aircraft

Tight Rope v0.6

2nd November 2015

1 ID Files

1.1 MissionIds

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

$$
\begin{array}{|l}
MainMissionID : MissionID \\
TakeOffMissionID : MissionID \\
CruiseMissionID : MissionID \\
LandMissionID : MissionID \\
\hline
distinct\langle nullMissionId, MainMissionID, TakeOffMissionID, \\
CruiseMissionID, LandMissionID\rangle
\end{array}
$$

1.2 SchedulablesIds

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

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

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

1.3 ThreadIds

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

$$
\begin{array}{l}
ACModeChangerThreadID : ThreadID \\
EnvironmentMonitorThreadID : ThreadID \\
ControlHandlerThreadID : ThreadID \\
FlightSensorsMonitorThreadID : ThreadID \\
CommunicationsHandlerThreadID : ThreadID \\
AperiodicSimulatorThreadID : ThreadID \\
LandingGearHandlerTakeOffThreadID : ThreadID \\
TakeOffMonitorThreadID : ThreadID \\
TakeOffFailureHandlerThreadID : ThreadID \\
BeginLandingHandlerThreadID : ThreadID \\
NavigationMonitorThreadID : ThreadID \\
GroundDistanceMonitorThreadID : ThreadID \\
LandingGearHandlerLandThreadID : ThreadID \\
InstrumentLandingSystemMonitorThreadID : ThreadID \\
SafeLandingHandlerThreadID : ThreadID
\end{array}
$$

$$
\begin{array}{l}
distinct\langle SafeletThreadId, nullThreadId, \\
ACModeChangerThreadID, EnvironmentMonitorThreadID, \\
ControlHandlerThreadID, FlightSensorsMonitorThreadID, \\
CommunicationsHandlerThreadID, AperiodicSimulatorThreadID, \\
LandingGearHandlerTakeOffThreadID, TakeOffMonitorThreadID, \\
TakeOffFailureHandlerThreadID, BeginLandingHandlerThreadID, \\
NavigationMonitorThreadID, GroundDistanceMonitorThreadID, \\
LandingGearHandlerLandThreadID, InstrumentLandingSystemMonitorThreadID, \\
SafeLandingHandlerThreadID\rangle
\end{array}
$$

1.4 ObjectIds

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

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

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

2 Network

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

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

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

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

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

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

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

channelset *AppSync* ==
$\bigcup$ {*SafeltAppSync*, *MissionSequencerAppSync*, *MissionAppSync*,
MTAppSync, *OSEHSync*, *APEHSync*,
{| *getSequencer*, *end_mission_app*, *end_managedThread_app*,
setCeilingPriority, *requestTerminationCall*, *requestTerminationRet*, *terminationPendingCall*,
terminationPendingRet, *handleAsyncEventCall*, *handleAsyncEventRet* |}}

channelset *ObjectSync* ==
{| |}

channelset *ThreadSync* ==
{| |}

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

channelset *Tier*0*Sync* ==
{| *done_toplevel_sequencer*, *done_safeletFW*,
start_mission . , *done_mission* . ,
initializeRet . , *requestTermination* . . ,
start_mission . , *done_mission* . ,
initializeRet . , *requestTermination* . . ,
start_mission . , *done_mission* . ,
initializeRet . , *requestTermination* . . |}

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

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

process $Tier0 \mathrel{\widehat=}$
$$\left(\begin{array}{l} MissionFW(MainMission) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} SchedulableMissionSequencerFW() \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} AperiodicEventHandlerFW() \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW() \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} PeriodicEventHandlerFW() \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW() \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW() \end{array}\right) \end{array}\right) \end{array}\right)$$

process $Tier1 \mathrel{\widehat=}$
$$\left(\begin{array}{l} MissionFW(TakeOffMission) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW() \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW() \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW() \end{array}\right) \end{array}\right)$$
$$[\![ClusterSync]\!]$$
$$\left(\begin{array}{l} MissionFW(CruiseMission) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} AperiodicEventHandlerFW() \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW() \end{array}\right) \end{array}\right)$$
$$[\![ClusterSync]\!]$$
$$\left(\begin{array}{l} MissionFW(LandMission) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} \left(\begin{array}{l} AperiodicEventHandlerFW() \\ \quad [\![SchedulablesSync]\!] \\ AperiodicEventHandlerFW() \end{array}\right) \\ \quad [\![SchedulablesSync]\!] \\ \left(\begin{array}{l} PeriodicEventHandlerFW() \\ \quad [\![SchedulablesSync]\!] \\ PeriodicEventHandlerFW() \end{array}\right) \end{array}\right) \end{array}\right)$$

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

$$
\begin{array}{l}
\textbf{process}\ Application\ \widehat{=} \\
\left(\begin{array}{l}
ACSafeletApp \\
|||\ MainMissionSequencerApp \\
|||\ MainMissionApp \\
|||\ ACModeChangerApp(MainMission) \\
|||\ ControlHandlerApp \\
|||\ CommunicationsHandlerApp \\
|||\ EnvironmentMonitorApp(MainMission) \\
|||\ FlightSensorsMonitorApp(MainMission) \\
|||\ AperiodicSimulatorApp(AperiodicEventHandler) \\
|||\ TakeOffMissionApp \\
|||\ LandingGearHandlerTakeOffApp(TakeOffMission) \\
|||\ TakeOffFailureHandlerApp(TakeOffMission) \\
|||\ TakeOffMonitorApp(TakeOffMission, AperiodicEventHandler) \\
|||\ CruiseMissionApp \\
|||\ BeginLandingHandlerApp(Mission) \\
|||\ NavigationMonitorApp(CruiseMission) \\
|||\ LandMissionApp \\
|||\ LandingGearHandlerLandApp(LandMission) \\
|||\ SafeLandingHandlerApp(LandMission) \\
|||\ GroundDistanceMonitorApp(LandMission) \\
|||\ InstrumentLandingSystemMonitorApp(LandMission)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Locking \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
\left(\begin{array}{l}
ThreadFW(LandingGearHandlerLandThread, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(SafeLandingHandlerThread, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(GroundDistanceMonitorThread, MinPriority) \\
\quad [\![ThreadSync]\!] \\
ThreadFW(InstrumentLandingSystemMonitorThread, MinPriority)
\end{array}\right) \\
|||
\left(\begin{array}{l}
ObjectFW(ACSafeletObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(MainMissionObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(ACModeChangerObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(EnvironmentMonitorObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(ControlHandlerObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(FlightSensorsMonitorObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(CommunicationsHandlerObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(AperiodicSimulatorObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(TakeOffMissionObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(LandingGearHandlerTakeOffObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(TakeOffMonitorObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(TakeOffFailureHandlerObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(CruiseMissionObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(BeginLandingHandlerObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(NavigationMonitorObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(LandMissionObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(GroundDistanceMonitorObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(LandingGearHandlerLandObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(InstrumentLandingSystemMonitorObjectIDObject) \\
\quad [\![ObjectSync]\!] \\
ObjectFW(SafeLandingHandlerObjectIDObject)
\end{array}\right)
\end{array}\right)
\end{array}
$$

process $Program \mathrel{\widehat{=}} Framework \; [\![\; AppSync \;]\!] \; Application \; [\![\; LockingSync \;]\!] \; Locking$

3 Safelet

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

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

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

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

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

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

end

4 Top Level Mission Sequencer

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

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

State

$this : \textbf{ref}\ MainMissionSequencerClass$

state *State*

Init

$State'$

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

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

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

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

end

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

state *State*

$$returnedMission : \mathbb{B}$$

state *State*

initial *Init*

$$State'$$

$$returnedMission' = false$$

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

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

• **Skip**

end

5 Missions

5.1 MainMission

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

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

State

$this : \mathbf{ref}\ MainMissionClass$

state *State*

Init

$State'$

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

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

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

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

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

$$getCabinPressureMeth \mathrel{\widehat=} \mathbf{var}\ ret : double \bullet \left(\begin{array}{l} getCabinPressureCall\,.\,MainMission \longrightarrow \\ ret := this\,.\,getCabinPressure(); \\ getCabinPressureRet\,.\,MainMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

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

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

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

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

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

$$setCabinPressureMeth \mathrel{\widehat=} \left(\begin{array}{l} setCabinPressureCall\,.\,MainMission\,?\,cabinPressure \longrightarrow \\ this\,.\,setCabinPressure(cabinPressure); \\ setCabinPressureRet\,.\,MainMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$setEmergencyOxygenMeth \mathrel{\widehat=} \left(\begin{array}{l} setEmergencyOxygenCall\,.\,MainMission\,?\,emergencyOxygen \longrightarrow \\ this\,.\,setEmergencyOxygen(emergencyOxygen); \\ setEmergencyOxygenRet\,.\,MainMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$setFuelRemainingMeth \mathrel{\widehat=} \left(\begin{array}{l} setFuelRemainingCall\,.\,MainMission\,?\,fuelRemaining \longrightarrow \\ this\,.\,setFuelRemaining(fuelRemaining); \\ setFuelRemainingRet\,.\,MainMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$setHeadingMeth \mathrel{\widehat=} \left(\begin{array}{l} setHeadingCall\,.\,MainMission\,?\,heading \longrightarrow \\ this\,.\,setHeading(heading); \\ setHeadingRet\,.\,MainMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

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

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

end

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

state *State*

$ALTITUDE\_READING\_ON\_GROUND : double$
$cabinPressure : double$
$emergencyOxygen : double$
$fuelRemaining : double$
$altitude : double$
$airSpeed : double$
$heading : double$

state *State*

initial *Init*

$State'$

$ALTITUDE\_READING\_ON\_GROUND' = 0.0$

public $getAirSpeed \mathrel{\widehat{=}} \textbf{var}\ ret : double \bullet$
$(ret := airSpeed)$

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

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

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

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

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

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

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

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

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

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

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

- **Skip**

end

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Schedulables of MainMission

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

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

State

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

state *State*

Init

$State'$

$modesLeft' = 3$
ref $currentModeClass' = \mathbf{new}\ ModeClass()$
ref $launchModeClass' = \mathbf{new}\ ModeClass()$
ref $cruiseModeClass' = \mathbf{new}\ ModeClass()$
ref $landModeClass' = \mathbf{new}\ ModeClass()$

$GetNextMission \mathrel{\widehat=} \mathbf{var}\ ret : MissionID \bullet$
$\begin{pmatrix} getNextMissionCall\,.\,ACModeChanger \longrightarrow \\ ret := this\,.\,getNextMission(); \\ getNextMissionRet\,.\,ACModeChanger\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$changeToMeth \mathrel{\widehat=}$
$\begin{pmatrix} changeToCall\,.\,ACModeChanger\,?\,newMode \longrightarrow \\ (this\,.\,currentMode := newMode)\,; \\ changeToRet\,.\,ACModeChanger \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$$
\begin{array}{l}
getNextMissionMeth \mathrel{\widehat=} \textbf{var}\ ret : MissionID \bullet \\
\left(\begin{array}{l}
getNextMissionCall\,.\,ACModeChanger \longrightarrow \\
\left(\begin{array}{l}
\textbf{if}\ (modesLeft = 3) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := TakeOffMission \end{array}\right) \\
[]\ \neg\ (modesLeft = 3) \longrightarrow \\
\quad \textbf{if}\ (modesLeft = 2) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := CruiseMission \end{array}\right) \\
[]\ \neg\ (modesLeft = 2) \longrightarrow \\
\quad \textbf{if}\ (modesLeft = 1) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ ret := LandMission \end{array}\right) \\
[]\ \neg\ (modesLeft = 1) \longrightarrow \\
\quad \big(ret := nullMissionId\big) \\
\textbf{fi} \\
\textbf{fi} \\
\textbf{fi}
\end{array}\right) \\
getNextMissionRet\,.\,ACModeChanger\,!\,ret \longrightarrow \\
\textbf{Skip}
\end{array}\right)
\end{array}
$$

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

$$
\begin{array}{l}
Methods \mathrel{\widehat=} \\
\left(\begin{array}{l}
GetNextMission \\
\Box \\
changeToMeth \\
\Box \\
getNextMissionMeth \\
\Box \\
advanceModeSyncMeth
\end{array}\right)\ ;\ Methods
\end{array}
$$

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

end

class $ACModeChangerClass \mathrel{\widehat{=}}$ **begin**

$$
\begin{array}{l}
\circ\ \textbf{sync}\ advanceMode\ \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
; \\
\textbf{if}\ (modesLeft = 3) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(launchMode) \end{array}\right) \\
[]\ \neg\ (modesLeft = 3) \longrightarrow \\
\quad \textbf{if}\ (modesLeft = 2) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(cruiseMode) \end{array}\right) \\
[]\ \neg\ (modesLeft = 2) \longrightarrow \\
\quad \textbf{if}\ (modesLeft = 1) \longrightarrow \\
\quad \left(\begin{array}{l} modesLeft := modesLeft - 1; \\ changeTo(landMode) \end{array}\right) \\
[]\ \neg\ (modesLeft = 1) \longrightarrow \\
\quad \big(changeTo(\textbf{null})\big) \\
\textbf{fi} \\
\textbf{fi} \\
\textbf{fi}
\end{array}\right)
\end{array}
$$

- **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* $\widehat{=}$ **begin**

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

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

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

end

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

- **Skip**

end

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

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

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

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

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

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

end

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

- **Skip**

end

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

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

process $EnvironmentMonitorApp \mathrel{\widehat=}$
$controllingMission : MainMission \bullet$ **begin**

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

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

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

end

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

- **Skip**

end

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

process $FlightSensorsMonitorApp \mathrel{\widehat=}$
$controllingMission : MainMission \bullet$ **begin**

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

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

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

end

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

- **Skip**

end

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

process $AperiodicSimulatorApp \mathrel{\widehat=}$
$\quad event : AperiodicEventHandler \bullet$ **begin**

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

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

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

end

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

- **Skip**

end

5.3 TakeOffMission

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

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

State

$this : \mathbf{ref}\ TakeOffMissionClass$

state *State*

Init

$State'$

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

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

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

$abortMeth \mathrel{\widehat=}$
$\left(\begin{array}{l} abortCall\,.\,TakeOffMission \longrightarrow \\ this\,.\,abort(); \\ abortRet\,.\,TakeOffMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$

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

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

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

$$
\begin{array}{l}
stowLandingGearMeth \mathrel{\widehat=} \\
\left(\begin{array}{l}
stowLandingGearCall\,.\,TakeOffMission \longrightarrow \\
this\,.\,stowLandingGear();\\
stowLandingGearRet\,.\,TakeOffMission \longrightarrow \\
\textbf{Skip}
\end{array}\right)
\end{array}
$$

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

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

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

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

end

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

state *State*

$SAFE\_AIRSPEED\_THRESHOLD : double$
$TAKEOFF\_ALTITUDE : double$
$abort : \mathbb{B}$
$landingGearDeployed : \mathbb{B}$

state *State*

initial *Init*

$State'$

$SAFE\_AIRSPEED\_THRESHOLD' = 10.0$
$TAKEOFF\_ALTITUDE' = 10.0$
$abort' = false$

public *abort* $\widehat{=}$
$\big(this\,.\,abort := true\big)$

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

public *setControllingMission* $\widehat{=}$
$\big(this\,.\,this.controllingMission := controllingMission\big)$

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

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

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

• **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 $LandingGearHandlerTakeOffApp \mathrel{\widehat=}$
$mission : TakeOffMission \bullet$ **begin**

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

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

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

$\bullet\ (Methods) \triangle (end\_aperiodic\_app\ .\ LandingGearHandlerTakeOff \longrightarrow$ **Skip**$)$

end

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

- **Skip**

end

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

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

process $TakeOffFailureHandlerApp \mathrel{\widehat=}$
$takeoffMission : TakeOffMission \bullet$ **begin**

State

$threshold : double$

state *State*

Init

$State'$

$threshold' = threshold$

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

var $currentSpeed : double \bullet currentSpeed := getAirSpeed$
if $(currentSpeed < threshold) \longrightarrow$
$\Big($ **Skip**;
$abortCall\,.\,takeoffMission \longrightarrow$
$abortRet\,.\,takeoffMission \longrightarrow$
Skip;
$requestTerminationCall\,.\,takeoffMission \longrightarrow$
$requestTerminationRet\,.\,takeoffMission\,?\,requestTermination \longrightarrow$
Skip $\Big)$
$[] \neg (currentSpeed < threshold) \longrightarrow$
$\big($**Skip**$\big)$
fi Skip $\Big)$;
$handleAsyncEventRet\,.\,TakeOffFailureHandler \longrightarrow$
Skip $\Big)$

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

$\bullet\ (Init\ ;\ Methods) \triangle (end\_aperiodic\_app\,.\,TakeOffFailureHandler \longrightarrow$ **Skip**$)$

end

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

- **Skip**

end

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

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

process $TakeOffMonitorApp \mathrel{\widehat=}$
$takeoffMission : TakeOffMission,$
$landingGearHandler : AperiodicEventHandler \bullet$ **begin**

State

$takeOffAltitude : double$

state *State*

Init

$State'$

$takeOffAltitude' = takeOffAltitude$

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

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

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

$\bullet\ (Init\ ;\ Methods) \triangle (end\_periodic\_app\,.\,TakeOffMonitor \longrightarrow$ **Skip**$)$

end

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

- **Skip**

end

5.5 CruiseMission

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

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

State

$this : \textbf{ref}\ CruiseMissionClass$

state *State*

Init

$State'$

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

$$InitializePhase \mathrel{\widehat=} \begin{pmatrix} initializeCall\,.\,CruiseMission \longrightarrow \\ register\,!\,BeginLandingHandler\,!\,CruiseMission \longrightarrow \\ register\,!\,NavigationMonitor\,!\,CruiseMission \longrightarrow \\ initializeRet\,.\,CruiseMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$CleanupPhase \mathrel{\widehat=} \begin{pmatrix} cleanupMissionCall\,.\,CruiseMission \longrightarrow \\ cleanupMissionRet\,.\,CruiseMission\,!\,\textbf{True} \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$getControllingMissionMeth \mathrel{\widehat=} \textbf{var}\ ret : MissionID \bullet \begin{pmatrix} getControllingMissionCall\,.\,CruiseMission \longrightarrow \\ ret := this\,.\,getControllingMission(); \\ getControllingMissionRet\,.\,CruiseMission\,!\,ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$Methods \mathrel{\widehat=} \begin{pmatrix} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ getControllingMissionMeth \end{pmatrix} ;\ Methods$$

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

end

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

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

$\bullet$ **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 $BeginLandingHandlerApp \mathrel{\widehat=}$
$controllingMission : Mission \bullet$ **begin**

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

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

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

end

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

- **Skip**

end

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

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

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

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

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

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

end

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

- **Skip**

end

5.7 LandMission

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

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

State

$this : \mathbf{ref}\ LandMissionClass$

state *State*

Init

$State'$

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

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

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

$stowLandingGearMeth \mathrel{\widehat=}$
$\left(\begin{array}{l} stowLandingGearCall\,.\,LandMission \longrightarrow \\ this\,.\,stowLandingGear(); \\ stowLandingGearRet\,.\,LandMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$

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

$getControllingMissionMeth \mathrel{\widehat=} \mathbf{var}\ ret : MissionID \bullet$
$\left(\begin{array}{l} getControllingMissionCall\,.\,LandMission \longrightarrow \\ ret := this\,.\,getControllingMission(); \\ getControllingMissionRet\,.\,LandMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$

$$abortMeth \mathrel{\widehat=} \left(\begin{array}{l} abortCall\,.\,LandMission \longrightarrow \\ this\,.\,abort(); \\ abortRet\,.\,LandMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

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

$$deployLandingGearSyncMeth \mathrel{\widehat=} \left(\begin{array}{l} deployLandingGearCall\,.\,LandMission\,?\,thread \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ lockAcquired\,.\,LandMissionObject\,.\,thread \longrightarrow \\ \big(this\,.\,landingGearDeployed := true\big)\,; \\ endSyncMeth\,.\,LandMissionObject\,.\,thread \longrightarrow \\ deployLandingGearRet\,.\,LandMission\,.\,thread \longrightarrow \\ \textbf{Skip} \end{array}\right) \end{array}\right)$$

$$Methods \mathrel{\widehat=} \left(\begin{array}{l} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ stowLandingGearMeth \\ \Box \\ isLandingGearDeployedMeth \\ \Box \\ getControllingMissionMeth \\ \Box \\ abortMeth \\ \Box \\ cleanUpMeth \\ \Box \\ deployLandingGearSyncMeth \end{array}\right) ;\ Methods$$

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

end

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

state *State*

$SAFE\_LANDING\_ALTITUDE : double$
$abort : \mathbb{B}$
$landingGearDeployed : \mathbb{B}$

state *State*

initial *Init*

$State'$

$SAFE\_LANDING\_ALTITUDE' = 10.0$
$abort' = false$

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

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

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

public *abort* $\widehat{=}$
$\big(this\,.\,abort := true\big)$

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

• **Skip**

end

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

channel *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 $LandingGearHandlerLandApp \mathrel{\widehat=}$
$mission : LandMission \bullet$ **begin**

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

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

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

$\bullet$ $(Methods) \triangle (end\_aperiodic\_app\ .\ LandingGearHandlerLand \longrightarrow$ **Skip**$)$

end

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

- **Skip**

end

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

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

process $SafeLandingHandlerApp \mathrel{\widehat=}$
$landMission : LandMission \bullet$ **begin**

State

$threshold : double$

state *State*

Init

$State'$

$threshold' = threshold$

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

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

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

$\bullet$ $(Init$; $Methods) \triangle (end\_aperiodic\_app \,.\, SafeLandingHandler \longrightarrow$ **Skip**$)$

end

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

- **Skip**

end

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

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

process $GroundDistanceMonitorApp \mathrel{\widehat=}$
$mission : LandMission \bullet$ **begin**

State

$readingOnGround : double$

state *State*

Init

$State'$

$readingOnGround' =$

$handlerAsyncEvent \mathrel{\widehat=}$

$handleAsyncEventCall \,.\, GroundDistanceMonitor \longrightarrow$
Skip;
$getControllingMissionCall \,.\, mission.getControllingMission() \longrightarrow$
$getControllingMissionRet \,.\, mission.getControllingMission() ? getControllingMission \longrightarrow$

var $distance : double \bullet distance := getAltitude$
if $(distance = readingOnGround) \longrightarrow$
Skip;
$requestTerminationCall \,.\, mission \longrightarrow$
$requestTerminationRet \,.\, mission ? requestTermination \longrightarrow$
Skip
$[] \neg (distance = readingOnGround) \longrightarrow$ **Skip**
fi ;
Skip
;
$handleAsyncEventRet \,.\, GroundDistanceMonitor \longrightarrow$
Skip

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

$\bullet$ $(Init$; $Methods) \triangle (end\_periodic\_app \,.\, GroundDistanceMonitor \longrightarrow$ **Skip**$)$

end

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

- **Skip**

end

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

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

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

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

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

end

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

- **Skip**

end