

aircraft

Tight Rope v0.75

18th February 2017

1 ID Files

1.1 MissionIds

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

MainMissionMID : *MissionID*
TakeOffMissionMID : *MissionID*
CruiseMissionMID : *MissionID*
LandMissionMID : *MissionID*

distinct(*nullMissionId*, *MainMissionMID*, *TakeOffMissionMID*,
CruiseMissionMID, *LandMissionMID*)

1.2 SchedulablesIds

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

MainMissionSequencerSID : *SchedulableID*
ACModeChanger2SID : *SchedulableID*
EnvironmentMonitorSID : *SchedulableID*
ControlHandlerSID : *SchedulableID*
FlightSensorsMonitorSID : *SchedulableID*
CommunicationsHandlerSID : *SchedulableID*
LandingGearHandlerSID : *SchedulableID*
TakeOffMonitorSID : *SchedulableID*
TakeOffFailureHandlerSID : *SchedulableID*
BeginLandingHandlerSID : *SchedulableID*
NavigationMonitorSID : *SchedulableID*
GroundDistanceMonitorSID : *SchedulableID*
LandingGearHandlerLandSID : *SchedulableID*
InstrumentLandingSystemMonitorSID : *SchedulableID*
SafeLandingHandlerSID : *SchedulableID*

distinct(*nullSequencerId*, *nullSchedulableId*, *MainMissionSequencerSID*,
ACModeChanger2SID, *EnvironmentMonitorSID*,
ControlHandlerSID, *FlightSensorsMonitorSID*,
CommunicationsHandlerSID, *LandingGearHandlerSID*,
TakeOffMonitorSID, *TakeOffFailureHandlerSID*,
BeginLandingHandlerSID, *NavigationMonitorSID*,
GroundDistanceMonitorSID, *LandingGearHandlerLandSID*,
InstrumentLandingSystemMonitorSID, *SafeLandingHandlerSID*)

1.3 Non-Paradigm Objects

1.4 ThreadIds

section *ThreadId* **parents** *scj_prelude, GlobalTypes*

SafeletTid : *ThreadID*
nullThreadId : *ThreadID*

distinct(*SafeletTid*, *nullThreadId*)

1.5 ObjectIds

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

$distinct \langle \rangle$

2 Network

2.1 Network Channel Sets

```
section NetworkChannels parents scj_prelude, MissionId, MissionIds,  
    SchedulableId, SchedulableIds, MissionChan, TopLevelMissionSequencerFWChan,  
    FrameworkChan, SafeletChan, AperiodicEventHandlerChan, ManagedThreadChan,  
    OneShotEventHandlerChan, PeriodicEventHandlerChan, MissionSequencerMethChan  
  
channelset TerminateSync ==  
    { schedulables_terminated, schedulables_stopped, get_activeSchedulables }  
  
channelset ControlTierSync ==  
    { start_toplevel_sequencer, done_toplevel_sequencer, done_safeletFW }  
  
channelset TierSync ==  
    { start_mission . MainMission, done_mission . MainMission,  
      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 SafeltAppSync  $\hat{=}$   
    { getSequencerCall, getSequencerRet, initializeApplicationCall, initializeApplicationRet, end_safelet_app }  
  
channelset MissionSequencerAppSync ==  
    { getNextMissionCall, getNextMissionRet, end_sequencer_app }  
  
channelset MissionAppSync ==  
    { initializeCall, register, initializeRet, cleanupMissionCall, cleanupMissionRet }  
  
channelset AppSync ==  
    { SafeltAppSync, MissionSequencerAppSync, MissionAppSync,  
      MTAppSync, OSEHSync, APEHSync, PEHSync,  
      { getSequencer, end_mission_app, end_managedThread_app,  
        setCeilingPriority, requestTerminationCall, requestTerminationRet, terminationPendingCall,  
        terminationPendingRet, handleAsyncEventCall, handleAsyncEventRet } }  
  
channelset ThreadSync ==  
    { raise_thread_priority, lower_thread_priority, isInterruptedCall, isInterruptedRet, get_priorityLevel }  
  
channelset LockingSync ==  
    { lockAcquired, startSyncMeth, endSyncMeth, waitCall, waitRet, notify, isInterruptedCall, isInterruptedRet,  
      interruptedCall, interruptedRet, done_toplevel_sequencer, get_priorityLevel }  
  
channelset Tier0Sync ==  
    { done_toplevel_sequencer, done_safeletFW,  
      start_mission . TakeOffMission, done_mission . TakeOffMission,  
      initializeRet . TakeOffMission, requestTermination . TakeOffMission . MainMissionSequencer,  
      start_mission . CruiseMission, done_mission . CruiseMission,  
      initializeRet . CruiseMission, requestTermination . CruiseMission . MainMissionSequencer,  
      start_mission . LandMission, done_mission . LandMission,  
      initializeRet . LandMission, requestTermination . LandMission . MainMissionSequencer }
```

2.2 MethodCallBinder

section *MethodCallBindingChannels* **parents** *scj_prelude, GlobalTypes, FrameworkChan, MissionId, MissionIds, SchedulableId, SchedulableIds, ThreadIds*

channel *binder_setCabinPressureCall* : *MissionID* \times *SchedulableID* \times $\mathbb{P} \mathbb{A}$

channel *binder_setCabinPressureRet* : *MissionID* \times *SchedulableID*

setCabinPressureLocs == {*MainMissionMID*}

setCabinPressureCallers == {*EnvironmentMonitorSID*}

channel *binder_setFuelRemainingCall* : *MissionID* \times *SchedulableID* \times $\mathbb{P} \mathbb{A}$

channel *binder_setFuelRemainingRet* : *MissionID* \times *SchedulableID*

setFuelRemainingLocs == {*MainMissionMID*}

setFuelRemainingCallers == {*EnvironmentMonitorSID*}

channel *binder_getAltitudeCall* : *MissionID* \times *SchedulableID*

channel *binder_getAltitudeRet* : *MissionID* \times *SchedulableID* \times $\mathbb{P} \mathbb{A}$

getAltitudeLocs == {*MainMissionMID*}

getAltitudeCallers == {*NavigationMonitorSID, TakeOffMonitorSID, GroundDistanceMonitorSID, SafeLandingHandlerS*}

channel *binder_setHeadingCall* : *MissionID* \times *SchedulableID* \times $\mathbb{P} \mathbb{A}$

channel *binder_setHeadingRet* : *MissionID* \times *SchedulableID*

setHeadingLocs == {*MainMissionMID*}

setHeadingCallers == {*FlightSensorsMonitorSID*}

channel *binder_stowLandingGearCall* : *MissionID* \times *SchedulableID*

channel *binder_stowLandingGearRet* : *MissionID* \times *SchedulableID*

stowLandingGearLocs == {*TakeOffMissionMID, LandMissionMID*}

stowLandingGearCallers == {*LandingGearHandlerSID, LandingGearHandlerLandSID*}

channel *binder_takeOffAbortCall* : *MissionID* \times *SchedulableID*

channel *binder_takeOffAbortRet* : *MissionID* \times *SchedulableID*

takeOffAbortLocs == {*TakeOffMissionMID*}

takeOffAbortCallers == {*TakeOffFailureHandlerSID*}

channel *binder_setAltitudeCall* : *MissionID* \times *SchedulableID* \times $\mathbb{P} \mathbb{A}$

channel *binder_setAltitudeRet* : *MissionID* \times *SchedulableID*

setAltitudeLocs == {*MainMissionMID*}

setAltitudeCallers == {*FlightSensorsMonitorSID*}

channel *binder_getHeadingCall* : *MissionID* \times *SchedulableID*
channel *binder_getHeadingRet* : *MissionID* \times *SchedulableID* \times $\mathbb{P}\mathbb{A}$

getHeadingLocs == { *MainMissionMID* }
getHeadingCallers == { *NavigationMonitorSID* }

channel *binder_getAirSpeedCall* : *MissionID* \times *SchedulableID*
channel *binder_getAirSpeedRet* : *MissionID* \times *SchedulableID* \times $\mathbb{P}\mathbb{A}$

getAirSpeedLocs == { *MainMissionMID* }
getAirSpeedCallers == { *NavigationMonitorSID*, *TakeOffFailureHandlerSID* }

channel *binder_deployLandingGearCall* : *MissionID* \times *SchedulableID*
channel *binder_deployLandingGearRet* : *MissionID* \times *SchedulableID*

deployLandingGearLocs == { *TakeOffMissionMID*, *LandMissionMID* }
deployLandingGearCallers == { *LandingGearHandlerSID*, *LandingGearHandlerLandSID* }

channel *binder_setEmergencyOxygenCall* : *MissionID* \times *SchedulableID* \times $\mathbb{P}\mathbb{A}$
channel *binder_setEmergencyOxygenRet* : *MissionID* \times *SchedulableID*

setEmergencyOxygenLocs == { *MainMissionMID* }
setEmergencyOxygenCallers == { *EnvironmentMonitorSID* }

channel *binder_setAirSpeedCall* : *MissionID* \times *SchedulableID* \times $\mathbb{P}\mathbb{A}$
channel *binder_setAirSpeedRet* : *MissionID* \times *SchedulableID*

setAirSpeedLocs == { *MainMissionMID* }
setAirSpeedCallers == { *FlightSensorsMonitorSID* }

channel *binder_isLandingGearDeployedCall* : *MissionID* \times *SchedulableID*
channel *binder_isLandingGearDeployedRet* : *MissionID* \times *SchedulableID* \times \mathbb{B}

isLandingGearDeployedLocs == { *TakeOffMissionMID*, *LandMissionMID* }
isLandingGearDeployedCallers == { *LandingGearHandlerSID*, *LandingGearHandlerLandSID* }

channelset *MethodCallBinderSync* == { *done_toplevel_sequencer*,
binder_setCabinPressureCall, *binder_setCabinPressureRet*,
binder_setFuelRemainingCall, *binder_setFuelRemainingRet*,
binder_getAltitudeCall, *binder_getAltitudeRet*,
binder_setHeadingCall, *binder_setHeadingRet*,
binder_stowLandingGearCall, *binder_stowLandingGearRet*,
binder_takeOffAbortCall, *binder_takeOffAbortRet*,
binder_setAltitudeCall, *binder_setAltitudeRet*,
binder_getHeadingCall, *binder_getHeadingRet*,
binder_getAirSpeedCall, *binder_getAirSpeedRet*,
binder_deployLandingGearCall, *binder_deployLandingGearRet*,
binder_setEmergencyOxygenCall, *binder_setEmergencyOxygenRet*,
binder_setAirSpeedCall, *binder_setAirSpeedRet*,
binder_isLandingGearDeployedCall, *binder_isLandingGearDeployedRet* }

section *MethodCallBinder* **parents** *scj_prelude, MissionId, MissionIds,*
SchedulableId, SchedulableIds, MethodCallBindingChannels
, MainMissionMethChan, LandMissionMethChan

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

setCabinPressure_MethodBinder $\hat{=}$
 $\left(\begin{array}{l} \text{binder_setCabinPressureCall} ? \text{loc} : (\text{loc} \in \text{setCabinPressureLocs}) ? \text{caller} : (\text{caller} \in \text{setCabinPressureCallers}) ? p1 \longrightarrow \\ \text{setCabinPressureCall} . \text{loc} . \text{caller} ! p1 \longrightarrow \\ \text{setCabinPressureRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_setCabinPressureRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{setCabinPressure_MethodBinder} \end{array} \right)$

setFuelRemaining_MethodBinder $\hat{=}$
 $\left(\begin{array}{l} \text{binder_setFuelRemainingCall} ? \text{loc} : (\text{loc} \in \text{setFuelRemainingLocs}) ? \text{caller} : (\text{caller} \in \text{setFuelRemainingCallers}) ? p1 \longrightarrow \\ \text{setFuelRemainingCall} . \text{loc} . \text{caller} ! p1 \longrightarrow \\ \text{setFuelRemainingRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_setFuelRemainingRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{setFuelRemaining_MethodBinder} \end{array} \right)$

getAltitude_MethodBinder $\hat{=}$
 $\left(\begin{array}{l} \text{binder_getAltitudeCall} ? \text{loc} : (\text{loc} \in \text{getAltitudeLocs}) ? \text{caller} : (\text{caller} \in \text{getAltitudeCallers}) \longrightarrow \\ \text{getAltitudeCall} . \text{loc} . \text{caller} \longrightarrow \\ \text{getAltitudeRet} . \text{loc} . \text{caller} ? \text{ret} \longrightarrow \\ \text{binder_getAltitudeRet} . \text{loc} . \text{caller} ! \text{ret} \longrightarrow \\ \text{getAltitude_MethodBinder} \end{array} \right)$

setHeading_MethodBinder $\hat{=}$
 $\left(\begin{array}{l} \text{binder_setHeadingCall} ? \text{loc} : (\text{loc} \in \text{setHeadingLocs}) ? \text{caller} : (\text{caller} \in \text{setHeadingCallers}) ? p1 \longrightarrow \\ \text{setHeadingCall} . \text{loc} . \text{caller} ! p1 \longrightarrow \\ \text{setHeadingRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_setHeadingRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{setHeading_MethodBinder} \end{array} \right)$

stowLandingGear_MethodBinder $\hat{=}$
 $\left(\begin{array}{l} \text{binder_stowLandingGearCall} ? \text{loc} : (\text{loc} \in \text{stowLandingGearLocs}) ? \text{caller} : (\text{caller} \in \text{stowLandingGearCallers}) \longrightarrow \\ \text{stowLandingGearCall} . \text{loc} . \text{caller} \longrightarrow \\ \text{stowLandingGearRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_stowLandingGearRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{stowLandingGear_MethodBinder} \end{array} \right)$

takeOffAbort_MethodBinder $\hat{=}$
 $\left(\begin{array}{l} \text{binder_takeOffAbortCall} ? \text{loc} : (\text{loc} \in \text{takeOffAbortLocs}) ? \text{caller} : (\text{caller} \in \text{takeOffAbortCallers}) \longrightarrow \\ \text{takeOffAbortCall} . \text{loc} . \text{caller} \longrightarrow \\ \text{takeOffAbortRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_takeOffAbortRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{takeOffAbort_MethodBinder} \end{array} \right)$

setAltitude_MethodBinder $\hat{=}$
 $\left(\begin{array}{l} \text{binder_setAltitudeCall} ? \text{loc} : (\text{loc} \in \text{setAltitudeLocs}) ? \text{caller} : (\text{caller} \in \text{setAltitudeCallers}) ? p1 \longrightarrow \\ \text{setAltitudeCall} . \text{loc} . \text{caller} ! p1 \longrightarrow \\ \text{setAltitudeRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_setAltitudeRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{setAltitude_MethodBinder} \end{array} \right)$

$$\text{getHeading_MethodBinder} \hat{=} \left(\begin{array}{l} \text{binder_getHeadingCall} ? \text{loc} : (\text{loc} \in \text{getHeadingLocs}) ? \text{caller} : (\text{caller} \in \text{getHeadingCallers}) \longrightarrow \\ \text{getHeadingCall} . \text{loc} . \text{caller} \longrightarrow \\ \text{getHeadingRet} . \text{loc} . \text{caller} ? \text{ret} \longrightarrow \\ \text{binder_getHeadingRet} . \text{loc} . \text{caller} ! \text{ret} \longrightarrow \\ \text{getHeading_MethodBinder} \end{array} \right)$$

$$\text{getAirSpeed_MethodBinder} \hat{=} \left(\begin{array}{l} \text{binder_getAirSpeedCall} ? \text{loc} : (\text{loc} \in \text{getAirSpeedLocs}) ? \text{caller} : (\text{caller} \in \text{getAirSpeedCallers}) \longrightarrow \\ \text{getAirSpeedCall} . \text{loc} . \text{caller} \longrightarrow \\ \text{getAirSpeedRet} . \text{loc} . \text{caller} ? \text{ret} \longrightarrow \\ \text{binder_getAirSpeedRet} . \text{loc} . \text{caller} ! \text{ret} \longrightarrow \\ \text{getAirSpeed_MethodBinder} \end{array} \right)$$

$$\text{deployLandingGear_MethodBinder} \hat{=} \left(\begin{array}{l} \text{binder_deployLandingGearCall} ? \text{loc} : (\text{loc} \in \text{deployLandingGearLocs}) ? \text{caller} : (\text{caller} \in \text{deployLandingGearCallers}) \longrightarrow \\ \text{deployLandingGearCall} . \text{loc} . \text{caller} \longrightarrow \\ \text{deployLandingGearRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_deployLandingGearRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{deployLandingGear_MethodBinder} \end{array} \right)$$

$$\text{setEmergencyOxygen_MethodBinder} \hat{=} \left(\begin{array}{l} \text{binder_setEmergencyOxygenCall} ? \text{loc} : (\text{loc} \in \text{setEmergencyOxygenLocs}) ? \text{caller} : (\text{caller} \in \text{setEmergencyOxygenCallers}) \longrightarrow \\ \text{setEmergencyOxygenCall} . \text{loc} . \text{caller} ! p1 \longrightarrow \\ \text{setEmergencyOxygenRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_setEmergencyOxygenRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{setEmergencyOxygen_MethodBinder} \end{array} \right)$$

$$\text{setAirSpeed_MethodBinder} \hat{=} \left(\begin{array}{l} \text{binder_setAirSpeedCall} ? \text{loc} : (\text{loc} \in \text{setAirSpeedLocs}) ? \text{caller} : (\text{caller} \in \text{setAirSpeedCallers}) ? p1 \longrightarrow \\ \text{setAirSpeedCall} . \text{loc} . \text{caller} ! p1 \longrightarrow \\ \text{setAirSpeedRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{binder_setAirSpeedRet} . \text{loc} . \text{caller} \longrightarrow \\ \text{setAirSpeed_MethodBinder} \end{array} \right)$$

$$\text{isLandingGearDeployed_MethodBinder} \hat{=} \left(\begin{array}{l} \text{binder_isLandingGearDeployedCall} ? \text{loc} : (\text{loc} \in \text{isLandingGearDeployedLocs}) ? \text{caller} : (\text{caller} \in \text{isLandingGearDeployedCallers}) \longrightarrow \\ \text{isLandingGearDeployedCall} . \text{loc} . \text{caller} \longrightarrow \\ \text{isLandingGearDeployedRet} . \text{loc} . \text{caller} ? \text{ret} \longrightarrow \\ \text{binder_isLandingGearDeployedRet} . \text{loc} . \text{caller} ! \text{ret} \longrightarrow \\ \text{isLandingGearDeployed_MethodBinder} \end{array} \right)$$

$$\begin{array}{l}
\text{BinderActions} \triangleq \\
\left(\begin{array}{l}
\text{setCabinPressure_MethodBinder} \\
||| \\
\text{setFuelRemaining_MethodBinder} \\
||| \\
\text{getAltitude_MethodBinder} \\
||| \\
\text{setHeading_MethodBinder} \\
||| \\
\text{stowLandingGear_MethodBinder} \\
||| \\
\text{takeOffAbort_MethodBinder} \\
||| \\
\text{setAltitude_MethodBinder} \\
||| \\
\text{getHeading_MethodBinder} \\
||| \\
\text{getAirSpeed_MethodBinder} \\
||| \\
\text{deployLandingGear_MethodBinder} \\
||| \\
\text{setEmergencyOxygen_MethodBinder} \\
||| \\
\text{setAirSpeed_MethodBinder} \\
||| \\
\text{isLandingGearDeployed_MethodBinder}
\end{array} \right)
\end{array}$$

- $\text{BinderActions} \triangleq (\text{done_toplevel_sequencer} \longrightarrow \mathbf{Skip})$

end

2.3 Locking

section *NetworkLocking* **parents** *scj_prelude, GlobalTypes, FrameworkChan, MissionId, MissionIds, ThreadIds, NetworkChannels, ObjectFW, ThreadFW, Priority*

process *Threads* $\hat{=}$
(**Skip**)

process *Objects* $\hat{=}$
(**Skip**)

process *Locking* $\hat{=}$ *Threads* [*ThreadSync*] *Objects*

2.4 Program

section *Program* **parents** *scj_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, MissionFW, SafeletFW, TopLevelMissionSequencerFW, NetworkChannels, ManagedThreadFW, SchedulableMissionSequencerFW, PeriodicEventHandlerFW, OneShotEventHandlerFW, AperiodicEventHandlerFW, ObjectFW, ThreadFW, ACSafeletApp, MainMissionSequencerApp, MainMissionApp, ACModeChanger2App, ControlHandlerApp, CommunicationsHandlerApp, EnvironmentMonitorApp, FlightSensorsMonitorApp, TakeOffMissionApp, LandingGearHandlerApp, TakeOffFailureHandlerApp, TakeOffMonitorApp, CruiseMissionApp, BeginLandingHandlerApp, NavigationMonitorApp, LandMissionApp, LandingGearHandlerLandApp, SafeLandingHandlerApp, GroundDistanceMonitorApp, InstrumentLandingSystemMonitorApp*

process *ControlTier* $\hat{=}$

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

process *Tier0* $\hat{=}$

$$\left(\begin{array}{l} \text{MissionFW}(\text{MainMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{SchedulableMissionSequencerFW}(\text{ACModeChanger2ID}) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{AperiodicEventHandlerFW}(\text{ControlHandlerID}, \text{aperiodic}, (\text{time}(10, 0), \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{AperiodicEventHandlerFW}(\text{CommunicationsHandlerID}, \text{aperiodic}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}(\text{EnvironmentMonitorID}, (\text{time}(10, 0), \text{NULL}, \text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}(\text{FlightSensorsMonitorID}, (\text{time}(10, 0), \text{NULL}, \text{NULL}, \text{nullSchedulableId})) \end{array} \right) \end{array} \right)$$

process *Tier1* $\hat{=}$

$$\left(\begin{array}{l} \text{MissionFW}(\text{TakeOffMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{AperiodicEventHandlerFW}(\text{LandingGearHandlerID}, \text{aperiodic}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{AperiodicEventHandlerFW}(\text{TakeOffFailureHandlerID}, \text{aperiodic}, (\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 \\ \left(\begin{array}{l} \text{MissionFW}(\text{CruiseMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{AperiodicEventHandlerFW}(\text{BeginLandingHandlerID}, \text{aperiodic}, (\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 \\ \left(\begin{array}{l} \text{MissionFW}(\text{LandMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{AperiodicEventHandlerFW}(\text{LandingGearHandlerLandID}, \text{aperiodic}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{AperiodicEventHandlerFW}(\text{SafeLandingHandlerID}, \text{aperiodic}, (\text{NULL}, \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \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)$$

$$\text{process Framework} \hat{=} \left(\begin{array}{c} \text{ControlTier} \\ \llbracket \text{TierSync} \rrbracket \\ \left(\begin{array}{c} \text{Tier0} \\ \llbracket \text{Tier0Sync} \rrbracket \end{array} \right) \\ \text{Tier1} \end{array} \right)$$

$$\text{process Application} \hat{=} \left(\begin{array}{l} \text{ACSafeletApp} \\ ||| \\ \text{MainMissionSequencerApp} \\ ||| \\ \text{MainMissionApp} \\ ||| \\ \text{ACModeChanger2App}(\text{MainMissionID}) \\ ||| \\ \text{ControlHandlerApp} \\ ||| \\ \text{CommunicationsHandlerApp} \\ ||| \\ \text{EnvironmentMonitorApp}(\text{MainMissionID}) \\ ||| \\ \text{FlightSensorsMonitorApp}(\text{MainMissionID}) \\ ||| \\ \text{TakeOffMissionApp} \\ ||| \\ \text{LandingGearHandlerApp}(\text{TakeOffMissionID}) \\ ||| \\ \text{TakeOffFailureHandlerApp}(\text{MissionID}, \text{TakeOffMissionID}, 10.0) \\ ||| \\ \text{TakeOffMonitorApp}(\text{MissionID}, \text{TakeOffMissionID}, 10.0, \text{landingGearHandlerID}) \\ ||| \\ \text{CruiseMissionApp} \\ ||| \\ \text{BeginLandingHandlerApp}(\text{MissionID}) \\ ||| \\ \text{NavigationMonitorApp}(\text{MissionID}) \\ ||| \\ \text{LandMissionApp} \\ ||| \\ \text{LandingGearHandlerLandApp}(\text{LandMissionID}) \\ ||| \\ \text{SafeLandingHandlerApp}(\text{MissionID}, 10.0) \\ ||| \\ \text{GroundDistanceMonitorApp}(\text{MissionID}) \\ ||| \\ \text{InstrumentLandingSystemMonitorApp}(\text{LandMissionID}) \end{array} \right)$$

$$\begin{aligned} \text{process Bound_Application} &\hat{=} \text{Application} \llbracket \text{MethodCallBinderSync} \rrbracket \text{MethodCallBinder} \\ \text{process Program} &\hat{=} (\text{Framework} \llbracket \text{AppSync} \rrbracket \text{Bound_Application}) \llbracket \text{LockingSync} \rrbracket \text{Locking} \end{aligned}$$

3 Safelet

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

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

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

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

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*, *SchedulableIds*, *MainMissionSequencerClass*, *MethodCallBindingChannels*

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

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

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

end

section *MainMissionSequencerClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *MethodCallBindingChannels*, *MissionId*, *MissionIds*

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

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

state *State*

initial <i>Init</i> <i>State</i> '
<i>returnedMission</i> ' = False

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

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

• **Skip**

end

5 Missions

5.1 MainMission

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

process *MainMissionApp* $\hat{=}$ **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 $\hat{=}$

$$\left(\begin{array}{l} \textit{initializeCall} . \textit{MainMissionMID} \longrightarrow \\ \textit{register} ! \textit{ACModeChanger2SID} ! \textit{MainMissionMID} \longrightarrow \\ \textit{register} ! \textit{EnvironmentMonitorSID} ! \textit{MainMissionMID} \longrightarrow \\ \textit{register} ! \textit{ControlHandlerSID} ! \textit{MainMissionMID} \longrightarrow \\ \textit{register} ! \textit{FlightSensorsMonitorSID} ! \textit{MainMissionMID} \longrightarrow \\ \textit{register} ! \textit{CommunicationsHandlerSID} ! \textit{MainMissionMID} \longrightarrow \\ \textit{initializeRet} . \textit{MainMissionMID} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

CleanupPhase $\hat{=}$

$$\left(\begin{array}{l} \textbf{var } \mathbb{B} : \textit{ret} \bullet \textit{cleanupMissionCall} . \textit{MainMissionMID} \longrightarrow \\ \textit{cleanupMissionRet} . \textit{MainMissionMID} ! \textbf{True} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

getAirSpeedMeth $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A} \bullet$

$$\left(\begin{array}{l} \textit{getAirSpeedCall} . \textit{MainMissionMID} ? \textit{caller} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getAirSpeed}(); \\ \textit{getAirSpeedRet} . \textit{MainMissionMID} . \textit{caller} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

getAltitudeMeth $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A} \bullet$

$$\left(\begin{array}{l} \textit{getAltitudeCall} . \textit{MainMissionMID} ? \textit{caller} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getAltitude}(); \\ \textit{getAltitudeRet} . \textit{MainMissionMID} . \textit{caller} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

getCabinPressureMeth $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A} \bullet$

$$\left(\begin{array}{l} \textit{getCabinPressureCall} . \textit{MainMissionMID} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getCabinPressure}(); \\ \textit{getCabinPressureRet} . \textit{MainMissionMID} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

$$\text{getEmergencyOxygenMeth} \hat{=} \mathbf{var} \text{ ret} : \mathbb{P} \mathbb{A} \bullet \left(\begin{array}{l} \text{getEmergencyOxygenCall} . \text{MainMissionMID} \longrightarrow \\ \text{ret} := \text{this} . \text{getEmergencyOxygen}(); \\ \text{getEmergencyOxygenRet} . \text{MainMissionMID} ! \text{ret} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$\text{getFuelRemainingMeth} \hat{=} \mathbf{var} \text{ ret} : \mathbb{P} \mathbb{A} \bullet \left(\begin{array}{l} \text{getFuelRemainingCall} . \text{MainMissionMID} \longrightarrow \\ \text{ret} := \text{this} . \text{getFuelRemaining}(); \\ \text{getFuelRemainingRet} . \text{MainMissionMID} ! \text{ret} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$\text{getHeadingMeth} \hat{=} \mathbf{var} \text{ ret} : \mathbb{P} \mathbb{A} \bullet \left(\begin{array}{l} \text{getHeadingCall} . \text{MainMissionMID} ? \text{caller} \longrightarrow \\ \text{ret} := \text{this} . \text{getHeading}(); \\ \text{getHeadingRet} . \text{MainMissionMID} . \text{caller} ! \text{ret} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$\text{setAirSpeedMeth} \hat{=} \left(\begin{array}{l} \text{setAirSpeedCall} . \text{MainMissionMID} ? \text{caller} ? \text{airSpeed} \longrightarrow \\ \text{this} . \text{setAirSpeed}(\text{airSpeed}); \\ \text{setAirSpeedRet} . \text{MainMissionMID} . \text{caller} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$\text{setAltitudeMeth} \hat{=} \left(\begin{array}{l} \text{setAltitudeCall} . \text{MainMissionMID} ? \text{caller} ? \text{altitude} \longrightarrow \\ \text{this} . \text{setAltitude}(\text{altitude}); \\ \text{setAltitudeRet} . \text{MainMissionMID} . \text{caller} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$\text{setCabinPressureMeth} \hat{=} \left(\begin{array}{l} \text{setCabinPressureCall} . \text{MainMissionMID} ? \text{caller} ? \text{cabinPressure} \longrightarrow \\ \text{this} . \text{setCabinPressure}(\text{cabinPressure}); \\ \text{setCabinPressureRet} . \text{MainMissionMID} . \text{caller} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$\text{setEmergencyOxygenMeth} \hat{=} \left(\begin{array}{l} \text{setEmergencyOxygenCall} . \text{MainMissionMID} ? \text{caller} ? \text{emergencyOxygen} \longrightarrow \\ \text{this} . \text{setEmergencyOxygen}(\text{emergencyOxygen}); \\ \text{setEmergencyOxygenRet} . \text{MainMissionMID} . \text{caller} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$\text{setFuelRemainingMeth} \hat{=} \left(\begin{array}{l} \text{setFuelRemainingCall} . \text{MainMissionMID} ? \text{caller} ? \text{fuelRemaining} \longrightarrow \\ \text{this} . \text{setFuelRemaining}(\text{fuelRemaining}); \\ \text{setFuelRemainingRet} . \text{MainMissionMID} . \text{caller} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$\text{setHeadingMeth} \hat{=} \left(\begin{array}{l} \text{setHeadingCall} . \text{MainMissionMID} ? \text{caller} ? \text{heading} \longrightarrow \\ \text{this} . \text{setHeading}(\text{heading}); \\ \text{setHeadingRet} . \text{MainMissionMID} . \text{caller} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

section *MainMissionClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *MethodCallBindingChannels*

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

state *State* _____

ALTITUDE_READING_ON_GROUND : $\mathbb{P} \mathbb{A}$
cabinPressure : $\mathbb{P} \mathbb{A}$
emergencyOxygen : $\mathbb{P} \mathbb{A}$
fuelRemaining : $\mathbb{P} \mathbb{A}$
altitude : $\mathbb{P} \mathbb{A}$
airSpeed : $\mathbb{P} \mathbb{A}$
heading : $\mathbb{P} \mathbb{A}$

state *State*

initial *Init* _____

State'
ALTITUDE_READING_ON_GROUND' = 0.0

public *getAirSpeed* $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A}$ •
(*ret* := *airSpeed*)

public *getAltitude* $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A}$ •
(*ret* := *altitude*)

public *getCabinPressure* $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A}$ •
(*ret* := *cabinPressure*)

public *getEmergencyOxygen* $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A}$ •
(*ret* := *emergencyOxygen*)

public *getFuelRemaining* $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A}$ •
(*ret* := *fuelRemaining*)

public *getHeading* $\hat{=}$ **var** *ret* : $\mathbb{P} \mathbb{A}$ •
(*ret* := *heading*)

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

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

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

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

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

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

- **Skip**

```
end
```

5.2 Schedulables of MainMission

section *ACModeChanger2App* **parents** *TopLevelMissionSequencerChan*,
MissionId, *MissionIds*, *SchedulableId*, *SchedulableIds*, *ACModeChanger2Class*, *MethodCallBindingChannels*

process *ACModeChanger2App* $\hat{=}$
controllingMission : *MissionID* • **begin**

<i>State</i> <i>this</i> : <i>ACModeChanger2Class</i>
--

state *State*

<i>Init</i> <i>State'</i>
<i>this'</i> = new <i>ACModeChanger2Class</i> ()

GetNextMission $\hat{=}$ **var** *ret* : *MissionID* •
 $\left(\begin{array}{l} \textit{getNextMissionCall} . \textit{ACModeChanger2SID} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getNextMission}(); \\ \textit{getNextMissionRet} . \textit{ACModeChanger2SID} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

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

• (*Init* ; *Methods*) \triangle (*end_sequencer_app* . *ACModeChanger2SID* \longrightarrow **Skip**)

end

section *ACModeChanger2Class* **parents** *scj_prelude, SchedulableId, SchedulableIds, SafeletChan*
, MethodCallBindingChannels, MissionId, MissionIds

class *ACModeChanger2Class* $\hat{=}$ **begin**

state *State*

controllingMission : *MainMission*
modesLeft : \mathbb{Z}

state *State*

initial *Init*

State'

modesLeft' = 3

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

$$\left(\begin{array}{l} \text{if } (modesLeft = 3) \longrightarrow \\ \quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := TakeOffMissionMID \end{array} \right) \\ \square \neg (modesLeft = 3) \longrightarrow \\ \quad \text{if } (modesLeft = 2) \longrightarrow \\ \quad \quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := CruiseMissionMID \end{array} \right) \\ \square \neg (modesLeft = 2) \longrightarrow \\ \quad \text{if } (modesLeft = 1) \longrightarrow \\ \quad \quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := LandMissionMID \end{array} \right) \\ \square \neg (modesLeft = 1) \longrightarrow \\ \quad (ret := nullMissionId) \\ \text{fi} \\ \text{fi} \\ \text{fi} \end{array} \right)$$

• **Skip**

end

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

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

handleAsyncEvent $\hat{=}$

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

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

$\bullet (\text{Methods}) \triangle (\text{end_aperiodic_app} . \text{ControlHandlerSID} \longrightarrow \mathbf{Skip})$

end

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

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

handleAsyncEvent $\hat{=}$

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

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

$\bullet (\text{Methods}) \triangle (\text{end_aperiodic_app} . \text{CommunicationsHandlerSID} \longrightarrow \mathbf{Skip})$

end

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

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

<i>State</i> <i>this</i> : <i>EnvironmentMonitorClass</i>
--

state *State*

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{EnvironmentMonitorSID} \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{binder_setCabinPressureCall} . \text{controllingMission} . \text{EnvironmentMonitorSID} ! 0 \longrightarrow \\ \text{binder_setCabinPressureRet} . \text{controllingMission} . \text{EnvironmentMonitorSID} \longrightarrow \\ \mathbf{Skip}; \\ \text{binder_setEmergencyOxygenCall} . \text{controllingMission} . \text{EnvironmentMonitorSID} ! 0 \longrightarrow \\ \text{binder_setEmergencyOxygenRet} . \text{controllingMission} . \text{EnvironmentMonitorSID} \longrightarrow \\ \mathbf{Skip}; \\ \text{binder_setFuelRemainingCall} . \text{controllingMission} . \text{EnvironmentMonitorSID} ! 0 \longrightarrow \\ \text{binder_setFuelRemainingRet} . \text{controllingMission} . \text{EnvironmentMonitorSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right) ; \\ \text{handleAsyncEventRet} . \text{EnvironmentMonitorSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

section *EnvironmentMonitorClass* **parents** *scj_prelude, SchedulableId, SchedulableIds, SafeletChan*
, MethodCallBindingChannels

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

state <i>State</i> <i>controllingMission : MainMission</i>
--

state *State*

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

• **Skip**

end

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

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

<i>State</i> <i>this</i> : <i>FlightSensorsMonitorClass</i>
--

state *State*

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{FlightSensorsMonitorSID} \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{binder_setAirSpeedCall} . \text{controllingMission} . \text{FlightSensorsMonitorSID} ! 0 \longrightarrow \\ \text{binder_setAirSpeedRet} . \text{controllingMission} . \text{FlightSensorsMonitorSID} \longrightarrow \\ \mathbf{Skip}; \\ \text{binder_setAltitudeCall} . \text{controllingMission} . \text{FlightSensorsMonitorSID} ! 0 \longrightarrow \\ \text{binder_setAltitudeRet} . \text{controllingMission} . \text{FlightSensorsMonitorSID} \longrightarrow \\ \mathbf{Skip}; \\ \text{binder_setHeadingCall} . \text{controllingMission} . \text{FlightSensorsMonitorSID} ! 0 \longrightarrow \\ \text{binder_setHeadingRet} . \text{controllingMission} . \text{FlightSensorsMonitorSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right) ; \\ \text{handleAsyncEventRet} . \text{FlightSensorsMonitorSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

section *FlightSensorsMonitorClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*
, MethodCallBindingChannels

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

state <i>State</i> <i>controllingMission</i> : <i>MainMission</i>

state *State*

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

• **Skip**

end

5.3 TakeOffMission

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

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

State
this : **ref** *TakeOffMissionClass*

state *State*

Init
State'

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

InitializePhase $\hat{=}$

$$\left(\begin{array}{l} \textit{initializeCall} . \textit{TakeOffMissionMID} \longrightarrow \\ \textit{register! LandingGearHandlerSID! TakeOffMissionMID} \longrightarrow \\ \textit{register! TakeOffMonitorSID! TakeOffMissionMID} \longrightarrow \\ \textit{register! TakeOffFailureHandlerSID! TakeOffMissionMID} \longrightarrow \\ \textit{initializeRet} . \textit{TakeOffMissionMID} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

CleanupPhase $\hat{=}$

$$\left(\begin{array}{l} \textbf{var } \mathbb{B} : \textit{ret} \bullet \textit{cleanupMissionCall} . \textit{TakeOffMissionMID} \longrightarrow \\ \textit{cleanupMissionRet} . \textit{TakeOffMissionMID! True} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

takeOffAbortMeth $\hat{=}$

$$\left(\begin{array}{l} \textit{takeOffAbortCall} . \textit{TakeOffMissionMID? caller} \longrightarrow \\ \textit{this} . \textit{takeOffAbort}(); \\ \textit{takeOffAbortRet} . \textit{TakeOffMissionMID} . \textit{caller} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

deployLandingGearMeth $\hat{=}$

$$\left(\begin{array}{l} \textit{deployLandingGearCall} . \textit{TakeOffMissionMID? caller} \longrightarrow \\ \textit{this} . \textit{deployLandingGear}(); \\ \textit{deployLandingGearRet} . \textit{TakeOffMissionMID} . \textit{caller} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

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

$$\left(\begin{array}{l} \textit{cleanUpCall} . \textit{TakeOffMissionMID} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{cleanUp}(); \\ \textit{cleanUpRet} . \textit{TakeOffMissionMID! ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

$$stowLandingGearMeth \hat{=} \left(\begin{array}{l} stowLandingGearCall . TakeOffMissionMID ? caller \longrightarrow \\ this . stowLandingGear(); \\ stowLandingGearRet . TakeOffMissionMID . caller \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

$$Methods \hat{=} \left(\begin{array}{l} InitializePhase \\ \square \\ CleanupPhase \\ \square \\ takeOffAbortMeth \\ \square \\ deployLandingGearMeth \\ \square \\ cleanUpMeth \\ \square \\ stowLandingGearMeth \\ \square \\ isLandingGearDeployedMeth \end{array} \right) ; Methods$$

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

end

section *TakeOffMissionClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *MethodCallBindingChannels*

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

state *State*

SAFE_AIRSPEED_THRESHOLD : $\mathbb{P} \mathbb{A}$
TAKEOFF_ALTITUDE : $\mathbb{P} \mathbb{A}$
controllingMission : *MainMission*
abort : \mathbb{B}
landingGearDeployed : \mathbb{B}

state *State*

initial *Init*

State'

SAFE_AIRSPEED_THRESHOLD' = 10.0
TAKEOFF_ALTITUDE' = 10.0
abort' = *false*

public *takeOffAbort* $\hat{=}$
(*abort* := **True**)

public *deployLandingGear* $\hat{=}$
(*landingGearDeployed* := **True**)

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

public *stowLandingGear* $\hat{=}$
(*landingGearDeployed* := **False**)

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

• **Skip**

end

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

channel *takeOffAbortCall* : *MissionID* \times *SchedulableID*
channel *takeOffAbortRet* : *MissionID* \times *SchedulableID*

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

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

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

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

5.4 Schedulables of TakeOffMission

section *LandingGearHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBinder*, *TakeOffMissionMethChan*

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{LandingGearHandlerSID} \longrightarrow \\ \left(\begin{array}{l} \text{Skip}; \\ \text{binder_isLandingGearDeployedCall} . \text{mission} . \text{LandingGearHandlerSID} \longrightarrow \\ \text{binder_isLandingGearDeployedRet} . \text{mission} . \text{LandingGearHandlerSID} ? \text{isLandingGearDeployed} \longrightarrow \\ \text{Skip}; \text{var landingGearIsDeployed} : \mathbb{B} \bullet \text{landingGearIsDeployed} := \text{isLandingGearDeployed}; \\ \text{if landingGearIsDeployed} = \text{True} \longrightarrow \\ \left(\begin{array}{l} \text{binder_stowLandingGearCall} . \text{mission} . \text{LandingGearHandlerSID} \longrightarrow \\ \text{binder_stowLandingGearRet} . \text{mission} . \text{LandingGearHandlerSID} \longrightarrow \end{array} \right) \\ \text{Skip} \\ \square \neg \text{landingGearIsDeployed} = \text{True} \longrightarrow \\ \left(\begin{array}{l} \text{binder_deployLandingGearCall} . \text{mission} . \text{LandingGearHandlerSID} \longrightarrow \\ \text{binder_deployLandingGearRet} . \text{mission} . \text{LandingGearHandlerSID} \longrightarrow \end{array} \right) \\ \text{Skip} \end{array} \right) \\ \text{fi} \\ \text{handleAsyncEventRet} . \text{LandingGearHandlerSID} \longrightarrow \\ \text{Skip} \end{array} \right) ;$$

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

• $(\text{Methods}) \triangle (\text{end_aperiodic_app} . \text{LandingGearHandlerSID} \longrightarrow \text{Skip})$

end

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

process *TakeOffFailureHandlerApp* $\hat{=}$
mainMission : *MissionID*,
takeoffMission : *MissionID*,
threshold : $\mathbb{P}\mathbb{A}$ • **begin**

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

state *State*

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{TakeOffFailureHandlerSID} \longrightarrow \\ \left(\begin{array}{l} \text{binder_getAirSpeedCall} . \text{mainMission} . \text{TakeOffFailureHandlerSID} \longrightarrow \\ \text{binder_getAirSpeedRet} . \text{mainMission} . \text{TakeOffFailureHandlerSID} ? \text{getAirSpeed} \longrightarrow \\ \mathbf{Skip}; \text{var } \text{currentSpeed} : \mathbb{P}\mathbb{A} \bullet \text{currentSpeed} := \text{getAirSpeed}; \\ \text{if } (\text{currentSpeed} < \text{threshold}) \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{binder_takeOffAbortCall} . \text{takeoffMission} . \text{TakeOffFailureHandlerSID} \longrightarrow \\ \text{binder_takeOffAbortRet} . \text{takeoffMission} . \text{TakeOffFailureHandlerSID} \longrightarrow \\ \mathbf{Skip}; \\ \text{requestTerminationCall} . \text{takeoffMission} . \text{TakeOffFailureHandlerSID} \longrightarrow \\ \text{requestTerminationRet} . \text{takeoffMission} . \text{TakeOffFailureHandlerSID} ? \text{requestTermination} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\ \sqcap \neg (\text{currentSpeed} < \text{threshold}) \longrightarrow \\ (\mathbf{Skip}) \end{array} \right) \\ \text{fi} \\ \text{handleAsyncEventRet} . \text{TakeOffFailureHandlerSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right) ;$$

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

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

end

section *TakeOffFailureHandlerClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChannels*, *MethodCallBindingChannels*

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

state *State*
threshold : $\mathbb{P} \mathbb{A}$

state *State*

initial *Init*
State '

• **Skip**

end

section *TakeOffMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChan*, *MainMissionMethChan*

process *TakeOffMonitorApp* $\hat{=}$
 mainMission : *MissionID*,
 takeOffMission : *MissionID*,
 takeOffAltitude : $\mathbb{P} \mathbb{A}$,
 landingGearHandler : *SchedulableID* • **begin**

<i>State</i> <i>this</i> : <i>TakeOffMonitorClass</i>
--

state *State*

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{TakeOffMonitorSID} \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{binder_getAltitudeCall} . \text{mainMission} . \text{TakeOffMonitorSID} \longrightarrow \\ \text{binder_getAltitudeRet} . \text{mainMission} . \text{TakeOffMonitorSID} ? \text{getAltitude} \longrightarrow \\ \mathbf{Skip}; \text{ var altitude} : \mathbb{P} \mathbb{A} \bullet \text{altitude} := \text{getAltitude}; \\ \text{if } (\text{altitude} > \text{takeOffAltitude}) \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{release} . \text{landingGearHandler} \longrightarrow \\ \mathbf{Skip}; \\ \text{requestTerminationCall} . \text{takeoffMission} . \text{TakeOffMonitorSID} \longrightarrow \\ \text{requestTerminationRet} . \text{takeoffMission} . \text{TakeOffMonitorSID} ? \text{requestTermination} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\ \parallel \neg (\text{altitude} > \text{takeOffAltitude}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array} \right) ; \\ \text{handleAsyncEventRet} . \text{TakeOffMonitorSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

section *TakeOffMonitorClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*
, MethodCallBindingChannels

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

state *State*

takeoffMission : *TakeOffMission*

takeOffAltitude : $\mathbb{P} \mathbb{A}$

state *State*

initial *Init*

State '

• **Skip**

end

5.5 CruiseMission

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

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 $\hat{=}$

$$\left(\begin{array}{l} \textit{initializeCall} . \textit{CruiseMissionMID} \longrightarrow \\ \textit{register} ! \textit{BeginLandingHandlerSID} ! \textit{CruiseMissionMID} \longrightarrow \\ \textit{register} ! \textit{NavigationMonitorSID} ! \textit{CruiseMissionMID} \longrightarrow \\ \textit{initializeRet} . \textit{CruiseMissionMID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

CleanupPhase $\hat{=}$

$$\left(\begin{array}{l} \mathbf{var} \ \mathbb{B} : \textit{ret} \bullet \textit{cleanupMissionCall} . \textit{CruiseMissionMID} \longrightarrow \\ \textit{cleanupMissionRet} . \textit{CruiseMissionMID} ! \mathbf{True} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

section *CruiseMissionClass* **parents** *scj_prelude, SchedulableId, SchedulableIds, SafeletChannels, MethodCallBindingChannels*

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

state <i>State</i> <i>controllingMission : MainMission</i>
--

state *State*

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

• **Skip**

end

5.6 Schedulables of CruiseMission

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

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

handleAsyncEvent $\hat{=}$

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

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

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

• (*Methods*) \triangle (*end_aperiodic_app* . *BeginLandingHandlerSID* \longrightarrow **Skip**)

end

section *NavigationMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBinding*
, MainMissionMethChan

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{NavigationMonitorSID} \longrightarrow \\ \left(\begin{array}{l} \text{binder_getHeadingCall} . \text{mainMission} . \text{NavigationMonitorSID} \longrightarrow \\ \text{binder_getHeadingRet} . \text{mainMission} . \text{NavigationMonitorSID} ? \text{getHeading} \longrightarrow \\ \mathbf{Skip} ; \mathbf{var} \text{ heading} : \mathbb{P} \mathbb{A} \bullet \text{heading} := \text{getHeading}; \\ \text{binder_getAirSpeedCall} . \text{mainMission} . \text{NavigationMonitorSID} \longrightarrow \\ \text{binder_getAirSpeedRet} . \text{mainMission} . \text{NavigationMonitorSID} ? \text{getAirSpeed} \longrightarrow \\ \mathbf{Skip} ; \mathbf{var} \text{ airSpeed} : \mathbb{P} \mathbb{A} \bullet \text{airSpeed} := \text{getAirSpeed}; \\ \text{binder_getAltitudeCall} . \text{mainMission} . \text{NavigationMonitorSID} \longrightarrow \\ \text{binder_getAltitudeRet} . \text{mainMission} . \text{NavigationMonitorSID} ? \text{getAltitude} \longrightarrow \\ \mathbf{Skip} ; \mathbf{var} \text{ altitude} : \mathbb{P} \mathbb{A} \bullet \text{altitude} := \text{getAltitude} \end{array} \right) ; \\ \text{handleAsyncEventRet} . \text{NavigationMonitorSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

5.7 LandMission

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

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

State
this : **ref** *LandMissionClass*

state *State*

Init
State'
this' = **new** *LandMissionClass*()

InitializePhase $\hat{=}$
 $\left(\begin{array}{l} \text{initializeCall} . \text{LandMissionMID} \longrightarrow \\ \text{register} ! \text{GroundDistanceMonitorSID} ! \text{LandMissionMID} \longrightarrow \\ \text{register} ! \text{LandingGearHandlerLandSID} ! \text{LandMissionMID} \longrightarrow \\ \text{register} ! \text{InstrumentLandingSystemMonitorSID} ! \text{LandMissionMID} \longrightarrow \\ \text{register} ! \text{SafeLandingHandlerSID} ! \text{LandMissionMID} \longrightarrow \\ \text{initializeRet} . \text{LandMissionMID} \longrightarrow \\ \text{Skip} \end{array} \right)$

CleanupPhase $\hat{=}$
 $\left(\begin{array}{l} \text{var } \mathbb{B} : \text{ret} \bullet \text{cleanupMissionCall} . \text{LandMissionMID} \longrightarrow \\ \text{cleanupMissionRet} . \text{LandMissionMID} ! \text{True} \longrightarrow \\ \text{Skip} \end{array} \right)$

deployLandingGearMeth $\hat{=}$
 $\left(\begin{array}{l} \text{deployLandingGearCall} . \text{LandMissionMID} ? \text{caller} \longrightarrow \\ \text{this} . \text{deployLandingGear}(); \\ \text{deployLandingGearRet} . \text{LandMissionMID} . \text{caller} \longrightarrow \\ \text{Skip} \end{array} \right)$

stowLandingGearMeth $\hat{=}$
 $\left(\begin{array}{l} \text{stowLandingGearCall} . \text{LandMissionMID} ? \text{caller} \longrightarrow \\ \text{this} . \text{stowLandingGear}(); \\ \text{stowLandingGearRet} . \text{LandMissionMID} . \text{caller} \longrightarrow \\ \text{Skip} \end{array} \right)$

isLandingGearDeployedMeth $\hat{=}$ **var** *ret* : \mathbb{B} •
 $\left(\begin{array}{l} \text{isLandingGearDeployedCall} . \text{LandMissionMID} ? \text{caller} \longrightarrow \\ \text{ret} := \text{this} . \text{isLandingGearDeployed}(); \\ \text{isLandingGearDeployedRet} . \text{LandMissionMID} . \text{caller} ! \text{ret} \longrightarrow \\ \text{Skip} \end{array} \right)$

$cleanUpMeth \hat{=} \mathbf{var} \ ret : \mathbb{B} \bullet$
 $\left(\begin{array}{l} cleanUpCall . LandMissionMID \longrightarrow \\ ret := this . cleanUp(); \\ cleanUpRet . LandMissionMID ! ret \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

$Methods \hat{=} \left(\begin{array}{l} InitializePhase \\ \square \\ CleanupPhase \\ \square \\ deployLandingGearMeth \\ \square \\ stowLandingGearMeth \\ \square \\ isLandingGearDeployedMeth \\ \square \\ cleanUpMeth \end{array} \right) ; Methods$

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

end

section *LandMissionClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChannels*, *MethodCallBindingChannels*

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

state *State*

controllingMission : *MainMission*
SAFE_LANDING_ALTITUDE : $\mathbb{P} \mathbb{A}$
abort : \mathbb{B}
landingGearDeployed : \mathbb{B}

state *State*

initial *Init*

State'

SAFE_LANDING_ALTITUDE' = 10.0
abort' = *false*

public *deployLandingGear* $\hat{=}$
(*landingGearDeployed* := **True**)

public *stowLandingGear* $\hat{=}$
(*landingGearDeployed* := **False**)

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

public *cleanUp* $\hat{=}$ **var** *ret* : \mathbb{B} •
(**Skip**;
ret := **False**)

• **Skip**

end

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

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

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

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

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

5.8 Schedulables of LandMission

section *LandingGearHandlerLandApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*, *MethodCall*, *LandMissionMethChan*

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{LandingGearHandlerLandSID} \longrightarrow \\ \left(\begin{array}{l} \text{Skip}; \\ \text{binder_isLandingGearDeployedCall} . \text{mission} . \text{LandingGearHandlerLandSID} \longrightarrow \\ \text{binder_isLandingGearDeployedRet} . \text{mission} . \text{LandingGearHandlerLandSID} ? \text{isLandingGearDeployed} \longrightarrow \\ \text{Skip}; \text{var landingGearIsDeployed} : \mathbb{B} \bullet \text{landingGearIsDeployed} := \text{isLandingGearDeployed}; \\ \text{if landingGearIsDeployed} = \text{True} \longrightarrow \\ \quad \left(\begin{array}{l} \text{binder_stowLandingGearCall} . \text{mission} . \text{LandingGearHandlerLandSID} \longrightarrow \\ \text{binder_stowLandingGearRet} . \text{mission} . \text{LandingGearHandlerLandSID} \longrightarrow \\ \text{Skip} \end{array} \right) \\ \quad \square \neg \text{landingGearIsDeployed} = \text{True} \longrightarrow \\ \quad \quad \left(\begin{array}{l} \text{binder_deployLandingGearCall} . \text{mission} . \text{LandingGearHandlerLandSID} \longrightarrow \\ \text{binder_deployLandingGearRet} . \text{mission} . \text{LandingGearHandlerLandSID} \longrightarrow \\ \text{Skip} \end{array} \right) \\ \text{fi} \end{array} \right) \\ \text{handleAsyncEventRet} . \text{LandingGearHandlerLandSID} \longrightarrow \\ \text{Skip} \end{array} \right);$$

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

• (*Methods*) \triangle (*end_aperiodic_app* . *LandingGearHandlerLandSID* \longrightarrow **Skip**)

end

section *SafeLandingHandlerApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBinding*, *MainMissionMethChan*

process *SafeLandingHandlerApp* $\hat{=}$
 mainMission : *MissionID*,
 threshold : $\mathbb{P} \mathbb{A}$ • **begin**

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

state *State*

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{SafeLandingHandlerSID} \longrightarrow \\ \left(\begin{array}{l} \text{binder_getAltitudeCall} . \text{mainMission} . \text{SafeLandingHandlerSID} \longrightarrow \\ \text{binder_getAltitudeRet} . \text{mainMission} . \text{SafeLandingHandlerSID} ? \text{getAltitude} \longrightarrow \\ \mathbf{Skip} ; \mathbf{var} \text{altitude} : \mathbb{P} \mathbb{A} \bullet \text{altitude} := \text{getAltitude}; \\ \mathbf{if} (\text{altitude} < \text{threshold}) \longrightarrow \\ \quad (\mathbf{Skip}) \\ \quad \neg (\text{altitude} < \text{threshold}) \longrightarrow \\ \quad \quad (\mathbf{Skip}) \\ \mathbf{fi} \end{array} \right) ; \\ \text{handleAsyncEventRet} . \text{SafeLandingHandlerSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

section *SafeLandingHandlerClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *MethodCallBindingChannels*

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

state *State*
threshold : $\mathbb{P} \mathbb{A}$

state *State*

initial *Init*
State '

• **Skip**

end

section *GroundDistanceMonitorApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBin*, *MainMissionMethChan*

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

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

state *State*

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

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} \text{handleAsyncEventCall} . \text{GroundDistanceMonitorSID} \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{binder_getAltitudeCall} . \text{mainMission} . \text{GroundDistanceMonitorSID} \longrightarrow \\ \text{binder_getAltitudeRet} . \text{mainMission} . \text{GroundDistanceMonitorSID} ? \text{getAltitude} \longrightarrow \\ \mathbf{Skip}; \text{var } \text{distance} : \mathbb{P} \mathbb{A} \bullet \text{distance} := \text{getAltitude}; \\ \text{if } (\text{distance} = \text{readingOnGround}) \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \text{requestTerminationCall} . \text{mainMission} . \text{GroundDistanceMonitorSID} \longrightarrow \\ \text{requestTerminationRet} . \text{mainMission} . \text{GroundDistanceMonitorSID} ? \text{requestTermination} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\ \Box \neg (\text{distance} = \text{readingOnGround}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array} \right) ; \\ \text{handleAsyncEventRet} . \text{GroundDistanceMonitorSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

section *GroundDistanceMonitorClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChannels*, *MethodCallBindingChannels*

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

state <i>State</i> <i>readingOnGround</i> : $\mathbb{P} \mathbb{A}$

state *State*

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

• **Skip**

end

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

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

handleAsyncEvent $\hat{=}$

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

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

• $(\text{Methods}) \triangle (\text{end_periodic_app} . \text{InstrumentLandingSystemMonitorSID} \longrightarrow \mathbf{Skip})$

end