

aircraft

Tight Rope v0.6

1st 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 *ThreadId* **parents** *scj_prelude, GlobalTypes*

distinct(*SafeletThreadId, nullThreadId,*
)

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* ==
 { *SafeltAppSync, MissionSequencerAppSync, MissionAppSync, MTAAppSync, OSEHSync, APEHSync, getSequencer, end_mission_app, end_managedThread_app, setCeilingPriority, requestTerminationCall, requestTerminationRet, terminationPendingCall, terminationPendingRet, handleAsyncEventCall, handleAsyncEventRet* }

channelset *ObjectSync* ==
 { }

channelset *ThreadSync* ==
 { }

channelset *LockingSync* ==
 { *lockAcquired, startSyncMeth, endSyncMeth, waitCall, waitRet, notify* }

channelset *Tier0Sync* ==
 { *done_toplevel_sequencer, done_safeletFW, start_mission ., done_mission ., initializeRet ., requestTermination . ., start_mission ., done_mission ., initializeRet ., requestTermination . ., start_mission ., done_mission ., initializeRet ., requestTermination . .* }

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

process *ControlTier* $\hat{=}$

$$\left(\begin{array}{l} \text{SafeletFW} \\ \llbracket \text{ControlTierSync} \rrbracket \\ \text{TopLevelMissionSequencerFW}(\text{MainMissionSequencer}) \end{array} \right)$$

process *Tier0* $\hat{=}$

$$\left(\begin{array}{l} \text{MissionFW}(\text{MainMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{SchedulableMissionSequencerFW}(\text{ACModeChangerID}) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \left(\begin{array}{l} \text{AperiodicEventHandlerFW}(\text{ControlHandlerID}, (\text{time}(10, 0), \text{null})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{AperiodicEventHandlerFW}(\text{CommunicationsHandlerID}, (\text{NULL}, \text{nullSchedulableId})) \end{array} \right) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \left(\begin{array}{l} \text{PeriodicEventHandlerFW}(\text{EnvironmentMonitorID}, (\text{time}(10, 0), \text{NULL}, \text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}(\text{FlightSensorsMonitorID}, (\text{time}(10, 0), \text{NULL}, \text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}(\text{AperiodicSimulatorID}, (\text{time}(10, 0), \text{NULL}, \text{NULL}, \text{nullSchedulableId})) \end{array} \right) \end{array} \right) \end{array} \right)$$

process *Tier1* $\hat{=}$

$$\left(\begin{array}{l} \text{MissionFW}(\text{TakeOffMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \left(\begin{array}{l} \text{AperiodicEventHandlerFW}(\text{LandingGearHandlerTakeOffID}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \end{array} \right) \\ \text{AperiodicEventHandlerFW}(\text{TakeOffFailureHandlerID}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}(\text{TakeOffMonitorID}, (\text{time}(0, 0), \text{time}(500, 0), \text{NULL}, \text{nullSchedulableId})) \end{array} \right) \\ \llbracket \text{ClusterSync} \rrbracket \\ \text{MissionFW}(\text{CruiseMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{AperiodicEventHandlerFW}(\text{BeginLandingHandlerID}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}(\text{NavigationMonitorID}, (\text{time}(0, 0), \text{time}(10, 0), \text{NULL}, \text{nullSchedulableId})) \end{array} \right) \\ \llbracket \text{ClusterSync} \rrbracket \\ \text{MissionFW}(\text{LandMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \left(\begin{array}{l} \text{AperiodicEventHandlerFW}(\text{LandingGearHandlerLandID}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \end{array} \right) \\ \text{AperiodicEventHandlerFW}(\text{SafeLandingHandlerID}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \left(\begin{array}{l} \text{PeriodicEventHandlerFW}(\text{GroundDistanceMonitorID}, (\text{time}(0, 0), \text{time}(10, 0), \text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}(\text{InstrumentLandingSystemMonitorID}, (\text{time}(0, 0), \text{time}(10, 0), \text{NULL}, \text{nullSchedulableId})) \end{array} \right) \end{array} \right) \end{array} \right)$$

process *Framework* $\hat{=}$

$$\left(\begin{array}{l} \text{ControlTier} \\ \llbracket \text{TierSync} \rrbracket \\ \left(\begin{array}{l} \text{Tier0} \\ \llbracket \text{Tier0Sync} \rrbracket \end{array} \right) \\ \text{Tier1} \end{array} \right)$$

process *Application* $\hat{=}$

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*)

$$\begin{aligned}
& \text{Threads} \triangleq \\
& \left(\begin{array}{l}
\text{ThreadFW}(\text{SafeLandingHandlerThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{ACModeChangerThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{TakeOffFailureHandlerThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{InstrumentLandingSystemMonitorThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{FlightSensorsMonitorThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{TakeOffMonitorThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{AperiodicSimulatorThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{LandingGearHandlerLandThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{LandingGearHandlerTakeOffThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{GroundDistanceMonitorThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{ControlHandlerThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{CommunicationsHandlerThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{BeginLandingHandlerThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{NavigationMonitorThreadID}, 5) \\
\quad \llbracket \text{ThreadSync} \rrbracket \\
\text{ThreadFW}(\text{EnvironmentMonitorThreadID}, 5)
\end{array} \right)
\end{aligned}$$

$$\begin{aligned}
\text{Objects} \hat{=} & \left(\begin{array}{l}
\text{ObjectFW}(\text{ACSafeletObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{MainMissionObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{ACModeChangerObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{EnvironmentMonitorObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{ControlHandlerObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{FlightSensorsMonitorObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{CommunicationsHandlerObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{AperiodicSimulatorObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{TakeOffMissionObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{LandingGearHandlerTakeOffObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{TakeOffMonitorObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{TakeOffFailureHandlerObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{CruiseMissionObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{BeginLandingHandlerObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{NavigationMonitorObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{LandMissionObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{GroundDistanceMonitorObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{LandingGearHandlerLandObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{InstrumentLandingSystemMonitorObjectID}) \\
\quad \llbracket \text{ObjectSync} \rrbracket \\
\text{ObjectFW}(\text{SafeLandingHandlerObjectID})
\end{array} \right)
\end{aligned}$$

$$\text{Locking} \hat{=} \text{Threads} \parallel \text{Objects}$$

$$\mathbf{process} \text{ Program} \hat{=} (\text{Framework} \llbracket \text{AppSync} \rrbracket \text{Application}) \llbracket \text{LockingSync} \rrbracket \text{Locking}$$

3 Safelet

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

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

InitializeApplication $\hat{=}$
 $\left(\begin{array}{l} \textit{initializeApplicationCall} \longrightarrow \\ \textit{initializeApplicationRet} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

GetSequencer $\hat{=}$
 $\left(\begin{array}{l} \textit{getSequencerCall} \longrightarrow \\ \textit{getSequencerRet} ! \textit{MainMissionSequencer} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

Methods $\hat{=}$
 $\left(\begin{array}{l} \textit{GetSequencer} \\ \square \\ \textit{InitializeApplication} \end{array} \right); \textit{Methods}$

• $(\textit{Methods}) \triangle (\textit{end_safelet_app} \longrightarrow \mathbf{Skip})$

end

4 Top Level Mission Sequencer

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

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

<i>State</i> <i>this</i> : ref <i>MainMissionSequencerClass</i>

state *State*

<i>Init</i> <i>State</i> '
<i>this</i> ' = new <i>MainMissionSequencerClass</i> ()

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

Methods $\hat{=}$
 $(\textit{GetNextMission}) ; \textit{Methods}$

• $(\textit{Init} ; \textit{Methods}) \triangle (\textit{end_sequencer_app} . \textit{MainMissionSequencer} \longrightarrow \mathbf{Skip})$

end

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

state <i>State</i> <i>returnedMission</i> : \mathbb{B}
--

state *State*

initial <i>Init</i> <i>State</i> '
--

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

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

• **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* $\hat{=}$
test : \mathbb{Z} • **begin**

<i>State</i> <i>this</i> : ref <i>MainMissionClass</i>
--

state *State*

<i>Init</i> <i>State'</i>
<i>this'</i> = new <i>MainMissionClass</i> ()

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

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

getAirSpeedSyncMeth $\hat{=}$ **var** *ret* : double •
 $\left(\begin{array}{l} \textit{getAirSpeedCall} . \textit{MainMission} ? \textit{thread} \longrightarrow \\ \left(\begin{array}{l} \textit{startSyncMeth} . \textit{MainMissionObject} . \textit{thread} \longrightarrow \\ \textit{lockAcquired} . \textit{MainMissionObject} . \textit{thread} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getAirSpeed}(); \\ \textit{endSyncMeth} . \textit{MainMissionObject} . \textit{thread} \longrightarrow \\ \textit{getAirSpeedRet} . \textit{MainMission} ! \textit{thread} ! \textit{ret} \longrightarrow \end{array} \right) \\ \textbf{Skip} \end{array} \right)$

getAltitudeSyncMeth $\hat{=}$ **var** *ret* : double •
 $\left(\begin{array}{l} \textit{getAltitudeCall} . \textit{MainMission} ? \textit{thread} \longrightarrow \\ \left(\begin{array}{l} \textit{startSyncMeth} . \textit{MainMissionObject} . \textit{thread} \longrightarrow \\ \textit{lockAcquired} . \textit{MainMissionObject} . \textit{thread} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getAltitude}(); \\ \textit{endSyncMeth} . \textit{MainMissionObject} . \textit{thread} \longrightarrow \\ \textit{getAltitudeRet} . \textit{MainMission} ! \textit{thread} ! \textit{ret} \longrightarrow \end{array} \right) \\ \textbf{Skip} \end{array} \right)$

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

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

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

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

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

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

$$\text{setCabinPressureSyncMeth} \hat{=} \left(\text{setCabinPressureCall} . \text{MainMission} ? \text{thread} ? \text{cabinPressure} \longrightarrow \right. \\ \left. \left(\text{startSyncMeth} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \right. \\ \left. \left. \text{lockAcquired} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \right. \\ \left. \left. \text{this} . \text{setCabinPressure}(\text{cabinPressure}); \right. \right. \\ \left. \left. \text{endSyncMeth} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \right. \\ \left. \left. \text{setCabinPressureRet} . \text{MainMission} . \text{thread} \longrightarrow \right. \right. \\ \left. \text{Skip} \right)$$

$$\text{setEmergencyOxygenSyncMeth} \hat{=} \left(\text{setEmergencyOxygenCall} . \text{MainMission} ? \text{thread} ? \text{emergencyOxygen} \longrightarrow \right. \\ \left(\text{startSyncMeth} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{lockAcquired} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{this} . \text{setEmergencyOxygen}(\text{emergencyOxygen}); \right. \\ \left. \text{endSyncMeth} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{setEmergencyOxygenRet} . \text{MainMission} . \text{thread} \longrightarrow \right. \\ \left. \text{Skip} \right)$$

$$\text{setFuelRemainingSyncMeth} \hat{=} \left(\text{setFuelRemainingCall} . \text{MainMission} ? \text{thread} ? \text{fuelRemaining} \longrightarrow \right. \\ \left(\text{startSyncMeth} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{lockAcquired} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{this} . \text{setFuelRemaining}(\text{fuelRemaining}); \right. \\ \left. \text{endSyncMeth} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{setFuelRemainingRet} . \text{MainMission} . \text{thread} \longrightarrow \right. \\ \left. \text{Skip} \right)$$

$$\text{setHeadingSyncMeth} \hat{=} \left(\text{setHeadingCall} . \text{MainMission} ? \text{thread} ? \text{heading} \longrightarrow \right. \\ \left(\text{startSyncMeth} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{lockAcquired} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{this} . \text{setHeading}(\text{heading}); \right. \\ \left. \text{endSyncMeth} . \text{MainMissionObject} . \text{thread} \longrightarrow \right. \\ \left. \text{setHeadingRet} . \text{MainMission} . \text{thread} \longrightarrow \right. \\ \left. \text{Skip} \right)$$

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

- $(Init ; Methods) \triangle (end_mission_app . MainMission \longrightarrow \mathbf{Skip})$

end

class *MainMissionClass* $\hat{=}$ **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* $\hat{=}$ **var** *ret* : *double* •
(*ret* := *airSpeed*)

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

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

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

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

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

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

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

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

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

```
public sync setFuelRemaining  $\hat{=}$   
(this.this.fuelRemaining := fuelRemaining)
```

```
public sync setHeading  $\hat{=}$   
(this.this.heading := heading)
```

- **Skip**

```
end
```

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

channel *getAirSpeedCall* : *MissionID* \times *ThreadID*
channel *getAirSpeedRet* : *MissionID* \times *ThreadID* \times *double*

channel *getAltitudeCall* : *MissionID* \times *ThreadID*
channel *getAltitudeRet* : *MissionID* \times *ThreadID* \times *double*

channel *getCabinPressureCall* : *MissionID* \times *ThreadID*
channel *getCabinPressureRet* : *MissionID* \times *ThreadID* \times *double*

channel *getEmergencyOxygenCall* : *MissionID* \times *ThreadID*
channel *getEmergencyOxygenRet* : *MissionID* \times *ThreadID* \times *double*

channel *getFuelRemainingCall* : *MissionID* \times *ThreadID*
channel *getFuelRemainingRet* : *MissionID* \times *ThreadID* \times *double*

channel *getHeadingCall* : *MissionID* \times *ThreadID*
channel *getHeadingRet* : *MissionID* \times *ThreadID* \times *double*

channel *setAirSpeedCall* : *MissionID* \times *ThreadID* \times *double*
channel *setAirSpeedRet* : *MissionID* \times *ThreadID*

channel *setAltitudeCall* : *MissionID* \times *ThreadID* \times *double*
channel *setAltitudeRet* : *MissionID* \times *ThreadID*

channel *setCabinPressureCall* : *MissionID* \times *ThreadID* \times *double*
channel *setCabinPressureRet* : *MissionID* \times *ThreadID*

channel *setEmergencyOxygenCall* : *MissionID* \times *ThreadID* \times *double*
channel *setEmergencyOxygenRet* : *MissionID* \times *ThreadID*

channel *setFuelRemainingCall* : *MissionID* \times *ThreadID* \times *double*
channel *setFuelRemainingRet* : *MissionID* \times *ThreadID*

channel *setHeadingCall* : *MissionID* \times *ThreadID* \times *double*
channel *setHeadingRet* : *MissionID* \times *ThreadID*

5.2 Schedulables of MainMission

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

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

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

advanceModeMeth $\hat{=}$
 $\left(\begin{array}{l} \text{advanceModeCall} . \text{ACModeChanger} \longrightarrow \\ \left(\begin{array}{l} \text{Skip}; \\ \text{if } (\text{modesLeft} = 3) \longrightarrow \\ \quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{changeTo}(\text{launchMode}) \end{array} \right) \\ \square \neg (\text{modesLeft} = 3) \longrightarrow \\ \quad \text{if } (\text{modesLeft} = 2) \longrightarrow \\ \quad \quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{changeTo}(\text{cruiseMode}) \end{array} \right) \\ \square \neg (\text{modesLeft} = 2) \longrightarrow \\ \quad \quad \text{if } (\text{modesLeft} = 1) \longrightarrow \\ \quad \quad \quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{changeTo}(\text{landMode}) \end{array} \right) \\ \square \neg (\text{modesLeft} = 1) \longrightarrow \\ \quad \quad \quad \left(\text{changeTo}(\text{null}) \right) \\ \text{fi} \\ \text{fi} \\ \text{fi} \end{array} \right) ; \\ \text{advanceModeRet} . \text{ACModeChanger} \longrightarrow \\ \text{Skip} \end{array} \right)$

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

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

• (*Methods*) \triangle (*end_sequencer_app* . *ACModeChanger* \longrightarrow **Skip**)

end

```
class ACModeChangerClass  $\hat{=}$  begin
```

```
  state State
```

```
    modesLeft :  $\mathbb{Z}$   
    ref currentModeClass : ModeClass  
    ref launchModeClass : ModeClass  
    ref cruiseModeClass : ModeClass  
    ref landModeClass : ModeClass
```

```
state State
```

```
  initial Init
```

```
    State'  
  
    modesLeft' = 3  
    ref currentModeClass' = new ModeClass()  
    ref launchModeClass' = new ModeClass()  
    ref cruiseModeClass' = new ModeClass()  
    ref landModeClass' = new ModeClass()
```

```
protected sync getNextMission  $\hat{=}$  var ret : MissionID •
```

```
  ( if (modesLeft = 3)  $\longrightarrow$   
    ( modesLeft := modesLeft - 1;  
      ret := TakeOffMission  
    )  
  ||  $\neg$  (modesLeft = 3)  $\longrightarrow$   
    if (modesLeft = 2)  $\longrightarrow$   
      ( modesLeft := modesLeft - 1;  
        ret := CruiseMission  
      )  
  ||  $\neg$  (modesLeft = 2)  $\longrightarrow$   
    if (modesLeft = 1)  $\longrightarrow$   
      ( modesLeft := modesLeft - 1;  
        ret := LandMission  
      )  
  ||  $\neg$  (modesLeft = 1)  $\longrightarrow$   
    ( ret := nullMissionId )  
  fi  
  fi  
  fi )
```

```
• Skip
```

```
end
```

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

channel *advanceModeCall* : *SchedulableID*

channel *advanceModeRet* : *SchedulableID*

channel *changeToCall* : *SchedulableID* \times *ThreadID* \times *Mode*

channel *changeToRet* : *SchedulableID* \times *ThreadID*

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

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

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

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

• $(\text{Methods}) \triangle (\text{end_app} . \text{ControlHandler} \longrightarrow \mathbf{Skip})$

end

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

- **Skip**

end

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

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

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

handlerAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{CommunicationsHandler} \longrightarrow \\ (\mathbf{Skip}) ; \\ \text{handleAsyncEventRet} . \text{CommunicationsHandler} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

• $(\text{Methods}) \triangle (\text{end_app} . \text{CommunicationsHandler} \longrightarrow \mathbf{Skip})$

end

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

- **Skip**

end

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

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

process *FlightSensorsMonitorApp* $\hat{=}$
mainMission : *MissionID* • **begin**

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

process *AperiodicSimulatorApp* $\hat{=}$
aperiodicEvent : *SchedulableID* • **begin**

handlerAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{AperiodicSimulator} \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{releaseCall} . \text{event} \longrightarrow \\ \text{releaseRet} . \text{event} ? \text{release} \longrightarrow \end{array} \right); \\ \mathbf{Skip} \\ \text{handleAsyncEventRet} . \text{AperiodicSimulator} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

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

- **Skip**

```
end
```

5.3 TakeOffMission

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

process *TakeOffMissionApp* $\hat{=}$
controllingMission : *MissionID* • **begin**

State
this : **ref** *TakeOffMissionClass*

state *State*

Init
State'

this' = **new** *TakeOffMissionClass*()

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

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

deployLandingGearMeth $\hat{=}$
 $\left(\begin{array}{l} \textit{deployLandingGearCall} . \textit{TakeOffMission} \longrightarrow \\ (\textit{this} . \textit{landingGearDeployed} := \textit{true}); \\ \textit{deployLandingGearRet} . \textit{TakeOffMission} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

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

getControllingMissionSyncMeth $\hat{=}$ **var** *ret* : *MissionID* •
 $\left(\begin{array}{l} \textit{getControllingMissionCall} . \textit{TakeOffMission} ? \textit{thread} \longrightarrow \\ \left(\begin{array}{l} \textit{startSyncMeth} . \textit{TakeOffMissionObject} . \textit{thread} \longrightarrow \\ \textit{lockAcquired} . \textit{TakeOffMissionObject} . \textit{thread} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getControllingMission}(); \\ \textit{endSyncMeth} . \textit{TakeOffMissionObject} . \textit{thread} \longrightarrow \\ \textit{getControllingMissionRet} . \textit{TakeOffMission} ! \textit{thread} ! \textit{ret} \longrightarrow \end{array} \right) \\ \textbf{Skip} \end{array} \right)$

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

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

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

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

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

$$\bullet (Init ; Methods) \triangle (end_mission_app . TakeOffMission \rightarrow \text{Skip})$$

end

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

state *State*

SAFE_AIRSPPEED_THRESHOLD : *double*
TAKEOFF_ALTITUDE : *double*
abort : \mathbb{B}
landingGearDeployed : \mathbb{B}

state *State*

initial *Init*

State'

SAFE_AIRSPPEED_THRESHOLD' = 10.0
TAKEOFF_ALTITUDE' = 10.0
abort' = *false*

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

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

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

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

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

public sync *isLandingGearDeployed* $\hat{=}$ **var** *ret* : \mathbb{B} •
(*ret* := *landingGearDeployed* = **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* $\hat{=}$
mission : *MissionID* • **begin**

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

process *TakeOffFailureHandlerApp* $\hat{=}$
takeoffMission : *MissionID*,
threshold : *Double* • **begin**

handlerAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{TakeOffFailureHandler} \longrightarrow \\ \left(\begin{array}{l} \text{getControllingMissionCall} . \text{takeoffMission} . \text{getControllingMission}() \longrightarrow \\ \text{getControllingMissionRet} . \text{takeoffMission} . \text{getControllingMission}() ? \text{getControllingMission} \longrightarrow \end{array} \right) \\ \\ \mathbf{var} \text{ currentSpeed} : \text{double} \bullet \text{currentSpeed} := \text{getAirSpeed} \\ \mathbf{if} (\text{currentSpeed} < \text{threshold}) \longrightarrow \\ \quad \left(\begin{array}{l} \mathbf{Skip}; \\ \text{abortCall} . \text{takeoffMission} \longrightarrow \\ \text{abortRet} . \text{takeoffMission} \longrightarrow \\ \mathbf{Skip}; \\ \text{requestTerminationCall} . \text{takeoffMission} \longrightarrow \\ \text{requestTerminationRet} . \text{takeoffMission} ? \text{requestTermination} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\ \quad \Box \neg (\text{currentSpeed} < \text{threshold}) \longrightarrow \\ \quad \quad (\mathbf{Skip}) \\ \mathbf{fi} \mathbf{Skip} \end{array} \right) ;$$

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

• (*Methods*) \triangle (*end_app* . *TakeOffFailureHandler* \longrightarrow **Skip**)

end

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

state <i>State</i> <i>threshold</i> : <i>double</i>

state *State*

initial <i>Init</i> <i>State'</i>

- **Skip**

end

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

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

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

handlerAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{TakeOffMonitor} \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{getControllingMissionCall} . \text{takeoffMission} . \text{getControllingMission}() \longrightarrow \\ \text{getControllingMissionRet} . \text{takeoffMission} . \text{getControllingMission}() ? \text{getControllingMission} \longrightarrow \\ \\ \mathbf{var} \text{altitude} : \text{double} \bullet \text{altitude} := \text{getAltitude} \\ \mathbf{if} (\text{altitude} > \text{takeOffAltitude}) \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{releaseCall} . \text{landingGearHandler} \longrightarrow \\ \text{releaseRet} . \text{landingGearHandler} ? \text{release} \longrightarrow \\ \text{requestTerminationCall} . \text{takeoffMission} \longrightarrow \\ \text{requestTerminationRet} . \text{takeoffMission} ? \text{requestTermination} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\ \mathbb{I} \neg (\text{altitude} > \text{takeOffAltitude}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi}; \\ \mathbf{Skip} \end{array} \right) ; \\ \text{handleAsyncEventRet} . \text{TakeOffMonitor} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

class *TakeOffMonitorClass* $\hat{=}$ **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* $\hat{=}$
controllingMission : *MissionID* • **begin**

<i>State</i> <i>this</i> : ref <i>CruiseMissionClass</i>
--

state *State*

<i>Init</i> <i>State'</i>
<i>this'</i> = new <i>CruiseMissionClass</i> ()

InitializePhase $\hat{=}$

$$\left(\begin{array}{l} \textit{initializeCall} . \textit{CruiseMission} \longrightarrow \\ \textit{register} ! \textit{BeginLandingHandler} ! \textit{CruiseMission} \longrightarrow \\ \textit{register} ! \textit{NavigationMonitor} ! \textit{CruiseMission} \longrightarrow \\ \textit{initializeRet} . \textit{CruiseMission} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

CleanupPhase $\hat{=}$

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

getControllingMissionSyncMeth $\hat{=}$ **var** *ret* : *MissionID* •

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

Methods $\hat{=}$
$$\left(\begin{array}{l} \textit{InitializePhase} \\ \square \\ \textit{CleanupPhase} \\ \square \\ \textit{getControllingMissionSyncMeth} \end{array} \right); \textit{Methods}$$

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

end

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

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

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

handlerAsyncEvent $\hat{=}$

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

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

• (*Methods*) \triangle (*end_periodic_app* . *NavigationMonitor* \longrightarrow **Skip**)

end

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

- **Skip**

end

5.7 LandMission

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

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

<i>State</i> <i>this</i> : ref <i>LandMissionClass</i>
--

state *State*

<i>Init</i> <i>State'</i>
<i>this'</i> = new <i>LandMissionClass</i> ()

InitializePhase $\hat{=}$

$$\left(\begin{array}{l} \text{initializeCall} . \text{LandMission} \longrightarrow \\ \text{register} ! \text{GroundDistanceMonitor} ! \text{LandMission} \longrightarrow \\ \text{register} ! \text{LandingGearHandlerLand} ! \text{LandMission} \longrightarrow \\ \text{register} ! \text{InstrumentLandingSystemMonitor} ! \text{LandMission} \longrightarrow \\ \text{register} ! \text{SafeLandingHandler} ! \text{LandMission} \longrightarrow \\ \text{initializeRet} . \text{LandMission} \longrightarrow \\ \text{Skip} \end{array} \right)$$

CleanupPhase $\hat{=}$

$$\left(\begin{array}{l} \text{cleanupMissionCall} . \text{LandMission} \longrightarrow \\ \text{cleanupMissionRet} . \text{LandMission} ! \text{True} \longrightarrow \\ \text{Skip} \end{array} \right)$$

deployLandingGearMeth $\hat{=}$

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

stowLandingGearSyncMeth $\hat{=}$

$$\left(\begin{array}{l} \text{stowLandingGearCall} . \text{LandMission} ? \text{thread} \longrightarrow \\ \left(\begin{array}{l} \text{startSyncMeth} . \text{LandMissionObject} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{LandMissionObject} . \text{thread} \longrightarrow \\ \text{this} . \text{stowLandingGear}(); \\ \text{endSyncMeth} . \text{LandMissionObject} . \text{thread} \longrightarrow \\ \text{stowLandingGearRet} . \text{LandMission} . \text{thread} \longrightarrow \end{array} \right) \\ \text{Skip} \end{array} \right)$$

isLandingGearDeployedSyncMeth $\hat{=}$ **var** *ret* : \mathbb{B} •

$$\left(\begin{array}{l} \text{isLandingGearDeployedCall} . \text{LandMission} ? \text{thread} \longrightarrow \\ \left(\begin{array}{l} \text{startSyncMeth} . \text{LandMissionObject} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{LandMissionObject} . \text{thread} \longrightarrow \\ \text{ret} := \text{this} . \text{isLandingGearDeployed}(); \\ \text{endSyncMeth} . \text{LandMissionObject} . \text{thread} \longrightarrow \\ \text{isLandingGearDeployedRet} . \text{LandMission} ! \text{thread} ! \text{ret} \longrightarrow \end{array} \right) \\ \text{Skip} \end{array} \right)$$

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

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

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

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

$$\bullet (\text{Init} ; \text{Methods}) \triangle (\text{end_mission_app} . \text{LandMission} \longrightarrow \mathbf{Skip})$$

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

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

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

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

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

• **Skip**

end

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

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

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

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

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

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

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

5.8 Schedulables of LandMission

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

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

handlerAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{SafeLandingHandler} \longrightarrow \\ \left(\begin{array}{l} \text{getControllingMissionCall} . \text{landMission} . \text{getControllingMission}() \longrightarrow \\ \text{getControllingMissionRet} . \text{landMission} . \text{getControllingMission}() ? \text{getControllingMission} \longrightarrow \end{array} \right) \\ \\ \text{var } \text{altitude} : \text{double} \bullet \text{altitude} := \text{getAltitude} \\ \text{if } (\text{altitude} < \text{threshold}) \longrightarrow \\ \quad (\text{Skip}) \\ \quad \square \neg (\text{altitude} < \text{threshold}) \longrightarrow \\ \quad (\text{Skip}) \\ \text{fi} \\ \text{handleAsyncEventRet} . \text{SafeLandingHandler} \longrightarrow \\ \text{Skip} \end{array} \right) ;$$

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

• (*Methods*) \triangle (*end_app* . *SafeLandingHandler* \longrightarrow **Skip**)

end

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

state <i>State</i> <i>threshold</i> : <i>double</i>

state *State*

initial <i>Init</i> <i>State</i> '
--

- **Skip**

end

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

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

process *GroundDistanceMonitorApp* $\hat{=}$
landMission : *MissionID* • **begin**

handlerAsyncEvent $\hat{=}$

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

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

• (*Methods*) \triangle (*end_periodic_app* . *GroundDistanceMonitor* \longrightarrow **Skip**)

end

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

state *State*

readingOnGround : *double*

state *State*

initial *Init*

State'

• **Skip**

end

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

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

handlerAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{InstrumentLandingSystemMonitor} \longrightarrow \\ (\mathbf{Skip}) ; \\ \text{handleAsyncEventRet} . \text{InstrumentLandingSystemMonitor} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

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

- **Skip**

end