aircraft

Tight Rope v0.6

19th November 2015

1 ID Files

1.1 MissionIds

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

$MainMissionID : MissionID$
$TakeOffMissionID : MissionID$
$CruiseMissionID : MissionID$
$LandMissionID : MissionID$

$distinct\langle nullMissionId, MainMissionID, TakeOffMissionID,$
$CruiseMissionID, LandMissionID\rangle$

1.2 SchedulablesIds

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

MainMissionSequencerID : *SchedulableID*
ACModeChangerID : *SchedulableID*
EnvironmentMonitorID : *SchedulableID*
ControlHandlerID : *SchedulableID*
FlightSensorsMonitorID : *SchedulableID*
CommunicationsHandlerID : *SchedulableID*
AperiodicSimulatorID : *SchedulableID*
LandingGearHandlerTakeOffID : *SchedulableID*
TakeOffMonitorID : *SchedulableID*
TakeOffFailureHandlerID : *SchedulableID*
BeginLandingHandlerID : *SchedulableID*
NavigationMonitorID : *SchedulableID*
GroundDistanceMonitorID : *SchedulableID*
LandingGearHandlerLandID : *SchedulableID*
InstrumentLandingSystemMonitorID : *SchedulableID*
SafeLandingHandlerID : *SchedulableID*

---

distinct⟨nullSequencerId, nullSchedulableId, MainMissionSequencerID,
ACModeChangerID, EnvironmentMonitorID,
ControlHandlerID, FlightSensorsMonitorID,
CommunicationsHandlerID, AperiodicSimulatorID,
LandingGearHandlerTakeOffID, TakeOffMonitorID,
TakeOffFailureHandlerID, BeginLandingHandlerID,
NavigationMonitorID, GroundDistanceMonitorID,
LandingGearHandlerLandID, InstrumentLandingSystemMonitorID,
SafeLandingHandlerID⟩

1.3 ThreadIds

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

ACModeChangerThreadID : *ThreadID*
EnvironmentMonitorThreadID : *ThreadID*
ControlHandlerThreadID : *ThreadID*
FlightSensorsMonitorThreadID : *ThreadID*
CommunicationsHandlerThreadID : *ThreadID*
AperiodicSimulatorThreadID : *ThreadID*
LandingGearHandlerTakeOffThreadID : *ThreadID*
TakeOffMonitorThreadID : *ThreadID*
TakeOffFailureHandlerThreadID : *ThreadID*
BeginLandingHandlerThreadID : *ThreadID*
NavigationMonitorThreadID : *ThreadID*
GroundDistanceMonitorThreadID : *ThreadID*
LandingGearHandlerLandThreadID : *ThreadID*
InstrumentLandingSystemMonitorThreadID : *ThreadID*
SafeLandingHandlerThreadID : *ThreadID*

---

distinct⟨SafeletThreadId, nullThreadId,
ACModeChangerThreadID, EnvironmentMonitorThreadID,
ControlHandlerThreadID, FlightSensorsMonitorThreadID,
CommunicationsHandlerThreadID, AperiodicSimulatorThreadID,
LandingGearHandlerTakeOffThreadID, TakeOffMonitorThreadID,
TakeOffFailureHandlerThreadID, BeginLandingHandlerThreadID,
NavigationMonitorThreadID, GroundDistanceMonitorThreadID,
LandingGearHandlerLandThreadID, InstrumentLandingSystemMonitorThreadID,
SafeLandingHandlerThreadID⟩

1.4 ObjectIds

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

ACSafeletObjectID : *ObjectID*
MainMissionObjectID : *ObjectID*
ACModeChangerObjectID : *ObjectID*
EnvironmentMonitorObjectID : *ObjectID*
ControlHandlerObjectID : *ObjectID*
FlightSensorsMonitorObjectID : *ObjectID*
CommunicationsHandlerObjectID : *ObjectID*
AperiodicSimulatorObjectID : *ObjectID*
TakeOffMissionObjectID : *ObjectID*
LandingGearHandlerTakeOffObjectID : *ObjectID*
TakeOffMonitorObjectID : *ObjectID*
TakeOffFailureHandlerObjectID : *ObjectID*
CruiseMissionObjectID : *ObjectID*
BeginLandingHandlerObjectID : *ObjectID*
NavigationMonitorObjectID : *ObjectID*
LandMissionObjectID : *ObjectID*
GroundDistanceMonitorObjectID : *ObjectID*
LandingGearHandlerLandObjectID : *ObjectID*
InstrumentLandingSystemMonitorObjectID : *ObjectID*
SafeLandingHandlerObjectID : *ObjectID*

---

distinct⟨ACSafeletObjectID, MainMissionObjectID,
ACModeChangerObjectID, EnvironmentMonitorObjectID,
ControlHandlerObjectID, FlightSensorsMonitorObjectID,
CommunicationsHandlerObjectID, AperiodicSimulatorObjectID,
TakeOffMissionObjectID, LandingGearHandlerTakeOffObjectID,
TakeOffMonitorObjectID, TakeOffFailureHandlerObjectID,
CruiseMissionObjectID, BeginLandingHandlerObjectID,
NavigationMonitorObjectID, LandMissionObjectID,
GroundDistanceMonitorObjectID, LandingGearHandlerLandObjectID,
InstrumentLandingSystemMonitorObjectID, SafeLandingHandlerObjectID⟩

2 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* ==
$\bigcup\{$*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* . . $|\!\}$

section *Program* **parents** $scj\_prelude$, $MissionId$, $MissionIds$,
    $SchedulableId$, $SchedulableIds$, $MissionChan$, $SchedulableMethChan$, $MissionFW$,
    $SafeletFW$, $TopLevelMissionSequencerFW$, $NetworkChannels$, $ManagedThreadFW$,
    $SchedulableMissionSequencerFW$, $PeriodicEventHandlerFW$, $OneShotEventHandlerFW$,
    $AperiodicEventHandlerFW$, $ACSafeletApp$, $MainMissionSequencerApp$,
    $ObjectFW$, $ThreadFW$, $MainMissionApp$, $ACModeChangerApp$, $ControlHandlerApp$, $CommunicationsHandlerApp$

process $ControlTier \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} SafeletFW \\ \quad [\![ControlTierSync]\!] \\ TopLevelMissionSequencerFW(MainMissionSequencer) \end{array} \right)$$

process $Tier0 \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} MissionFW(MainMission) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} SchedulableMissionSequencerFW(ACModeChanger) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} AperiodicEventHandlerFW(ControlHandler) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW(CommunicationsHandler) \end{array} \right) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} PeriodicEventHandlerFW(EnvironmentMonitor) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(FlightSensorsMonitor) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(AperiodicSimulator) \end{array} \right) \end{array} \right) \end{array} \right)$$

process $Tier1 \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} MissionFW(TakeOffMission) \\ \quad [\![MissionSync]\!] \\ \left(\left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW(LandingGearHandlerTakeOff) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW(TakeOffFailureHandler) \end{array} \right) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(TakeOffMonitor) \end{array} \right) \right) \\ \quad [\![ClusterSync]\!] \\ \left(\begin{array}{l} MissionFW(CruiseMission) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} AperiodicEventHandlerFW(BeginLandingHandler) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(NavigationMonitor) \end{array} \right) \end{array} \right) \\ \quad [\![ClusterSync]\!] \\ \left(\begin{array}{l} MissionFW(LandMission) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW(LandingGearHandlerLand) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW(SafeLandingHandler) \end{array} \right) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} PeriodicEventHandlerFW(GroundDistanceMonitor) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(InstrumentLandingSystemMonitor) \end{array} \right) \end{array} \right) \end{array} \right) \end{array} \right)$$

process $Framework \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} ControlTier \\ \quad [\![TierSync]\!] \\ \left(\begin{array}{l} Tier0 \\ \quad [\![Tier0Sync]\!] \\ Tier1 \end{array} \right) \end{array} \right)$$

process *Application* $\hat{=}$

$\left(\begin{array}{l}\right.$ *ACSafeletApp*

⦀

MainMissionSequencerApp

⦀

MainMissionApp

⦀

ACModeChangerApp(5, *MainMissionID*)

⦀

ControlHandlerApp(5, ,)

⦀

CommunicationsHandlerApp(5, ,)

⦀

EnvironmentMonitorApp(5, (*newRelativeTime*(10, 0), *null*), , *MainMissionID*)

⦀

FlightSensorsMonitorApp(5, (*newRelativeTime*(10, 0), *null*), , *MainMissionID*)

⦀

AperiodicSimulatorApp(5, (*newRelativeTime*(10, 0), *null*), *controlHandler*, 5, (*newRelativeTime*(10, 0), *null*), *commsHan*

⦀

TakeOffMissionApp

⦀

LandingGearHandlerTakeOffApp(5, , , *TakeOffMissionID*)

⦀

TakeOffFailureHandlerApp(5, , , *TakeOffMissionID*,)

⦀

TakeOffMonitorApp(*PriorityScheduler.instance*().*getMaxPriority*(), (*newRelativeTime*(0, 0), *newRelativeTime*(500, 0)),

⦀

CruiseMissionApp

⦀

BeginLandingHandlerApp(5, , , *CruiseMissionID*)

⦀

NavigationMonitorApp(*PriorityScheduler.instance*().*getMaxPriority*(), (*newRelativeTime*(0, 0), *newRelativeTime*(10, 0)

⦀

LandMissionApp

⦀

LandingGearHandlerLandApp(5, , , *LandMissionID*)

⦀

SafeLandingHandlerApp(5, , , *LandMissionID*,)

⦀

GroundDistanceMonitorApp(5, (*newRelativeTime*(0, 0), *newRelativeTime*(10, 0)), *LandMissionID*)

⦀

InstrumentLandingSystemMonitorApp(5, (*newRelativeTime*(0, 0), *newRelativeTime*(10, 0)), , *LandMissionID*)

$Locking \mathrel{\widehat{=}}$

$\left(\left(\begin{array}{l}
ThreadFW(ACModeChangerThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(EnvironmentMonitorThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(ControlHandlerThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(FlightSensorsMonitorThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(CommunicationsHandlerThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(AperiodicSimulatorThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(LandingGearHandlerTakeOffThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(TakeOffMonitorThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(TakeOffFailureHandlerThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(BeginLandingHandlerThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(NavigationMonitorThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(GroundDistanceMonitorThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(LandingGearHandlerLandThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(InstrumentLandingSystemMonitorThreadID, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(SafeLandingHandlerThreadID, MinPriority)
\end{array}\right)\right.$

$|||$

$\left(\begin{array}{l}
ObjectFW(ACSafeletObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(MainMissionObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(ACModeChangerObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(EnvironmentMonitorObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(ControlHandlerObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(FlightSensorsMonitorObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(CommunicationsHandlerObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(AperiodicSimulatorObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(TakeOffMissionObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(LandingGearHandlerTakeOffObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(TakeOffMonitorObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(TakeOffFailureHandlerObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(CruiseMissionObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(BeginLandingHandlerObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(NavigationMonitorObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(LandMissionObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(GroundDistanceMonitorObjectID) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(LandingGearHandlerLandObjectID)
\end{array}\right)$

8

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

3 Safelet

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

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

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

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

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

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

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

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

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

end

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

---
state *State* ─────────────────────────────
  returnedMission : $\mathbb{B}$
---

state *State*

---
initial *Init* ─────────────────────────────
  State'
─────────────────────────────
---

protected *getNextMission* $\,\widehat{=}\,$ **var** *ret* : *MissionID* •
$$
\begin{pmatrix}
\mathbf{if}\,(\neg\ returnedMission = \mathbf{True}) \longrightarrow \\
\quad \begin{pmatrix} this\,.\,returnedMission := true; \\ ret := MainMission \end{pmatrix} \\
[\!]\ \neg\,(\neg\ returnedMission = \mathbf{True}) \longrightarrow \\
\quad \begin{pmatrix} ret := nullMissionId \end{pmatrix} \\
\mathbf{fi}
\end{pmatrix}
$$

• **Skip**

end

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

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

5 Missions

5.1 MainMission

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

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

```
┌─ State ─────────────────────────────────────────────────────────────────
│  this : ref MainMissionClass
└──────────────────────────────────────────────────────────────────────────
```

state *State*

```
┌─ Init ──────────────────────────────────────────────────────────────────
│  State′
│ ─────────────────
│  this′ = new MainMissionClass()
└──────────────────────────────────────────────────────────────────────────
```

InitializePhase $\widehat{=}$
$$\left(\begin{array}{l} initializeCall\,.\,MainMission \longrightarrow \\ register\,!\,ACModeChanger\,!\,MainMission \longrightarrow \\ register\,!\,EnvironmentMonitor\,!\,MainMission \longrightarrow \\ register\,!\,ControlHandler\,!\,MainMission \longrightarrow \\ register\,!\,FlightSensorsMonitor\,!\,MainMission \longrightarrow \\ register\,!\,CommunicationsHandler\,!\,MainMission \longrightarrow \\ register\,!\,AperiodicSimulator\,!\,MainMission \longrightarrow \\ initializeRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

CleanupPhase $\widehat{=}$
$$\left(\begin{array}{l} cleanupMissionCall\,.\,MainMission \longrightarrow \\ cleanupMissionRet\,.\,MainMission\,!\,\mathbf{True} \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

getAirSpeedMeth $\widehat{=}$ **var** *ret : double* •
$$\left(\begin{array}{l} getAirSpeedCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getAirSpeed(); \\ getAirSpeedRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

getAltitudeMeth $\widehat{=}$ **var** *ret : double* •
$$\left(\begin{array}{l} getAltitudeCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getAltitude(); \\ getAltitudeRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

getCabinPressureMeth $\widehat{=}$ **var** *ret : double* •
$$\left(\begin{array}{l} getCabinPressureCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getCabinPressure(); \\ getCabinPressureRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$getEmergencyOxygenMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : double\ \bullet$
$$\left(\begin{array}{l} getEmergencyOxygenCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getEmergencyOxygen(); \\ getEmergencyOxygenRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$getFuelRemainingMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : double\ \bullet$
$$\left(\begin{array}{l} getFuelRemainingCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getFuelRemaining(); \\ getFuelRemainingRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$getHeadingMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : double\ \bullet$
$$\left(\begin{array}{l} getHeadingCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getHeading(); \\ getHeadingRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$setAirSpeedMeth \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} setAirSpeedCall\,.\,MainMission\,?\,airSpeed \longrightarrow \\ this\,.\,setAirSpeed(airSpeed); \\ setAirSpeedRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$setAltitudeMeth \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} setAltitudeCall\,.\,MainMission\,?\,altitude \longrightarrow \\ this\,.\,setAltitude(altitude); \\ setAltitudeRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$setCabinPressureMeth \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} setCabinPressureCall\,.\,MainMission\,?\,cabinPressure \longrightarrow \\ this\,.\,setCabinPressure(cabinPressure); \\ setCabinPressureRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$setEmergencyOxygenMeth \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} setEmergencyOxygenCall\,.\,MainMission\,?\,emergencyOxygen \longrightarrow \\ this\,.\,setEmergencyOxygen(emergencyOxygen); \\ setEmergencyOxygenRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$setFuelRemainingMeth \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} setFuelRemainingCall\,.\,MainMission\,?\,fuelRemaining \longrightarrow \\ this\,.\,setFuelRemaining(fuelRemaining); \\ setFuelRemainingRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$setHeadingMeth \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} setHeadingCall\,.\,MainMission\,?\,heading \longrightarrow \\ this\,.\,setHeading(heading); \\ setHeadingRet\,.\,MainMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

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

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

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* $\;\widehat{=}\;$
$\big(\,this\,.\,this.fuelRemaining := fuelRemaining\,\big)$

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

- **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 MainMission

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

process *ACModeChangerApp* $\widehat{=}$
  PriorityParameters :,
  : *ProbablyMission* ● **begin**

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

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

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

$Methods \widehat{=}$
$$\left(\begin{array}{l} GetNextMission \\ \Box \\ changeToMeth \\ \Box \\ advanceModeSyncMeth \end{array} \right) ; \; Methods$$

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

end

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

---
state *State*
modesLeft : $\mathbb{Z}$
ref *currentModeClass* : *ModeClass*
ref *launchModeClass* : *ModeClass*
ref *cruiseModeClass* : *ModeClass*
ref *landModeClass* : *ModeClass*

---

state *State*

---
initial *Init*
State$'$

---
modesLeft$'$ = 3
ref *currentModeClass$'$* = **new** *ModeClass*()
ref *launchModeClass$'$* = **new** *ModeClass*()
ref *cruiseModeClass$'$* = **new** *ModeClass*()
ref *landModeClass$'$* = **new** *ModeClass*()

---

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

$$
\begin{pmatrix}
\textbf{if } (modesLeft = 3) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ ret := TakeOffMission \end{pmatrix} \\
[\!] \neg (modesLeft = 3) \longrightarrow \\
\quad \textbf{if } (modesLeft = 2) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ ret := CruiseMission \end{pmatrix} \\
[\!] \neg (modesLeft = 2) \longrightarrow \\
\quad \textbf{if } (modesLeft = 1) \longrightarrow \\
\quad \begin{pmatrix} modesLeft := modesLeft - 1; \\ ret := LandMission \end{pmatrix} \\
[\!] \neg (modesLeft = 1) \longrightarrow \\
\quad \begin{pmatrix} ret := nullMissionId \end{pmatrix} \\
\textbf{fi} \\
\textbf{fi} \\
\textbf{fi}
\end{pmatrix}
$$

• **Skip**

end

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

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

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

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

process *ControlHandlerApp* $\widehat{=}$
 PriorityParameters :,
 AperiodicParameters :,
 MainMission :• **begin**

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

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

• (*Methods*) $\triangle$ (*end__app . ControlHandler* $\longrightarrow$ **Skip**)

end

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

- **Skip**

end

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

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

process *CommunicationsHandlerApp* $\widehat{=}$
    PriorityParameters :,
    AperiodicParameters :,
    MainMission :● **begin**

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

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

● (*Methods*) $\triangle$ (*end__app . CommunicationsHandler* $\longrightarrow$ **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* $\widehat{=}$
    PriorityParameters :,
    PeriodicParameters :,
    MainMission :,
    : *ProbablyMission* • **begin**

handlerAsyncEvent $\widehat{=}$

$$
\left(
\begin{array}{l}
handleAsyncEventCall \,.\, EnvironmentMonitor \longrightarrow \\
\left(
\begin{array}{l}
\mathbf{Skip;} \\
setCabinPressureCall \,.\, controllingMission\,!\,0 \longrightarrow \\
setCabinPressureRet \,.\, controllingMission \longrightarrow \\
\mathbf{Skip;} \\
setEmergencyOxygenCall \,.\, controllingMission\,!\,0 \longrightarrow \\
setEmergencyOxygenRet \,.\, controllingMission \longrightarrow \\
\mathbf{Skip;} \\
setFuelRemainingCall \,.\, controllingMission\,!\,0 \longrightarrow \\
setFuelRemainingRet \,.\, controllingMission \longrightarrow \\
\mathbf{Skip}
\end{array}
\right) ; \\
handleAsyncEventRet \,.\, EnvironmentMonitor \longrightarrow \\
\mathbf{Skip}
\end{array}
\right)
$$

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* $\widehat{=}$
    PriorityParameters :,
    PeriodicParameters :,
    MainMission :,
    : *ProbablyMission* • **begin**

handlerAsyncEvent $\widehat{=}$
$$
\begin{pmatrix}
handleAsyncEventCall \, . \, FlightSensorsMonitor \longrightarrow \\
\begin{pmatrix}
\mathbf{Skip;} \\
setAirSpeedCall \, . \, controllingMission \,!\, 0 \longrightarrow \\
setAirSpeedRet \, . \, controllingMission \longrightarrow \\
\mathbf{Skip;} \\
setAltitudeCall \, . \, controllingMission \,!\, 0 \longrightarrow \\
setAltitudeRet \, . \, controllingMission \longrightarrow \\
\mathbf{Skip;} \\
setHeadingCall \, . \, controllingMission \,!\, 0 \longrightarrow \\
setHeadingRet \, . \, controllingMission \longrightarrow \\
\mathbf{Skip}
\end{pmatrix} \; ; \\
handleAsyncEventRet \, . \, FlightSensorsMonitor \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
$$

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

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

end

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

- **Skip**

end

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

process *AperiodicSimulatorApp* $\widehat{=}$
$\quad$ *PriorityParameters* :,
$\quad$ *PeriodicParameters* :,
$\quad$ *controlHandler* :,
$\quad$ *PriorityParameters* :,
$\quad$ *PeriodicParameters* :,
$\quad$ *commsHandler* :,
$\quad$ *PriorityParameters* :,
$\quad$ *PeriodicParameters* :,
$\quad$ *beginLandingHandler* :$\bullet$ **begin**

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

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

$\bullet$ (*Methods*) $\triangle$ (*end_periodic_app* . *AperiodicSimulator* $\longrightarrow$ **Skip**)

end

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

- **Skip**

end

5.3 TakeOffMission

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

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

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

state *State*

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

InitializePhase $\widehat{=}$
$\left(\begin{array}{l} \textit{initializeCall . TakeOffMission} \longrightarrow \\ \textit{register ! LandingGearHandlerTakeOff ! TakeOffMission} \longrightarrow \\ \textit{register ! TakeOffMonitor ! TakeOffMission} \longrightarrow \\ \textit{register ! TakeOffFailureHandler ! TakeOffMission} \longrightarrow \\ \textit{initializeRet . TakeOffMission} \longrightarrow \\ \textbf{Skip} \end{array}\right)$

CleanupPhase $\widehat{=}$
$\left(\begin{array}{l} \textit{cleanupMissionCall . TakeOffMission} \longrightarrow \\ \textit{cleanupMissionRet . TakeOffMission ! } \textbf{True} \longrightarrow \\ \textbf{Skip} \end{array}\right)$

abortMeth $\widehat{=}$
$\left(\begin{array}{l} \textit{abortCall . TakeOffMission} \longrightarrow \\ \textit{this . abort}(); \\ \textit{abortRet . TakeOffMission} \longrightarrow \\ \textbf{Skip} \end{array}\right)$

getControllingMissionMeth $\widehat{=}$ **var** *ret* : *MissionID* •
$\left(\begin{array}{l} \textit{getControllingMissionCall . TakeOffMission} \longrightarrow \\ \textit{ret := this . getControllingMission}(); \\ \textit{getControllingMissionRet . TakeOffMission ! ret} \longrightarrow \\ \textbf{Skip} \end{array}\right)$

setControllingMissionMeth $\widehat{=}$
$\left(\begin{array}{l} \textit{setControllingMissionCall . TakeOffMission ? controllingMission} \longrightarrow \\ \textit{this . setControllingMission}(\textit{controllingMission}); \\ \textit{setControllingMissionRet . TakeOffMission} \longrightarrow \\ \textbf{Skip} \end{array}\right)$

cleanUpMeth $\widehat{=}$ **var** *ret* : $\mathbb{B}$ •
$\left(\begin{array}{l} \textit{cleanUpCall . TakeOffMission} \longrightarrow \\ \textit{ret := this . cleanUp}(); \\ \textit{cleanUpRet . TakeOffMission ! ret} \longrightarrow \\ \textbf{Skip} \end{array}\right)$

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

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

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

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

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

$\mathbf{end}$

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

__ **state** *State* _____
$SAFE\_AIRSPEED\_THRESHOLD : double$
$TAKEOFF\_ALTITUDE : double$
$abort : \mathbb{B}$
$landingGearDeployed : \mathbb{B}$
_____

state *State*

__ **initial** *Init* _____
$State'$
_____
$SAFE\_AIRSPEED\_THRESHOLD' = 10.0$
$TAKEOFF\_ALTITUDE' = 10.0$
$abort' = false$
_____

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

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

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

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

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

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

• **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 TakeOffMission

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

process *LandingGearHandlerTakeOffApp* $\widehat{=}$
    PriorityParameters :,
    AperiodicParameters :,
    TakeOffMission :,
    : *ProbablyMission* • **begin**

handlerAsyncEvent $\widehat{=}$

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

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

• (*Methods*) $\triangle$ (*end__app* . *LandingGearHandlerTakeOff* $\longrightarrow$ **Skip**)

end

40

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

- **Skip**

end

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

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

process *TakeOffFailureHandlerApp* $\hat{=}$
    PriorityParameters :,
    AperiodicParameters :,
    TakeOffMission :,
    : *ProbablyMission*,
    :• **begin**

$handlerAsyncEvent \hat{=}$
$$
\left(
\begin{array}{l}
handleAsyncEventCall \, . \, TakeOffFailureHandler \longrightarrow \\
\left(
\begin{array}{l}
getControllingMissionCall \, . \, takeoffMission.getControllingMission() \longrightarrow \\
getControllingMissionRet \, . \, takeoffMission.getControllingMission() \, ? \, getControllingMission \longrightarrow \\
\\
\textbf{var}\ currentSpeed : double \bullet currentSpeed := getAirSpeed \\
\textbf{if}\ (currentSpeed < threshold) \longrightarrow \\
\qquad \left(
\begin{array}{l}
\textbf{Skip}; \\
abortCall \, . \, takeoffMission \longrightarrow \\
abortRet \, . \, takeoffMission \longrightarrow \\
\textbf{Skip}; \\
requestTerminationCall \, . \, takeoffMission \longrightarrow \\
requestTerminationRet \, . \, takeoffMission \, ? \, requestTermination \longrightarrow \\
\textbf{Skip}
\end{array}
\right) \\
[\!] \, \neg \, (currentSpeed < threshold) \longrightarrow \\
\qquad \big(\textbf{Skip}\big) \\
\textbf{fi}\ \textbf{Skip}
\end{array}
\right) \\
handleAsyncEventRet \, . \, TakeOffFailureHandler \longrightarrow \\
\textbf{Skip}
\end{array}
\right) \, ;
$$

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

• $\big(Methods\big) \, \triangle \, \big(end\_\_app \, . \, TakeOffFailureHandler \longrightarrow \textbf{Skip}\big)$

end

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

---
state *State*
  threshold : *double*

---

state *State*

---
initial *Init*
  State′

- **Skip**

end

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

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

process *TakeOffMonitorApp* $\widehat{=}$
 PriorityParameters :,
 PeriodicParameters :,
 : *ProbablyMission*,
 :,
 landingGearHandler :● **begin**

$handlerAsyncEvent \widehat{=}$
$$
\left(
\begin{array}{l}
handleAsyncEventCall \, . \, TakeOffMonitor \longrightarrow \\
\left(
\begin{array}{l}
\textbf{Skip}; \\
getControllingMissionCall \, . \, takeoffMission.getControllingMission() \longrightarrow \\
getControllingMissionRet \, . \, takeoffMission.getControllingMission() \, ? \, getControllingMission \longrightarrow \\
\\
\textbf{var} \; altitude : double \; \bullet \; altitude := getAltitude \\
\textbf{if} \; (altitude > takeOffAltitude) \longrightarrow \\
\quad \left(
\begin{array}{l}
\textbf{Skip}; \\
releaseCall \, . \, landingGearHandler \longrightarrow \\
releaseRet \, . \, landingGearHandler \, ? \, release \longrightarrow \\
requestTerminationCall \, . \, takeoffMission \longrightarrow \\
requestTerminationRet \, . \, takeoffMission \, ? \, requestTermination \longrightarrow \\
\textbf{Skip}
\end{array}
\right) \\
[\!] \, \neg \, (altitude > takeOffAltitude) \longrightarrow \textbf{Skip} \\
\textbf{fi} \,; \\
\textbf{Skip}
\end{array}
\right) \\
handleAsyncEventRet \, . \, TakeOffMonitor \longrightarrow \\
\textbf{Skip}
\end{array}
\right) ;
$$

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

● (*Methods*) $\triangle$ (*end_periodic_app* . *TakeOffMonitor* $\longrightarrow$ **Skip**)

end

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

---**state** *State*---
  takeOffAltitude : *double*

----

state *State*

---**initial** *Init*---
  State′

• **Skip**

end

5.5 CruiseMission

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

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

 State
 _____
 this : **ref** *CruiseMissionClass*
---

state *State*

 Init
 _____
 State'
 _____
 this' = **new** *CruiseMissionClass*()

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

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

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

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

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

end

class *CruiseMissionClass* $\hat{=}$ **begin**

public *getControllingMission* $\hat{=}$ **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 CruiseMission

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

process *BeginLandingHandlerApp* $\widehat{=}$
    PriorityParameters :,
    AperiodicParameters :,
    CruiseMission :,
    : *ProbablyMission* ● **begin**

$handlerAsyncEvent \widehat{=}$
$$\begin{pmatrix} handleAsyncEventCall \,.\, BeginLandingHandler \longrightarrow \\ \begin{pmatrix} \mathbf{Skip}; \\ requestTerminationCall \,.\, controllingMission \longrightarrow \\ requestTerminationRet \,.\, controllingMission \,?\, requestTermination \longrightarrow \\ \mathbf{Skip} \end{pmatrix} ; \\ handleAsyncEventRet \,.\, BeginLandingHandler \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

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

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

end

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

- **Skip**

end

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

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

process *NavigationMonitorApp* $\widehat{=}$
    PriorityParameters :,
    PeriodicParameters :,
    CruiseMission :,
    : *ProbablyMission* ● **begin**

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

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

● (*Methods*) $\triangle$ (*end_periodic_app* . *NavigationMonitor* $\longrightarrow$ **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* $\widehat{=}$ **begin**

State _____
  this : **ref** *LandMissionClass*
_____

state *State*

Init _____
  State$'$
  ————————————
  this$'$ = **new** *LandMissionClass*()
_____

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

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

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

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

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

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

$cleanUpMeth \,\widehat{=} \mathbf{var}\ ret : \mathbb{B} \bullet$
$$\begin{pmatrix} cleanUpCall \,.\, LandMission \longrightarrow \\ ret := this \,.\, cleanUp(); \\ cleanUpRet \,.\, LandMission \,!\, ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

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

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

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

$\mathbf{end}$

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

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

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

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

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

• **Skip**

end

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

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

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

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

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

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

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

5.8 Schedulables of LandMission

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

process *LandingGearHandlerLandApp* $\widehat{=}$
    PriorityParameters :,
    AperiodicParameters :,
    LandMission :,
    : *ProbablyMission* • **begin**

handlerAsyncEvent $\widehat{=}$

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

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

• (*Methods*) $\triangle$ (*end__app* . *LandingGearHandlerLand* $\longrightarrow$ **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* $\widehat{=}$
    PriorityParameters :,
    AperiodicParameters :,
    LandMission :,
    : *ProbablyMission*,
    :● **begin**

$handlerAsyncEvent \widehat{=}$

$$
\begin{pmatrix}
handleAsyncEventCall \,.\, SafeLandingHandler \longrightarrow \\
\begin{pmatrix}
getControllingMissionCall \,.\, landMission.getControllingMission() \longrightarrow \\
getControllingMissionRet \,.\, landMission.getControllingMission() \,?\, getControllingMission \longrightarrow \\
\\
\textbf{var } altitude : double \bullet altitude := getAltitude \\
\textbf{if } (altitude < threshold) \longrightarrow \\
\quad (\textbf{Skip}) \\
[\!] \neg (altitude < threshold) \longrightarrow \\
\quad (\textbf{Skip}) \\
\textbf{fi}
\end{pmatrix} \\
handleAsyncEventRet \,.\, SafeLandingHandler \longrightarrow \\
\textbf{Skip}
\end{pmatrix} \;;
$$

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

● $(Methods) \,\triangle\, (end\_\!\_app \,.\, SafeLandingHandler \longrightarrow \textbf{Skip})$

end

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

    ┌─ **state** *State* ──
    │ *threshold* : *double*
    └──

  state *State*

    ┌─ **initial** *Init* ──
    │ *State′*
    │─────────────────
    └──

  • **Skip**

  end

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

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

process *GroundDistanceMonitorApp* $\widehat{=}$
　　PriorityParameters :,
　　PeriodicParameters :,
　　: *ProbablyMission* ● **begin**

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

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

● (*Methods*) △ (*end_periodic_app* . *GroundDistanceMonitor* ⟶ **Skip**)

end

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

---
state *State*

  readingOnGround : *double*

---

state *State*

---
initial *Init*

  State′

- **Skip**

end

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

process *InstrumentLandingSystemMonitorApp* $\widehat{=}$
    PriorityParameters :,
    PeriodicParameters :,
    LandMission :,
    : *ProbablyMission* ● **begin**

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

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

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

end

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

- **Skip**

end