

# Contents

CONTEXT <b>discCtx</b>	<b>2</b>
CONTEXT <b>CruiseCtx</b>	<b>3</b>
CONTEXT <b>CruiseCtx1</b>	<b>4</b>
CONTEXT <b>CruiseCtx2</b>	<b>5</b>
MACHINE <b>CruiseControl</b>	<b>6</b>

**CONTEXT** discCtx

**CONSTANTS**

DT

**AXIOMS**

axm1:  $DT \in \mathbb{N}_1$

**END**

**CONTEXT** CruiseCtx

**CONSTANTS**

CRUISE\_MAX\_SPEED viteza max admisa a controlului de croaziera

CRUISE\_MIN\_SPEED viteza min admisa a controlului de croaziera

**AXIOMS**

axm1:  $CRUISE\_MAX\_SPEED \in \mathbb{N}$

axm2:  $CRUISE\_MAX\_SPEED = 180000$   
180 km/h exprimat în m/s \* 1000

axm3:  $CRUISE\_MIN\_SPEED \in \mathbb{N}$

axm4:  $CRUISE\_MIN\_SPEED = 10000$   
10 km/h exprimat în m/s \* 1000

**END**

**CONTEXT** CruiseCtx1

**EXTENDS** CruiseCtx

**CONSTANTS**

PEDAL\_COMMANDS comenzi posibile pentru apasarea pedalelor

SAFETY\_DISTANCE stabilirea unei distante de siguranta

**AXIOMS**

**axm1:**  $PEDAL\_COMMANDS = 0 \dots 3$

comenzi pentru pedale: 0=nicio actiune, 1=acceleratie, 2=franare, 3=resetare

**axm2:**  $SAFETY\_DISTANCE \in \mathbb{N}$

**axm3:**  $SAFETY\_DISTANCE = 50$

distanta de siguranta are valoarea prestabilita de 50 de metri

**END**

**CONTEXT** CruiseCtx2

**EXTENDS** CruiseCtx1

**CONSTANTS**

VEHICLE\_MAX\_SPEED viteza max a vehiculului

MAX\_INACTIVITY limita max a timpului de inactivitate permis înainte ca sistemul să ia măsuri, cum ar fi revenirea automată la modul Cruise Mode

**AXIOMS**

axm1:  $VEHICLE\_MAX\_SPEED = 200000$   
200 km/h exprimat în unități (m/s x 1000)

axm2:  $MAX\_INACTIVITY \in \mathbb{N}$

axm3:  $MAX\_INACTIVITY = 30$   
30 secunde de inactivitate

**END**

**MACHINE** CruiseControl**SEES** CruiseCtx2, discCtx**VARIABLES**

cruise\_mode booleana care indica daca modul cruise este activ  
 follow\_mode booleana care indica daca modul follow este activ  
 pedal\_command  
 emergency\_mode booleana care indica daca modul emergency este activ  
 cruise\_speed viteza curenta a vehiculului in modul cruise (limitata intre CRUISE\_MIN\_SPEED si CRUISE\_MAX\_SPEED)  
 engine\_state booleana care indica daca motorul este pornit sau oprit  
 distance\_sensor valoare intreaga care reprezinta distanta pana la un obstacol (-1 indica lipsa datelor)  
 vehicle\_speed valoare intreaga care reprezinta viteza curenta a vehiculului (limitata intre 0 si VEHICLE\_MAX\_SPEED)  
 safety\_distance distanta de siguranta pentru modul follow (numar natural)  
 warning\_alert booleana care indica daca un avertisment este activ  
 t variabila de timp folosita pentru esantionare (numar natural)  
 canRead booleana care indica daca datele pot fi citite  
 workingClock booleana care indica daca ceasul de esantionare functioneaza  
 inactivity\_timer

**INVARIANTS**

**inv1:**  $cruise\_mode \in \text{BOOL}$   
 Modul de croaziera este activ sau inactiv  
**inv2:**  $follow\_mode \in \text{BOOL}$   
 Modul de urmarire este activ sau inactiv  
**inv3:**  $emergency\_mode \in \text{BOOL}$   
 Modul de urgenta este activ sau inactiv  
**inv4:**  $cruise\_speed \in \text{CRUISE\_MIN\_SPEED} .. \text{CRUISE\_MAX\_SPEED}$   
 Viteza de croaziera trebuie sa fie intre limitele stabilite  
**inv5:**  $engine\_state \in \text{BOOL}$   
 Motorul poate fi pornit sau oprit  
**inv6:**  $distance\_sensor \in -1 .. \text{SAFETY\_DISTANCE}$   
 Senzorul de distanta poate fi intre -1 si distanta de siguranta  
**inv7:**  $vehicle\_speed \in 0 .. \text{VEHICLE\_MAX\_SPEED}$   
 Viteza vehiculului trebuie sa fie intre 0 si viteza maxima permisa  
**inv8:**  $safety\_distance \in \mathbb{N}$   
 Distanța de siguranță este un număr natural  
**inv9:**  $warning\_alert \in \text{BOOL}$   
 Alerta de avertizare este activa sau inactiva  
**inv10:**  $t \in \mathbb{N}$   
 Cronometrul de esantionare este un număr natural  
**inv11:**  $canRead \in \text{BOOL}$   
 Indica daca se poate efectua o citire  
**inv12:**  $workingClock \in \text{BOOL}$   
 Starea ceasului de lucru  
**inv13:**  $pedal\_command \in \text{PEDAL\_COMMANDS}$   
 Comenzile pedalei (0: fara actiune, 1: accelereaza, 2: franeaza, 3: reseteaza)  
**inv14:**  $inactivity\_timer \in 0 .. \text{MAX\_INACTIVITY}$   
 Cronometrul de inactivitate este intre 0 si limita maxima

**EVENTS****Initialisation**

configureaza toate variabilele sistemului in starile lor initiale

**begin**

**act1:**  $cruise\_mode := \text{FALSE}$   
 Modul de croaziera este dezactivat  
**act2:**  $follow\_mode := \text{FALSE}$   
 Modul de urmarire este dezactivat

```

    act3: emergency_mode := FALSE
        Modul de urgenta este dezactivat
    act4: cruise_speed := 50000
        Viteza de croaziera implicita
    act5: engine_state := FALSE
        Motorul este oprit
    act6: distance_sensor := -1
        Senzorul de distanta este inactiv
    act7: vehicle_speed := 0
        Viteza vehiculului este 0
    act8: safety_distance := 50
        Distanța de siguranță inițială este 50
    act9: warning_alert := FALSE
        Alerta este dezactivată
    act10: t := 0
        Cronometrul începe de la 0
    act11: canRead := FALSE
        Nu este permisă citirea inițială
    act12: workingClock := FALSE
        Ceasul de lucru este dezactivat
    act13: pedal_command := 0
        Fără nicio comandă activă pe pedala
    act14: inactivity_timer := 0
        Cronometrul de inactivitate este inițializat
end
Event StartEngine ⟨ordinary⟩ ≐
    porneste motorul vehiculului dacă acesta este oprit
when
    grd1: engine_state = FALSE
        Motorul trebuie să fie oprit pentru a putea fi pornit
then
    act1: engine_state := TRUE
        Motorul este pornit
end
Event StopEngine ⟨ordinary⟩ ≐
    oprește motorul vehiculului dacă acesta este pornit. Toate modulele sunt dezactivate
when
    grd1: engine_state = TRUE
        Motorul trebuie să fie pornit pentru a putea fi oprit
then
    act1: engine_state := FALSE
        oprește motorul
    act2: cruise_mode := FALSE
        resetează cruise_mode
    act3: follow_mode := FALSE
        resetează follow_mode
    act4: emergency_mode := FALSE
        resetează emergency_mode
end
Event EnterCruiseMode ⟨ordinary⟩ ≐
    activează modul de croazieră atunci când motorul este pornit, nu există vehicule în apropiere (distance_sensor = -1), iar vehiculul are o viteză mai mare de 0
when
    grd1: engine_state = TRUE
        Motorul este pornit
    grd2: distance_sensor = -1
    grd3: vehicle_speed > 0
then

```

```

    act1: cruise_mode := TRUE
           Modul de croaziera este activat
    act2: follow_mode := FALSE
           Modul de urmarire este dezactivat
    act3: emergency_mode := FALSE
           Modul de urgenta este dezactivat
    act4: inactivity_timer := 0
           reseteaza cronometrul de inactivitate
end
Event EnterFollowMode ⟨ordinary⟩ ≐
ajusteaza comportamentul vehiculului pentru a mentine distanta fata de alte vehicule
when
    grd1: engine_state = TRUE
    grd2: distance_sensor ≤ SAFETY_DISTANCE
    grd3: distance_sensor > 0
then
    act1: cruise_mode := FALSE
    act2: follow_mode := TRUE
    act3: emergency_mode := FALSE
    act4: inactivity_timer := 0
end
Event EnterEmergencyMode ⟨ordinary⟩ ≐
activeaza modul de urgenta atunci când distanta fata de un alt vehicul este 0 (distance_sensor = 0), iar
sistemul emite o alerta
when
    grd1: engine_state = TRUE
           conditii pentru a intra in modul emergency
    grd2: distance_sensor = 0
then
    act1: cruise_mode := FALSE
    act2: follow_mode := FALSE
    act3: emergency_mode := TRUE
    act4: warning_alert := TRUE
    act5: inactivity_timer := 0
end
Event IncreaseCruiseSpeed ⟨ordinary⟩ ≐
when
    grd1: cruise_mode = TRUE
    grd2: cruise_speed + 2500 ≤ CRUISE_MAX_SPEED
then
    act1: cruise_speed := cruise_speed + 2500
           creste cruise_speed cu 2500 unitati
end
Event DecreaseCruiseSpeed ⟨ordinary⟩ ≐
when
    grd1: cruise_mode = TRUE
    grd2: cruise_speed - 2500 ≥ CRUISE_MIN_SPEED
then
    act1: cruise_speed := cruise_speed - 2500
end
Event UpdateSafetyDistance ⟨ordinary⟩ ≐
when
    grd1: engine_state = TRUE
           actualizeaza distanta de siguranta doar cand motorul este pornit
then
    act1: safety_distance := SAFETY_DISTANCE
           seteaza safety_distance la constanta predefinita
end

```



**Event** TriggerDriverWarning  $\langle \text{ordinary} \rangle \hat{=}$

```

when
  grd1: warning_alert = TRUE
        declansare doar daca warning_alert este activ
then
  act1: warning_alert := TRUE
        mentine avertismentul activ
end

```

**Event** MonitorDistanceAboveSafety  $\langle \text{ordinary} \rangle \hat{=}$

```

distanta mai mare decat pragul de siguranta
when
  grd1: engine_state = TRUE
  grd2: distance_sensor > SAFETY_DISTANCE
then
  act1: cruise_mode := TRUE
  act2: follow_mode := FALSE
  act3: emergency_mode := FALSE
  act4: warning_alert := FALSE
end

```

**Event** MonitorDistanceWithinSafety  $\langle \text{ordinary} \rangle \hat{=}$

```

when
  grd1: engine_state = TRUE
  grd2: distance_sensor ≤ SAFETY_DISTANCE
  grd3: distance_sensor > 0
then
  act1: cruise_mode := FALSE
  act2: follow_mode := TRUE
  act3: emergency_mode := FALSE
  act4: warning_alert := FALSE
end

```

**Event** MonitorDistanceCritical  $\langle \text{ordinary} \rangle \hat{=}$

```

when
  grd1: engine_state = TRUE
  grd2: distance_sensor = 0
then
  act1: cruise_mode := FALSE
  act2: follow_mode := FALSE
  act3: emergency_mode := TRUE
  act4: warning_alert := TRUE
end

```

**Event** StartSampling  $\langle \text{ordinary} \rangle \hat{=}$

```

when
  grd1: workingClock = FALSE
        porneste esantionarea doar daca ceasul nu functioneaza si datele nu pot fi citite
  grd2: canRead = FALSE
then
  act1: workingClock := TRUE
        porneste ceasul de esantionare
end

```

**Event** StopSampling  $\langle \text{ordinary} \rangle \hat{=}$

```

when
  grd1: workingClock = TRUE
        opreste esantionarea cand ceasul functioneaza si timpul ajunge la pragul definit DT
  grd2: t = DT
then
  act1: canRead := TRUE
        permite citirea datelor
end

```

```

    act2:  $t := 0$ 
           reseteaza contorul de timp  $t$  la 0
    act3:  $workingClock := FALSE$ 
           opreste ceasul de esantionare
end
Event Sampling ⟨ordinary⟩  $\hat{=}$ 
when
    grd1:  $workingClock = TRUE$ 
           esantioneaza cat timp ceasul functioneaza si timpul este mai mic decat  $DT$ 
    grd2:  $t < DT$ 
then
    act1:  $t := t + 1$ 
           incrementeaza contorul de timp  $t$ 
end
Event ApplyPedalCommand ⟨ordinary⟩  $\hat{=}$ 
any
    cmd
where
    grd1:  $cmd \in \{0, 1, 2, 3\}$ 
           0: no action, 1: accelerate, 2: brake, 3: reset
then
    act1:  $pedal\_command := cmd$ 
end
Event Accelerate ⟨ordinary⟩  $\hat{=}$ 
when
    grd1:  $pedal\_command = 1$ 
    grd2:  $vehicle\_speed + 5000 \leq VEHICLE\_MAX\_SPEED$ 
then
    act1:  $vehicle\_speed := vehicle\_speed + 5000$ 
    act2:  $inactivity\_timer := 0$ 
end
Event BrakeDecrease ⟨ordinary⟩  $\hat{=}$ 
when
    grd1:  $pedal\_command = 2$ 
    grd2:  $vehicle\_speed > 5000$ 
then
    act1:  $vehicle\_speed := vehicle\_speed - 5000$ 
    act2:  $inactivity\_timer := 0$ 
end
Event ResetPedalCommand ⟨ordinary⟩  $\hat{=}$ 
    reseteaza comenzile pedalei ( $pedal\_command$ ) la starea initiala
when
    grd1:  $pedal\_command = 3$ 
then
    act1:  $pedal\_command := 0$ 
    act2:  $inactivity\_timer := 0$ 
end
Event BrakeStop ⟨ordinary⟩  $\hat{=}$ 
when
    grd1:  $pedal\_command = 2$ 
    grd2:  $vehicle\_speed \leq 5000$ 
then
    act1:  $vehicle\_speed := 0$ 
    act2:  $inactivity\_timer := 0$ 
end
Event CheckInactivity ⟨ordinary⟩  $\hat{=}$ 
    daca vehiculul este inactiv pentru o perioada maxima definita ( $MAX\_INACTIVITY$ ), revine automat la
    modul de croaziera

```

```
when
  grd1: inactivity_timer = MAX_INACTIVITY
then
  act1: cruise_mode := TRUE
  act2: follow_mode := FALSE
  act3: emergency_mode := FALSE
  act4: warning_alert := FALSE
  act5: inactivity_timer := 0
end
END
```