

producerConsumer

Tight Rope v0.88

5th March 2017

1 ID Files

1.1 MissionIds

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

$PCMissionMID : MissionID$
$distinct\langle nullMissionId, PCMissionMID \rangle$

1.2 SchedulablesIds

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

PCMissionSequencerSID : SchedulableID

ProducerSID : SchedulableID

ConsumerSID : SchedulableID

*distinct⟨nullSequencerId, nullSchedulableId, PCMissionSequencerSID,
ProducerSID, ConsumerSID⟩*

1.3 Non-Paradigm Objects

section *BufferApp* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*,
BufferClass, *MethodCallBindingChannels*, *ObjectChan*, *ObjectIds*, *ThreadIds*, *ObjectFWChan*, *ObjectIds*

| *BufferID* : *NonParadigmID*

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

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

state *State*

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

writeSyncMeth $\hat{=}$

$$\left(\begin{array}{l} \text{writeCall} . \text{BufferID} ? \text{caller} ? \text{thread} ? \text{update} \longrightarrow \\ \left(\begin{array}{l} \text{startSyncMeth} . \text{BufferOID} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{BufferOID} . \text{thread} \longrightarrow \\ \left(\begin{array}{l} \mu X \bullet \\ \left(\begin{array}{l} \text{var } \text{loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{this} . \text{bufferEmpty}()); \\ \text{if } (\text{loopVar} = \mathbf{True}) \longrightarrow \\ \left(\begin{array}{l} \text{waitCall} . \text{BufferOID} . \text{thread} \longrightarrow \\ \text{waitRet} . \text{BufferOID} . \text{thread} \longrightarrow \end{array} \right) ; X \\ \mathbf{Skip} \end{array} \right) \\ \parallel (\text{loopVar} = \mathbf{False}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array} \right) ; \\ \text{theBuffer} := \text{update}; \\ \text{notify} . \text{BufferOID} ! \text{thread} \longrightarrow \\ \mathbf{Skip} \end{array} \right) ; \\ \text{endSyncMeth} . \text{BufferOID} . \text{thread} \longrightarrow \\ \text{writeRet} . \text{BufferID} . \text{caller} . \text{thread} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \end{array} \right)$$

$$\begin{aligned}
& \text{readSyncMeth} \triangleq \mathbf{var} \text{ ret} : \mathbb{Z} \bullet \\
& \left(\begin{array}{l}
\text{readCall} . \text{BufferID} ? \text{caller} ? \text{thread} \longrightarrow \\
\left(\begin{array}{l}
\text{startSyncMeth} . \text{BufferOID} . \text{thread} \longrightarrow \\
\text{lockAcquired} . \text{BufferOID} . \text{thread} \longrightarrow \\
\left(\mu X \bullet \right. \\
\left(\begin{array}{l}
\mathbf{var} \text{ loopVar} : \mathbb{B} \bullet \text{loopVar} := \text{this} . \text{bufferEmpty}(); \\
\mathbf{if} (\text{loopVar} = \mathbf{True}) \longrightarrow \\
\left(\begin{array}{l}
\text{waitCall} . \text{BufferOID} . \text{thread} \longrightarrow \\
\text{waitRet} . \text{BufferOID} . \text{thread} \longrightarrow
\end{array} \right) ; X \\
\mathbf{Skip} \\
\left[\text{loopVar} = \mathbf{False} \right] \longrightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{array} \right) \\
; \\
\mathbf{var} \text{ out} : \mathbb{Z} \bullet \text{out} := \text{this} . \text{theBuffer}; \\
\text{this} . \text{theBuffer} := 0; \\
\text{notify} . \text{BufferOID} ! \text{thread} \longrightarrow \\
\mathbf{Skip}; \\
\text{ret} := \text{out} \\
\text{endSyncMeth} . \text{BufferOID} . \text{thread} \longrightarrow \\
\text{readRet} . \text{BufferID} . \text{caller} . \text{thread} ! \text{ret} \longrightarrow \\
\mathbf{Skip}
\end{array} \right) ;
\end{array} \right)
\end{aligned}$$

$$\begin{aligned}
& \text{Methods} \triangleq \\
& \left(\begin{array}{l}
\text{GetSequencer} \\
\Box \\
\text{InitializeApplication} \\
\Box \\
\text{writeSyncMeth} \\
\Box \\
\text{readSyncMeth}
\end{array} \right) ; \text{Methods}
\end{aligned}$$

$$\bullet (\text{Init} ; \text{Methods}) \triangle (\text{end_safelet_app} \longrightarrow \mathbf{Skip})$$

end

```
section BufferClass parents scj_prelude, SchedulableId, SchedulableIds, SafeletChan  
  , MethodCallBindingChannels
```

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

state <i>State</i> <i>theBuffer</i> : \mathbb{Z}
--

```
state State
```

initial <i>Init</i> <i>State</i> '
<i>theBuffer</i> ' = 0

```
public bufferEmpty  $\hat{=}$   

 $\left( \begin{array}{l} \mathbf{Skip}; \\ \mathbf{if} \ (theBuffer = 0) \longrightarrow \\ \quad ret := \mathbf{True} \\ \quad \square \neg (theBuffer = 0) \longrightarrow \\ \quad \quad ret := \mathbf{False} \\ \mathbf{fi} \end{array} \right)$ 
```

- **Skip**

```
end
```

1.4 ThreadIds

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

SafeletTid : *ThreadID*
nullThreadId : *ThreadID*
ProducerTID : *ThreadID*
ConsumerTID : *ThreadID*

distinct(*SafeletTid*, *nullThreadId*,
ProducerTID, *ConsumerTID*)

1.5 ObjectIds

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

BufferOID : *ObjectID*

distinct \langle *BufferOID* \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.PCMission, done_mission.PCMission, 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 MethodCallBinder

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

channel *binder_readCall* : *blankID* \times *SchedulableID* \times *ThreadID*
channel *binder_readRet* : *blankID* \times *SchedulableID* \times *ThreadID* \times \mathbb{Z}

readLocs == { *BufferID* }
readCallers == { *ConsumerSID* }

channel *binder_terminationPendingCall* : \times *SchedulableID*
channel *binder_terminationPendingRet* : \times *SchedulableID* \times *boolean*

terminationPendingLocs == { *PCMissionMID* }
terminationPendingCallers == { *ProducerSID, ConsumerSID* }

channel *binder_writeCall* : *blankID* \times *SchedulableID* \times *ThreadID* \times \mathbb{Z}
channel *binder_writeRet* : *blankID* \times *SchedulableID* \times *ThreadID*

writeLocs == { *BufferID* }
writeCallers == { *ProducerSID* }

channelset *MethodCallBinderSync* == { *done_toplevel_sequencer,*
binder_readCall, binder_readRet,
binder_terminationPendingCall, binder_terminationPendingRet,
binder_writeCall, binder_writeRet }

section *MethodCallBinder* **parents** *scj_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MethodCallBindingChannels, BufferMethChan, PCMissionMethChan*

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

read_MethodBinder $\hat{=}$

$$\left(\begin{array}{l} \text{binder_readCall ? loc : (loc \in readLocs) ? caller : (caller \in readCallers) ? callingThread} \longrightarrow \\ \text{readCall . loc . caller . callingThread} \longrightarrow \\ \text{readRet . loc . caller . callingThread ? ret} \longrightarrow \\ \text{binder_readRet . loc . caller . callingThread ! ret} \longrightarrow \\ \text{read_MethodBinder} \end{array} \right)$$

terminationPending_MethodBinder $\hat{=}$

$$\left(\begin{array}{l} \text{binder_terminationPendingCall ? loc : (loc \in terminationPendingLocs) ? caller : (caller \in terminationPendingCallers)} \longrightarrow \\ \text{terminationPendingCall . loc . caller} \longrightarrow \\ \text{terminationPendingRet . loc . caller ? ret} \longrightarrow \\ \text{binder_terminationPendingRet . loc . caller ! ret} \longrightarrow \\ \text{terminationPending_MethodBinder} \end{array} \right)$$

$$write_MethodBinder \hat{=} \left(\begin{array}{l} binder_writeCall ? loc : (loc \in writeLocs) ? caller : (caller \in writeCallers) ? callingThread ? p1 \longrightarrow \\ writeCall . loc . caller . callingThread ! p1 \longrightarrow \\ writeRet . loc . caller . callingThread \longrightarrow \\ binder_writeRet . loc . caller . callingThread \longrightarrow \\ write_MethodBinder \end{array} \right)$$

$$BinderActions \hat{=} \left(\begin{array}{l} read_MethodBinder \\ ||| \\ terminationPending_MethodBinder \\ ||| \\ write_MethodBinder \end{array} \right)$$

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

end

2.3 Locking

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

process *Threads* $\hat{=}$
 $\left(\begin{array}{c} \text{ThreadFW}(\text{ProducerTID}, 10) \\ \parallel \\ \text{ThreadFW}(\text{ConsumerTID}, 10) \end{array} \right)$

process *Objects* $\hat{=}$
 $(\text{ObjectFW}(\text{BufferOID}))$

process *Locking* $\hat{=}$ $(\text{Threads} \parallel \text{ThreadSync} \parallel \text{Objects}) \triangle (\text{done_toplevel_sequencer} \longrightarrow \mathbf{Skip})$

2.4 Program

section *Program* **parents** *scj_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, MissionFW, SafeletFW, TopLevelMissionSequencerFW, NetworkChannels, ManagedThreadFW, SchedulableMissionSequencerFW, PeriodicEventHandlerFW, OneShotEventHandlerFW, AperiodicEventHandlerFW, ObjectFW, ThreadFW, PCSafeletApp, PCMissionSequencerApp, PCMissionApp, ProducerApp, ConsumerApp*

process *ControlTier* $\hat{=}$

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

process *Tier0* $\hat{=}$

$$\left(\begin{array}{l} \text{MissionFW}(\text{PCMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{ManagedThreadFW}(\text{ProducerID}) \\ \llbracket \text{SchedulablesSync} \rrbracket \end{array} \right) \\ \text{ManagedThreadFW}(\text{ConsumerID}) \end{array} \right)$$

process *Framework* $\hat{=}$

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

process *Application* $\hat{=}$

$$\left(\begin{array}{l} \text{PCSafeletApp} \\ ||| \\ \text{PCMissionSequencerApp} \\ ||| \\ \text{PCMissionApp} \\ ||| \\ \text{ProducerApp}(\text{PCMissionID}) \\ ||| \\ \text{ConsumerApp}(\text{PCMissionID}) \\ ||| \\ \text{BufferApp} \end{array} \right)$$

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

3 Safelet

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

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

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

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

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 *PCMissionSequencerApp* **parents** *TopLevelMissionSequencerChan*,
MissionId, *MissionIds*, *SchedulableId*, *SchedulableIds*, *PCMissionSequencerClass*, *MethodCallBindingChannels*

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

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

state *State*

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

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

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

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

end

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

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

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

state *State*

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

protected *getNextMission* $\hat{=}$

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

• **Skip**

end

5 Missions

5.1 PCMission

section *PCMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *PCMissionMethChan*,
MethodCallBindingChannels

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

InitializePhase $\hat{=}$

$$\left(\begin{array}{l} \text{initializeCall} . \text{PCMissionMID} \longrightarrow \\ \text{register} ! \text{ProducerSID} ! \text{PCMissionMID} \longrightarrow \\ \text{register} ! \text{ConsumerSID} ! \text{PCMissionMID} \longrightarrow \\ \text{initializeRet} . \text{PCMissionMID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

CleanupPhase $\hat{=}$ **var** $\mathbb{B} : \text{ret} \bullet$

$$\left(\begin{array}{l} \text{cleanupMissionCall} . \text{PCMissionMID} \longrightarrow \\ (\text{ret} := \mathbf{False}) \\ \text{cleanupMissionRet} . \text{PCMissionMID} ! \text{ret} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

$\bullet (\text{Methods}) \triangle (\text{end_mission_app} . \text{PCMissionMID} \longrightarrow \mathbf{Skip})$

end

5.2 Schedulables of PCMission

section *ProducerApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*, *MissionMethChan*, *BufferMethChan*, *ObjectIds*, *ThreadIds*

process *ProducerApp* $\hat{=}$
pcMission : *MissionID* • **begin**

Run $\hat{=}$

$$\left(\begin{array}{l} \text{runCall} . \text{ProducerSID} \longrightarrow \\ \left(\begin{array}{l} \mathbf{var} \ i : \mathbb{Z} \bullet i := 1; \\ \mu X \bullet \\ \left(\begin{array}{l} \text{binder_terminationPendingCall} . \text{pcMission} \longrightarrow \\ \text{binder_terminationPendingRet} . \text{pcMission} ? \text{terminationPending} \longrightarrow \\ \mathbf{var} \ \text{loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \mathbf{if} \ (\text{loopVar} = \mathbf{True}) \longrightarrow \\ \left(\begin{array}{l} \text{binder_writeCall} . \text{BufferID} . \text{ProducerSID} . \text{ProducerTID} ! i \longrightarrow \\ \text{binder_writeRet} . \text{BufferID} . \text{ProducerSID} . \text{ProducerTID} \longrightarrow \\ \mathbf{Skip}; \\ i := i + 1; \\ \mathbf{if} \ (i \geq 5) \longrightarrow \\ \left(\begin{array}{l} \text{requestTerminationCall} . \text{pcMission} . \text{ProducerSID} \longrightarrow \\ \text{requestTerminationRet} . \text{pcMission} . \text{ProducerSID} ? \text{requestTermination} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\ \parallel \neg (i \geq 5) \longrightarrow \mathbf{Skip} \end{array} \right) \\ \mathbf{fi} \\ \parallel (\text{loopVar} = \mathbf{False}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array} \right) ; X \end{array} \right) ; \end{array} \right)$$

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

• (*Methods*) \triangle (*end_managedThread_app* . *ProducerSID* \longrightarrow **Skip**)

end

section *ConsumerApp* **parents** *ManagedThreadChan, SchedulableId, SchedulableIds, MethodCallBindingChannels, MissionMethChan, BufferMethChan, ObjectIds, ThreadIds*

process *ConsumerApp* $\hat{=}$
 pcMission : *MissionID* • **begin**

Run $\hat{=}$

$$\left(\begin{array}{l} \text{runCall} . \text{ConsumerSID} \longrightarrow \\ \left(\mu X \bullet \begin{array}{l} \text{binder_terminationPendingCall} . \text{pcMission} \longrightarrow \\ \text{binder_terminationPendingRet} . \text{pcMission} ? \text{terminationPending} \longrightarrow \\ \text{var } \text{loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \text{if } (\text{loopVar} = \mathbf{True}) \longrightarrow \\ \quad \left(\begin{array}{l} \text{var } \text{result} : \mathbb{Z} \bullet \text{result} := 999; \\ \text{binder_readCall} . \text{BufferID} . \text{ConsumerSID} . \text{ConsumerTID} \longrightarrow \\ \text{binder_readRet} . \text{BufferID} . \text{ConsumerSID} . \text{ConsumerTID} ? \text{read} \longrightarrow \end{array} \right); X \\ \mathbf{Skip} \end{array} \right) \\ \parallel (\text{loopVar} = \mathbf{False}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array} \right) \end{array} \right);$$

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

• (*Methods*) \triangle (*end_managedThread_app* . *ConsumerSID* \longrightarrow **Skip**)

end