPERATOR,SUPERVISOR}) ∧ CurrentMode = MONITORED))
THEN
 act1   :   CurrentMode ≔ AUTOMATED
END

SetHeaterAutomated   ≙
 extended
STATUS
 ordinary
REFINES
 SetHeater
WHEN
 grd1   :   Pressure ∈ ℕ
 grd2   :   (Pressure ≥ 61) ⇒ (NextHeater = Off)
 grd3   :   (Pressure ∈ {56, 57, 58, 59, 60}) ⇒ NextHeater = Low
 grd4   :   (Pressure ∈ {50, 51, 52, 53, 54, 55}) ⇒ NextHeater = High
 grd5   :   Flag = Cont
 grd8   :   SensorAddress ∈ LegitimateAddresses      //   Assure adress from sensor is Legitimate, REQ 11
 grd7   :   TimeStamp > ControllerClock      //   Should be a new Timestamp
 grd10   :   (CurrentMode = AUTOMATED)      //   REQ 20
THEN
 act1   :   Heater ≔ NextHeater
 act2   :   Flag ≔ Sens
 act3   :   ControllerClock ≔ TimeStamp      //   Update controller clock
END

SafeShutDown   ≙
 extended
STATUS
 ordinary
REFINES
 SafeShutDown
WHEN
 grd1   :   Flag = Cont
 grd2   :   (TimeStamp < ControllerClock) ∨ (SensorAddress ∉ LegitimateAddresses)      //   REQ 12, 13, If a non valid timestamp
THEN
 act1   :   Heater ≔ Off
 act2   :   Flag ≔ Sens
END

PressureSens   ≙
 extended
STATUS
 ordinary
REFINES
 PressureSens
WHEN
 grd1   :   Flag = Sens
 grd2   :   (Heater = High) ⇒ (Delta ∈ {0, 1, 2, 3})
 grd3   :   (Heater = Low) ⇒ (Delta ∈ {−2, −1, 0})
 grd4   :   (Heater = Off) ⇒ (Delta ∈ {−1, −2})
 grd5   :   Pressure + Delta ∈ ℕ
 grd6   :   DeltaTime ∈ 1‥TMAX      //   Time between 1 and TMAX
 grd7   :   SensorClock + DeltaTime ∈ ℕ
THEN
 act1   :   Flag ≔ Cont
 act2   :   Pressure ≔ Pressure + Delta
 act4   :   SensorClock ≔ SensorClock + DeltaTime      //   Update SensorClock
```

```
     act5   :   TimeStamp ≔ SensorClock      //   Update Timestamp, part of REQ 9
     act6   :   SensorAddress ≔ 1      //   Valid sensorAdress, REQ 10
   END

 END
```