

# 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> : <b>ref</b> <i>BufferClass</i>
---

**state** *State*

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

*writeSyncMeth*  $\hat{=}$

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

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

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

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

**end**

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

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

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

```
state State
```

<b>initial</b> <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*  $\hat{=}$  (*Threads*  $\parallel$  *ThreadSync*  $\parallel$  *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*  $\hat{=}$   

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

**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> : <b>ref</b> <i>PCMissionSequencerClass</i>
---

**state** *State*

<i>Init</i> <i>State'</i>
<i>this'</i> = <b>new</b> <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**

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

**state** *State*

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

**protected** *getNextMission*  $\hat{=}$

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

• **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*  $\hat{=}$   

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

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

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

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

$\bullet (\textit{Methods}) \triangle (\textit{end\_mission\_app} . \textit{PCMissionMID} \longrightarrow \mathbf{Skip})$

**end**



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

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

```
public getBuffer  $\hat{=}$   
(ret := buffer)
```

- **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*  $\hat{=}$

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

*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*  $\hat{=}$   

$$\left( \begin{array}{l} \text{runCall} . \text{ConsumerSID} \longrightarrow \\ \left( \mu X \bullet \begin{array}{l} \text{binder\_terminationPendingCall} . \text{PcMission} \longrightarrow \\ \text{binder\_terminationPendingRet} . \text{PcMission} ? \text{terminationPending} \longrightarrow \\ \text{var } \text{loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \text{if } (\text{loopVar} = \mathbf{True}) \longrightarrow \\ \left( \begin{array}{l} \text{var } \text{result} : \mathbb{Z} \bullet \text{result} := 999; \\ \text{binder\_readCall} . \text{BufferID} . \text{ConsumerSID} . \text{ConsumerTID} \longrightarrow \\ \text{binder\_readRet} . \text{BufferID} . \text{ConsumerSID} . \text{ConsumerTID} ? \text{read} \longrightarrow \end{array} \right); X \\ \mathbf{Skip} \end{array} \right) \\ \parallel (\text{loopVar} = \mathbf{False}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array} \right) \\ \text{runRet} . \text{ConsumerSID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

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

•  $(\text{Methods}) \triangle (\text{end\_managedThread\_app} . \text{ConsumerSID} \longrightarrow \mathbf{Skip})$

**end**