

aircraft

Tight Rope v0.6

4th November 2015

1 ID Files

1.1 MissionIds

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

MainMissionID : *MissionID*
TakeOffMissionID : *MissionID*
CruiseMissionID : *MissionID*
LandMissionID : *MissionID*

distinct(*nullMissionId*, *MainMissionID*, *TakeOffMissionID*,
CruiseMissionID, *LandMissionID*)

1.2 SchedulablesIds

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

MainMissionSequencerID : *SchedulableID*
ACModeChangerID : *SchedulableID*
EnvironmentMonitorID : *SchedulableID*
ControlHandlerID : *SchedulableID*
FlightSensorsMonitorID : *SchedulableID*
CommunicationsHandlerID : *SchedulableID*
AperiodicSimulatorID : *SchedulableID*
LandingGearHandlerTakeOffID : *SchedulableID*
TakeOffMonitorID : *SchedulableID*
TakeOffFailureHandlerID : *SchedulableID*
BeginLandingHandlerID : *SchedulableID*
NavigationMonitorID : *SchedulableID*
GroundDistanceMonitorID : *SchedulableID*
LandingGearHandlerLandID : *SchedulableID*
InstrumentLandingSystemMonitorID : *SchedulableID*
SafeLandingHandlerID : *SchedulableID*

distinct(*nullSequencerId*, *nullSchedulableId*, *MainMissionSequencerID*,
ACModeChangerID, *EnvironmentMonitorID*,
ControlHandlerID, *FlightSensorsMonitorID*,
CommunicationsHandlerID, *AperiodicSimulatorID*,
LandingGearHandlerTakeOffID, *TakeOffMonitorID*,
TakeOffFailureHandlerID, *BeginLandingHandlerID*,
NavigationMonitorID, *GroundDistanceMonitorID*,
LandingGearHandlerLandID, *InstrumentLandingSystemMonitorID*,
SafeLandingHandlerID)

1.3 ThreadIDs

section *ThreadIDs* **parents** *scj_prelude, GlobalTypes*

ACModeChangerThreadID : ThreadID
EnvironmentMonitorThreadID : ThreadID
ControlHandlerThreadID : ThreadID
FlightSensorsMonitorThreadID : ThreadID
CommunicationsHandlerThreadID : ThreadID
AperiodicSimulatorThreadID : ThreadID
LandingGearHandlerTakeOffThreadID : ThreadID
TakeOffMonitorThreadID : ThreadID
TakeOffFailureHandlerThreadID : ThreadID
BeginLandingHandlerThreadID : ThreadID
NavigationMonitorThreadID : ThreadID
GroundDistanceMonitorThreadID : ThreadID
LandingGearHandlerLandThreadID : ThreadID
InstrumentLandingSystemMonitorThreadID : ThreadID
SafeLandingHandlerThreadID : ThreadID

distinct(SafeletThreadId, nullThreadId,
ACModeChangerThreadID, EnvironmentMonitorThreadID,
ControlHandlerThreadID, FlightSensorsMonitorThreadID,
CommunicationsHandlerThreadID, AperiodicSimulatorThreadID,
LandingGearHandlerTakeOffThreadID, TakeOffMonitorThreadID,
TakeOffFailureHandlerThreadID, BeginLandingHandlerThreadID,
NavigationMonitorThreadID, GroundDistanceMonitorThreadID,
LandingGearHandlerLandThreadID, InstrumentLandingSystemMonitorThreadID,
SafeLandingHandlerThreadID)

1.4 ObjectIds

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

ACSafeletObjectID : *ObjectID*
MainMissionObjectID : *ObjectID*
ACModeChangerObjectID : *ObjectID*
EnvironmentMonitorObjectID : *ObjectID*
ControlHandlerObjectID : *ObjectID*
FlightSensorsMonitorObjectID : *ObjectID*
CommunicationsHandlerObjectID : *ObjectID*
AperiodicSimulatorObjectID : *ObjectID*
TakeOffMissionObjectID : *ObjectID*
LandingGearHandlerTakeOffObjectID : *ObjectID*
TakeOffMonitorObjectID : *ObjectID*
TakeOffFailureHandlerObjectID : *ObjectID*
CruiseMissionObjectID : *ObjectID*
BeginLandingHandlerObjectID : *ObjectID*
NavigationMonitorObjectID : *ObjectID*
LandMissionObjectID : *ObjectID*
GroundDistanceMonitorObjectID : *ObjectID*
LandingGearHandlerLandObjectID : *ObjectID*
InstrumentLandingSystemMonitorObjectID : *ObjectID*
SafeLandingHandlerObjectID : *ObjectID*

distinct(*ACSafeletObjectID*, *MainMissionObjectID*,
ACModeChangerObjectID, *EnvironmentMonitorObjectID*,
ControlHandlerObjectID, *FlightSensorsMonitorObjectID*,
CommunicationsHandlerObjectID, *AperiodicSimulatorObjectID*,
TakeOffMissionObjectID, *LandingGearHandlerTakeOffObjectID*,
TakeOffMonitorObjectID, *TakeOffFailureHandlerObjectID*,
CruiseMissionObjectID, *BeginLandingHandlerObjectID*,
NavigationMonitorObjectID, *LandMissionObjectID*,
GroundDistanceMonitorObjectID, *LandingGearHandlerLandObjectID*,
InstrumentLandingSystemMonitorObjectID, *SafeLandingHandlerObjectID*)

2 Network

section *NetworkChannels* **parents** *scj_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableChan, TopLevelMissionSequencerFWChan, FrameworkChan, SafeletChan*

channelset *TerminateSync* ==
 { *schedulables_terminated, schedulables_stopped, get_activeSchedulables* }

channelset *ControlTierSync* ==
 { *start_toplevel_sequencer, done_toplevel_sequencer, done_safeletFW* }

channelset *TierSync* ==
 { *start_mission ., done_mission ., done_safeletFW, done_toplevel_sequencer* }

channelset *MissionSync* ==
 { *done_safeletFW, done_toplevel_sequencer, register, signalTerminationCall, signalTerminationRet, activate_schedulables, done_schedulable, cleanupSchedulableCall, cleanupSchedulableRet* }

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

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

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

channelset *ObjectSync* ==
 { }

channelset *ThreadSync* ==
 { }

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

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

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

process *ControlTier* $\hat{=}$

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

process *Tier0* $\hat{=}$

$$\left(\begin{array}{c} \text{MissionFW}(\text{MainMission}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{c} \text{SchedulableMissionSequencerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \left(\begin{array}{c} \text{AperiodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{AperiodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \left(\begin{array}{c} \text{PeriodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}() \end{array} \right) \end{array} \right) \end{array} \right)$$

process *Tier1* $\hat{=}$

$$\left(\begin{array}{c} \text{MissionFW}(\text{TakeOffMission}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{c} \left(\begin{array}{c} \text{AperiodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \end{array} \right) \\ \text{AperiodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}() \end{array} \right) \\ \llbracket \text{ClusterSync} \rrbracket \\ \left(\begin{array}{c} \text{MissionFW}(\text{CruiseMission}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{c} \text{AperiodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \end{array} \right) \\ \text{PeriodicEventHandlerFW}() \end{array} \right) \\ \llbracket \text{ClusterSync} \rrbracket \\ \left(\begin{array}{c} \text{MissionFW}(\text{LandMission}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{c} \left(\begin{array}{c} \text{AperiodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \end{array} \right) \\ \text{AperiodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \left(\begin{array}{c} \text{PeriodicEventHandlerFW}() \\ \llbracket \text{SchedulablesSync} \rrbracket \end{array} \right) \\ \text{PeriodicEventHandlerFW}() \end{array} \right) \end{array} \right)$$

process *Framework* $\hat{=}$

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

process *Application* \triangleq

ACSafeletApp
|||
MainMissionSequencerApp
|||
MainMissionApp
|||
ACModeChangerApp(*MainMissionID*)
|||
ControlHandlerApp
|||
CommunicationsHandlerApp
|||
EnvironmentMonitorApp(*MainMissionID*)
|||
FlightSensorsMonitorApp(*MainMissionID*)
|||
AperiodicSimulatorApp(*EventID*)
|||
TakeOffMissionApp
|||
LandingGearHandlerTakeOffApp(*TakeOffMissionID*)
|||
TakeOffFailureHandlerApp(*TakeOffMissionID*)
|||
TakeOffMonitorApp(*TakeOffMissionID*, *LandingGearHandlerID*)
|||
CruiseMissionApp
|||
BeginLandingHandlerApp(*ControllingMissionID*)
|||
NavigationMonitorApp(*CruiseMissionID*)
|||
LandMissionApp
|||
LandingGearHandlerLandApp(*LandMissionID*)
|||
SafeLandingHandlerApp(*LandMissionID*)
|||
GroundDistanceMonitorApp(*LandMissionID*)
|||
InstrumentLandingSystemMonitorApp(*LandMissionID*)

Locking \triangleq

(
 ThreadFW(*ACModeChangerThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*EnvironmentMonitorThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*ControlHandlerThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*FlightSensorsMonitorThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*CommunicationsHandlerThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*AperiodicSimulatorThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*LandingGearHandlerTakeOffThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*TakeOffMonitorThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*TakeOffFailureHandlerThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*BeginLandingHandlerThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*NavigationMonitorThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*GroundDistanceMonitorThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*LandingGearHandlerLandThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*InstrumentLandingSystemMonitorThreadID*, *MinPriority*)
 [[*ThreadSync*]]
 ThreadFW(*SafeLandingHandlerThreadID*, *MinPriority*)
)

|||

(
 ObjectFW(*ACSafeletObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*MainMissionObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*ACModeChangerObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*EnvironmentMonitorObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*ControlHandlerObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*FlightSensorsMonitorObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*CommunicationsHandlerObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*AperiodicSimulatorObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*TakeOffMissionObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*LandingGearHandlerTakeOffObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*TakeOffMonitorObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*TakeOffFailureHandlerObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*CruiseMissionObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*BeginLandingHandlerObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*NavigationMonitorObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*LandMissionObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*GroundDistanceMonitorObjectID*)
 [[*ObjectSync*]]
 ObjectFW(*LandingGearHandlerLandObjectID*)
)

process $Program \hat{=} Framework \llbracket AppSync \rrbracket Application \llbracket LockingSync \rrbracket Locking$

3 Safelet

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

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

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

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

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

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

end

4 Top Level Mission Sequencer

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

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

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

state *State*

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

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

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

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

end

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

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

state *State*

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

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

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

• **Skip**

end

5 Missions

5.1 MainMission

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

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

State
this : **ref** *MainMissionClass*

state *State*

Init
State'

this' = **new** *MainMissionClass*()

InitializePhase $\hat{=}$

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

CleanupPhase $\hat{=}$

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

getAirSpeedMeth $\hat{=}$ **var** *ret* : double •

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

getAltitudeMeth $\hat{=}$ **var** *ret* : double •

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

getCabinPressureMeth $\hat{=}$ **var** *ret* : double •

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

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

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

$$\text{getHeadingMeth} \hat{=} \mathbf{var} \text{ ret} : \text{double} \bullet \left(\begin{array}{l} \text{getHeadingCall} . \text{MainMission} \longrightarrow \\ \text{ret} := \text{this} . \text{getHeading}(); \\ \text{getHeadingRet} . \text{MainMission} ! \text{ret} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

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

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

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

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

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

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

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

end

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

state *State*

ALTITUDE_READING_ON_GROUND : *double*
cabinPressure : *double*
emergencyOxygen : *double*
fuelRemaining : *double*
altitude : *double*
airSpeed : *double*
heading : *double*

state *State*

initial *Init*

State '

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Schedulables of MainMission

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

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

State

```

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

```

state *State*

Init

State'

```

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

```

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

changeToMeth $\hat{=}$
 $\left(\begin{array}{l} \textit{changeToCall} . \textit{ACModeChanger} ? \textit{newMode} \longrightarrow \\ (\textit{this} . \textit{currentMode} := \textit{newMode}); \\ \textit{changeToRet} . \textit{ACModeChanger} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

$$\begin{aligned}
& \text{getNextMissionMeth} \hat{=} \mathbf{var} \text{ ret} : \text{MissionID} \bullet \\
& \left(\begin{array}{l}
\text{getNextMissionCall} . \text{ACModeChanger} \longrightarrow \\
\left(\begin{array}{l}
\mathbf{if} (\text{modesLeft} = 3) \longrightarrow \\
\quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{ret} := \text{TakeOffMission} \end{array} \right) \\
\Box \neg (\text{modesLeft} = 3) \longrightarrow \\
\quad \mathbf{if} (\text{modesLeft} = 2) \longrightarrow \\
\quad \quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{ret} := \text{CruiseMission} \end{array} \right) \\
\Box \neg (\text{modesLeft} = 2) \longrightarrow \\
\quad \mathbf{if} (\text{modesLeft} = 1) \longrightarrow \\
\quad \quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{ret} := \text{LandMission} \end{array} \right) \\
\Box \neg (\text{modesLeft} = 1) \longrightarrow \\
\quad \quad (\text{ret} := \text{nullMissionId}) \\
\mathbf{fi} \\
\mathbf{fi} \\
\mathbf{fi}
\end{array} \right) ; \\
\text{getNextMissionRet} . \text{ACModeChanger} ! \text{ret} \longrightarrow \\
\mathbf{Skip}
\end{array} \right)
\end{aligned}$$

$$\begin{aligned}
& \text{advanceModeSyncMeth} \hat{=} \\
& \left(\begin{array}{l}
\text{advanceModeCall} . \text{ACModeChanger} ? \text{thread} \longrightarrow \\
\left(\begin{array}{l}
\text{startSyncMeth} . \text{ACModeChangerObject} . \text{thread} \longrightarrow \\
\text{lockAcquired} . \text{ACModeChangerObject} . \text{thread} \longrightarrow \\
\left(\begin{array}{l}
\mathbf{Skip}; \\
\mathbf{if} (\text{modesLeft} = 3) \longrightarrow \\
\quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{changeTo}(\text{launchMode}) \end{array} \right) \\
\Box \neg (\text{modesLeft} = 3) \longrightarrow \\
\quad \mathbf{if} (\text{modesLeft} = 2) \longrightarrow \\
\quad \quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{changeTo}(\text{cruiseMode}) \end{array} \right) \\
\Box \neg (\text{modesLeft} = 2) \longrightarrow \\
\quad \mathbf{if} (\text{modesLeft} = 1) \longrightarrow \\
\quad \quad \left(\begin{array}{l} \text{modesLeft} := \text{modesLeft} - 1; \\ \text{changeTo}(\text{landMode}) \end{array} \right) \\
\Box \neg (\text{modesLeft} = 1) \longrightarrow \\
\quad \quad (\text{changeTo}(\mathbf{null})) \\
\mathbf{fi} \\
\mathbf{fi} \\
\mathbf{fi}
\end{array} \right) ; \\
\text{endSyncMeth} . \text{ACModeChangerObject} . \text{thread} \longrightarrow \\
\text{advanceModeRet} . \text{ACModeChanger} . \text{thread} \longrightarrow \\
\mathbf{Skip}
\end{array} \right)
\end{aligned}$$

$$\begin{aligned}
& \text{Methods} \hat{=} \\
& \left(\begin{array}{l}
\text{GetNextMission} \\
\Box \\
\text{changeToMeth} \\
\Box \\
\text{getNextMissionMeth} \\
\Box \\
\text{advanceModeSyncMeth}
\end{array} \right) ; \text{Methods}
\end{aligned}$$

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

end

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

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

channel *getNextMissionCall* : *SchedulableID*
channel *getNextMissionRet* : *SchedulableID* × *MissionID*

channel *advanceModeCall* : *SchedulableID* × *ThreadID*
channel *advanceModeRet* : *SchedulableID* × *ThreadID*

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

process *EnvironmentMonitorApp* $\hat{=}$
controllingMission : *MainMission* • **begin**

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

process *FlightSensorsMonitorApp* $\hat{=}$
controllingMission : *MainMission* • **begin**

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

process *AperiodicSimulatorApp* $\hat{=}$
event : *AperiodicEventHandler* • **begin**

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

```
end
```


5.3 TakeOffMission

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

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

State
this : **ref** *TakeOffMissionClass*

state *State*

Init
State'

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

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

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

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

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

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

cleanUpMeth $\hat{=}$ **var** *ret* : \mathbb{B} •
 $\left(\begin{array}{l} \textit{cleanUpCall} . \textit{TakeOffMission} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{cleanUp}(); \\ \textit{cleanUpRet} . \textit{TakeOffMission} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

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

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

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

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

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

end

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

state *State*

SAFE_AIRSPPEED_THRESHOLD : *double*

TAKEOFF_ALTITUDE : *double*

abort : \mathbb{B}

landingGearDeployed : \mathbb{B}

state *State*

initial *Init*

State'

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} •
(**Skip**;
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*
channel *abortRet* : *MissionID*

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*
channel *stowLandingGearRet* : *MissionID*

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

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

5.4 Schedulables of TakeOffMission

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

process *TakeOffFailureHandlerApp* $\hat{=}$
takeoffMission : *TakeOffMission* • **begin**

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

state *State*

<i>Init</i> <i>State'</i>
<i>threshold' = threshold</i>

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

process *TakeOffMonitorApp* $\hat{=}$
takeoffMission : *TakeOffMission*,
landingGearHandler : *AperiodicEventHandler* • **begin**

<i>State</i> <i>takeOffAltitude</i> : double

state *State*

<i>Init</i> <i>State</i> '
<i>takeOffAltitude</i> ' = <i>takeOffAltitude</i>

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

5.5 CruiseMission

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

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

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

state *State*

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

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

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

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

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* . *CruiseMission* \longrightarrow **Skip**)

end

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

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

• **Skip**

end

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

channel *getControllingMissionCall* : *MissionID*

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

5.6 Schedulables of CruiseMission

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

5.7 LandMission

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

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

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

state *State*

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

InitializePhase $\hat{=}$

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

CleanupPhase $\hat{=}$

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

stowLandingGearMeth $\hat{=}$

$$\left(\begin{array}{l} \text{stowLandingGearCall} . \text{LandMission} \longrightarrow \\ \text{this} . \text{stowLandingGear}(); \\ \text{stowLandingGearRet} . \text{LandMission} \longrightarrow \\ \text{Skip} \end{array} \right)$$

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

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

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

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

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

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

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

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

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

end

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

state *State*

SAFE_LANDING_ALTITUDE : *double*

abort : \mathbb{B}

landingGearDeployed : \mathbb{B}

state *State*

initial *Init*

State'

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} •

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

• **Skip**

end

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

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

channel *isLandingGearDeployedCall* : *MissionID*
channel *isLandingGearDeployedRet* : *MissionID* \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 *ThreadID*
channel *deployLandingGearRet* : *MissionID* \times *ThreadID*

5.8 Schedulables of LandMission

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

process *SafeLandingHandlerApp* $\hat{=}$
landMission : *LandMission* • **begin**

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

state *State*

<i>Init</i> <i>State'</i>
<i>threshold' = threshold</i>

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

process *GroundDistanceMonitorApp* $\hat{=}$
mission : *LandMission* • **begin**

<i>State</i> <i>readingOnGround</i> : double

state *State*

<i>Init</i> <i>State</i> '
<i>readingOnGround</i> ' =

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end

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

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

handlerAsyncEvent $\hat{=}$

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

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

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

end

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

- **Skip**

end