aircraft

Tight Rope v0.6

30th 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*

$$
\begin{array}{l}
MainMissionSequencerID : SchedulableID \\
ACModeChangerID : SchedulableID \\
EnvironmentMonitorID : SchedulableID \\
ControlHandlerID : SchedulableID \\
FlightSensorsMonitorID : SchedulableID \\
CommunicationsHandlerID : SchedulableID \\
AperiodicSimulatorID : SchedulableID \\
LandingGearHandlerTakeOffID : SchedulableID \\
TakeOffMonitorID : SchedulableID \\
TakeOffFailureHandlerID : SchedulableID \\
BeginLandingHandlerID : SchedulableID \\
NavigationMonitorID : SchedulableID \\
GroundDistanceMonitorID : SchedulableID \\
LandingGearHandlerLandID : SchedulableID \\
InstrumentLandingSystemMonitorID : SchedulableID \\
SafeLandingHandlerID : SchedulableID
\end{array}
$$

$$
\begin{array}{l}
distinct\langle nullSequencerId, nullSchedulableId, MainMissionSequencerID, \\
ACModeChangerID, EnvironmentMonitorID, \\
ControlHandlerID, FlightSensorsMonitorID, \\
CommunicationsHandlerID, AperiodicSimulatorID, \\
LandingGearHandlerTakeOffID, TakeOffMonitorID, \\
TakeOffFailureHandlerID, BeginLandingHandlerID, \\
NavigationMonitorID, GroundDistanceMonitorID, \\
LandingGearHandlerLandID, InstrumentLandingSystemMonitorID, \\
SafeLandingHandlerID\rangle
\end{array}
$$

1.3 ThreadIds

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

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

1.4 ObjectIds

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

$$
\begin{array}{l}
ACSafeletObjectID : ObjectID \\
MainMissionObjectID : ObjectID \\
ACModeChangerObjectID : ObjectID \\
EnvironmentMonitorObjectID : ObjectID \\
ControlHandlerObjectID : ObjectID \\
FlightSensorsMonitorObjectID : ObjectID \\
CommunicationsHandlerObjectID : ObjectID \\
AperiodicSimulatorObjectID : ObjectID \\
TakeOffMissionObjectID : ObjectID \\
LandingGearHandlerTakeOffObjectID : ObjectID \\
TakeOffMonitorObjectID : ObjectID \\
TakeOffFailureHandlerObjectID : ObjectID \\
CruiseMissionObjectID : ObjectID \\
BeginLandingHandlerObjectID : ObjectID \\
NavigationMonitorObjectID : ObjectID \\
LandMissionObjectID : ObjectID \\
GroundDistanceMonitorObjectID : ObjectID \\
LandingGearHandlerLandObjectID : ObjectID \\
InstrumentLandingSystemMonitorObjectID : ObjectID \\
SafeLandingHandlerObjectID : ObjectID
\end{array}
$$

$$
\begin{array}{l}
distinct\langle ACSafeletObjectID, MainMissionObjectID, \\
ACModeChangerObjectID, EnvironmentMonitorObjectID, \\
ControlHandlerObjectID, FlightSensorsMonitorObjectID, \\
CommunicationsHandlerObjectID, AperiodicSimulatorObjectID, \\
TakeOffMissionObjectID, LandingGearHandlerTakeOffObjectID, \\
TakeOffMonitorObjectID, TakeOffFailureHandlerObjectID, \\
CruiseMissionObjectID, BeginLandingHandlerObjectID, \\
NavigationMonitorObjectID, LandMissionObjectID, \\
GroundDistanceMonitorObjectID, LandingGearHandlerLandObjectID, \\
InstrumentLandingSystemMonitorObjectID, SafeLandingHandlerObjectID\rangle
\end{array}
$$

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 *Tier*0*Sync* ==
{| *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*, *CommunicationsHandlerAp*

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

process $Tier0 \mathrel{\widehat=}$
$$\left(\begin{array}{l} MissionFW(MainMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} SchedulableMissionSequencerFW(ACModeChangerID) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} AperiodicEventHandlerFW(ControlHandlerID, (time(10,0), null)) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW(CommunicationsHandlerID, (NULL, nullSchedulableId)) \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} PeriodicEventHandlerFW(EnvironmentMonitorID, (time(10,0), NULL, NULL, nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(FlightSensorsMonitorID, (time(10,0), NULL, NULL, nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(AperiodicSimulatorID, (time(10,0), NULL, NULL, nullSchedulableId)) \end{array}\right) \end{array}\right) \end{array}\right)$$

process $Tier1 \mathrel{\widehat=}$
$$\left(\begin{array}{l} MissionFW(TakeOffMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW(LandingGearHandlerTakeOffID, (NULL, nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW(TakeOffFailureHandlerID, (NULL, nullSchedulableId)) \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(TakeOffMonitorID, (time(0,0), time(500,0), NULL, nullSchedulableId)) \end{array}\right) \end{array}\right)$$
$$[\![ClusterSync]\!]$$
$$\left(\begin{array}{l} MissionFW(CruiseMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} AperiodicEventHandlerFW(BeginLandingHandlerID, (NULL, nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(NavigationMonitorID, (time(0,0), time(10,0), NULL, nullSchedulableId)) \end{array}\right) \end{array}\right)$$
$$[\![ClusterSync]\!]$$
$$\left(\begin{array}{l} MissionFW(LandMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW(LandingGearHandlerLandID, (NULL, nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW(SafeLandingHandlerID, (NULL, nullSchedulableId)) \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} PeriodicEventHandlerFW(GroundDistanceMonitorID, (time(0,0), time(10,0), NULL, nullSchedulableId)) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW(InstrumentLandingSystemMonitorID, (time(0,0), time(10,0), NULL, nullSchedulableId) \end{array}\right. \end{array}\right) \end{array}\right)$$

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

$$
\begin{array}{l}
\textbf{process}\ Application \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
ACSafeletApp \\
|||\ MainMissionSequencerApp \\
|||\ MainMissionApp \\
|||\ ACModeChangerApp(MainMissionID) \\
|||\ ControlHandlerApp \\
|||\ CommunicationsHandlerApp \\
|||\ EnvironmentMonitorApp(MainMissionID) \\
|||\ FlightSensorsMonitorApp(MainMissionID) \\
|||\ AperiodicSimulatorApp(controlHandlerID) \\
|||\ TakeOffMissionApp \\
|||\ LandingGearHandlerTakeOffApp(TakeOffMissionID) \\
|||\ TakeOffFailureHandlerApp(TakeOffMissionID, 10.0) \\
|||\ TakeOffMonitorApp(TakeOffMissionID, 10.0, landingGearHandlerID) \\
|||\ CruiseMissionApp \\
|||\ BeginLandingHandlerApp(CruiseMissionID) \\
|||\ NavigationMonitorApp(CruiseMissionID) \\
|||\ LandMissionApp \\
|||\ LandingGearHandlerLandApp(LandMissionID) \\
|||\ SafeLandingHandlerApp(LandMissionID, 10.0) \\
|||\ GroundDistanceMonitorApp(LandMissionID) \\
|||\ InstrumentLandingSystemMonitorApp(LandMissionID)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Threads \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
ThreadFW(SafeLandingHandlerThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(ACModeChangerThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(TakeOffFailureHandlerThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(InstrumentLandingSystemMonitorThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(FlightSensorsMonitorThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(TakeOffMonitorThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(AperiodicSimulatorThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(LandingGearHandlerLandThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(LandingGearHandlerTakeOffThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(GroundDistanceMonitorThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(ControlHandlerThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(CommunicationsHandlerThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(BeginLandingHandlerThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(NavigationMonitorThreadID, 5) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(EnvironmentMonitorThreadID, 5)
\end{array}\right)
\end{array}
$$

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

$$Locking \mathrel{\widehat{=}} Threads \;|||\; Objects$$

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

3 Safelet

section $ACSafeletApp$ **parents** $scj\_prelude, SchedulableId, SchedulableIds, SafeletChan$

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

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

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

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

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

end

4 Top Level Mission Sequencer

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

process $MainMissionSequencerApp \mathrel{\widehat=} \textbf{begin}$

State

$this : \textbf{ref}\ MainMissionSequencerClass$

state *State*

Init

$State'$

$this' = \textbf{new}\ MainMissionSequencerClass()$

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

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

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

end

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

state *State*

$$returnedMission : \mathbb{B}$$

state *State*

initial *Init*

$$State'$$

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

$$\left(\begin{array}{l} \textbf{if } (\neg\ returnedMission = \textbf{True}) \longrightarrow \\ \quad \left(\begin{array}{l} this\,.\,returnedMission := true; \\ ret := MainMission \end{array}\right) \\ [] \neg\ (\neg\ returnedMission = \textbf{True}) \longrightarrow \\ \quad \left(ret := nullMissionId\right) \\ \textbf{fi} \end{array}\right)$$

• **Skip**

end

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

channel $getNextMissionCall : SchedulableID$
channel $getNextMissionRet : SchedulableID \times MissionID$

5 Missions

5.1 MainMission

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

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

State

$this : \mathbf{ref}\ MainMissionClass$

state *State*

Init

$State'$

$this' = \mathbf{new}\ MainMissionClass()$

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

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

$$getAirSpeedSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \left(\begin{array}{l} getAirSpeedCall\,.\,MainMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,getAirSpeed(); \\ endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ getAirSpeedRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right) \end{array}\right)$$

$$getAltitudeSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \left(\begin{array}{l} getAltitudeCall\,.\,MainMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,MainMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,getAltitude(); \\ endSyncMeth\,.\,MainMissionObject\,.\,thread \longrightarrow \\ getAltitudeRet\,.\,MainMission\,!\,thread\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right) \end{array}\right)$$

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

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

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

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

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

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

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

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

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

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

$$Methods \mathrel{\widehat{=}} \left(\begin{array}{l} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ getAirSpeedSyncMeth \\ \Box \\ getAltitudeSyncMeth \\ \Box \\ getCabinPressureSyncMeth \\ \Box \\ getEmergencyOxygenSyncMeth \\ \Box \\ getFuelRemainingSyncMeth \\ \Box \\ getHeadingSyncMeth \\ \Box \\ setAirSpeedSyncMeth \\ \Box \\ setAltitudeSyncMeth \\ \Box \\ setCabinPressureSyncMeth \\ \Box \\ setEmergencyOxygenSyncMeth \\ \Box \\ setFuelRemainingSyncMeth \\ \Box \\ setHeadingSyncMeth \end{array} \right) ; \; Methods$$

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

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* $\bullet$
$(ret := airSpeed)$

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

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

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

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

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

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

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

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

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

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

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

- **Skip**

end

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Schedulables of MainMission

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

process $ACModeChangerApp \mathrel{\widehat=}$
$ControllingMission : MainMissionID \bullet$ **begin**

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

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

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

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

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

end

class $ACModeChangerClass \mathrel{\widehat=}$ **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 $currentModeClass' = \textbf{new}\ ModeClass()$
ref $launchModeClass' = \textbf{new}\ ModeClass()$
ref $cruiseModeClass' = \textbf{new}\ ModeClass()$
ref $landModeClass' = \textbf{new}\ ModeClass()$

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

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

• **Skip**

end

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

channel *advanceModeCall* : *SchedulableID*
channel *advanceModeRet* : *SchedulableID*

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

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

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

$$
\begin{array}{l}
handlerAsyncEvent \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
handleAsyncEventCall\,.\,ControlHandler \longrightarrow \\
(\mathbf{Skip})\,; \\
handleAsyncEventRet\,.\,ControlHandler \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
(handlerAsyncEvent)\,;\ Methods
\end{array}
$$

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

end

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

- **Skip**

end

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

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

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

$$handlerAsyncEvent \mathrel{\widehat=} \left(\begin{array}{l} handleAsyncEventCall\,.\,CommunicationsHandler \longrightarrow \\ (\mathbf{Skip})\,; \\ handleAsyncEventRet\,.\,CommunicationsHandler \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

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

- **Skip**

end

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

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

process $EnvironmentMonitorApp \mathrel{\widehat=}$
$ControllingMission : MainMissionID \bullet$ **begin**

$handlerAsyncEvent \mathrel{\widehat=}$
$\Big($ $handleAsyncEventCall\,.\,EnvironmentMonitor \longrightarrow$
$\Big($ **Skip**;
$setCabinPressureCall\,.\,controllingMission\,!\,0 \longrightarrow$
$setCabinPressureRet\,.\,controllingMission \longrightarrow$
Skip;
$setEmergencyOxygenCall\,.\,controllingMission\,!\,0 \longrightarrow$
$setEmergencyOxygenRet\,.\,controllingMission \longrightarrow$
Skip;
$setFuelRemainingCall\,.\,controllingMission\,!\,0 \longrightarrow$
$setFuelRemainingRet\,.\,controllingMission \longrightarrow$
Skip $\Big)$;
$handleAsyncEventRet\,.\,EnvironmentMonitor \longrightarrow$
Skip $\Big)$

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

$\bullet\ (Methods) \triangle (end\_periodic\_app\,.\,EnvironmentMonitor \longrightarrow$ **Skip**$)$

end

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

- **Skip**

end

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

process $FlightSensorsMonitorApp \mathrel{\widehat=}$
$ControllingMission : MainMissionID \bullet \textbf{begin}$

$handlerAsyncEvent \mathrel{\widehat=}$
$\Big(handleAsyncEventCall\,.\,FlightSensorsMonitor \longrightarrow$
$\Big(\textbf{Skip};$
$setAirSpeedCall\,.\,controllingMission\,!\,0 \longrightarrow$
$setAirSpeedRet\,.\,controllingMission \longrightarrow$
$\textbf{Skip};$
$setAltitudeCall\,.\,controllingMission\,!\,0 \longrightarrow$
$setAltitudeRet\,.\,controllingMission \longrightarrow$
$\textbf{Skip};$
$setHeadingCall\,.\,controllingMission\,!\,0 \longrightarrow$
$setHeadingRet\,.\,controllingMission \longrightarrow$
$\textbf{Skip}\Big)$
$;$
$handleAsyncEventRet\,.\,FlightSensorsMonitor \longrightarrow$
$\textbf{Skip}\Big)$

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

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

end

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

- **Skip**

end

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

process $AperiodicSimulatorApp \mathrel{\widehat=}$
$controlHandlerID : controlHandlerID \bullet$ **begin**

$$handlerAsyncEvent \mathrel{\widehat=}$$
$$\left(\begin{array}{l} handleAsyncEventCall \,.\, AperiodicSimulator \longrightarrow \\ \left(\begin{array}{l} \textbf{Skip}; \\ releaseCall \,.\, event \longrightarrow \\ releaseRet \,.\, event \,?\, release \longrightarrow \\ \textbf{Skip} \end{array}\right); \\ handleAsyncEventRet \,.\, AperiodicSimulator \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

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

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

end

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

- **Skip**

end

5.3 TakeOffMission

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

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

State

$this : \mathbf{ref}\ TakeOffMissionClass$

state *State*

Init

$State'$

$this' = \mathbf{new}\ TakeOffMissionClass()$

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

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

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

$abortSyncMeth \mathrel{\widehat=}$
$\left(\begin{array}{l} abortCall\,.\,TakeOffMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\ this\,.\,abort(); \\ endSyncMeth\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\ abortRet\,.\,TakeOffMission\,.\,thread \longrightarrow \\ \mathbf{Skip} \end{array}\right) \end{array}\right)$

$getControllingMissionSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : MissionID \bullet$
$\left(\begin{array}{l} getControllingMissionCall\,.\,TakeOffMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,getControllingMission(); \\ endSyncMeth\,.\,TakeOffMissionObject\,.\,thread \longrightarrow \\ getControllingMissionRet\,.\,TakeOffMission\,!\,thread\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right) \end{array}\right)$

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

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

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

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

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

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

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 sync $abort \mathrel{\widehat{=}}$
$(this\,.\,abort := true)$

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

public sync $setControllingMission \mathrel{\widehat{=}}$
$(this\,.\,this.controllingMission := controllingMission)$

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

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

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

• **Skip**

end

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

channel $deployLandingGearCall : MissionID$
channel $deployLandingGearRet : MissionID$

channel $abortCall : MissionID \times ThreadID$
channel $abortRet : MissionID \times ThreadID$

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

channel $setControllingMissionCall : MissionID \times ThreadID \times MissionID$
channel $setControllingMissionRet : MissionID \times ThreadID$

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

channel $stowLandingGearCall : MissionID \times ThreadID$
channel $stowLandingGearRet : MissionID \times ThreadID$

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

5.4 Schedulables of TakeOffMission

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

process $LandingGearHandlerTakeOffApp \mathrel{\widehat=}$
$ControllingMission : TakeOffMissionID \bullet$ **begin**

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

$$
\begin{array}{l}
Methods \mathrel{\widehat=} \\
\left(handlerAsyncEvent\right);\ Methods
\end{array}
$$

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

end

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

- **Skip**

end

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

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

process $TakeOffFailureHandlerApp \mathrel{\widehat=}$
$ControllingMission : TakeOffMissionID,$
$SAFE\_AIRSPEED\_THRESHOLD : 10.0 \bullet$ **begin**

$handlerAsyncEvent \mathrel{\widehat=}$
$\Big($ $handleAsyncEventCall\,.\,TakeOffFailureHandler \longrightarrow$
$\Big($ $getControllingMissionCall\,.\,takeoffMission.getControllingMission() \longrightarrow$
$getControllingMissionRet\,.\,takeoffMission.getControllingMission()\,?\,getControllingMission \longrightarrow$
var $currentSpeed : double \bullet currentSpeed := getAirSpeed$
if $(currentSpeed < threshold) \longrightarrow$
$\Big($ **Skip**;
$abortCall\,.\,takeoffMission \longrightarrow$
$abortRet\,.\,takeoffMission \longrightarrow$
Skip;
$requestTerminationCall\,.\,takeoffMission \longrightarrow$
$requestTerminationRet\,.\,takeoffMission\,?\,requestTermination \longrightarrow$
Skip $\Big)$
$[] \neg (currentSpeed < threshold) \longrightarrow$
$\big($**Skip**$\big)$
fi Skip $\Big)$;
$handleAsyncEventRet\,.\,TakeOffFailureHandler \longrightarrow$
Skip $\Big)$

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

$\bullet\ (Methods) \triangle (end\_\_app\,.\,TakeOffFailureHandler \longrightarrow$ **Skip**$)$

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 $TakeOffMonitorApp \mathrel{\widehat=}$
$ControllingMission : TakeOffMissionID,$
$TAKEOFF\_ALTITUDE : 10.0,$
$landingGearHandlerID : landingGearHandlerID \bullet$ **begin**

$handlerAsyncEvent \mathrel{\widehat=}$
$\Big($
$handleAsyncEventCall\,.\,TakeOffMonitor \longrightarrow$
$\Big($
$\Big($
Skip;
$getControllingMissionCall\,.\,takeoffMission.getControllingMission() \longrightarrow$
$getControllingMissionRet\,.\,takeoffMission.getControllingMission()\,?\,getControllingMission \longrightarrow$

var $altitude : double \bullet altitude := getAltitude$
if $(altitude > takeOffAltitude) \longrightarrow$
$\Big($
Skip;
$releaseCall\,.\,landingGearHandler \longrightarrow$
$releaseRet\,.\,landingGearHandler\,?\,release \longrightarrow$
$requestTerminationCall\,.\,takeoffMission \longrightarrow$
$requestTerminationRet\,.\,takeoffMission\,?\,requestTermination \longrightarrow$
Skip
$\Big)$
$[] \neg (altitude > takeOffAltitude) \longrightarrow$ **Skip**
fi ;
Skip
$\Big)$
$\Big)$
;
$handleAsyncEventRet\,.\,TakeOffMonitor \longrightarrow$
Skip
$\Big)$

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

$\bullet\ (Methods) \triangle (end\_periodic\_app\,.\,TakeOffMonitor \longrightarrow$ **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 $CruiseMissionApp \mathrel{\widehat=} \textbf{begin}$

State

$this : \textbf{ref}\ CruiseMissionClass$

state *State*

Init

$State'$

$this' = \textbf{new}\ CruiseMissionClass()$

$$InitializePhase \mathrel{\widehat=} \left(\begin{array}{l} initializeCall\,.\,CruiseMission \longrightarrow \\ register\,!\,BeginLandingHandler\,!\,CruiseMission \longrightarrow \\ register\,!\,NavigationMonitor\,!\,CruiseMission \longrightarrow \\ initializeRet\,.\,CruiseMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

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

$$getControllingMissionSyncMeth \mathrel{\widehat=} \textbf{var}\ ret : MissionID \bullet \left(\begin{array}{l} getControllingMissionCall\,.\,CruiseMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,CruiseMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,CruiseMissionObject\,.\,thread \longrightarrow \\ ret := this\,.\,getControllingMission(); \\ endSyncMeth\,.\,CruiseMissionObject\,.\,thread \longrightarrow \\ getControllingMissionRet\,.\,CruiseMission\,!\,thread\,!\,ret \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$$Methods \mathrel{\widehat=} \left(\begin{array}{l} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ getControllingMissionSyncMeth \end{array}\right) ;\ Methods$$

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

end

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

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

$\bullet$ **Skip**

end

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

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

5.6 Schedulables of CruiseMission

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

process *BeginLandingHandlerApp* $\widehat{=}$
ControllingMission : *CruiseMissionID* • **begin**

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

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
\big(handlerAsyncEvent\big)\ ;\ Methods
\end{array}
$$

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

end

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

- **Skip**

end

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

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

process $NavigationMonitorApp \mathrel{\widehat=}$
$ControllingMission : CruiseMissionID \bullet$ **begin**

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

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

$\bullet\ (Methods) \triangle (end\_periodic\_app \,.\, NavigationMonitor \longrightarrow$ **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 $LandMissionApp \mathrel{\widehat=} \textbf{begin}$

State

$this : \textbf{ref}\ LandMissionClass$

state *State*

Init

$State'$

$this' = \textbf{new}\ LandMissionClass()$

$InitializePhase \mathrel{\widehat=}$
$\left(\begin{array}{l} initializeCall\,.\,LandMission \longrightarrow \\ register\,!\,GroundDistanceMonitor\,!\,LandMission \longrightarrow \\ register\,!\,LandingGearHandlerLand\,!\,LandMission \longrightarrow \\ register\,!\,InstrumentLandingSystemMonitor\,!\,LandMission \longrightarrow \\ register\,!\,SafeLandingHandler\,!\,LandMission \longrightarrow \\ initializeRet\,.\,LandMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$

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

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

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

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

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

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

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

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

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

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 sync $stowLandingGear \mathrel{\widehat{=}}$
$(this\,.\,landingGearDeployed := false)$

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

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

public sync $abort \mathrel{\widehat{=}}$
$(this\,.\,abort := true)$

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

• **Skip**

end

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

channel *deployLandingGearCall* : *MissionID*
channel *deployLandingGearRet* : *MissionID*

channel *stowLandingGearCall* : *MissionID* × *ThreadID*
channel *stowLandingGearRet* : *MissionID* × *ThreadID*

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

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

channel *abortCall* : *MissionID* × *ThreadID*
channel *abortRet* : *MissionID* × *ThreadID*

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

5.8 Schedulables of LandMission

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

process $LandingGearHandlerLandApp \mathrel{\widehat=}$
$ControllingMission : LandMissionID \bullet$ **begin**

$handlerAsyncEvent \mathrel{\widehat=}$
$\Big($ *handleAsyncEventCall . LandingGearHandlerLand* $\longrightarrow$
$\Big($ **Skip**;
isLandingGearDeployedCall . mission $\longrightarrow$
isLandingGearDeployedRet . mission ? *isLandingGearDeployed* $\longrightarrow$

var *landingGearIsDeployed* : $\mathbb{B} \bullet$ *landingGearIsDeployed* := *isLandingGearDeployed*
if *landingGearIsDeployed* = **True** $\longrightarrow$
$\Big($ *stowLandingGearCall . mission* $\longrightarrow$
stowLandingGearRet . mission $\longrightarrow$
Skip $\Big)$
$[]\ \neg$ *landingGearIsDeployed* = **True** $\longrightarrow$
$\Big($ *deployLandingGearCall . mission . LandingGearHandlerLandThread* $\longrightarrow$
deployLandingGearRet . mission . LandingGearHandlerLandThread $\longrightarrow$
Skip $\Big)$
fi $\Big)$;
handleAsyncEventRet . LandingGearHandlerLand $\longrightarrow$
Skip $\Big)$

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

$\bullet$ $(Methods) \triangle (end\_app\ .\ LandingGearHandlerLand \longrightarrow$ **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* $\widehat{=}$
ControllingMission : *LandMissionID*,
SAFE_LANDING_ALTITUDE : 10.0 • **begin**

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

var *altitude* : *double* • *altitude* := *getAltitude*
if (*altitude* < *threshold*) $\longrightarrow$
$\big($**Skip**$\big)$
$[]$ ¬ (*altitude* < *threshold*) $\longrightarrow$
$\big($**Skip**$\big)$
fi
$\Big)$;
handleAsyncEventRet . *SafeLandingHandler* $\longrightarrow$
Skip $\Big)$

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

• (*Methods*) $\triangle$ (*end__app* . *SafeLandingHandler* $\longrightarrow$ **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* $\widehat{=}$
ControllingMission : *LandMissionID* • **begin**

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

$$
\begin{array}{l}
\textit{Methods} \mathrel{\widehat{=}} \\
\big(\textit{handlerAsyncEvent}\big)\,;\ \textit{Methods}
\end{array}
$$

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

end

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

state *State*

$readingOnGround : double$

state *State*

initial *Init*

$State'$

- **Skip**

end

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

process *InstrumentLandingSystemMonitorApp* $\widehat{=}$
$\quad$ *ControllingMission* : *LandMissionID* • **begin**

$$
\begin{array}{l}
handlerAsyncEvent \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
handleAsyncEventCall\,.\,InstrumentLandingSystemMonitor \longrightarrow \\
(\mathbf{Skip})\,; \\
handleAsyncEventRet\,.\,InstrumentLandingSystemMonitor \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
\big(handlerAsyncEvent\big)\,;\ Methods
\end{array}
$$

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

end

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

- **Skip**

end