

# Flatbuffer

Tight Rope v0.6

24th January 2016

## 1 ID Files

### 1.1 MissionIds

**section** *MissionIds* **parents** *scj\_prelude*, *MissionId*

<i>FlatBufferMissionID</i> : <i>MissionID</i>
---

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

## 1.2 SchedulablesIds

**section** *SchedulableIds* **parents** *scj\_prelude*, *SchedulableId*

*FlatBufferMissionSequencerID* : *SchedulableID*

*ReaderID* : *SchedulableID*

*WriterID* : *SchedulableID*

---

*distinct*  $\langle$ *nullSequencerId*, *nullSchedulableId*, *FlatBufferMissionSequencerID*,  
*ReaderID*, *WriterID* $\rangle$

### 1.3 ThreadIds

**section** *ThreadId* **parents** *scj\_prelude, GlobalTypes*

*ReaderThreadID* : *ThreadID*

*WriterThreadID* : *ThreadID*

---

*distinct*(*SafeletThreadId*, *nullThreadId*,  
*ReaderThreadID*, *WriterThreadID*)

## 1.4 ObjectIds

**section** *ObjectIds* **parents** *scj\_prelude, GlobalTypes*

*FlatBufferObjectID : ObjectID*

*FlatBufferMissionObjectID : ObjectID*

*ReaderObjectID : ObjectID*

*WriterObjectID : ObjectID*

---

*distinct*  $\langle$  *FlatBufferObjectID*, *FlatBufferMissionObjectID*,  
*ReaderObjectID*, *WriterObjectID*  $\rangle$

## 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.FlatBufferMission, done\_mission.FlatBufferMission,*  
      *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* ==  
    { *SafeltAppSync, MissionSequencerAppSync, MissionAppSync,*  
      *MtAppSync, OSEHSync, APEHSync,*  
      { *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* }

**section** *Program parents* *scj\_prelude*, *MissionId*, *MissionIds*,  
*SchedulableId*, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *MissionFW*,  
*SafeletFW*, *TopLevelMissionSequencerFW*, *NetworkChannels*, *ManagedThreadFW*,  
*SchedulableMissionSequencerFW*, *PeriodicEventHandlerFW*, *OneShotEventHandlerFW*,  
*AperiodicEventHandlerFW*, *ObjectFW*, *ThreadFW*,  
*FlatBufferApp*, *FlatBufferMissionSequencerApp*, *FlatBufferMissionApp*, *ReaderApp*, *WriterApp*

**process** *ControlTier*  $\hat{=}$   

$$\left( \begin{array}{c} \textit{SafeletFW} \\ \llbracket \textit{ControlTierSync} \rrbracket \\ \textit{TopLevelMissionSequencerFW}(\textit{FlatBufferMissionSequencer}) \end{array} \right)$$

**process** *Tier0*  $\hat{=}$   

$$\left( \begin{array}{c} \textit{MissionFW}(\textit{FlatBufferMissionID}) \\ \llbracket \textit{MissionSync} \rrbracket \\ \left( \begin{array}{c} \textit{ManagedThreadFW}(\textit{ReaderID}) \\ \llbracket \textit{SchedulablesSync} \rrbracket \\ \textit{ManagedThreadFW}(\textit{WriterID}) \end{array} \right) \end{array} \right)$$

**process** *Framework*  $\hat{=}$   

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

**process** *Application*  $\hat{=}$   

$$\left( \begin{array}{c} \textit{FlatBufferApp} \\ ||| \\ \textit{FlatBufferMissionSequencerApp} \\ ||| \\ \textit{FlatBufferMissionApp} \\ ||| \\ \textit{ReaderApp}(\textit{FlatBufferMissionID}) \\ ||| \\ \textit{WriterApp}(\textit{FlatBufferMissionID}) \end{array} \right)$$

**channel** *binder\_readCall* : *MissionID* × *SchedulableID*  
**channel** *binder\_readRet* : *MissionID* × *SchedulableID* ×  $\mathbb{Z}$

*readLocs* == {*FlatBufferMission*}  
*readCallers* == {*Reader*}

**channel** *binder\_writeCall* : *MissionID* × *SchedulableID* ×  $\mathbb{Z}$   
**channel** *binder\_writeRet* : *MissionID* × *SchedulableID*

*writeLocs* == {*FlatBufferMission*}  
*writeCallers* == {*Writer*}

**channelset** *MethodCallBinderSync* == { *done\_toplevel\_sequencer*, *binder\_readCall*, *binder\_readRet*,  
*binder\_writeCall*, *binder\_writeRet* }

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

*read\_MethodBinder*  $\hat{=}$   

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

*write\_MethodBinder*  $\hat{=}$   

$$\left( \begin{array}{l} \textit{binder\_writeCall} \\ \quad ? \textit{loc} : (\textit{loc} \in \textit{writeLocs}) \\ \quad ? \textit{caller} : (\textit{caller} \in \textit{writeCallers}) \times \mathbb{Z} \longrightarrow \\ \textit{writeCall} . \textit{loc} . \textit{caller} \times \mathbb{Z} \longrightarrow \\ \textit{writeRet} . \textit{loc} . \textit{caller} \longrightarrow \\ \textit{binder\_writeRet} . \textit{loc} . \textit{caller} \longrightarrow \\ \textit{write\_MethodBinder} \end{array} \right)$$

*BinderActions*  $\hat{=}$   

$$\left( \begin{array}{l} \textit{read\_MethodBinder} \\ ||| \\ \textit{write\_MethodBinder} \end{array} \right)$$

• *BinderActions*  $\triangle$  (*done\_toplevel\_sequencer*  $\longrightarrow$  **Skip**)

**end**

**process** *ApplicationB*  $\hat{=}$  *Application* [ *MethodCallBinderSync* ] *MethodCallBinder*

**process** *Threads*  $\hat{=}$   
 $\left( \begin{array}{l} \textit{ThreadFW}(\textit{ReaderThreadID}, 10) \\ ||| \\ \textit{ThreadFW}(\textit{WriterThreadID}, 10) \end{array} \right)$

**process** *Objects*  $\hat{=}$   
 $\left( \begin{array}{l} \textit{ObjectFW}(\textit{FlatBufferObjectID}) \\ ||| \\ \textit{ObjectFW}(\textit{FlatBufferMissionObjectID}) \\ ||| \\ \textit{ObjectFW}(\textit{ReaderObjectID}) \\ ||| \\ \textit{ObjectFW}(\textit{WriterObjectID}) \end{array} \right)$

**process** *Locking*  $\hat{=}$  *Threads*  $\llbracket$  *ThreadSync*  $\rrbracket$  *Objects*

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



### 3 Safelet

**section** *FlatBufferApp* **parents** *scj\_prelude, SchedulableId, SchedulableIds, SafeletChan*

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

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

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

*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** *FlatBufferMissionSequencerApp* **parents** *TopLevelMissionSequencerChan*,  
*MissionId*, *MissionIds*, *SchedulableId*, *FlatBufferMissionSequencerClass*

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

<i>State</i> <i>this</i> : <b>ref</b> <i>FlatBufferMissionSequencerClass</i>
---

**state** *State*

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

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

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

•  $( \textit{Init} ; \textit{Methods} ) \triangle ( \textit{end\_sequencer\_app} . \textit{FlatBufferMissionSequencer} \longrightarrow \mathbf{Skip} )$

**end**

**class** *FlatBufferMissionSequencerClass*  $\hat{=}$  **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> ' = <i>false</i>

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

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

• **Skip**

**end**

## 5 Missions

### 5.1 FlatBufferMission

**section** *FlatBufferMissionApp* **parents** *scj\_prelude, MissionId, MissionIds,*  
*SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, FlatBufferMissionClass*  
*,*  
*ObjectChan, ObjectIds, ThreadIds, FlatBufferMissionMethChan*

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

---

*State*  
*this* : **ref** *FlatBufferMissionClass*

---

**state** *State*

---

*Init*  
*State'*  


---

*this'* = **new** *FlatBufferMissionClass*()

---

*InitializePhase*  $\hat{=}$   
 $\left( \begin{array}{l} \textit{initializeCall} . \textit{FlatBufferMission} \longrightarrow \\ \textit{register} ! \textit{Reader} ! \textit{FlatBufferMission} \longrightarrow \\ \textit{register} ! \textit{Writer} ! \textit{FlatBufferMission} \longrightarrow \\ \textit{initializeRet} . \textit{FlatBufferMission} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

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

*bufferEmptyMeth*  $\hat{=}$  **var** *ret* :  $\mathbb{B}$  •  
 $\left( \begin{array}{l} \textit{bufferEmptyCall} . \textit{FlatBufferMission} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{bufferEmpty}(); \\ \textit{bufferEmptyRet} . \textit{FlatBufferMission} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

*cleanUpMeth*  $\hat{=}$  **var** *ret* :  $\mathbb{B}$  •  
 $\left( \begin{array}{l} \textit{cleanUpCall} . \textit{FlatBufferMission} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{cleanUp}(); \\ \textit{cleanUpRet} . \textit{FlatBufferMission} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

$$\begin{aligned}
\text{writeSyncMeth} \triangleq & \left( \begin{array}{l} \text{writeCall} . \text{FlatBufferMission} ? \text{thread} \longrightarrow \\ \left( \begin{array}{l} \text{startSyncMeth} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \left( \begin{array}{l} \mu X \bullet \\ \left( \begin{array}{l} \text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{bufferEmpty}()); \\ \text{if} (\text{loopVar}) \longrightarrow \\ \left( \begin{array}{l} \text{waitCall} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \\ \text{waitRet} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \end{array} \right) ; X \\ \text{Skip} \end{array} \right) \\ \square \neg (\text{loopVar}) \longrightarrow \text{Skip} \\ \text{fi} \end{array} \right) \end{array} \right) ; \\ \text{endSyncMeth} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \text{writeRet} . \text{FlatBufferMission} . \text{thread} \longrightarrow \\ \text{Skip} \end{array} \right)
\end{aligned}$$

$$\begin{aligned}
\text{readSyncMeth} \triangleq & \text{var ret} : \mathbb{Z} \bullet \left( \begin{array}{l} \text{readCall} . \text{FlatBufferMission} ? \text{thread} \longrightarrow \\ \left( \begin{array}{l} \text{startSyncMeth} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \left( \begin{array}{l} \mu X \bullet \\ \left( \begin{array}{l} \text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := \text{bufferEmpty}(); \\ \text{if} (\text{loopVar}) \longrightarrow \\ \left( \begin{array}{l} \text{waitCall} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \\ \text{waitRet} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \end{array} \right) ; X \\ \text{Skip} \end{array} \right) \\ \square \neg (\text{loopVar}) \longrightarrow \text{Skip} \\ \text{fi} \end{array} \right) \end{array} \right) ; \\ \text{var out} : \mathbb{Z} \bullet \text{out} := \text{buffer} ; \\ \text{this} . \text{buffer} := 0 ; \\ \text{notify} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \\ \text{Skip} ; \\ \text{ret} := \text{out} \\ \text{endSyncMeth} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \text{readRet} . \text{FlatBufferMission} ! \text{thread} ! \text{ret} \longrightarrow \\ \text{Skip} \end{array} \right)
\end{aligned}$$

$$\text{Methods} \triangleq \left( \begin{array}{l} \text{InitializePhase} \\ \square \\ \text{CleanupPhase} \\ \square \\ \text{bufferEmptyMeth} \\ \square \\ \text{cleanUpMeth} \\ \square \\ \text{writeSyncMeth} \\ \square \\ \text{readSyncMeth} \end{array} \right) ; \text{Methods}$$

$$\bullet (\text{Init} ; \text{Methods}) \triangle (\text{end\_mission\_app} . \text{FlatBufferMission} \longrightarrow \text{Skip})$$

end

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

**state** *State*

*buffer* :  $\mathbb{Z}$   
*t* : *testClass*

**state** *State*

**initial** *Init*

*State'*

*buffer'* = 0  
*t'* = *testClass*

**public** *bufferEmpty*  $\hat{=}$  **var** *ret* :  $\mathbb{B}$  •

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

**public** *cleanUp*  $\hat{=}$  **var** *ret* :  $\mathbb{B}$  •

(*ret* := **False**)

• **Skip**

**end**

**section** *FlatBufferMissionMethChan* **parents** *scj\_prelude, GlobalTypes, MissionId, SchedulableId*

**channel** *bufferEmptyCall* : *MissionID*  
**channel** *bufferEmptyRet* : *MissionID*  $\times$   $\mathbb{B}$

**channel** *cleanUpCall* : *MissionID*  
**channel** *cleanUpRet* : *MissionID*  $\times$   $\mathbb{B}$

**channel** *writeCall* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  $\times$   $\mathbb{Z}$   
**channel** *writeRet* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*

**channel** *readCall* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  
**channel** *readRet* : *MissionID*  $\times$  *SchedulableID*  $\times$  *ThreadID*  $\times$   $\mathbb{Z}$

## 5.2 Schedulables of FlatBufferMission

**section** *ReaderApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*  
*MissionMethChan*, *FlatBufferMissionMethChan*, *ObjectIds*, *ThreadIds*

**process** *ReaderApp*  $\hat{=}$   
*fbMission* : *MissionID* • **begin**

*Run*  $\hat{=}$   

$$\left( \begin{array}{l} \text{runCall} . \text{Reader} \longrightarrow \\ \left( \begin{array}{l} \mu X \bullet \\ \left( \begin{array}{l} \text{terminationPendingCall} . \text{fbMission} . \text{Reader} \longrightarrow \\ \text{terminationPendingRet} . \text{fbMission} . \text{Reader} ? \text{terminationPending} \longrightarrow \\ \mathbf{var} \text{ loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \mathbf{if} (\text{loopVar}) \longrightarrow \\ \left( \begin{array}{l} \mathbf{var} \text{ result} : \mathbb{Z} \bullet \text{result} := 999; \\ \left( \begin{array}{l} \text{binder\_readCall} . \text{fbMission} . \text{Reader} . \text{ReaderThread} \longrightarrow \\ \text{binder\_readRet} . \text{fbMission} . \text{Reader} . \text{ReaderThread} ? \text{read} \longrightarrow \end{array} \right) \mathbf{Skip} \\ \mathbf{Skip} \end{array} \right) \\ \parallel \neg (\text{loopVar}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \\ \mathbf{Skip} \end{array} \right) ; X \end{array} \right) \end{array} \right) ; \end{array} \right)$$

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

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

**end**



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

**state** *State*

*fbMission* : *FlatBufferMission*

**state** *State*

**initial** *Init*

*State'*

• **Skip**

**end**

**section** *WriterApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*

*MissionMethChan*, *FlatBufferMissionMethChan*, *ObjectIds*, *ThreadIds*

**process** *WriterApp*  $\hat{=}$   
*fbMission* : *MissionID* • **begin**

*Run*  $\hat{=}$

$$\left( \begin{array}{l} \text{runCall} . \text{Writer} \longrightarrow \\ \left( \begin{array}{l} \mu X \bullet \\ \left( \begin{array}{l} \text{terminationPendingCall} . \text{fbMission} . \text{Writer} \longrightarrow \\ \text{terminationPendingRet} . \text{fbMission} . \text{Writer} ? \text{terminationPending} \longrightarrow \\ \text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \text{if } (\text{loopVar}) \longrightarrow \\ \left( \begin{array}{l} \left( \begin{array}{l} \text{binder\_writeCall} . \text{fbMission} . \text{Writer} . \text{WriterThread} ! i \longrightarrow \\ \text{binder\_writeRet} . \text{fbMission} . \text{Writer} . \text{WriterThread} \longrightarrow \end{array} \right) ; \\ \text{Skip} \\ i := i + 1; \\ \text{var keepWriting} : \mathbb{B} \bullet \text{keepWriting} := (i \geq 5); \\ \text{if } (\neg \text{keepWriting} = \text{True}) \longrightarrow \\ \left( \begin{array}{l} \text{requestTerminationCall} . \text{fbMission} . \text{Writer} \longrightarrow \\ \text{requestTerminationRet} . \text{fbMission} . \text{Writer} ? \text{requestTermination} \longrightarrow \end{array} \right) ; X \\ \text{Skip} \\ \parallel \neg (\neg \text{keepWriting} = \text{True}) \longrightarrow \text{Skip} \\ \text{fi}; \\ \text{Skip} \\ \parallel \neg (\text{loopVar}) \longrightarrow \text{Skip} \\ \text{fi} \\ \text{Skip} \end{array} \right) \end{array} \right) \end{array} \right) \end{array} \right) ;$$

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

• (*Methods*)  $\triangle$  (*end\_managedThread\_app* . *Writer*  $\longrightarrow$  **Skip**)

**end**

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

**state** *State*

*fbMission* : *FlatBufferMission*

*i* :  $\mathbb{Z}$

**state** *State*

**initial** *Init*

*State*'

*i*' = 1

• **Skip**

**end**