producerConsumer

Tight Rope v0.75

1st 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*

> $PCMissionSequencerSID : SchedulableID$
> $ProducerSID : SchedulableID$
> $ConsumerSID : SchedulableID$
> ───────────────
> $distinct \langle nullSequencerId, nullSchedulableId, PCMissionSequencerSID,$
> $ProducerSID, ConsumerSID \rangle$

1.3 Non-Paradigm Objects

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

   | *BufferID* : *NonParadigmID*

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

   ── *State* ─────────────────────────
     this : **ref** *BufferClass*
   ────────────────────────────────

state *State*

   ── *Init* ──────────────────────────
     State'
     ──────────
     this' = **new** *BufferClass*()
   ────────────────────────────────

$writeSyncMeth \ \widehat{=}$
$$
\begin{pmatrix}
writeCall . BufferID ? caller ? thread ? update \longrightarrow \\
\begin{pmatrix}
startSyncMeth . BufferOID . thread \longrightarrow \\
lockAcquired . BufferOID . thread \longrightarrow \\
\begin{pmatrix}
\mu X \bullet \\
\begin{pmatrix}
\mathbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ this . this . bufferEmpty()); \\
\mathbf{if}\ (loopVar = \mathbf{True}) \longrightarrow \\
\quad \begin{pmatrix}
\mathbf{Skip}; \\
waitCall . BufferOID . thread \longrightarrow \\
waitRet . BufferOID . thread \longrightarrow \\
\mathbf{Skip}
\end{pmatrix} ;\ X \\
[\!]\ (loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{pmatrix} \\
; \\
\mathbf{Skip}; \\
theBuffer := update; \\
notify . BufferOID\ !\ thread \longrightarrow \\
\mathbf{Skip}
\end{pmatrix} \\
endSyncMeth . BufferOID . thread \longrightarrow \\
writeRet . BufferID . caller . thread \longrightarrow \\
\mathbf{Skip}
\end{pmatrix} ;
\end{pmatrix}
$$

$readSyncMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : \mathbb{Z} \bullet$

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

$Methods \mathrel{\widehat{=}}$

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

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

$\mathbf{end}$

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

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

---
state *State*
$theBuffer : \mathbb{Z}$

---

state *State*

---
initial *Init*
$State'$

$theBuffer' = 0$

---

public *bufferEmpty* $\widehat{=}$

$$\begin{pmatrix} \textbf{Skip;} \\ \textbf{if } (theBuffer = 0) \longrightarrow \\ \quad ret := \textbf{True} \\ [\!] \neg (theBuffer = 0) \longrightarrow \\ \quad ret := \textbf{False} \\ \textbf{fi} \end{pmatrix}$$

• **Skip**

end

1.4   ThreadIds

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

$SafeletTId : ThreadID$
$nullThreadId : ThreadID$
$ProducerTID : ThreadID$
$ConsumerTID : ThreadID$

$distinct\langle SafeletTId, nullThreadId,$
$ProducerTID, ConsumerTID\rangle$

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* ==
    $\lmoustache$ *schedulables_terminated*, *schedulables_stopped*, *get_activeSchedulables* $\rmoustache$

channelset *ControlTierSync* ==
    $\lmoustache$ *start_toplevel_sequencer*, *done_toplevel_sequencer*, *done_safeletFW* $\rmoustache$

channelset *TierSync* ==
    $\lmoustache$ *start_mission . PCMission*, *done_mission . PCMission*,
    done_safeletFW, *done_toplevel_sequencer* $\rmoustache$

channelset *MissionSync* ==
    $\lmoustache$ *done_safeletFW*, *done_toplevel_sequencer*, *register*,
signalTerminationCall, *signalTerminationRet*, *activate_schedulables*, *done_schedulable*,
cleanupSchedulableCall, *cleanupSchedulableRet* $\rmoustache$

channelset *SchedulablesSync* ==
    $\lmoustache$ *activate_schedulables*, *done_safeletFW*, *done_toplevel_sequencer* $\rmoustache$

channelset *ClusterSync* ==
    $\lmoustache$ *done_toplevel_sequencer*, *done_safeletFW* $\rmoustache$

channelset *SafeltAppSync* $\widehat{=}$
$\lmoustache$ *getSequencerCall*, *getSequencerRet*, *initializeApplicationCall*, *initializeApplicationRet*, *end_safelet_app* $\rmoustache$

channelset *MissionSequencerAppSync* ==
$\lmoustache$ *getNextMissionCall*, *getNextMissionRet*, *end_sequencer_app* $\rmoustache$

channelset *MissionAppSync* ==
$\lmoustache$ *initializeCall*, *register*, *initializeRet*, *cleanupMissionCall*, *cleanupMissionRet* $\rmoustache$

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

channelset *ThreadSync* ==
    $\lmoustache$ *raise_thread_priority*, *lower_thread_priority*, *isInterruptedCall*, *isInterruptedRet*, *get_priorityLevel* $\rmoustache$

channelset *LockingSync* ==
    $\lmoustache$ *lockAcquired*, *startSyncMeth*, *endSyncMeth*, *waitCall*, *waitRet*, *notify*, *isInterruptedCall*, *isInterruptedRet*,
    interruptedCall, *interruptedRet*, *done_toplevel_sequencer*, *get_priorityLevel* $\rmoustache$

2.2 MethodCallBinder

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

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

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

channel *binder_terminationPendingCall* : ×*SchedulableID*
channel *binder_terminationPendingRet* : ×*SchedulableID* × *boolean*

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

channel *binder_writeCall* : *blankID* × *SchedulableID* × *ThreadID* × $\mathbb{Z}$
channel *binder_writeRet* : *blankID* × *SchedulableID* × *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*
, *BuffeMethChan*, *PCMissionMethChan*

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

read_MethodBinder $\widehat{=}$
$$\left(\begin{array}{l} binder\_readCall\,?\,loc : (loc \in readLocs)\,?\,caller : (caller \in readCallers)\,?\,callingThread \longrightarrow \\ readCall\,.\,loc\,.\,caller\,.\,callingThread \longrightarrow \\ readRet\,.\,loc\,.\,caller\,.\,callingThread\,?\,ret \longrightarrow \\ binder\_readRet\,.\,loc\,.\,caller\,.\,callingThread\,!\,ret \longrightarrow \\ read\_MethodBinder \end{array}\right)$$

terminationPending_MethodBinder $\widehat{=}$
$$\left(\begin{array}{l} binder\_terminationPendingCall\,?\,loc : (loc \in terminationPendingLocs)\,?\,caller : (caller \in terminationPendingCaller \\ terminationPendingCall\,.\,loc\,.\,caller \longrightarrow \\ terminationPendingRet\,.\,loc\,.\,caller\,?\,ret \longrightarrow \\ binder\_terminationPendingRet\,.\,loc\,.\,caller\,!\,ret \longrightarrow \\ terminationPending\_MethodBinder \end{array}\right.$$

$write\_MethodBinder \mathrel{\widehat{=}}$

$$\begin{pmatrix} binder\_writeCall\,?\,loc : (loc \in writeLocs)\,?\,caller : (caller \in writeCallers)\,?\,callingThread\,?\,p1 \longrightarrow \\ writeCall\,.\,loc\,.\,caller\,.\,callingThread\,!\,p1 \longrightarrow \\ writeRet\,.\,loc\,.\,caller\,.\,callingThread \longrightarrow \\ binder\_writeRet\,.\,loc\,.\,caller\,.\,callingThread \longrightarrow \\ write\_MethodBinder \end{pmatrix}$$

$BinderActions \mathrel{\widehat{=}}$

$$\begin{pmatrix} read\_MethodBinder \\ ||| \\ terminationPending\_MethodBinder \\ ||| \\ write\_MethodBinder \end{pmatrix}$$

$\bullet\ BinderActions \mathrel{\triangle} (done\_toplevel\_sequencer \longrightarrow \mathbf{Skip})$

end

10

2.3 Locking

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

process *Threads* $\widehat{=}$

$$\begin{pmatrix} ThreadFW(ProducerTID, 10) \\ ||| \\ ThreadFW(ConsumerTID, 10) \end{pmatrix}$$

process *Objects* $\widehat{=}$
$$\big(ObjectFW(BufferOID) \big)$$

process *Locking* $\widehat{=}$ (*Threads* ⟦ *ThreadSync* ⟧ *Objects*) $\triangle$ (*done_toplevel_sequencer* $\longrightarrow$ **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* $\widehat{=}$
$$\begin{pmatrix} SafeletFW \\ \quad [\![ControlTierSync]\!] \\ TopLevelMissionSequencerFW\,(PCMissionSequencer) \end{pmatrix}$$

process *Tier0* $\widehat{=}$
$$\begin{pmatrix} MissionFW\,(PCMissionID) \\ \quad [\![MissionSync]\!] \\ \begin{pmatrix} ManagedThreadFW\,(ProducerID) \\ \quad [\![SchedulablesSync]\!] \\ ManagedThreadFW\,(ConsumerID) \end{pmatrix} \end{pmatrix}$$

process *Framework* $\widehat{=}$
$$\begin{pmatrix} ControlTier \\ \quad [\![TierSync]\!] \\ \left(Tier0 \right) \end{pmatrix}$$

process *Application* $\widehat{=}$
$$\begin{pmatrix} PCSafeletApp \\ ||| \\ PCMissionSequencerApp \\ ||| \\ PCMissionApp \\ ||| \\ ProducerApp(PCMissionID) \\ ||| \\ ConsumerApp(PCMissionID) \\ ||| \\ BufferApp \end{pmatrix}$$

process *Bound_Application* $\widehat{=}$ *Application* $[\![$ *MethodCallBinderSync* $]\!]$ *MethodCallBinder*
process *Program* $\widehat{=}$ $\left(\right.$ *Framework* $[\![$ *AppSync* $]\!]$ *Bound_Application* $\left.\right)$ $[\![$ *LockingSync* $]\!]$ *Locking*

3 Safelet

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

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

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

GetSequencer $\widehat{=}$
$$\begin{pmatrix} getSequencerCall \longrightarrow \\ getSequencerRet \, ! \, PCMissionSequencerSID \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

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

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

end

4 Top Level Mission Sequencer

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

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

```
┌─ State ──────────────────────────────────────────────
│  this : ref PCMissionSequencerClass
└──────────────────────────────────────────────────────
```

state *State*

```
┌─ Init ───────────────────────────────────────────────
│  State′
│ ┌──────────────
│  this′ = new PCMissionSequencerClass()
└──────────────────────────────────────────────────────
```

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

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

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

end

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

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

```
┌─ state State ──────────────────────────────────────────
│  returnedMission : 𝔹
└────────────────────────────────────────────────────────
```

state *State*

```
┌─ initial Init ─────────────────────────────────────────
│  State′
│ ───────────────────
│  returnedMission′ = False
└────────────────────────────────────────────────────────
```

protected *getNextMission* $\widehat{=}$

$$
\begin{pmatrix}
\textbf{Skip}; \\
\textbf{if } (\neg\ returnedMission) \longrightarrow \\
\qquad \begin{pmatrix} \textbf{Skip}; \\ returnedMission := \textbf{True}; \\ ret := PCMissionMID \end{pmatrix} \\
[\!] \neg\ (\neg\ returnedMission) \longrightarrow \\
\qquad \big(ret := nullMissionId \big) \\
\textbf{fi}
\end{pmatrix}
$$

- **Skip**

end

5 Missions

5.1 PCMission

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

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

```
┌─ State ─────────────────────────────────────────────
│  this : ref PCMissionClass
│  buffer : Buffer
```

state *State*

```
┌─ Init ──────────────────────────────────────────────
│  State′
│ ────────────
│  this′ = new PCMissionClass()
│  buffer′ =
```

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

CleanupPhase $\widehat{=}$ **var** $\mathbb{B}$: *ret* •
$$\left(\begin{array}{l} cleanupMissionCall \, . \, PCMissionMID \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ ret := \mathbf{False} \end{array} \right) \\ cleanupMissionRet \, . \, PCMissionMID \, ! \, ret \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

getBufferMeth $\widehat{=}$ **var** *ret* : *Buffer* •
$$\left(\begin{array}{l} getBufferCall \, . \, PCMissionMID \longrightarrow \\ ret := this \, . \, getBuffer(); \\ getBufferRet \, . \, PCMissionMID \, ! \, ret \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$$Methods \ \widehat{=}\ \left(\begin{array}{l} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ getBufferMeth \end{array} \right) ;\ \ Methods$$

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

end

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

class *PCMissionClass* $\cong$ **begin**

public *getBuffer* $\cong$
$(\ ret := buffer\ )$

• **Skip**

end

5.2 Schedulables of PCMission

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

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

State
---
  buffer : *Buffer*
---

state *State*

Init

  State'
---
  buffer' =

$Run \widehat{=}$
$\Big(\ runCall\ .\ ProducerSID \longrightarrow$
  $\Big(\ \mathbf{Skip};$
    $\mu X \bullet$
    $\Big(\ binder\_terminationPendingCall\ .\ pcMission \longrightarrow$
      $binder\_terminationPendingRet\ .\ pcMission\ ?\ terminationPending \longrightarrow$
      $\mathbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ terminationPending);$
      $\mathbf{if}\ (loopVar = \mathbf{True}) \longrightarrow$
        $\Big(\ binder\_writeCall\ .\ bufferID\ .\ ProducerSID\ .\ ProducerTID\ !\ i \longrightarrow$
          $binder\_writeRet\ .\ bufferID\ .\ ProducerSID\ .\ ProducerTID \longrightarrow$
          $\mathbf{Skip};$
          $this\ .\ i := this\ .\ i + 1;$
          $\mathbf{var}\ keepWriting : \mathbb{B} \bullet keepWriting := this\ .\ i >= 5;$
          $\mathbf{if}\ (\neg\ keepWriting) \longrightarrow$
            $\Big(\ requestTerminationCall\ .\ pcMission\ .\ ProducerSID \longrightarrow$
              $requestTerminationRet\ .\ pcMission\ .\ ProducerSID\ ?\ requestTermination \longrightarrow \Big)$
              $\mathbf{Skip}$
           $[\!]\ \neg\ (\neg\ keepWriting) \longrightarrow \mathbf{Skip}$
          $\mathbf{fi}\ \Big)\ ;\ X$
      $[\!]\ (loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip}$
      $\mathbf{fi}\ \Big)\Big)\ ;$
  $runRet\ .\ ProducerSID \longrightarrow$
  $\mathbf{Skip}\ \Big)$

$Methods \widehat{=}$
$\big(\ Run\ \big)\ ;\ Methods$

• $(Init\ ;\ Methods) \triangle (end\_managedThread\_app\ .\ ProducerSID \longrightarrow \mathbf{Skip})$

end

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

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

---
state *State*
$i : \mathbb{Z}$

---

state *State*

---
initial *Init*
$State'$

$i' = 1$

- **Skip**

end

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

process *ConsumerApp* $\widehat{=}$
    pcMission : *MissionID* ● **begin**

```
┌─ State ──────────────────────────────────────────────────────────
│  buffer : Buffer
└──────────────────────────────────────────────────────────────────
```

state *State*

```
┌─ Init ───────────────────────────────────────────────────────────
│  State′
│ ─────────
│  buffer′ =
└──────────────────────────────────────────────────────────────────
```

$Run \widehat{=}$
$\left(\begin{array}{l} runCall \,.\, ConsumerSID \longrightarrow \\ \left(\begin{array}{l} \mathbf{Skip}; \\ \mu X \bullet \\ \left(\begin{array}{l} binder\_terminationPendingCall \,.\, pcMission \longrightarrow \\ binder\_terminationPendingRet \,.\, pcMission \,?\, terminationPending \longrightarrow \\ \mathbf{var}\; loopVar : \mathbb{B} \bullet loopVar := (\neg\; terminationPending); \\ \mathbf{if}\; (loopVar = \mathbf{True}) \longrightarrow \\ \quad \left(\begin{array}{l} \mathbf{var}\; result : \mathbb{Z} \bullet result := 999; \\ binder\_readCall \,.\, bufferID \,.\, ConsumerSID \,.\, ConsumerTID \longrightarrow \\ binder\_readRet \,.\, bufferID \,.\, ConsumerSID \,.\, ConsumerTID \,?\, read \longrightarrow \\ \mathbf{Skip}; \\ \mathbf{Skip} \end{array}\right)\; ;\; X \\ [] \; (loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array}\right) \end{array}\right) \; ; \\ runRet \,.\, ConsumerSID \longrightarrow \\ \mathbf{Skip} \end{array}\right)$

$Methods \widehat{=}$
$\left(\; Run \;\right) \; ; \; Methods$

● (*Init* ; *Methods*) $\triangle$ (*end_managedThread_app* . *ConsumerSID* $\longrightarrow$ **Skip**)

end

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

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

```
  ┌─ state State ──────────────────────────────────────────────
  │  buffer : Buffer
  └──────────────────────────────────────────────────────────
```

state *State*

```
  ┌─ initial Init ─────────────────────────────────────────────
  │  State'
  └──────────────────────────────────────────────────────────
```

• **Skip**

end