

aircraft

Tight Rope v0.65

26th February 2016

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*
ACModeChangerSID : *SchedulableID*
EnvironmentMonitorSID : *SchedulableID*
ControlHandlerSID : *SchedulableID*
FlightSensorsMonitorSID : *SchedulableID*
CommunicationsHandlerSID : *SchedulableID*
AperiodicSimulatorSID : *SchedulableID*
LandingGearHandlerTakeOffSID : *SchedulableID*
TakeOffMonitorSID : *SchedulableID*
TakeOffFailureHandlerSID : *SchedulableID*
BeginLandingHandlerSID : *SchedulableID*
NavigationMonitorSID : *SchedulableID*
GroundDistanceMonitorSID : *SchedulableID*
LandingGearHandlerLandSID : *SchedulableID*
InstrumentLandingSystemMonitorSID : *SchedulableID*
SafeLandingHandlerSID : *SchedulableID*

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

1.3 ThreadIds

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

InstrumentLandingSystemMonitorTID : *ThreadId*
SafeLandingHandlerTID : *ThreadId*
GroundDistanceMonitorTID : *ThreadId*
CommunicationsHandlerTID : *ThreadId*
ControlHandlerTID : *ThreadId*
AperiodicSimulatorTID : *ThreadId*
TakeOffFailureHandlerTID : *ThreadId*
LandingGearHandlerLandTID : *ThreadId*
EnvironmentMonitorTID : *ThreadId*
FlightSensorsMonitorTID : *ThreadId*
NavigationMonitorTID : *ThreadId*
ACModeChangerTID : *ThreadId*
BeginLandingHandlerTID : *ThreadId*
LandingGearHandlerTakeOffTID : *ThreadId*
TakeOffMonitorTID : *ThreadId*

distinct(*SafeletTid*, *nullThreadId*,
InstrumentLandingSystemMonitorTID, *SafeLandingHandlerTID*,
GroundDistanceMonitorTID, *CommunicationsHandlerTID*,
ControlHandlerTID, *AperiodicSimulatorTID*,
TakeOffFailureHandlerTID, *LandingGearHandlerLandTID*,
EnvironmentMonitorTID, *FlightSensorsMonitorTID*,
NavigationMonitorTID, *ACModeChangerTID*,
BeginLandingHandlerTID, *LandingGearHandlerTakeOffTID*,
TakeOffMonitorTID)

1.4 ObjectIds

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

TakeOffMissionOID : *ObjectID*

LandMissionOID : *ObjectID*

distinct \langle *TakeOffMissionOID*, *LandMissionOID* \rangle

2 Network

2.1 Network Channel Sets

section *NetworkChannels* **parents** *scj_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, FrameworkChan, SafeletChan, MissionChan, TopLevelMissionSequencerFWChan, AperiodicEventHandlerChan, ManagedThreadChan, OneShotEventHandlerChan, PeriodicEventHandlerChan, MissionSequencerChan, ObjecChan, ThreadChan*

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 *SchedulablesSync* ==
 { *activate_schedulables, done_safeletFW, done_toplevel_sequencer* }

channelset *ClusterSync* ==
 { *done_toplevel_sequencer, done_safeletFW* }

channelset *AppSync* ==
 { *getSequencerCall, getSequencerRet, initializeApplicationCall, initializeApplicationRet, end_safelet_app, initializeCall, initializeRet, register, cleanupMissionCall, cleanupMissionRet, end_mission_app, getNextMissionCall, getNextMissionRet, end_sequencer_app, handleAsyncEventCall, handleAsyncEventRet, end_periodic_app, handleAsyncLongEventCall, handleAsyncLongEventRet, end_aperiodic_app, descheduleCall, descheduleRet, scheduleNextRelease, getNextReleaseTimeCall, getNextReleaseTimeRet, end_oneShot_runCall, runRet, end_managedThread_app, setCeilingPriority, requestTerminationCall, requestTerminationRet, terminationPendingCall, terminationPendingRet, done_safeletFW, done_toplevel_sequencer, signalTerminationCall, signalTerminationRet, activate_schedulables, done_schedulable, cleanupSchedulableCall, cleanupSchedulableRet* }

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

channel *binder_setEmergencyOxygenRet* : *MissionID* \times *SchedulableID*

setEmergencyOxygenLocs == {*MainMissionMID*}

setEmergencyOxygenCallers == {*EnvironmentMonitorSID*}

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

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

setFuelRemainingLocs == {*MainMissionMID*}

setFuelRemainingCallers == {*EnvironmentMonitorSID*}

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

channel *binder_setAirSpeedRet* : *MissionID* \times *SchedulableID*

setAirSpeedLocs == {*MainMissionMID*}

setAirSpeedCallers == {*FlightSensorsMonitorSID*}

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

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

setAltitudeLocs == {*MainMissionMID*}

setAltitudeCallers == {*FlightSensorsMonitorSID*}

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

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

setHeadingLocs == {*MainMissionMID*}

setHeadingCallers == {*FlightSensorsMonitorSID*}

channel *binder_isLandingGearDeployedCall* : *MissionID* \times *SchedulableID*

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

isLandingGearDeployedLocs == {*TakeOffMissionMID*}

isLandingGearDeployedCallers == {*LandingGearHandlerTakeOffSID*}

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

stowLandingGearLocs == { *TakeOffMissionMID* }
stowLandingGearCallers == { *LandingGearHandlerTakeOffSID* }

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

deployLandingGearLocs == { *TakeOffMissionMID* }
deployLandingGearCallers == { *LandingGearHandlerTakeOffSID* }

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

getAltitudeLocs == { *MainMissionMID* }
getAltitudeCallers == { *NavigationMonitorSID*, *TakeOffMonitorSID*, *GroundDistanceMonitorSID*, *SafeLandingHandlerSID* }

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

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

channel *binder_abortCall* : *MissionID* \times *SchedulableID*
channel *binder_abortRet* : *MissionID* \times *SchedulableID*

abortLocs == { *TakeOffMissionMID* }
abortCallers == { *TakeOffFailureHandlerSID* }

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

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

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

isLandingGearDeployedLocs == { *LandMissionMID* }
isLandingGearDeployedCallers == { *LandingGearHandlerLandSID* }

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

$stowLandingGearLocs == \{LandMissionMID\}$
 $stowLandingGearCallers == \{LandingGearHandlerLandSID\}$

channel $binder_deployLandingGearCall : MissionID \times SchedulableID \times ThreadID$
channel $binder_deployLandingGearRet : MissionID \times SchedulableID \times ThreadID$

$deployLandingGearLocs == \{LandMissionMID\}$
 $deployLandingGearCallers == \{LandingGearHandlerLandSID\}$

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

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

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

$setCabinPressure_MethodBinder \hat{=}$
 $\left(binder_setCabinPressureCall ? loc : (loc \in setCabinPressureLocs) ? caller : (caller \in setCabinPressureCallers) ? p1 \rightarrow \right.$
 $\quad setCabinPressureCall . loc . caller ! p1 \rightarrow$
 $\quad setCabinPressureRet . loc . caller \rightarrow$
 $\quad binder_setCabinPressureRet . loc . caller \rightarrow$
 $\quad \left. setCabinPressure_MethodBinder \right)$

$setEmergencyOxygen_MethodBinder \hat{=}$
 $\left(binder_setEmergencyOxygenCall ? loc : (loc \in setEmergencyOxygenLocs) ? caller : (caller \in setEmergencyOxygenCallers) ? p1 \rightarrow \right.$
 $\quad setEmergencyOxygenCall . loc . caller ! p1 \rightarrow$
 $\quad setEmergencyOxygenRet . loc . caller \rightarrow$
 $\quad binder_setEmergencyOxygenRet . loc . caller \rightarrow$
 $\quad \left. setEmergencyOxygen_MethodBinder \right)$

$setFuelRemaining_MethodBinder \hat{=}$
 $\left(binder_setFuelRemainingCall ? loc : (loc \in setFuelRemainingLocs) ? caller : (caller \in setFuelRemainingCallers) ? p1 \rightarrow \right.$
 $\quad setFuelRemainingCall . loc . caller ! p1 \rightarrow$
 $\quad setFuelRemainingRet . loc . caller \rightarrow$
 $\quad binder_setFuelRemainingRet . loc . caller \rightarrow$
 $\quad \left. setFuelRemaining_MethodBinder \right)$

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

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

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

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

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

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

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

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

$abort_MethodBinder \hat{=}$

$$\left(\begin{array}{l} binder_abortCall ? loc : (loc \in abortLocs) ? caller : (caller \in abortCallers) \longrightarrow \\ abortCall . loc . caller \longrightarrow \\ abortRet . loc . caller \longrightarrow \\ binder_abortRet . loc . caller \longrightarrow \\ abort_MethodBinder \end{array} \right)$$

$getHeading_MethodBinder \hat{=}$

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

$isLandingGearDeployed_MethodBinder \hat{=}$

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

$stowLandingGear_MethodBinder \hat{=}$

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

$deployLandingGear_MethodBinder \hat{=}$

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

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

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

end

2.3 Locking

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

process *Threads* $\hat{=}$

$$\left(\begin{array}{l} \text{ThreadFW}(\text{InstrumentLandingSystemMonitorTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{SafeLandingHandlerTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{GroundDistanceMonitorTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{CommunicationsHandlerTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{ControlHandlerTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{AperiodicSimulatorTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{TakeOffFailureHandlerTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{LandingGearHandlerLandTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{EnvironmentMonitorTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{FlightSensorsMonitorTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{NavigationMonitorTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{ACModeChangerTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{BeginLandingHandlerTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{LandingGearHandlerTakeOffTID}, 5) \\ ||| \\ \text{ThreadFW}(\text{TakeOffMonitorTID}, 5) \end{array} \right)$$

process *Objects* $\hat{=}$

$$\left(\begin{array}{l} \text{ObjectFW}(\text{TakeOffMissionOID}) \\ ||| \\ \text{ObjectFW}(\text{LandMissionOID}) \end{array} \right)$$

process *Locking* $\hat{=}$ *Threads* \llbracket *ThreadSync* \rrbracket *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, ACModeChangerApp, ControlHandlerApp,
CommunicationsHandlerApp, EnvironmentMonitorApp, FlightSensorsMonitorApp,
AperiodicSimulatorApp, TakeOffMissionApp, LandingGearHandlerTakeOffApp, 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* $\hat{=}$

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

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

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

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

section Network parents *scj_prelude*,

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

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*, *SafeletChannels*, *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* •

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

• **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 $\hat{=}$

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

CleanupPhase $\hat{=}$

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

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 : $\mathbb{P} \mathbb{A}$
test : \mathbb{Z}
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'

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 *ACModeChangerApp* **parents** *TopLevelMissionSequencerChan*,
MissionId, *MissionIds*, *SchedulableId*, *SchedulableIds*, *ACModeChangerClass*, *MethodCallBindingChannels*

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

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

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

• $(\textit{Methods}) \triangle (\textit{end_sequencer_app} . \textit{ACModeChangerSID} \longrightarrow \mathbf{Skip})$

end

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

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

state *State*

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

state *State*

initial *Init*

State'

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

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

• **Skip**

end

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

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

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

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

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

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**

handleAsyncEvent $\hat{=}$

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

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

• (*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**

handleAsyncEvent $\hat{=}$

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

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

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

end

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

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

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

state *State*

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

• **Skip**

end

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

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

handleAsyncEvent $\hat{=}$

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

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

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

end

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

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

state <i>State</i> <i>event</i> : <i>AperiodicEventHandler</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, ObjectFWChan, ObjectIds*

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

State
this : **ref** *TakeOffMissionClass*

state *State*

Init
State'

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

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

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

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

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

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

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

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

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

$$\text{deployLandingGearSyncMeth} \hat{=} \left(\begin{array}{l} \text{deployLandingGearCall} . \text{TakeOffMissionMID} ? \text{caller} ? \text{thread} \longrightarrow \\ \left(\begin{array}{l} \text{startSyncMeth} . \text{TakeOffMissionOID} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{TakeOffMissionOID} . \text{thread} \longrightarrow \\ (\text{this} . \text{landingGearDeployed} := \mathbf{True}); \\ \text{endSyncMeth} . \text{TakeOffMissionOID} . \text{thread} \longrightarrow \\ \text{deployLandingGearRet} . \text{TakeOffMissionMID} . \text{caller} . \text{thread} \longrightarrow \end{array} \right) \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

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

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

state *State*

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

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

controllingMission : *MainMission*

abort : \mathbb{B}

landingGearDeployed : \mathbb{B}

state *State*

initial *Init*

State'

public *abort* $\hat{=}$

(*this* . *abort* := **True**)

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

(*ret* := *controllingMission*)

public *setControllingMission* $\hat{=}$

(*this* . *this* . *controllingMission* := *controllingMission*)

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

(*ret* := (\neg *abort* = **True**))

public *stowLandingGear* $\hat{=}$

(*this* . *landingGearDeployed* := **False**)

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

(*ret* := *landingGearDeployed* = **True**)

• **Skip**

end

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

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

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

channel *setControllingMissionCall* : *MissionID* \times *MissionID*
channel *setControllingMissionRet* : *MissionID*

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

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

handleAsyncEvent $\hat{=}$

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

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

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

end

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

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

handleAsyncEvent $\hat{=}$

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

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

\bullet (*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**

handleAsyncEvent $\hat{=}$

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

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

• (*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*,
CruiseMissionClass, *MethodCallBindingChannels*

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

State
this : **ref** *CruiseMissionClass*

state *State*

Init
State'

this' = **new** *CruiseMissionClass*()

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 $\hat{=}$

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

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

$$\left(\begin{array}{l} \textit{getControllingMissionCall} . \textit{CruiseMissionMID} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{getControllingMission}(); \\ \textit{getControllingMissionRet} . \textit{CruiseMissionMID} ! \textit{ret} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

```
section CruiseMissionClass parents scj_prelude, SchedulableId, SchedulableIds, SafeletChan  

, MethodCallBindingChannels
```

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

```
  state State _____  

  controllingMission : MainMission
```

```
state State
```

```
  initial Init _____  

  State '  

  _____
```

```
public getControllingMission  $\hat{=}$  var ret : MissionID •  

(ret := controllingMission)
```

```
• Skip
```

```
end
```

5.6 Schedulables of CruiseMission

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

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

handleAsyncEvent $\hat{=}$

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

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 $\hat{=}$

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

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*, *ObjectFWChan*, *ObjectIds*

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

State
this : **ref** *LandMissionClass*

state *State*

Init
State'

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

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

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

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

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

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

$$\text{abortMeth} \hat{=} \left(\begin{array}{l} \text{abortCall} . \text{LandMissionMID} \longrightarrow \\ \text{this} . \text{abort}(); \\ \text{abortRet} . \text{LandMissionMID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

$$\text{deployLandingGearSyncMeth} \hat{=} \left(\begin{array}{l} \text{deployLandingGearCall} . \text{LandMissionMID} ? \text{caller} ? \text{thread} \longrightarrow \\ \left(\begin{array}{l} \text{startSyncMeth} . \text{LandMissionOID} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{LandMissionOID} . \text{thread} \longrightarrow \\ (\text{this} . \text{landingGearDeployed} := \mathbf{True}) ; \\ \text{endSyncMeth} . \text{LandMissionOID} . \text{thread} \longrightarrow \\ \text{deployLandingGearRet} . \text{LandMissionMID} . \text{caller} . \text{thread} \longrightarrow \end{array} \right) \\ \mathbf{Skip} \end{array} \right)$$

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

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

end

section *LandMissionClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *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 '

public *stowLandingGear* $\hat{=}$

(*this* . *landingGearDeployed* := **False**)

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

(*ret* := *landingGearDeployed* = **True**)

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

(*ret* := *controllingMission*)

public *abort* $\hat{=}$

(*this* . *abort* := **True**)

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

(*ret* := (\neg *abort* = **True**))

• **Skip**

end

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

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

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

channel *isLandingGearDeployedCall* : *MissionID* \times *SchedulableID*

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

channel *getControllingMissionCall* : *MissionID*

channel *getControllingMissionRet* : *MissionID* \times *MissionID*

channel *abortCall* : *MissionID*

channel *abortRet* : *MissionID*

channel *cleanUpCall* : *MissionID*

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

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

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

5.8 Schedulables of LandMission

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

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

handleAsyncEvent $\hat{=}$

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

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 : $\mathbb{P} \mathbb{A} \bullet$ **begin**

handleAsyncEvent $\hat{=}$

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

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

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

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**

handleAsyncEvent $\hat{=}$

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

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

• (*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 $\hat{=}$

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

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

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

end