aircraft

Tight Rope v0.6

3rd December 2015

1 ID Files

1.1 MissionIds

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

> *MainMissionID* : *MissionID*
> *TakeOffMissionID* : *MissionID*
> *CruiseMissionID* : *MissionID*
> *LandMissionID* : *MissionID*
> _____
> *distinct⟨nullMissionId, MainMissionID, TakeOffMissionID,*
> *CruiseMissionID, LandMissionID⟩*

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*

$$distinct\langle SafeletThreadId, nullThreadId,$$
$$\rangle$$

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, *ObjectFW*, *ThreadFW*,
  ACSafeletApp, *MainMissionSequencerApp*, *MainMissionApp*, *ACModeChangerApp*, *ControlHandlerApp*,
  CommunicationsHandlerApp, *EnvironmentMonitorApp*, *FlightSensorsMonitorApp*,
  AperiodicSimulatorApp, *TakeOffMissionApp*, *LandingGearHandlerTakeOffApp*, *TakeOffFailureHandlerApp*,
  TakeOffMonitorApp, *CruiseMissionApp*, *BeginLandingHandlerApp*, *NavigationMonitorApp*
  , *LandMissionApp*, *LandingGearHandlerLandApp*, *SafeLandingHandlerApp*, *GroundDistanceMonitorApp*,
  InstrumentLandingSystemMonitorApp

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

process *Tier*0 $\widehat{=}$
$$\left(\begin{array}{l} MissionFW\,(MainMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} SchedulableMissionSequencerFW\,(ACModeChangerID) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} AperiodicEventHandlerFW\,(ControlHandlerID,(time(10,0),null)) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW\,(CommunicationsHandlerID,(NULL,nullSchedulableId)) \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} PeriodicEventHandlerFW\,(EnvironmentMonitorID,(time(10,0),NULL,NULL,nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW\,(FlightSensorsMonitorID,(time(10,0),NULL,NULL,nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW\,(AperiodicSimulatorID,(time(10,0),NULL,NULL,nullSchedulableId)) \end{array}\right) \end{array}\right) \end{array}\right)$$

process *Tier*1 $\widehat{=}$
$$\left(\begin{array}{l} MissionFW\,(TakeOffMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW\,(LandingGearHandlerTakeOffID,(NULL,nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW\,(TakeOffFailureHandlerID,(NULL,nullSchedulableId)) \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW\,(TakeOffMonitorID,(time(0,0),time(500,0),NULL,nullSchedulableId)) \end{array}\right) \\ \quad [\![ClusterSync]\!] \\ \left(\begin{array}{l} MissionFW\,(CruiseMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} AperiodicEventHandlerFW\,(BeginLandingHandlerID,(NULL,nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW\,(NavigationMonitorID,(time(0,0),time(10,0),NULL,nullSchedulableId)) \end{array}\right) \end{array}\right) \\ \quad [\![ClusterSync]\!] \\ \left(\begin{array}{l} MissionFW\,(LandMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW\,(LandingGearHandlerLandID,(NULL,nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW\,(SafeLandingHandlerID,(NULL,nullSchedulableId)) \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} PeriodicEventHandlerFW\,(GroundDistanceMonitorID,(time(0,0),time(10,0),NULL,nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW\,(InstrumentLandingSystemMonitorID,(time(0,0),time(10,0),NULL,nullSchedulableId) \end{array}\right) \end{array}\right) \end{array}\right) \end{array}\right)$$

process $Framework \mathrel{\widehat{=}}$

$$
\begin{pmatrix}
ControlTier \\
\qquad [\![TierSync]\!] \\
\begin{pmatrix}
Tier0 \\
\qquad [\![Tier0Sync]\!] \\
Tier1
\end{pmatrix}
\end{pmatrix}
$$

process $Application \mathrel{\widehat{=}}$

$$
\begin{pmatrix}
ACSafeletApp \\
\vertvert\vert \\
MainMissionSequencerApp \\
\vertvert\vert \\
MainMissionApp \\
\vertvert\vert \\
ACModeChangerApp(MainMissionID) \\
\vertvert\vert \\
ControlHandlerApp \\
\vertvert\vert \\
CommunicationsHandlerApp \\
\vertvert\vert \\
EnvironmentMonitorApp(MainMissionID) \\
\vertvert\vert \\
FlightSensorsMonitorApp(MainMissionID) \\
\vertvert\vert \\
AperiodicSimulatorApp(controlHandlerID) \\
\vertvert\vert \\
TakeOffMissionApp \\
\vertvert\vert \\
LandingGearHandlerTakeOffApp(TakeOffMissionID) \\
\vertvert\vert \\
TakeOffFailureHandlerApp(TakeOffMissionID, 10.0) \\
\vertvert\vert \\
TakeOffMonitorApp(TakeOffMissionID, 10.0, landingGearHandlerID) \\
\vertvert\vert \\
CruiseMissionApp \\
\vertvert\vert \\
BeginLandingHandlerApp(CruiseMissionID) \\
\vertvert\vert \\
NavigationMonitorApp(CruiseMissionID) \\
\vertvert\vert \\
LandMissionApp \\
\vertvert\vert \\
LandingGearHandlerLandApp(LandMissionID) \\
\vertvert\vert \\
SafeLandingHandlerApp(LandMissionID, 10.0) \\
\vertvert\vert \\
GroundDistanceMonitorApp(LandMissionID) \\
\vertvert\vert \\
InstrumentLandingSystemMonitorApp(LandMissionID)
\end{pmatrix}
$$

$Threads \;\widehat{=}$

$$\begin{pmatrix} ThreadFW\,(SafeLandingHandlerThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(ACModeChangerThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(TakeOffFailureHandlerThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(InstrumentLandingSystemMonitorThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(FlightSensorsMonitorThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(TakeOffMonitorThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(AperiodicSimulatorThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(LandingGearHandlerLandThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(LandingGearHandlerTakeOffThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(GroundDistanceMonitorThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(ControlHandlerThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(CommunicationsHandlerThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(BeginLandingHandlerThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(NavigationMonitorThreadID, 5) \\ \quad [\![\,ThreadSync\,]\!] \\ ThreadFW\,(EnvironmentMonitorThreadID, 5) \end{pmatrix}$$

$Objects \mathrel{\widehat{=}}$

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

$Locking \mathrel{\widehat{=}} Threads \mathbin{|\!|\!|} Objects$

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

3 Safelet

section *ACSafeletApp* **parents** *scj_prelude*, *ScheduableId*, *ScheduableIds*, *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 $\widehat{=}$
$$\begin{pmatrix} GetSequencer \\ \square \\ InitializeApplication \end{pmatrix} ; \; Methods$$

$\bullet$ (*Methods*) $\triangle$ (*end_safelet_app* $\longrightarrow$ **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 $\widehat{=}$
$\begin{pmatrix} GetNextMission \end{pmatrix}$; *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$'$
returnedMission$'$ = *false*

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

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

$\bullet$ **Skip**

end

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 $\widehat{=}$
$$
\begin{pmatrix}
initializeCall \cdot MainMission \longrightarrow \\
register\,!\,ACModeChanger\,!\,MainMission \longrightarrow \\
register\,!\,EnvironmentMonitor\,!\,MainMission \longrightarrow \\
register\,!\,ControlHandler\,!\,MainMission \longrightarrow \\
register\,!\,FlightSensorsMonitor\,!\,MainMission \longrightarrow \\
register\,!\,CommunicationsHandler\,!\,MainMission \longrightarrow \\
register\,!\,AperiodicSimulator\,!\,MainMission \longrightarrow \\
initializeRet \cdot MainMission \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
$$

CleanupPhase $\widehat{=}$
$$
\begin{pmatrix}
cleanupMissionCall \cdot MainMission \longrightarrow \\
cleanupMissionRet \cdot MainMission\,!\,\mathbf{True} \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
$$

getAirSpeedSyncMeth $\widehat{=}$ **var** *ret* : *double* $\bullet$
$$
\begin{pmatrix}
getAirSpeedCall \cdot MainMission\,?\,thread \longrightarrow \\
\begin{pmatrix}
startSyncMeth \cdot MainMissionObject \cdot thread \longrightarrow \\
lockAcquired \cdot MainMissionObject \cdot thread \longrightarrow \\
ret := this \cdot getAirSpeed(); \\
endSyncMeth \cdot MainMissionObject \cdot thread \longrightarrow \\
getAirSpeedRet \cdot MainMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
\end{pmatrix}
$$

getAltitudeSyncMeth $\widehat{=}$ **var** *ret* : *double* $\bullet$
$$
\begin{pmatrix}
getAltitudeCall \cdot MainMission\,?\,thread \longrightarrow \\
\begin{pmatrix}
startSyncMeth \cdot MainMissionObject \cdot thread \longrightarrow \\
lockAcquired \cdot MainMissionObject \cdot thread \longrightarrow \\
ret := this \cdot getAltitude(); \\
endSyncMeth \cdot MainMissionObject \cdot thread \longrightarrow \\
getAltitudeRet \cdot MainMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
\end{pmatrix}
$$

$getCabinPressureSyncMeth \cong \textbf{var}\ ret : double\ \bullet$
$$\left(\begin{array}{l} getCabinPressureCall\,.\,MainMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,getCabinPressure(); \\ endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ getCabinPressureRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$getEmergencyOxygenSyncMeth \cong \textbf{var}\ ret : double\ \bullet$
$$\left(\begin{array}{l} getEmergencyOxygenCall\,.\,MainMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,getEmergencyOxygen(); \\ endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ getEmergencyOxygenRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$getFuelRemainingSyncMeth \cong \textbf{var}\ ret : double\ \bullet$
$$\left(\begin{array}{l} getFuelRemainingCall\,.\,MainMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,getFuelRemaining(); \\ endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ getFuelRemainingRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$getHeadingSyncMeth \cong \textbf{var}\ ret : double\ \bullet$
$$\left(\begin{array}{l} getHeadingCall\,.\,MainMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,getHeading(); \\ endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ getHeadingRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$setAirSpeedSyncMeth \cong$
$$\left(\begin{array}{l} setAirSpeedCall\,.\,MainMission\,?\,thread\,?\,airSpeed \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\ this\,.\,setAirSpeed(airSpeed); \\ endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ setAirSpeedRet\,.\,MainMission\,.\,thread \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$setAltitudeSyncMeth \cong$
$$\left(\begin{array}{l} setAltitudeCall\,.\,MainMission\,?\,thread\,?\,altitude \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\ this\,.\,setAltitude(altitude); \\ endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ setAltitudeRet\,.\,MainMission\,.\,thread \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

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

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

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

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

$$Methods \mathrel{\widehat{=}} \begin{pmatrix} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ getAirSpeedSyncMeth \\ \Box \\ getAltitudeSyncMeth \\ \Box \\ getCabinPressureSyncMeth \\ \Box \\ getEmergencyOxygenSyncMeth \\ \Box \\ getFuelRemainingSyncMeth \\ \Box \\ getHeadingSyncMeth \\ \Box \\ setAirSpeedSyncMeth \\ \Box \\ setAltitudeSyncMeth \\ \Box \\ setCabinPressureSyncMeth \\ \Box \\ setEmergencyOxygenSyncMeth \\ \Box \\ setFuelRemainingSyncMeth \\ \Box \\ setHeadingSyncMeth \end{pmatrix} \; ; \; Methods$$

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

end

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

---
state *State*
> $ALTITUDE\_READING\_ON\_GROUND : double$
> $test : \mathbb{Z}$
> $cabinPressure : double$
> $emergencyOxygen : double$
> $fuelRemaining : double$
> $altitude : double$
> $airSpeed : double$
> $heading : double$
---

state *State*

---
initial *Init*
> $State'$
>
> ---
> $ALTITUDE\_READING\_ON\_GROUND' = 0.0$
> $test' = 0$
---

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

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

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

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

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

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

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

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

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

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

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

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

- **Skip**

end

5.2   Schedulables of MainMission

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

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

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

$advanceModeMeth \widehat{=}$
$$\left(\begin{array}{l} advanceModeCall\,.\,ACModeChanger \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \mathbf{if}\,(modesLeft = 3) \longrightarrow \\ \quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(launchMode) \end{array}\right) \\ [\!]\,\neg\,(modesLeft = 3) \longrightarrow \\ \quad \mathbf{if}\,(modesLeft = 2) \longrightarrow \\ \quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(cruiseMode) \end{array}\right) \\ \quad [\!]\,\neg\,(modesLeft = 2) \longrightarrow \\ \quad\quad \mathbf{if}\,(modesLeft = 1) \longrightarrow \\ \quad\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(landMode) \end{array}\right) \\ \quad\quad [\!]\,\neg\,(modesLeft = 1) \longrightarrow \\ \quad\quad\quad \big(changeTo(\mathbf{null})\big) \\ \quad\quad \mathbf{fi} \\ \quad \mathbf{fi} \\ \mathbf{fi} \end{array}\right); \\ advanceModeRet\,.\,ACModeChanger \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$changeToSyncMeth \widehat{=}$
$$\left(\begin{array}{l} changeToCall\,.\,ACModeChanger\,?\,thread\,?\,newMode \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,ACModeChangerObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,ACModeChangerObject\,.\,thread \longrightarrow \\ \big(this\,.\,currentMode := newMode\big)\,; \\ endSyncMeth\,.\,ACModeChangerObject\,.\,thread \longrightarrow \\ changeToRet\,.\,ACModeChanger\,.\,thread \longrightarrow \\ \mathbf{Skip} \end{array}\right) \end{array}\right)$$

$Methods \widehat{=}$
$$\left(\begin{array}{l} GetNextMission \\ \square \\ advanceModeMeth \\ \square \\ changeToSyncMeth \end{array}\right)\;;\;\; Methods$$

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

end

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

```
┌─ state State ─────────────────────────────────────────────────
│  modesLeft : 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 sync *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 *advanceModeCall* : *SchedulableID*
channel *advanceModeRet* : *SchedulableID*

channel *changeToCall* : *SchedulableID* × *ThreadID* × *Mode*
channel *changeToRet* : *SchedulableID* × *ThreadID*

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

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

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

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

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

end

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

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

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

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

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

end

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

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

handlerAsyncEvent $\widehat{=}$
$$\begin{pmatrix} handleAsyncEventCall \,.\, EnvironmentMonitor \longrightarrow \\ \begin{pmatrix} \textbf{Skip}; \\ setCabinPressureCall \,.\, controllingMission\,!\,0 \longrightarrow \\ setCabinPressureRet \,.\, controllingMission \longrightarrow \\ \textbf{Skip}; \\ setEmergencyOxygenCall \,.\, controllingMission\,!\,0 \longrightarrow \\ setEmergencyOxygenRet \,.\, controllingMission \longrightarrow \\ \textbf{Skip}; \\ setFuelRemainingCall \,.\, controllingMission\,!\,0 \longrightarrow \\ setFuelRemainingRet \,.\, controllingMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}; \\ handleAsyncEventRet \,.\, EnvironmentMonitor \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

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

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

end

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

process *FlightSensorsMonitorApp* $\,\widehat{=}\,$
  $mainMission : MissionID \bullet$ **begin**

$handlerAsyncEvent \,\widehat{=}\,$
$$
\left(
\begin{array}{l}
handleAsyncEventCall \,.\, FlightSensorsMonitor \longrightarrow \\
\left(
\begin{array}{l}
\textbf{Skip}; \\
setAirSpeedCall \,.\, controllingMission \,!\, 0 \longrightarrow \\
setAirSpeedRet \,.\, controllingMission \longrightarrow \\
\textbf{Skip}; \\
setAltitudeCall \,.\, controllingMission \,!\, 0 \longrightarrow \\
setAltitudeRet \,.\, controllingMission \longrightarrow \\
\textbf{Skip}; \\
setHeadingCall \,.\, controllingMission \,!\, 0 \longrightarrow \\
setHeadingRet \,.\, controllingMission \longrightarrow \\
\textbf{Skip}
\end{array}
\right) ; \\
handleAsyncEventRet \,.\, FlightSensorsMonitor \longrightarrow \\
\textbf{Skip}
\end{array}
\right)
$$

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

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

end

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

process *AperiodicSimulatorApp* $\widehat{=}$
    aperiodicEvent : *SchedulableID* • **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 $\widehat{=}$
$\big(handlerAsyncEvent \big)$; *Methods*

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

end

5.3   TakeOffMission

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

process *TakeOffMissionApp* $\widehat{=}$
   controllingMission : *MissionID* ● **begin**

─── *State* ───
  this : **ref** *TakeOffMissionClass*

state *State*

─── *Init* ──
  State′
  ──────────
  this′ = **new** *TakeOffMissionClass*()

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

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

deployLandingGearMeth $\widehat{=}$
$$\begin{pmatrix} deployLandingGearCall \,.\, TakeOffMission \longrightarrow \\ \left(this \,.\, landingGearDeployed := true \right); \\ deployLandingGearRet \,.\, TakeOffMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

abortSyncMeth $\widehat{=}$
$$\begin{pmatrix} abortCall \,.\, TakeOffMission \,?\, thread \longrightarrow \\ \begin{pmatrix} startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\ lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\ this \,.\, abort(); \\ endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\ abortRet \,.\, TakeOffMission \,.\, thread \longrightarrow \\ \textbf{Skip} \end{pmatrix} \end{pmatrix}$$

getControllingMissionSyncMeth $\widehat{=}$ **var** *ret* : *MissionID* ●
$$\begin{pmatrix} getControllingMissionCall \,.\, TakeOffMission \,?\, thread \longrightarrow \\ \begin{pmatrix} startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\ lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\ ret := this \,.\, getControllingMission(); \\ endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\ getControllingMissionRet \,.\, TakeOffMission \,!\, thread \,!\, ret \longrightarrow \\ \textbf{Skip} \end{pmatrix} \end{pmatrix}$$

$setControllingMissionSyncMeth \,\widehat{=}$
$$
\left(
\begin{array}{l}
setControllingMissionCall \,.\, TakeOffMission\,?\,thread\,?\,controllingMission \longrightarrow \\
\left(
\begin{array}{l}
startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
this \,.\, setControllingMission(controllingMission); \\
endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
setControllingMissionRet \,.\, TakeOffMission \,.\, thread \longrightarrow \\
\mathbf{Skip}
\end{array}
\right)
\end{array}
\right)
$$

$cleanUpSyncMeth \,\widehat{=}\, \mathbf{var}\, ret : \mathbb{B} \,\bullet$
$$
\left(
\begin{array}{l}
cleanUpCall \,.\, TakeOffMission\,?\,thread \longrightarrow \\
\left(
\begin{array}{l}
startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
ret := this \,.\, cleanUp(); \\
endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
cleanUpRet \,.\, TakeOffMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}
\right)
\end{array}
\right)
$$

$stowLandingGearSyncMeth \,\widehat{=}$
$$
\left(
\begin{array}{l}
stowLandingGearCall \,.\, TakeOffMission\,?\,thread \longrightarrow \\
\left(
\begin{array}{l}
startSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
lockAcquired \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
this \,.\, stowLandingGear(); \\
endSyncMeth \,.\, TakeOffMissionObject \,.\, thread \longrightarrow \\
stowLandingGearRet \,.\, TakeOffMission \,.\, thread \longrightarrow \\
\mathbf{Skip}
\end{array}
\right)
\end{array}
\right)
$$

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

$$
Methods \,\widehat{=}\,
\left(
\begin{array}{l}
InitializePhase \\
\square \\
CleanupPhase \\
\square \\
deployLandingGearMeth \\
\square \\
abortSyncMeth \\
\square \\
getControllingMissionSyncMeth \\
\square \\
setControllingMissionSyncMeth \\
\square \\
cleanUpSyncMeth \\
\square \\
stowLandingGearSyncMeth \\
\square \\
isLandingGearDeployedSyncMeth
\end{array}
\right)
\,;\ Methods
$$

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

end

class *TakeOffMissionClass* $\hat{=}$ **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 sync *abort* $\hat{=}$
$\big(this \,.\, abort := true \big)$

public sync *getControllingMission* $\hat{=}$ **var** *ret* : *MissionID* $\bullet$
$\big(ret := controllingMission \big)$

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

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

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

public sync *isLandingGearDeployed* $\hat{=}$ **var** *ret* : $\mathbb{B}$ $\bullet$
$\big(ret := landingGearDeployed = \textbf{True} \big)$

$\bullet$ **Skip**

end

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

channel *deployLandingGearCall* : *SchedulableID*
channel *deployLandingGearRet* : *SchedulableID*

channel *abortCall* : *SchedulableID* × *ThreadID*
channel *abortRet* : *SchedulableID* × *ThreadID*

channel *getControllingMissionCall* : *SchedulableID* × *ThreadID*
channel *getControllingMissionRet* : *SchedulableID* × *ThreadID* × *MissionID*

channel *setControllingMissionCall* : *SchedulableID* × *ThreadID* × *MissionID*
channel *setControllingMissionRet* : *SchedulableID* × *ThreadID*

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

channel *stowLandingGearCall* : *SchedulableID* × *ThreadID*
channel *stowLandingGearRet* : *SchedulableID* × *ThreadID*

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

5.4   Schedulables of TakeOffMission

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

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

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

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

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

end

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

process *TakeOffFailureHandlerApp* $\widehat{=}$
    takeoffMission : *MissionID*,
threshold : *Double* ● **begin**

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

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

● $\left(Methods \right) \, \triangle \, \left(end\_app \, . \, TakeOffFailureHandler \longrightarrow \mathbf{Skip} \right)$

end

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

```
─ state State ──────────────────────────────────────────────
   threshold : double
```

state *State*

```
─ initial Init ─────────────────────────────────────────────
   State′
────────────────
```

• **Skip**

end

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

process *TakeOffMonitorApp* $\widehat{=}$
  takeoffMission : *MissionID*,
takeOffAltitude : *double*,
landingGearHandler : *SchedulableID* • **begin**

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

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

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

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

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

state *State*

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

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

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

getControllingMissionSyncMeth $\widehat{=}$ **var** *ret* : *MissionID* •
$$\begin{pmatrix} getControllingMissionCall \,.\, CruiseMission\,?\,thread \longrightarrow \\ \begin{pmatrix} startSyncMeth \,.\, CruiseMissionObject \,.\, thread \longrightarrow \\ lockAcquired \,.\, CruiseMissionObject \,.\, thread \longrightarrow \\ ret := this \,.\, getControllingMission(); \\ endSyncMeth \,.\, CruiseMissionObject \,.\, thread \longrightarrow \\ getControllingMissionRet \,.\, CruiseMission\,!\,thread\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix} \end{pmatrix}$$

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

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

end

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

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

$\bullet$ **Skip**

end

5.6   Schedulables of CruiseMission

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

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

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

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

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

end

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

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

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

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

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

end

5.7 LandMission

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

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

─── *State* ─────────────────────────────────────
  this : **ref** *LandMissionClass*
───

state *State*

─── *Init* ──────────────────────────────────────
  State′
  ───────────
  this′ = **new** *LandMissionClass*()
───

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

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

deployLandingGearMeth $\widehat{=}$
$$\begin{pmatrix} deployLandingGearCall\,.\,LandMission \longrightarrow \\ \left(this\,.\,landingGearDeployed := true \right); \\ deployLandingGearRet\,.\,LandMission \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

stowLandingGearSyncMeth $\widehat{=}$
$$\begin{pmatrix} stowLandingGearCall\,.\,LandMission\,?\,thread \longrightarrow \\ \begin{pmatrix} startSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,LandMissionObject\,.\,thread \longrightarrow \\ this\,.\,stowLandingGear(); \\ endSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ stowLandingGearRet\,.\,LandMission\,.\,thread \longrightarrow \\ \mathbf{Skip} \end{pmatrix} \end{pmatrix}$$

isLandingGearDeployedSyncMeth $\widehat{=}$ **var** *ret* : $\mathbb{B}$ •
$$\begin{pmatrix} isLandingGearDeployedCall\,.\,LandMission\,?\,thread \longrightarrow \\ \begin{pmatrix} startSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,LandMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,isLandingGearDeployed(); \\ endSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ isLandingGearDeployedRet\,.\,LandMission\,!\,thread\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix} \end{pmatrix}$$

$getControllingMissionSyncMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : MissionID\ \bullet$

$$
\left(
\begin{array}{l}
getControllingMissionCall\,.\,LandMission\,?\,thread \longrightarrow \\
\left(
\begin{array}{l}
startSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,LandMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,getControllingMission(); \\
endSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\
getControllingMissionRet\,.\,LandMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}
\right)
\end{array}
\right)
$$

$abortSyncMeth \mathrel{\widehat{=}}$

$$
\left(
\begin{array}{l}
abortCall\,.\,LandMission\,?\,thread \longrightarrow \\
\left(
\begin{array}{l}
startSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,LandMissionObject\,.\,thread \longrightarrow \\
this\,.\,abort(); \\
endSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\
abortRet\,.\,LandMission\,.\,thread \longrightarrow \\
\mathbf{Skip}
\end{array}
\right)
\end{array}
\right)
$$

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

$$
\left(
\begin{array}{l}
cleanUpCall\,.\,LandMission\,?\,thread \longrightarrow \\
\left(
\begin{array}{l}
startSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\
lockAcquired\,.\,LandMissionObject\,.\,thread \longrightarrow \\
ret := this\,.\,cleanUp(); \\
endSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\
cleanUpRet\,.\,LandMission\,!\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}
\right)
\end{array}
\right)
$$

$$
Methods \mathrel{\widehat{=}}
\left(
\begin{array}{l}
InitializePhase \\
\square \\
CleanupPhase \\
\square \\
deployLandingGearMeth \\
\square \\
stowLandingGearSyncMeth \\
\square \\
isLandingGearDeployedSyncMeth \\
\square \\
getControllingMissionSyncMeth \\
\square \\
abortSyncMeth \\
\square \\
cleanUpSyncMeth
\end{array}
\right)
\ ;\ Methods
$$

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

$\mathbf{end}$

class *LandMissionClass* $\hat{=}$ **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 sync *stowLandingGear* $\hat{=}$
$\left(this . landingGearDeployed := false \right)$

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

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

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

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

• **Skip**

end

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

channel *deployLandingGearCall* : *SchedulableID*
channel *deployLandingGearRet* : *SchedulableID*

channel *stowLandingGearCall* : *SchedulableID* × *ThreadID*
channel *stowLandingGearRet* : *SchedulableID* × *ThreadID*

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

channel *getControllingMissionCall* : *SchedulableID* × *ThreadID*
channel *getControllingMissionRet* : *SchedulableID* × *ThreadID* × *MissionID*

channel *abortCall* : *SchedulableID* × *ThreadID*
channel *abortRet* : *SchedulableID* × *ThreadID*

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

5.8 Schedulables of LandMission

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

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

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

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

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

end

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

process *SafeLandingHandlerApp* $\hat{=}$
    landMission : *MissionID,*
threshold : *Double* ● **begin**

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

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

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

end

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

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

state *State*

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

- **Skip**

end

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

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

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

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

$\bullet$ (*Methods*) $\triangle$ (*end_periodic_app* . *GroundDistanceMonitor* $\longrightarrow$ **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* $\widehat{=}$
    mission : *MissionID* • **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