aircraft

Tight Rope v0.6

November 1, 2015

1 Network

section *NetworkChannels* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableChan*, *TopLevelMissionSequencerFWChan*,
FrameworkChan, *SafeletChan*

channelset *TerminateSync* ==
⦃ *schedulables_terminated*, *schedulables_stopped*, *get_activeSchedulables* ⦄

channelset *ControlTierSync* ==
⦃ *start_toplevel_sequencer*, *done_toplevel_sequencer*, *done_safeletFW* ⦄

channelset *TierSync* ==
⦃ *start_mission* . , *done_mission* . ,
done_safeletFW , *done_toplevel_sequencer* ⦄

channelset *MissionSync* ==
⦃ *done_safeletFW* , *done_toplevel_sequencer*, *register* ,
signalTerminationCall, *signalTerminationRet*, *activate_schedulables*, *done_schedulable*,
cleanupSchedulableCall, *cleanupSchedulableRet* ⦄

channelset *SchedulablesSync* ==
⦃ *activate_schedulables*, *done_safeletFW* , *done_toplevel_sequencer* ⦄

channelset *ClusterSync* ==
⦃ *done_toplevel_sequencer*, *done_safeletFW* ⦄

channelset *AppSync* ==
⋃{*SafeltAppSync*, *MissionSequencerAppSync*, *MissionAppSync*,
MTAppSync, *OSEHSync*, *APEHSync*,
⦃ *getSequencer*, *end_mission_app*, *end_managedThread_app*,
setCeilingPriority, *requestTerminationCall*, *requestTerminationRet*, *terminationPendingCall*,
terminationPendingRet, *handleAsyncEventCall*, *handleAsyncEventRet* ⦄}

channelset *ObjectSync* ==
⦃ ⦄

channelset *ThreadSync* ==
⦃ ⦄

channelset *LockingSync* ==
⦃ *lockAcquired*, *startSyncMeth*, *endSyncMeth*, *waitCall*, *waitRet*, *notify* ⦄

channelset $Tier0Sync ==$
    $\{\![\, done\_toplevel\_sequencer, done\_safeletFW,$
$start\_mission\,.\,, done\_mission\,.\,,$
    $initializeRet\,.\,, requestTermination\,.\,.\,,$
$start\_mission\,.\,, done\_mission\,.\,,$
    $initializeRet\,.\,, requestTermination\,.\,.\,,$
$start\_mission\,.\,, done\_mission\,.\,,$
    $initializeRet\,.\,, requestTermination\,.\,.\,]\!\}$

$start\_mission\,.\,, done\_mission\,.\,,$

section *Program* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
   SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *MissionFW*,
   SafeletFW, *TopLevelMissionSequencerFW*, *NetworkChannels*, *ManagedThreadFW*,
   SchedulableMissionSequencerFW, *PeriodicEventHandlerFW*, *OneShotEventHandlerFW*,
   AperiodicEventHandlerFW, *ACSafeletApp*, *MainMissionSequencerApp*,
   ObjectFW, *ThreadFW*,   *MainMissionApp*, *ACModeChangerApp*, *ControlHandlerApp*, *CommunicationsHandlerAp*

process *ControlTier* $\widehat{=}$
$$\begin{pmatrix} SafeletFW \\ \quad \llbracket ControlTierSync \rrbracket \\ TopLevelMissionSequencerFW\,(MainMissionSequencer) \end{pmatrix}$$

process *Tier*0 $\widehat{=}$
$$\begin{pmatrix} MissionFW\,(MainMission) \\ \quad \llbracket MissionSync \rrbracket \\ \begin{pmatrix} SchedulableMissionSequencerFW\,() \\ \quad \llbracket SchedulablesSync \rrbracket \\ \begin{pmatrix} AperiodicEventHandlerFW\,() \\ \quad \llbracket SchedulablesSync \rrbracket \\ AperiodicEventHandlerFW\,() \end{pmatrix} \\ \quad \llbracket SchedulablesSync \rrbracket \\ \begin{pmatrix} PeriodicEventHandlerFW\,() \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW\,() \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW\,() \end{pmatrix} \end{pmatrix} \end{pmatrix}$$

process *Tier*1 $\widehat{=}$
$$\begin{pmatrix} \begin{pmatrix} MissionFW\,(TakeOffMission) \\ \quad \llbracket MissionSync \rrbracket \\ \begin{pmatrix} \begin{pmatrix} AperiodicEventHandlerFW\,() \\ \quad \llbracket SchedulablesSync \rrbracket \\ AperiodicEventHandlerFW\,() \end{pmatrix} \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW\,() \end{pmatrix} \end{pmatrix} \\ \quad \llbracket ClusterSync \rrbracket \\ \begin{pmatrix} MissionFW\,(CruiseMission) \\ \quad \llbracket MissionSync \rrbracket \\ \begin{pmatrix} AperiodicEventHandlerFW\,() \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW\,() \end{pmatrix} \end{pmatrix} \\ \quad \llbracket ClusterSync \rrbracket \\ \begin{pmatrix} MissionFW\,(LandMission) \\ \quad \llbracket MissionSync \rrbracket \\ \begin{pmatrix} \begin{pmatrix} AperiodicEventHandlerFW\,() \\ \quad \llbracket SchedulablesSync \rrbracket \\ AperiodicEventHandlerFW\,() \end{pmatrix} \\ \quad \llbracket SchedulablesSync \rrbracket \\ \begin{pmatrix} PeriodicEventHandlerFW\,() \\ \quad \llbracket SchedulablesSync \rrbracket \\ PeriodicEventHandlerFW\,() \end{pmatrix} \end{pmatrix} \end{pmatrix} \end{pmatrix}$$

process *Framework* $\widehat{=}$
$$\begin{pmatrix} ControlTier \\ \quad \llbracket TierSync \rrbracket \\ \begin{pmatrix} Tier0 \\ \quad \llbracket Tier0Sync \rrbracket \\ Tier1 \end{pmatrix} \end{pmatrix}$$

3

process *Application* $\widehat{=}$

$\Big($ *ACSafeletApp*

⫴

MainMissionSequencerApp

⫴

MainMissionApp(*hijac.tools.tightrope.environments.VariableEnv* • 71*c27ee8, hijac.tools.tightrope.environments.Variab*

⫴

ACModeChangerApp(*MainMission*)

⫴

ControlHandlerApp

⫴

CommunicationsHandlerApp

⫴

EnvironmentMonitorApp(*MainMission*)

⫴

FlightSensorsMonitorApp(*MainMission*)

⫴

AperiodicSimulatorApp(*AperiodicEventHandler*)

⫴

TakeOffMissionApp(*hijac.tools.tightrope.environments.VariableEnv* • 6*e950bcf, hijac.tools.tightrope.environments.Var*

⫴

LandingGearHandlerTakeOffApp(*TakeOffMission*)

⫴

TakeOffFailureHandlerApp(*TakeOffMission*)

⫴

TakeOffMonitorApp(*TakeOffMission, AperiodicEventHandler*)

⫴

CruiseMissionApp(*hijac.tools.tightrope.environments.VariableEnv* • 46*dffdc3, hijac.tools.tightrope.environments.Varia*

⫴

BeginLandingHandlerApp(*Mission*)

⫴

NavigationMonitorApp(*CruiseMission*)

⫴

LandMissionApp(*hijac.tools.tightrope.environments.VariableEnv* • 53*dbe163, hijac.tools.tightrope.environments.Variab*

⫴

LandingGearHandlerLandApp(*LandMission*)

⫴

SafeLandingHandlerApp(*LandMission*)

⫴

GroundDistanceMonitorApp(*LandMission*)

⫴

InstrumentLandingSystemMonitorApp(*LandMission*)

$Locking \mathrel{\widehat{=}}$
$$
\left(
\begin{array}{l}
\left(
\begin{array}{l}
ThreadFW\,(LandingGearHandlerLandThread,\, MinPriority) \\
\quad [\![\,ThreadSync\,]\!] \\
ThreadFW\,(SafeLandingHandlerThread,\, MinPriority) \\
\quad [\![\,ThreadSync\,]\!] \\
ThreadFW\,(GroundDistanceMonitorThread,\, MinPriority) \\
\quad [\![\,ThreadSync\,]\!] \\
ThreadFW\,(InstrumentLandingSystemMonitorThread,\, MinPriority)
\end{array}
\right) \\
|\!|\!| \\
\left(
\begin{array}{l}
ObjectFW\,(ACSafeletObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(MainMissionObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(TakeOffMissionObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(CruiseMissionObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(LandMissionObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(ACModeChangerObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(EnvironmentMonitorObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(ControlHandlerObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(FlightSensorsMonitorObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(CommunicationsHandlerObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(AperiodicSimulatorObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(LandingGearHandlerTakeOffObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(TakeOffMonitorObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(TakeOffFailureHandlerObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(BeginLandingHandlerObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(NavigationMonitorObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(GroundDistanceMonitorObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(LandingGearHandlerLandObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(InstrumentLandingSystemMonitorObject) \\
\quad [\![\,ObjectSync\,]\!] \\
ObjectFW\,(SafeLandingHandlerObject)
\end{array}
\right)
\end{array}
\right)
$$

process $Program \mathrel{\widehat{=}} Framework \,[\![\,AppSync\,]\!]\, Application \,[\![\,LockingSync\,]\!]\, Locking$

2 ID Files

2.1 MissionIds

section *MissionIds* **parents** *scj_prelude*, *MissionId*

MainMission : *MissionID*
TakeOffMission : *MissionID*
CruiseMission : *MissionID*
LandMission : *MissionID*

---

distinct⟨nullMissionId, MainMission,
TakeOffMission,
CruiseMission,
LandMission⟩

2.2 SchedulablesIds

section *SchedulableIds* **parents** *scj_prelude*, *SchedulableId*

MainMissionSequencer : *SchedulableID*
ACModeChanger : *SchedulableID*
EnvironmentMonitor : *SchedulableID*
ControlHandler : *SchedulableID*
FlightSensorsMonitor : *SchedulableID*
CommunicationsHandler : *SchedulableID*
AperiodicSimulator : *SchedulableID*
LandingGearHandlerTakeOff : *SchedulableID*
TakeOffMonitor : *SchedulableID*
TakeOffFailureHandler : *SchedulableID*
BeginLandingHandler : *SchedulableID*
NavigationMonitor : *SchedulableID*
GroundDistanceMonitor : *SchedulableID*
LandingGearHandlerLand : *SchedulableID*
InstrumentLandingSystemMonitor : *SchedulableID*
SafeLandingHandler : *SchedulableID*

---

distinct⟨nullSequencerId, nullSchedulableId, ACModeChanger,
EnvironmentMonitor,
ControlHandler,
FlightSensorsMonitor,
CommunicationsHandler,
AperiodicSimulator,
LandingGearHandlerTakeOff,
TakeOffMonitor,
TakeOffFailureHandler,
BeginLandingHandler,
NavigationMonitor,
GroundDistanceMonitor,
LandingGearHandlerLand,
InstrumentLandingSystemMonitor,
SafeLandingHandler⟩

2.3 ThreadIds

section *ThreadIds* **parents** *scj_prelude*, *GlobalTypes*

$ACModeChangerThread : ThreadID$
$EnvironmentMonitorThread : ThreadID$
$ControlHandlerThread : ThreadID$
$FlightSensorsMonitorThread : ThreadID$
$CommunicationsHandlerThread : ThreadID$
$AperiodicSimulatorThread : ThreadID$
$LandingGearHandlerTakeOffThread : ThreadID$
$TakeOffMonitorThread : ThreadID$
$TakeOffFailureHandlerThread : ThreadID$
$BeginLandingHandlerThread : ThreadID$
$NavigationMonitorThread : ThreadID$
$GroundDistanceMonitorThread : ThreadID$
$LandingGearHandlerLandThread : ThreadID$
$InstrumentLandingSystemMonitorThread : ThreadID$
$SafeLandingHandlerThread : ThreadID$

$distinct \langle SafeletThreadId, nullThreadId,$
$ACModeChangerThread,$
$EnvironmentMonitorThread,$
$ControlHandlerThread,$
$FlightSensorsMonitorThread,$
$CommunicationsHandlerThread,$
$AperiodicSimulatorThread,$
$LandingGearHandlerTakeOffThread,$
$TakeOffMonitorThread,$
$TakeOffFailureHandlerThread,$
$BeginLandingHandlerThread,$
$NavigationMonitorThread,$
$GroundDistanceMonitorThread,$
$LandingGearHandlerLandThread,$
$InstrumentLandingSystemMonitorThread,$
$SafeLandingHandlerThread \rangle$

2.4 ObjectIds

section *ObjectIds* **parents** *scj_prelude, GlobalTypes*

7

$ACSafeletObject : ObjectID$
$MainMissionObject : ObjectID$
$TakeOffMissionObject : ObjectID$
$CruiseMissionObject : ObjectID$
$LandMissionObject : ObjectID$
$ACModeChangerObject : ObjectID$
$EnvironmentMonitorObject : ObjectID$
$ControlHandlerObject : ObjectID$
$FlightSensorsMonitorObject : ObjectID$
$CommunicationsHandlerObject : ObjectID$
$AperiodicSimulatorObject : ObjectID$
$LandingGearHandlerTakeOffObject : ObjectID$
$TakeOffMonitorObject : ObjectID$
$TakeOffFailureHandlerObject : ObjectID$
$BeginLandingHandlerObject : ObjectID$
$NavigationMonitorObject : ObjectID$
$GroundDistanceMonitorObject : ObjectID$
$LandingGearHandlerLandObject : ObjectID$
$InstrumentLandingSystemMonitorObject : ObjectID$
$SafeLandingHandlerObject : ObjectID$

$distinct\langle ACSafeletObject,$
$MainMissionObject,$
$TakeOffMissionObject,$
$CruiseMissionObject,$
$LandMissionObject,$
$ACModeChangerObject,$
$EnvironmentMonitorObject,$
$ControlHandlerObject,$
$FlightSensorsMonitorObject,$
$CommunicationsHandlerObject,$
$AperiodicSimulatorObject,$
$LandingGearHandlerTakeOffObject,$
$TakeOffMonitorObject,$
$TakeOffFailureHandlerObject,$
$BeginLandingHandlerObject,$
$NavigationMonitorObject,$
$GroundDistanceMonitorObject,$
$LandingGearHandlerLandObject,$
$InstrumentLandingSystemMonitorObject,$
$SafeLandingHandlerObject\rangle$

3 Safelet

section *ACSafeletApp* **parents** *scj_prelude, SchedulableId, SchedulableIds, SafeletChan*

process *ACSafeletApp* $\widehat{=}$ **begin**

$InitializeApplication \widehat{=}$
$$\begin{pmatrix} initializeApplicationCall \longrightarrow \\ initializeApplicationRet \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$GetSequencer \widehat{=}$
$$\begin{pmatrix} getSequencerCall \longrightarrow \\ getSequencerRet\,!\,MainMissionSequencer \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$Methods \widehat{=}$
$$\begin{pmatrix} GetSequencer \\ \Box \\ InitializeApplication \end{pmatrix} ;\ Methods$$

$\bullet\ (Methods) \triangle (end\_safelet\_app \longrightarrow \textbf{Skip})$

end

4 Top Level Mission Sequencer

section *MainMissionSequencerApp* **parents** *TopLevelMissionSequencerChan,*
  MissionId, MissionIds, SchedulableId, MainMissionSequencerClass

process *MainMissionSequencerApp* $\widehat{=}$ **begin**

State _____
  this : **ref** *MainMissionSequencerClass*
_____

state *State*

Init _____
  State′
  _____
  this′ = **new** *MainMissionSequencerClass*()
_____

GetNextMission $\widehat{=}$ **var** *ret* : *MissionID* $\bullet$
$\begin{pmatrix} getNextMissionCall . MainMissionSequencer \longrightarrow \\ ret := this . getNextMission(); \\ getNextMissionRet . MainMissionSequencer \,!\, ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

Methods $\widehat{=}$
$\begin{pmatrix} GetNextMission \end{pmatrix}$; *Methods*

$\bullet$ (*Init* ; *Methods*) $\triangle$ (*end_sequencer_app . MainMissionSequencer* $\longrightarrow$ **Skip**)

end

class *MainMissionSequencerClass* $\widehat{=}$ **begin**

state *State*
> *returnedMission* : $\mathbb{B}$

state *State*

initial *Init*
> *State′*
>
> *returnedMission′* = *false*

protected *getNextMission* $\widehat{=}$ **var** *ret* : *MissionID* •

$$
\begin{pmatrix}
\textbf{if} \, (\neg \, returnedMission = \textbf{True}) \longrightarrow \\
\quad \begin{pmatrix} returnedMission := \textbf{True} \, ; \\ ret := MainMission \end{pmatrix} \\
[\!] \, \neg \, (\neg \, returnedMission = \textbf{True}) \longrightarrow \\
\quad \begin{pmatrix} ret := nullMissionId \end{pmatrix} \\
\textbf{fi}
\end{pmatrix}
$$

• **Skip**

end

5 Missions

5.1 MainMission

section *MainMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *MainMissionClass*   , *MainMissionMethChan*

process *MainMissionApp* $\cong$ *storageParameters* : *MissionID*, *storageParametersSchedulable* : *MissionID*, *aCModeChange*

State
$\quad$ *this* : **ref** *MainMissionClass*

---

state *State*

Init
$\quad$ *State*$'$
$\quad$ *this*$'$ = **new** *MainMissionClass*()

InitializePhase $\cong$
$\begin{pmatrix} initializeCall\,.\,MainMission \longrightarrow \\ register\,!\,ACModeChanger\,!\,MainMission \longrightarrow \\ register\,!\,EnvironmentMonitor\,!\,MainMission \longrightarrow \\ register\,!\,ControlHandler\,!\,MainMission \longrightarrow \\ register\,!\,FlightSensorsMonitor\,!\,MainMission \longrightarrow \\ register\,!\,CommunicationsHandler\,!\,MainMission \longrightarrow \\ register\,!\,AperiodicSimulator\,!\,MainMission \longrightarrow \\ initializeRet\,.\,MainMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

CleanupPhase $\cong$
$\begin{pmatrix} cleanupMissionCall\,.\,MainMission \longrightarrow \\ cleanupMissionRet\,.\,MainMission\,!\,\textbf{True} \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

getAirSpeedMeth $\cong$ **var** *ret* : *double* $\bullet$
$\begin{pmatrix} getAirSpeedCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getAirSpeed(); \\ getAirSpeedRet\,.\,MainMission\,!\,ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

getAltitudeMeth $\cong$ **var** *ret* : *double* $\bullet$
$\begin{pmatrix} getAltitudeCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getAltitude(); \\ getAltitudeRet\,.\,MainMission\,!\,ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

getCabinPressureMeth $\cong$ **var** *ret* : *double* $\bullet$
$\begin{pmatrix} getCabinPressureCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getCabinPressure(); \\ getCabinPressureRet\,.\,MainMission\,!\,ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

$getEmergencyOxygenMeth \cong \mathbf{var}\ ret : double \bullet$
$\begin{pmatrix} getEmergencyOxygenCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getEmergencyOxygen(); \\ getEmergencyOxygenRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$getFuelRemainingMeth \cong \mathbf{var}\ ret : double \bullet$
$\begin{pmatrix} getFuelRemainingCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getFuelRemaining(); \\ getFuelRemainingRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$getHeadingMeth \cong \mathbf{var}\ ret : double \bullet$
$\begin{pmatrix} getHeadingCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getHeading(); \\ getHeadingRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$setAirSpeedMeth \cong$
$\begin{pmatrix} setAirSpeedCall\,.\,MainMission\,?\,airSpeed \longrightarrow \\ this\,.\,setAirSpeed(airSpeed); \\ setAirSpeedRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$setAltitudeMeth \cong$
$\begin{pmatrix} setAltitudeCall\,.\,MainMission\,?\,altitude \longrightarrow \\ this\,.\,setAltitude(altitude); \\ setAltitudeRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$setCabinPressureMeth \cong$
$\begin{pmatrix} setCabinPressureCall\,.\,MainMission\,?\,cabinPressure \longrightarrow \\ this\,.\,setCabinPressure(cabinPressure); \\ setCabinPressureRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$setEmergencyOxygenMeth \cong$
$\begin{pmatrix} setEmergencyOxygenCall\,.\,MainMission\,?\,emergencyOxygen \longrightarrow \\ this\,.\,setEmergencyOxygen(emergencyOxygen); \\ setEmergencyOxygenRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$setFuelRemainingMeth \cong$
$\begin{pmatrix} setFuelRemainingCall\,.\,MainMission\,?\,fuelRemaining \longrightarrow \\ this\,.\,setFuelRemaining(fuelRemaining); \\ setFuelRemainingRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$setHeadingMeth \cong$
$\begin{pmatrix} setHeadingCall\,.\,MainMission\,?\,heading \longrightarrow \\ this\,.\,setHeading(heading); \\ setHeadingRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$$
Methods \mathrel{\widehat{=}} \left(
\begin{array}{l}
InitializePhase \\
\square \\
CleanupPhase \\
\square \\
getAirSpeedMeth \\
\square \\
getAltitudeMeth \\
\square \\
getCabinPressureMeth \\
\square \\
getEmergencyOxygenMeth \\
\square \\
getFuelRemainingMeth \\
\square \\
getHeadingMeth \\
\square \\
setAirSpeedMeth \\
\square \\
setAltitudeMeth \\
\square \\
setCabinPressureMeth \\
\square \\
setEmergencyOxygenMeth \\
\square \\
setFuelRemainingMeth \\
\square \\
setHeadingMeth
\end{array}
\right) \ ; \ Methods
$$

• $(Init \ ; \ Methods) \triangle (end\_mission\_app \,.\, MainMission \longrightarrow \textbf{Skip})$

end

class *MainMissionClass* $\widehat{=}$ **begin**

---
state *State*
> *ALTITUDE_READING_ON_GROUND* : *double*
> *cabinPressure* : *double*
> *emergencyOxygen* : *double*
> *fuelRemaining* : *double*
> *altitude* : *double*
> *airSpeed* : *double*
> *heading* : *double*
---

state *State*

---
initial *Init*
> *State'*
> ---
> *ALTITUDE_READING_ON_GROUND'* = 0.0
---

public *getAirSpeed* $\widehat{=}$ **var** *ret* : *double* •
$(ret := airSpeed)$

public *getAltitude* $\widehat{=}$ **var** *ret* : *double* •
$(ret := altitude)$

public *getCabinPressure* $\widehat{=}$ **var** *ret* : *double* •
$(ret := cabinPressure)$

public *getEmergencyOxygen* $\widehat{=}$ **var** *ret* : *double* •
$(ret := emergencyOxygen)$

public *getFuelRemaining* $\widehat{=}$ **var** *ret* : *double* •
$(ret := fuelRemaining)$

public *getHeading* $\widehat{=}$ **var** *ret* : *double* •
$(ret := heading)$

public *setAirSpeed* $\widehat{=}$
$(this \,.\, this.airSpeed := airSpeed)$

public *setAltitude* $\widehat{=}$
$(this \,.\, this.altitude := altitude)$

public *setCabinPressure* $\widehat{=}$
$(this \,.\, this.cabinPressure := cabinPressure)$

public *setEmergencyOxygen* $\widehat{=}$
$(this \,.\, this.emergencyOxygen := emergencyOxygen)$

public *setFuelRemaining* $\widehat{=}$
$\big(\textit{this . this.fuelRemaining := fuelRemaining}\big)$

public *setHeading* $\widehat{=}$
$\big(\textit{this . this.heading := heading}\big)$

• **Skip**

end

section *MainMissionMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel *getAirSpeedCall* : *MissionID*
channel *getAirSpeedRet* : *MissionID* × *double*

channel *getAltitudeCall* : *MissionID*
channel *getAltitudeRet* : *MissionID* × *double*

channel *getCabinPressureCall* : *MissionID*
channel *getCabinPressureRet* : *MissionID* × *double*

channel *getEmergencyOxygenCall* : *MissionID*
channel *getEmergencyOxygenRet* : *MissionID* × *double*

channel *getFuelRemainingCall* : *MissionID*
channel *getFuelRemainingRet* : *MissionID* × *double*

channel *getHeadingCall* : *MissionID*
channel *getHeadingRet* : *MissionID* × *double*

channel *setAirSpeedCall* : *MissionID* × *double*
channel *setAirSpeedRet* : *MissionID*

channel *setAltitudeCall* : *MissionID* × *double*
channel *setAltitudeRet* : *MissionID*

channel *setCabinPressureCall* : *MissionID* × *double*
channel *setCabinPressureRet* : *MissionID*

channel *setEmergencyOxygenCall* : *MissionID* × *double*
channel *setEmergencyOxygenRet* : *MissionID*

channel *setFuelRemainingCall* : *MissionID* × *double*
channel *setFuelRemainingRet* : *MissionID*

channel *setHeadingCall* : *MissionID* × *double*
channel *setHeadingRet* : *MissionID*

5.2   Schedulables of

section *ACModeChangerApp* **parents** *TopLevelMissionSequencerChan,*
   MissionId, MissionIds, SchedulableId

process *ACModeChangerApp* $\widehat{=}$ *controllingMission* : *MissionID* $\bullet$ **begin**

```
┌─ State ────────────────────────────────────────────────────
│  modesLeft : ℤ
│  ref currentModeClass : ModeClass
│  ref launchModeClass : ModeClass
│  ref cruiseModeClass : ModeClass
│  ref landModeClass : ModeClass
└────────────────────────────────────────────────────────────
```

state *State*

```
┌─ Init ─────────────────────────────────────────────────────
│  ┌────────────
│  │  State′
│  ├────────────
│  modesLeft′ = 3
│  ref currentModeClass′ = new ModeClass()
│  ref launchModeClass′ = new ModeClass()
│  ref cruiseModeClass′ = new ModeClass()
│  ref landModeClass′ = new ModeClass()
└────────────────────────────────────────────────────────────
```

GetNextMission $\widehat{=}$ **var** *ret* : *MissionID* $\bullet$
$$\begin{pmatrix} getNextMissionCall \,.\, ACModeChanger \longrightarrow \\ ret := this \,.\, getNextMission(); \\ getNextMissionRet \,.\, ACModeChanger \,!\, ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

changeToMeth $\widehat{=}$
$$\begin{pmatrix} changeToCall \,.\, ACModeChanger \,?\, newMode \longrightarrow \\ \left(this \,.\, currentMode := newMode \right); \\ changeToRet \,.\, ACModeChanger \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$getNextMissionMeth \; \widehat{=} \; \mathbf{var} \; ret : MissionID \; \bullet$
$$
\begin{pmatrix}
getNextMissionCall . ACModeChanger \longrightarrow \\
\begin{pmatrix}
\begin{pmatrix}
\mathbf{if} \; (modesLeft = 3) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ ret := TakeOffMission \end{pmatrix} \\
[\!]\, \neg \, (modesLeft = 3) \longrightarrow \\
\quad \mathbf{if} \; (modesLeft = 2) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ ret := CruiseMission \end{pmatrix} \\
[\!]\, \neg \, (modesLeft = 2) \longrightarrow \\
\quad \mathbf{if} \; (modesLeft = 1) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ ret := LandMission \end{pmatrix} \\
[\!]\, \neg \, (modesLeft = 1) \longrightarrow \\
\quad \big(ret := nullMissionId \big) \\
\mathbf{fi} \\
\mathbf{fi} \\
\mathbf{fi}
\end{pmatrix} \; ; \\
getNextMissionRet . ACModeChanger \, ! \, ret \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
\end{pmatrix}
$$

$advanceModeSyncMeth \; \widehat{=}$
$$
\begin{pmatrix}
advanceModeCall . ACModeChanger \, ? \, thread \longrightarrow \\
\begin{pmatrix}
startSyncMeth . ACModeChangerObject . thread \longrightarrow \\
lockAcquired . ACModeChangerObject . thread \longrightarrow \\
\begin{pmatrix}
; \\
\mathbf{if} \; (modesLeft = 3) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ changeTo(launchMode) \end{pmatrix} \\
[\!]\, \neg \, (modesLeft = 3) \longrightarrow \\
\quad \mathbf{if} \; (modesLeft = 2) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ changeTo(cruiseMode) \end{pmatrix} \\
[\!]\, \neg \, (modesLeft = 2) \longrightarrow \\
\quad \mathbf{if} \; (modesLeft = 1) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ changeTo(landMode) \end{pmatrix} \\
[\!]\, \neg \, (modesLeft = 1) \longrightarrow \\
\quad \big(changeTo(\mathbf{null}) \big) \\
\mathbf{fi} \\
\mathbf{fi} \\
\mathbf{fi}
\end{pmatrix} \; ; \\
endSyncMeth . ACModeChangerObject . thread \longrightarrow \\
advanceModeRet . ACModeChanger . thread \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
\end{pmatrix}
$$

$Methods \; \widehat{=}$
$$
\begin{pmatrix}
GetNextMission \\
\square \\
changeToMeth \\
\square \\
getNextMissionMeth \\
\square \\
advanceModeSyncMeth
\end{pmatrix} \; ; \; Methods
$$

$\bullet \; (Init \; ; \; Methods) \; \triangle \; (end\_sequencer\_app . ACModeChanger \longrightarrow \mathbf{Skip})$

$\mathbf{end}$

class $ACModeChangerClass \;\widehat{=}\;$ **begin**

$\circ$ **sync** $advanceMode \;\widehat{=}$

$$
\begin{pmatrix}
; \\
\textbf{if } (modesLeft = 3) \longrightarrow \\
\qquad \begin{pmatrix} modesLeft := modesLeft - 1; \\ changeTo(launchMode) \end{pmatrix} \\
[\!] \neg (modesLeft = 3) \longrightarrow \\
\qquad \textbf{if } (modesLeft = 2) \longrightarrow \\
\qquad \begin{pmatrix} modesLeft := modesLeft - 1; \\ changeTo(cruiseMode) \end{pmatrix} \\
[\!] \neg (modesLeft = 2) \longrightarrow \\
\qquad \textbf{if } (modesLeft = 1) \longrightarrow \\
\qquad \begin{pmatrix} modesLeft := modesLeft - 1; \\ changeTo(landMode) \end{pmatrix} \\
[\!] \neg (modesLeft = 1) \longrightarrow \\
\qquad \begin{pmatrix} changeTo(\textbf{null}) \end{pmatrix} \\
\textbf{fi} \\
\textbf{fi} \\
\textbf{fi}
\end{pmatrix}
$$

$\bullet$ **Skip**

end

section *ACModeChangerMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel *changeToCall* : *SchedulableID*×
channel *changeToRet* : *SchedulableID*

channel *getNextMissionCall* : *SchedulableID*
channel *getNextMissionRet* : *SchedulableID* × *MissionID*

channel *advanceModeCall* : *SchedulableID* × *ThreadID*
channel *advanceModeRet* : *SchedulableID* × *ThreadID*

section *ControlHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*

process *ControlHandlerApp* $\widehat{=}$ **begin**

handlerAsyncEvent $\widehat{=}$
$$\begin{pmatrix} handleAsyncEventCall \, . \, ControlHandler \longrightarrow \\ handleAsyncEventRet \, . \, ControlHandler \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

Methods $\widehat{=}$
$$\big(handlerAsyncEvent \big) \, ; \, Methods$$

● (*Methods*) $\triangle$ (*end_aperiodic_app* . *ControlHandler* $\longrightarrow$ **Skip**)

end

class *ControlHandlerClass* $\widehat{=}$ **begin**

- **Skip**

end

section *ControlHandlerMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *CommunicationsHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*

process *CommunicationsHandlerApp* $\widehat{=}$ **begin**

$handlerAsyncEvent \,\widehat{=}$
$$\begin{pmatrix} handleAsyncEventCall \,.\, CommunicationsHandler \longrightarrow \\ \wr handleAsyncEventRet \,.\, CommunicationsHandler \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$Methods \,\widehat{=}$
$$\begin{pmatrix} handlerAsyncEvent \end{pmatrix} ; \; Methods$$

● $(Methods) \,\triangle\, (end\_aperiodic\_app \,.\, CommunicationsHandler \longrightarrow \mathbf{Skip})$

end

class *CommunicationsHandlerClass* $\widehat{=}$ **begin**

- **Skip**

end

section *CommunicationsHandlerMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *EnvironmentMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*   ,
MainMissionMethChan

process *EnvironmentMonitorApp* $\widehat{=}$ *controllingMission* : *MissionID* • **begin**

handlerAsyncEvent $\widehat{=}$

$$
\left(
\begin{array}{l}
handleAsyncEventCall \, . \, EnvironmentMonitor \longrightarrow \\
\left(
\begin{array}{l}
; \\
setCabinPressureCall \, . \, controllingMission0 \longrightarrow \\
setCabinPressureRet \, . \, controllingMission \longrightarrow \\
\textbf{Skip}; \\
setEmergencyOxygenCall \, . \, controllingMission0 \longrightarrow \\
setEmergencyOxygenRet \, . \, controllingMission \longrightarrow \\
\textbf{Skip}; \\
setFuelRemainingCall \, . \, controllingMission0 \longrightarrow \\
setFuelRemainingRet \, . \, controllingMission \longrightarrow \\
\textbf{Skip}
\end{array}
\right) \quad handleAsyncEventRet \, . \, EnvironmentMonitor \longrightarrow \\
\textbf{Skip}
\end{array}
\right)
$$

Methods $\widehat{=}$
$\big(handlerAsyncEvent \big)$; *Methods*

• (*Methods*) $\triangle$ (*end_periodic_app* . *EnvironmentMonitor* $\longrightarrow$ **Skip**)

end

class *EnvironmentMonitorClass* $\widehat{=}$ **begin**

- **Skip**

end

section *FlightSensorsMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*   ,
MainMissionMethChan

process *FlightSensorsMonitorApp* $\;\widehat{=}\;$ *controllingMission* : *MissionID* • **begin**

handlerAsyncEvent $\;\widehat{=}\;$
$$
\begin{pmatrix}
handleAsyncEventCall \,.\, FlightSensorsMonitor \longrightarrow \\
\begin{pmatrix}
; \\
\begin{pmatrix}
setAirSpeedCall \,.\, controllingMission0 \longrightarrow \\
setAirSpeedRet \,.\, controllingMission \longrightarrow \\
\mathbf{Skip}; \\
setAltitudeCall \,.\, controllingMission0 \longrightarrow \\
setAltitudeRet \,.\, controllingMission \longrightarrow \\
\mathbf{Skip}; \\
setHeadingCall \,.\, controllingMission0 \longrightarrow \\
setHeadingRet \,.\, controllingMission \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
\quad handleAsyncEventRet \,.\, FlightSensorsMonitor \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
\end{pmatrix}
$$

Methods $\;\widehat{=}\;$
$\big(\,handlerAsyncEvent\,\big)\,;\;\;Methods$

• $(Methods)\;\triangle\;(end\_periodic\_app \,.\, FlightSensorsMonitor \longrightarrow \mathbf{Skip})$

end

class *FlightSensorsMonitorClass* $\,\widehat{=}\,$ **begin**

- **Skip**

end

section *AperiodicSimulatorApp* **parents** *PeriodicEventHandlerChan, SchedulableId, SchedulableIds*

process *AperiodicSimulatorApp* $\widehat{=}$ *event* : *MissionID* • **begin**

$handlerAsyncEvent \widehat{=}$
$$\begin{pmatrix} handleAsyncEventCall . AperiodicSimulator \longrightarrow \\ \begin{pmatrix} ; \\ releaseCall . event \longrightarrow \\ releaseRet . event ? release \longrightarrow \\ \textbf{Skip} \\ \textbf{Skip} \end{pmatrix} handleAsyncEventRet . AperiodicSimulator \longrightarrow \end{pmatrix}$$

$Methods \widehat{=}$
$$\begin{pmatrix} handlerAsyncEvent \end{pmatrix} ; \; Methods$$

• $(Methods) \triangle (end\_periodic\_app . AperiodicSimulator \longrightarrow \textbf{Skip})$

end

class *AperiodicSimulatorClass* $\widehat{=}$ **begin**

- **Skip**

end

5.3 TakeOffMission

section *TakeOffMissionApp* **parents** *scj_prelude, MissionId, MissionIds,*
  SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, TakeOffMissionClass   , TakeOffMissionMethCl

process *TakeOffMissionApp* $\cong$ *storageParametersSchedulable : MissionID, landingGearHandler : MissionID, takeOffMo*

__ *State* _____
  this : **ref** *TakeOffMissionClass*
_____

state *State*

__ *Init* _____
  State'
  _____
  this' = **new** *TakeOffMissionClass*()
_____

InitializePhase $\cong$
$\begin{pmatrix} initializeCall . TakeOffMission \longrightarrow \\ register ! LandingGearHandlerTakeOff ! TakeOffMission \longrightarrow \\ register ! TakeOffMonitor ! TakeOffMission \longrightarrow \\ register ! TakeOffFailureHandler ! TakeOffMission \longrightarrow \\ initializeRet . TakeOffMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

CleanupPhase $\cong$
$\begin{pmatrix} cleanupMissionCall . TakeOffMission \longrightarrow \\ cleanupMissionRet . TakeOffMission ! \textbf{True} \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

abortMeth $\cong$
$\begin{pmatrix} abortCall . TakeOffMission \longrightarrow \\ this . abort(); \\ abortRet . TakeOffMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

getControllingMissionMeth $\cong$ **var** *ret* : *MissionID* •
$\begin{pmatrix} getControllingMissionCall . TakeOffMission \longrightarrow \\ ret := this . getControllingMission(); \\ getControllingMissionRet . TakeOffMission ! ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

setControllingMissionMeth $\cong$
$\begin{pmatrix} setControllingMissionCall . TakeOffMission ? controllingMission \longrightarrow \\ this . setControllingMission(controllingMission); \\ setControllingMissionRet . TakeOffMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

cleanUpMeth $\cong$ **var** *ret* : $\mathbb{B}$ •
$\begin{pmatrix} cleanUpCall . TakeOffMission \longrightarrow \\ ret := this . cleanUp(); \\ cleanUpRet . TakeOffMission ! ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

$stowLandingGearMeth \;\widehat{=}$
$$\left(\begin{array}{l} stowLandingGearCall\,.\,TakeOffMission\longrightarrow \\ this\,.\,stowLandingGear(); \\ stowLandingGearRet\,.\,TakeOffMission\longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$isLandingGearDeployedMeth \;\widehat{=}\; \textbf{var}\; ret : \mathbb{B} \;\bullet$
$$\left(\begin{array}{l} isLandingGearDeployedCall\,.\,TakeOffMission\longrightarrow \\ ret := this\,.\,isLandingGearDeployed(); \\ isLandingGearDeployedRet\,.\,TakeOffMission\,!\,ret\longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$deployLandingGearSyncMeth \;\widehat{=}$
$$\left(\begin{array}{l} deployLandingGearCall\,.\,TakeOffMission\,?\,thread\longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,TakeOffMissionObject\,.\,thread\longrightarrow \\ lockAcquired\,.\,TakeOffMissionObject\,.\,thread\longrightarrow \\ \left(this\,.\,landingGearDeployed := true\right); \\ endSyncMeth\,.\,TakeOffMissionObject\,.\,thread\longrightarrow \\ deployLandingGearRet\,.\,TakeOffMission\,.\,thread\longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$$Methods \;\widehat{=}\; \left(\begin{array}{l} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ abortMeth \\ \Box \\ getControllingMissionMeth \\ \Box \\ setControllingMissionMeth \\ \Box \\ cleanUpMeth \\ \Box \\ stowLandingGearMeth \\ \Box \\ isLandingGearDeployedMeth \\ \Box \\ deployLandingGearSyncMeth \end{array}\right) \;;\; Methods$$

$\bullet\; (Init\;;\;\; Methods)\;\triangle\;(end\_mission\_app\,.\,TakeOffMission\longrightarrow \textbf{Skip})$

end

class *TakeOffMissionClass* $\widehat{=}$ **begin**

```
┌─ state State ──────────────────────────────────────────────
│  SAFE_AIRSPEED_THRESHOLD : double
│  TAKEOFF_ALTITUDE : double
│  abort : 𝔹
│  landingGearDeployed : 𝔹
└────────────────────────────────────────────────────────────
```

state *State*

```
┌─ initial Init ─────────────────────────────────────────────
│  State′
│ ───────────────────────────────────────────────────────────
│  SAFE_AIRSPEED_THRESHOLD′ = 10.0
│  TAKEOFF_ALTITUDE′ = 10.0
│  abort′ = false
└────────────────────────────────────────────────────────────
```

public *abort* $\widehat{=}$
$\big(this\,.\,abort := true\big)$

public *getControllingMission* $\widehat{=}$ **var** *ret* : *MissionID* •
$\big(ret := controllingMission\big)$

public *setControllingMission* $\widehat{=}$
$\big(this\,.\,this.controllingMission := controllingMission\big)$

public *cleanUp* $\widehat{=}$ **var** *ret* : 𝔹 •
$\begin{pmatrix} ; \\ ret := (\neg\ abort = \textbf{True}) \end{pmatrix}$

public *stowLandingGear* $\widehat{=}$
$\big(this\,.\,landingGearDeployed := false\big)$

public *isLandingGearDeployed* $\widehat{=}$ **var** *ret* : 𝔹 •
$\big(ret := landingGearDeployed = \textbf{True}\big)$

• **Skip**

end

section *TakeOffMissionMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel *abortCall* : *MissionID*
channel *abortRet* : *MissionID*

channel *getControllingMissionCall* : *MissionID*
channel *getControllingMissionRet* : *MissionID* × *MissionID*

channel *setControllingMissionCall* : *MissionID* × *MissionID*
channel *setControllingMissionRet* : *MissionID*

channel *cleanUpCall* : *MissionID*
channel *cleanUpRet* : *MissionID* × $\mathbb{B}$

channel *stowLandingGearCall* : *MissionID*
channel *stowLandingGearRet* : *MissionID*

channel *isLandingGearDeployedCall* : *MissionID*
channel *isLandingGearDeployedRet* : *MissionID* × $\mathbb{B}$

channel *deployLandingGearCall* : *MissionID* × *ThreadID*
channel *deployLandingGearRet* : *MissionID* × *ThreadID*

5.4 Schedulables of

section *LandingGearHandlerTakeOffApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds* ,
TakeOffMissionMethChan, *ObjectIds*, *ThreadIds*

process *LandingGearHandlerTakeOffApp* $\widehat{=}$ *mission* : *MissionID* $\bullet$ **begin**

$handlerAsyncEvent \widehat{=}$
$\begin{pmatrix} handleAsyncEventCall \, . \, LandingGearHandlerTakeOff \longrightarrow \\ \begin{pmatrix} ; \\ isLandingGearDeployedCall \, . \, mission \longrightarrow \\ isLandingGearDeployedRet \, . \, mission \, ? \, isLandingGearDeployed \longrightarrow \\ \\ \textbf{var} \, landingGearIsDeployed : \mathbb{B} \bullet landingGearIsDeployed := isLandingGearDeployed \\ \textbf{if} \, landingGearIsDeployed = \textbf{True} \longrightarrow \\ \quad \begin{pmatrix} stowLandingGearCall \, . \, mission \longrightarrow \\ stowLandingGearRet \, . \, mission \longrightarrow \\ \textbf{Skip} \end{pmatrix} \\ [\![\neg \, landingGearIsDeployed = \textbf{True} \longrightarrow \\ \quad \begin{pmatrix} deployLandingGearCall \, . \, mission \, . \, LandingGearHandlerTakeOffThread \longrightarrow \\ deployLandingGearRet \, . \, mission \, . \, LandingGearHandlerTakeOffThread \longrightarrow \\ \textbf{Skip} \end{pmatrix} \\ \textbf{fi} \end{pmatrix} \\ \textbf{Skip} \end{pmatrix}$ *handleAsyncEventRet . Landi*

$Methods \widehat{=}$
$\bigl(handlerAsyncEvent \bigr) \, ; \; Methods$

$\bullet$ $(Methods) \, \triangle \, (end\_aperiodic\_app \, . \, LandingGearHandlerTakeOff \longrightarrow \textbf{Skip})$

end

class *LandingGearHandlerTakeOffClass* $\widehat{=}$ **begin**

- **Skip**

end

section *LandingGearHandlerTakeOffMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *TakeOffFailureHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*   ,
TakeOffMissionMethChan

process *TakeOffFailureHandlerApp* $\widehat{=}$ *takeoffMission* : *MissionID* • **begin**

```
┌─ State ──────────────────────────────────────────────────────
│  threshold : double
└──────────────────────────────────────────────────────────────
```

state *State*

```
┌─ Init ───────────────────────────────────────────────────────
│  State′
│ ─────────────────
│  threshold′ = threshold
└──────────────────────────────────────────────────────────────
```

$handlerAsyncEvent \widehat{=}$
$\Big(handleAsyncEventCall . TakeOffFailureHandler \longrightarrow$
$\quad \Big(getControllingMissionCall . takeoffMission.getControllingMission() \longrightarrow$
$\quad getControllingMissionRet . takeoffMission.getControllingMission() ? getControllingMission \longrightarrow$

$\quad \textbf{var } currentSpeed : double \bullet currentSpeed := getAirSpeed$
$\quad \textbf{if } (currentSpeed < threshold) \longrightarrow$
$\qquad \Big(;$
$\qquad abortCall . takeoffMission \longrightarrow$
$\qquad abortRet . takeoffMission \longrightarrow$
$\qquad \textbf{Skip};$
$\qquad requestTerminationCall . takeoffMission \longrightarrow$
$\qquad requestTerminationRet . takeoffMission ? requestTermination \longrightarrow$
$\qquad \textbf{Skip} \Big)$
$\quad [\!] \neg (currentSpeed < threshold) \longrightarrow$
$\qquad) ($
$\quad \textbf{fi Skip} \Big)$ $\qquad handleAsyncEventR$
$\textbf{Skip} \Big)$

$Methods \widehat{=}$
$\big(handlerAsyncEvent \big) ; \ Methods$

• $(Init ; \ Methods) \triangle (end\_aperiodic\_app . TakeOffFailureHandler \longrightarrow \textbf{Skip})$

end

class *TakeOffFailureHandlerClass* $\widehat{=}$ **begin**

- **Skip**

end

section *TakeOffFailureHandlerMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *TakeOffMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*   ,
TakeOffMissionMethChan

process *TakeOffMonitorApp* $\hat{=}$ *takeoffMission* : *MissionID*, *landingGearHandler* : *MissionID* • **begin**

```
┌─ State ─────────────────────────────────────────────────
│  takeOffAltitude : double
└─────────────────────────────────────────────────────────
```

state *State*

```
┌─ Init ──────────────────────────────────────────────────
│  State′
│ ────────────────────────────────────────────────────────
│  takeOffAltitude′ = takeOffAltitude
└─────────────────────────────────────────────────────────
```

handlerAsyncEvent $\hat{=}$
$\Big($ *handleAsyncEventCall . TakeOffMonitor* $\longrightarrow$

$\quad\Big($;

$\qquad$ *getControllingMissionCall . takeoffMission.getControllingMission()* $\longrightarrow$

$\qquad$ *getControllingMissionRet . takeoffMission.getControllingMission() ? getControllingMission* $\longrightarrow$

$\qquad$ **var** *altitude* : *double* • *altitude* := *getAltitude*

$\qquad$ **if** (*altitude* > *takeOffAltitude*) $\longrightarrow$

$\qquad\quad\Big($;

$\qquad\qquad$ *releaseCall . landingGearHandler* $\longrightarrow$

$\qquad\qquad$ *releaseRet . landingGearHandler ? release* $\longrightarrow$

$\qquad\qquad$ *requestTerminationCall . takeoffMission* $\longrightarrow$

$\qquad\qquad$ *requestTerminationRet . takeoffMission ? requestTermination* $\longrightarrow$

$\qquad\qquad$ **Skip**

$\qquad$ [] ¬ (*altitude* > *takeOffAltitude*) $\longrightarrow$ **Skip**

$\qquad$ **fi** ;

$\qquad$ **Skip**

$\quad$ **Skip**

handleAsyncEventR ...

Methods $\hat{=}$
$\big($ *handlerAsyncEvent* $\big)$; *Methods*

• (*Init* ; *Methods*) $\triangle$ (*end_periodic_app . TakeOffMonitor* $\longrightarrow$ **Skip**)

end

class *TakeOffMonitorClass* $\widehat{=}$ **begin**

- **Skip**

end

5.5 CruiseMission

section *CruiseMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
 SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *CruiseMissionClass* , *CruiseMissionMethChan*

process *CruiseMissionApp* $\widehat{=}$ *storageParametersSchedulable* : *MissionID*, *beginLandingHandler* : *MissionID*, *navigationM*

State
this : **ref** *CruiseMissionClass*

---

state *State*

Init
State'

---
this' = **new** *CruiseMissionClass*()

InitializePhase $\widehat{=}$
$\begin{pmatrix} initializeCall \, . \, CruiseMission \longrightarrow \\ register \, ! \, BeginLandingHandler \, ! \, CruiseMission \longrightarrow \\ register \, ! \, NavigationMonitor \, ! \, CruiseMission \longrightarrow \\ initializeRet \, . \, CruiseMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

CleanupPhase $\widehat{=}$
$\begin{pmatrix} cleanupMissionCall \, . \, CruiseMission \longrightarrow \\ cleanupMissionRet \, . \, CruiseMission \, ! \, \textbf{True} \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

getControllingMissionMeth $\widehat{=}$ **var** *ret* : *MissionID* •
$\begin{pmatrix} getControllingMissionCall \, . \, CruiseMission \longrightarrow \\ ret := this \, . \, getControllingMission(); \\ getControllingMissionRet \, . \, CruiseMission \, ! \, ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

Methods $\widehat{=}$
$\begin{pmatrix} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ getControllingMissionMeth \end{pmatrix}$; *Methods*

• (*Init* ; *Methods*) $\triangle$ (*end_mission_app* . *CruiseMission* $\longrightarrow$ **Skip**)

end

class *CruiseMissionClass* $\,\widehat{=}\,$ **begin**

public *getControllingMission* $\,\widehat{=}\,$ **var** *ret* : *MissionID* •
$\big(ret := controllingMission\big)$

• **Skip**

end

section *CruiseMissionMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel *getControllingMissionCall* : *MissionID*
channel *getControllingMissionRet* : *MissionID* × *MissionID*

5.6 Schedulables of

section *BeginLandingHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*

process *BeginLandingHandlerApp* $\widehat{=}$ *controllingMission* : *MissionID* $\bullet$ **begin**

$handlerAsyncEvent \;\widehat{=}$
$$\left(\begin{array}{l} handleAsyncEventCall\,.\,BeginLandingHandler \longrightarrow \\ \left(\begin{array}{l} ; \\ requestTerminationCall\,.\,controllingMission \longrightarrow \\ requestTerminationRet\,.\,controllingMission\,?\,requestTermination \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\ \mathbf{Skip} \end{array} \right) handleAsyncEventRet\,.\,BeginLandingHandler\text{-}$$

$Methods \;\widehat{=}$
$$\big(handlerAsyncEvent \big)\;;\;\; Methods$$

$\bullet$ $(Methods) \mathbin{\triangle} (end\_aperiodic\_app\,.\,BeginLandingHandler \longrightarrow \mathbf{Skip})$

end

class *BeginLandingHandlerClass* $\widehat{=}$ **begin**

- **Skip**

end

section *BeginLandingHandlerMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *NavigationMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*   ,
CruiseMissionMethChan

process *NavigationMonitorApp* $\widehat{=}$ *mission* : *MissionID* $\bullet$ **begin**

$handlerAsyncEvent \widehat{=}$
$\left(\begin{array}{l} handleAsyncEventCall \,.\, NavigationMonitor \longrightarrow \\ \left(\begin{array}{l} getControllingMissionCall \,.\, mission.getControllingMission() \longrightarrow \\ getControllingMissionRet \,.\, mission.getControllingMission() \,?\, getControllingMission \longrightarrow \\ \\ \mathbf{var}\ heading : double \bullet heading := getHeading \\ getControllingMissionCall \,.\, mission.getControllingMission() \longrightarrow \\ getControllingMissionRet \,.\, mission.getControllingMission() \,?\, getControllingMission \longrightarrow \\ \\ \mathbf{var}\ airSpeed : double \bullet airSpeed := getAirSpeed \\ getControllingMissionCall \,.\, mission.getControllingMission() \longrightarrow \\ getControllingMissionRet \,.\, mission.getControllingMission() \,?\, getControllingMission \longrightarrow \\ \\ \mathbf{var}\ altitude : double \bullet altitude := getAltitude \\ \mathbf{Skip} \end{array}\right) \\ \mathbf{Skip} \end{array}\right)$ $handleAsyncEventRet \,.\, Nav$

$Methods \widehat{=}$
$\left(handlerAsyncEvent\right)\, ;\ Methods$

$\bullet\ (Methods) \bigtriangleup (end\_periodic\_app \,.\, NavigationMonitor \longrightarrow \mathbf{Skip})$

end

class *NavigationMonitorClass* $\widehat{=}$ **begin**

- **Skip**

end

5.7 LandMission

section *LandMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
   SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *LandMissionClass* , *LandMissionMethChan*

process *LandMissionApp* ≙ *storageParametersSchedulable* : *MissionID*, *groundDistanceMonitor* : *MissionID*, *landingHan*

State

this : **ref** *LandMissionClass*

---

state *State*

Init

State′

this′ = **new** *LandMissionClass*()

InitializePhase ≙
$\begin{pmatrix} initializeCall \,.\, LandMission \longrightarrow \\ register \,!\, GroundDistanceMonitor \,!\, LandMission \longrightarrow \\ register \,!\, LandingGearHandlerLand \,!\, LandMission \longrightarrow \\ register \,!\, InstrumentLandingSystemMonitor \,!\, LandMission \longrightarrow \\ register \,!\, SafeLandingHandler \,!\, LandMission \longrightarrow \\ initializeRet \,.\, LandMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

CleanupPhase ≙
$\begin{pmatrix} cleanupMissionCall \,.\, LandMission \longrightarrow \\ cleanupMissionRet \,.\, LandMission \,!\, \mathbf{True} \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

stowLandingGearMeth ≙
$\begin{pmatrix} stowLandingGearCall \,.\, LandMission \longrightarrow \\ this \,.\, stowLandingGear(); \\ stowLandingGearRet \,.\, LandMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

isLandingGearDeployedMeth ≙ **var** *ret* : $\mathbb{B}$ •
$\begin{pmatrix} isLandingGearDeployedCall \,.\, LandMission \longrightarrow \\ ret := this \,.\, isLandingGearDeployed(); \\ isLandingGearDeployedRet \,.\, LandMission \,!\, ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

getControllingMissionMeth ≙ **var** *ret* : *MissionID* •
$\begin{pmatrix} getControllingMissionCall \,.\, LandMission \longrightarrow \\ ret := this \,.\, getControllingMission(); \\ getControllingMissionRet \,.\, LandMission \,!\, ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

abortMeth ≙
$\begin{pmatrix} abortCall \,.\, LandMission \longrightarrow \\ this \,.\, abort(); \\ abortRet \,.\, LandMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$cleanUpMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : \mathbb{B} \bullet$

$$\begin{pmatrix} cleanUpCall\ .\ LandMission \longrightarrow \\ ret := this\ .\ cleanUp(); \\ cleanUpRet\ .\ LandMission\ !\ ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$deployLandingGearSyncMeth \mathrel{\widehat{=}}$

$$\begin{pmatrix} deployLandingGearCall\ .\ LandMission\ ?\ thread \longrightarrow \\ \begin{pmatrix} startSyncMeth\ .\ LandMissionObject\ .\ thread \longrightarrow \\ lockAcquired\ .\ LandMissionObject\ .\ thread \longrightarrow \\ \big(this\ .\ landingGearDeployed := true \big)\ ; \\ endSyncMeth\ .\ LandMissionObject\ .\ thread \longrightarrow \\ deployLandingGearRet\ .\ LandMission\ .\ thread \longrightarrow \\ \mathbf{Skip} \end{pmatrix} \end{pmatrix}$$

$$Methods \mathrel{\widehat{=}} \begin{pmatrix} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ stowLandingGearMeth \\ \Box \\ isLandingGearDeployedMeth \\ \Box \\ getControllingMissionMeth \\ \Box \\ abortMeth \\ \Box \\ cleanUpMeth \\ \Box \\ deployLandingGearSyncMeth \end{pmatrix} \ ;\ \ Methods$$

$\bullet\ (Init\ ;\ \ Methods)\ \triangle\ (end\_mission\_app\ .\ LandMission \longrightarrow \mathbf{Skip})$

$\mathbf{end}$

class *LandMissionClass* $\widehat{=}$ **begin**

---
state *State*
$SAFE\_LANDING\_ALTITUDE : double$
$abort : \mathbb{B}$
$landingGearDeployed : \mathbb{B}$

---

state *State*

---
initial *Init*

$State'$

$SAFE\_LANDING\_ALTITUDE' = 10.0$
$abort' = false$

---

public *stowLandingGear* $\widehat{=}$
$\left(this . landingGearDeployed := false \right)$

public *isLandingGearDeployed* $\widehat{=}$ **var** $ret : \mathbb{B} \bullet$
$\left(ret := landingGearDeployed = \mathbf{True} \right)$

public *getControllingMission* $\widehat{=}$ **var** $ret : MissionID \bullet$
$\left(ret := controllingMission \right)$

public *abort* $\widehat{=}$
$\left(this . abort := true \right)$

public *cleanUp* $\widehat{=}$ **var** $ret : \mathbb{B} \bullet$
$\begin{pmatrix} ; \\ ret := (\neg\ abort = \mathbf{True}) \end{pmatrix}$

$\bullet$ **Skip**

end

section *LandMissionMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel *stowLandingGearCall* : *MissionID*
channel *stowLandingGearRet* : *MissionID*

channel *isLandingGearDeployedCall* : *MissionID*
channel *isLandingGearDeployedRet* : $MissionID \times \mathbb{B}$

channel *getControllingMissionCall* : *MissionID*
channel *getControllingMissionRet* : $MissionID \times MissionID$

channel *abortCall* : *MissionID*
channel *abortRet* : *MissionID*

channel *cleanUpCall* : *MissionID*
channel *cleanUpRet* : $MissionID \times \mathbb{B}$

channel *deployLandingGearCall* : $MissionID \times ThreadID$
channel *deployLandingGearRet* : $MissionID \times ThreadID$

5.8 Schedulables of

section *LandingGearHandlerLandApp* **parents** *AperiodicEventHandlerChan, SchedulableId, SchedulableIds* ,
LandMissionMethChan, ObjectIds, ThreadIds

process *LandingGearHandlerLandApp* $\widehat{=}$ *mission* : *MissionID* • **begin**

$handlerAsyncEvent \widehat{=}$
$$
\left(
\begin{array}{l}
handleAsyncEventCall . LandingGearHandlerLand \longrightarrow \\
\left(
\begin{array}{l}
; \\
isLandingGearDeployedCall . mission \longrightarrow \\
isLandingGearDeployedRet . mission \, ? \, isLandingGearDeployed \longrightarrow \\
\\
\textbf{var } landingGearIsDeployed : \mathbb{B} \bullet landingGearIsDeployed := isLandingGearDeployed \\
\textbf{if } landingGearIsDeployed = \textbf{True} \longrightarrow \\
\quad \left(
\begin{array}{l}
stowLandingGearCall . mission \longrightarrow \\
stowLandingGearRet . mission \longrightarrow \\
\textbf{Skip}
\end{array}
\right) \\
[\!] \, \neg \, landingGearIsDeployed = \textbf{True} \longrightarrow \\
\quad \left(
\begin{array}{l}
deployLandingGearCall . mission . LandingGearHandlerLandThread \longrightarrow \\
deployLandingGearRet . mission . LandingGearHandlerLandThread \longrightarrow \\
\textbf{Skip}
\end{array}
\right) \\
\textbf{fi}
\end{array}
\right) \\
\textbf{Skip}
\end{array}
\right)
$$
$handleAsyncEventRet . Landi$

$Methods \widehat{=}$
$\big(handlerAsyncEvent \big) \, ; \; Methods$

• $(Methods) \, \triangle \, (end\_aperiodic\_app . LandingGearHandlerLand \longrightarrow \textbf{Skip})$

end

class *LandingGearHandlerLandClass* $\widehat{=}$ **begin**

- **Skip**

end

section *LandingGearHandlerLandMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *SafeLandingHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*   ,
LandMissionMethChan

process *SafeLandingHandlerApp* $\widehat{=}$ *landMission* : *MissionID* • **begin**

State ──────────────────────────────────
  threshold : *double*
───

state *State*

Init ──────────────────────────────────
  State′
  ─────────────
  threshold′ = *threshold*
───

$handlerAsyncEvent \widehat{=}$
$\Big(handleAsyncEventCall . SafeLandingHandler \longrightarrow$
  $\Big(getControllingMissionCall . landMission.getControllingMission() \longrightarrow$
  $getControllingMissionRet . landMission.getControllingMission() ? getControllingMission \longrightarrow$

  var *altitude* : *double* • *altitude* := *getAltitude*
  if (*altitude* < *threshold*) $\longrightarrow$
) (
  $\llbracket \neg$ (*altitude* < *threshold*) $\longrightarrow$
) (
  fi $\Big)$ *handleAsyncEventRet*
Skip $\Big)$

$Methods \widehat{=}$
$\big(handlerAsyncEvent\big) ; \ Methods$

• (*Init* ; *Methods*) $\triangle$ (*end_aperiodic_app . SafeLandingHandler* $\longrightarrow$ **Skip**)

end

class *SafeLandingHandlerClass* $\widehat{=}$ **begin**

- **Skip**

end

section *SafeLandingHandlerMethChan* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

section *GroundDistanceMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*   ,
LandMissionMethChan

process *GroundDistanceMonitorApp* $\widehat{=}$ *mission* : *MissionID* $\bullet$ **begin**

┌─ *State* ───
│ *readingOnGround* : *double*
└───

state *State*

┌─ *Init* ──
│ *State'*
│├──────
│ *readingOnGround'* =
└───

$handlerAsyncEvent \;\widehat{=}$

$\left(\begin{array}{l} handleAsyncEventCall \,.\, GroundDistanceMonitor \longrightarrow \\[4pt] \left(\begin{array}{l} ; \\ getControllingMissionCall \,.\, mission.getControllingMission() \longrightarrow \\ getControllingMissionRet \,.\, mission.getControllingMission() \,?\, getControllingMission \longrightarrow \\[6pt] \textbf{var}\; distance : double \bullet distance := getAltitude \\ \textbf{if}\; (distance = readingOnGround) \longrightarrow \\ \qquad \left(\begin{array}{l} ; \\ requestTerminationCall \,.\, mission \longrightarrow \\ requestTerminationRet \,.\, mission \,?\, requestTermination \longrightarrow \\ \textbf{Skip} \end{array}\right) \\ []\, \neg\, (distance = readingOnGround) \longrightarrow \textbf{Skip} \\ \textbf{fi}\; ; \\ \textbf{Skip} \end{array}\right) \\ \textbf{Skip} \end{array}\right)$ $handleAsyncEventRet \,.\, Gr$

$Methods \;\widehat{=}$
$\big(handlerAsyncEvent\big)\; ;\; Methods$

$\bullet\; (Init \;;\; Methods) \,\triangle\, (end\_periodic\_app \,.\, GroundDistanceMonitor \longrightarrow \textbf{Skip})$

end

class *GroundDistanceMonitorClass* $\widehat{=}$ **begin**

- **Skip**

end

section *InstrumentLandingSystemMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*

process *InstrumentLandingSystemMonitorApp* $\widehat{=}$ *mission* : *MissionID* • **begin**

handlerAsyncEvent $\widehat{=}$
$\begin{pmatrix} handleAsyncEventCall \, . \, InstrumentLandingSystemMonitor \longrightarrow \\ (handleAsyncEventRet \, . \, InstrumentLandingSystemMonitor \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

Methods $\widehat{=}$
$\begin{pmatrix} handlerAsyncEvent \end{pmatrix}$; *Methods*

• (*Methods*) $\triangle$ (*end_periodic_app* . *InstrumentLandingSystemMonitor* $\longrightarrow$ **Skip**)

end

class *InstrumentLandingSystemMonitorClass* $\,\widehat{=}\,$ **begin**

- **Skip**

end