

mission1

Tight Rope v0.88

1st March 2017

1 ID Files

1.1 MissionIds

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

<i>MyMissionMID</i> : <i>MissionID</i>
--

<i>distinct</i> $\langle \text{nullMissionId}, \text{MyMissionMID} \rangle$

1.2 SchedulablesIds

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

mainSequencerSID : *SchedulableID*

APEHSID : *SchedulableID*

PEHSID : *SchedulableID*

distinct (*nullSequencerId*, *nullSchedulableId*, *mainSequencerSID*,
APEHSID, *PEHSID*)

1.3 Non-Paradigm Objects

1.4 ThreadIds

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

SafeletTid : *ThreadID*
nullThreadId : *ThreadID*

distinct(*SafeletTid*, *nullThreadId*)

1.5 ObjectIds

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

$distinct \langle \rangle$

2 Network

2.1 Network Channel Sets

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

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

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

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

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

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

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

channelset *SafeltAppSync* $\hat{=}$
{ *getSequencerCall, getSequencerRet, initializeApplicationCall, initializeApplicationRet, end_safelet_app* }

channelset *MissionSequencerAppSync* ==
{ *getNextMissionCall, getNextMissionRet, end_sequencer_app* }

channelset *MissionAppSync* ==
{ *initializeCall, register, initializeRet, cleanupMissionCall, cleanupMissionRet* }

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

channelset *ThreadSync* ==
{ *raise_thread_priority, lower_thread_priority, isInterruptedCall, isInterruptedRet, get_priorityLevel* }

channelset *LockingSync* ==
{ *lockAcquired, startSyncMeth, endSyncMeth, waitCall, waitRet, notify, isInterruptedCall, isInterruptedRet, interruptedCall, interruptedRet, done_toplevel_sequencer, get_priorityLevel* }

2.2 Locking

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

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

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

process *Locking* $\hat{=}$ (*Threads* \llbracket *ThreadSync* \rrbracket *Objects*) \triangle (*done_toplevel_sequencer* \longrightarrow **Skip**)

2.3 Program

section *Program* **parents** *scj_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, MissionFW, SafeletFW, TopLevelMissionSequencerFW, NetworkChannels, ManagedThreadFW, SchedulableMissionSequencerFW, PeriodicEventHandlerFW, OneShotEventHandlerFW, AperiodicEventHandlerFW, ObjectFW, ThreadFW, MyAppApp, mainSequencerApp, MyMissionApp, APEHApp, PEHApp*

process *ControlTier* $\hat{=}$

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

process *Tier0* $\hat{=}$

$$\left(\begin{array}{l} \text{MissionFW}(\text{MyMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{AperiodicEventHandlerFW}(\text{APEHID}, \text{aperiodic}, (\text{time}(5, 0), \text{nullSchedulableId})) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{PeriodicEventHandlerFW}(\text{PEHID}, (\text{time}(60, 0), \text{time}(5, 0), \text{NULL}, \text{nullSchedulableId})) \end{array} \right) \end{array} \right)$$

process *Framework* $\hat{=}$

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

process *Application* $\hat{=}$

$$\left(\begin{array}{l} \text{MyAppApp} \\ ||| \\ \text{mainSequencerApp} \\ ||| \\ \text{MyMissionApp} \\ ||| \\ \text{APEHApp}(\text{MyMissionID}) \\ ||| \\ \text{PEHApp}(\text{apehID}) \end{array} \right)$$

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

3 Safelet

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

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

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

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

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 *mainSequencerApp* **parents** *TopLevelMissionSequencerChan*,
MissionId, *MissionIds*, *SchedulableId*, *SchedulableIds*, *mainSequencerClass*, *MethodCallBindingChannels*

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

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

state *State*

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

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

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

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

end

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

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

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

state *State*

initial <i>Init</i> <i>State</i> '
<i>notReleased</i> ' = <i>true</i>

protected *getNextMission* $\hat{=}$

$$\left(\begin{array}{l} \text{if } \textit{notReleased} \longrightarrow \\ \quad \left(\textit{notReleased} := \mathbf{False}; \right. \\ \quad \left. \textit{ret} := \textit{MyMissionMID} \right) \\ \quad \square \neg \textit{notReleased} \longrightarrow \\ \quad \quad \left(\textit{ret} := \textit{nullMissionId} \right) \\ \text{fi} \end{array} \right)$$

• **Skip**

end

5 Missions

5.1 MyMission

section *MyMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
 SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *MyMissionMethChan*
 , *MethodCallBindingChannels*

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

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

state *State*

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

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

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

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

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

end

5.2 Schedulables of MyMission

section *APEHApp* **parents** *AperiodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*

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

handleAsyncEvent $\hat{=}$

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

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

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

end

section *PEHApp* **parents** *PeriodicEventHandlerChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*

process *PEHApp* $\hat{=}$
 apeh : *SchedulableID* • **begin**

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

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

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

end