Flatbuffer

Tight Rope v0.65

10th February 2016

1 ID Files

1.1 MissionIds

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

> *FlatBufferMissionMID* : *MissionID*
>
> ─────────
>
> *distinct⟨nullMissionId, FlatBufferMissionMID⟩*

1.2 SchedulablesIds

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

$FlatBufferMissionSequencerSID : SchedulableID$
$ReaderSID : SchedulableID$
$WriterSID : SchedulableID$

$distinct\langle nullSequencerId, nullSchedulableId, FlatBufferMissionSequencerSID,$
$ReaderSID, WriterSID\rangle$

1.3 ThreadIds

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

$$
\begin{array}{|l}
WriterTID : ThreadID \\
ReaderTID : ThreadID \\
\hline
distinct\langle SafeletTID, nullTID, \\
WriterTID, ReaderTID\rangle
\end{array}
$$

1.4 ObjectIds

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

$FlatBufferMissionOID : ObjectID$

$distinct\langle FlatBufferMissionOID \rangle$

2 Network

2.1 Network Channel Sets

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* ==
  $\bigcup\{$*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* $|\!\}$

2.2 MethodCallBinder

section *MethodCallBindingChannels* **parents** *scj_prelude*, *GlobalTypes*, *MissionId*, *SchedulableId*

channel *binder_readCall* : *MissionID* × *SchedulableID*
channel *binder_readRet* : *MissionID* × *SchedulableID* × ℤ

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

channel *binder_writeCall* : *MissionID* × *SchedulableID* × 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* ≙ **begin**

$read\_MethodBinder \mathrel{\widehat{=}}$
$$
\begin{pmatrix}
binder\_readCall \\
\quad ?\,loc : (loc \in readLocs) \\
\quad ?\,caller : (caller \in readCallers) \longrightarrow \\
readCall\,.\,loc\,.\,caller \longrightarrow \\
readRet\,.\,loc\,.\,caller\,?\,ret \longrightarrow \\
binder\_readRet\,.\,loc\,.\,caller\,!\,ret \longrightarrow \\
read\_MethodBinder
\end{pmatrix}
$$

$write\_MethodBinder \mathrel{\widehat{=}}$
$$
\begin{pmatrix}
binder\_writeCall \\
\quad ?\,loc : (loc \in writeLocs) \\
\quad ?\,caller : (caller \in writeCallers) \times \mathbb{Z} \longrightarrow \\
writeCall\,.\,loc\,.\,caller \times \mathbb{Z} \longrightarrow \\
writeRet\,.\,loc\,.\,caller \longrightarrow \\
binder\_writeRet\,.\,loc\,.\,caller \longrightarrow \\
write\_MethodBinder
\end{pmatrix}
$$

$BinderActions \mathrel{\widehat{=}}$
$$
\begin{pmatrix}
read\_MethodBinder \\
||| \\
write\_MethodBinder
\end{pmatrix}
$$

• *BinderActions* △ (*done_toplevel_sequencer* ⟶ **Skip**)

end

process *ApplicationB* ≙ *Application* ⟦ *MethodCallBinderSync* ⟧ *MethodCallBinder*

2.3   Locking

process *Threads* $\widehat{=}$
$$
\begin{pmatrix}
\textit{ThreadFW}(\textit{WriterTID}, 10) \\
\| \| \| \\
\textit{ThreadFW}(\textit{ReaderTID}, 10)
\end{pmatrix}
$$

process *Objects* $\widehat{=}$
$$
\big(\, \textit{ObjectFW}(\textit{FlatBufferMissionOID}) \big)
$$

process *Locking* $\widehat{=}$ *Threads* $[\![\, \textit{ThreadSync} \,]\!]$ *Objects*

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*,
  FlatBufferApp, *FlatBufferMissionSequencerApp*, *FlatBufferMissionApp*, *ReaderApp*, *WriterApp*

process $ControlTier \;\widehat{=}$
$$\begin{pmatrix} SafeletFW \\ \quad [\![ControlTierSync]\!] \\ TopLevelMissionSequencerFW\,(FlatBufferMissionSequencer) \end{pmatrix}$$

process $Tier0 \;\widehat{=}$
$$\begin{pmatrix} MissionFW\,(FlatBufferMissionID) \\ \quad [\![MissionSync]\!] \\ \begin{pmatrix} ManagedThreadFW\,(ReaderID) \\ \quad [\![SchedulablesSync]\!] \\ ManagedThreadFW\,(WriterID) \end{pmatrix} \end{pmatrix}$$

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

process $Application \;\widehat{=}$
$$\begin{pmatrix} FlatBufferApp \\ ||| \\ FlatBufferMissionSequencerApp \\ ||| \\ FlatBufferMissionApp \\ ||| \\ ReaderApp(FlatBufferMissionID) \\ ||| \\ WriterApp(FlatBufferMissionID) \end{pmatrix}$$

process $Program \;\widehat{=}\; \big(Framework\; [\![\, AppSync \,]\!] \; ApplicationB \big) \; [\![\, LockingSync \,]\!] \; Locking$

3 Safelet

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

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

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

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

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

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

end

4 Top Level Mission Sequencer

section *FlatBufferMissionSequencerApp* **parents** *TopLevelMissionSequencerChan*,
  MissionId, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*, *FlatBufferMissionSequencerClass*

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

State ─────────────────────────────
  this : **ref** *FlatBufferMissionSequencerClass*
─────────────────────────────────────

state *State*

Init ─────────────────────────────
  State′
  ─────────
  this′ = **new** *FlatBufferMissionSequencerClass*()
─────────────────────────────────────

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

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

• (*Init* ; *Methods*) $\triangle$ (*end_sequencer_app* . *FlatBufferMissionSequencerSID* $\longrightarrow$ **Skip**)

end

section *FlatBufferMissionSequencerClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *MethodCallE* , *MissionId*, *MissionIds*

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

---
state *State*
$returnedMission : \mathbb{B}$

---

state *State*

---
initial *Init*
$State'$
$returnedMission' = false$

---

protected *getNextMission* $\widehat{=}$ **var** *ret* : *MissionID* •
$$\left(\begin{array}{l} \textbf{if} \, (\neg \, returnedMission = \textbf{True}) \longrightarrow \\ \qquad \left(\begin{array}{l} this \, . \, returnedMission := \textbf{True}; \\ ret := FlatBufferMissionMID \end{array} \right) \\ [\!]\, \neg \, (\neg \, returnedMission = \textbf{True}) \longrightarrow \\ \qquad \left(ret := nullMissionId \right) \\ \textbf{fi} \end{array} \right)$$

• **Skip**

end

5 Missions

5.1 FlatBufferMission

section *FlatBufferMissionApp* **parents** *scj_prelude, MissionId, MissionIds,*
    SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, MethodCallBindingChannels, FlatBufferMissionC
    , ObjectChan, ObjectIds, ThreadIds, FlatBufferMissionMethChan

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

$\boxed{\begin{array}{l} \textit{State} \\ \hline \quad \textit{this} : \textbf{ref } \textit{FlatBufferMissionClass} \end{array}}$

state *State*

$\boxed{\begin{array}{l} \textit{Init} \\ \hline \quad \textit{State}' \\ \hline \quad \textit{this}' = \textbf{new } \textit{FlatBufferMissionClass}() \end{array}}$

InitializePhase $\widehat{=}$
$\begin{pmatrix} \textit{initializeCall . FlatBufferMissionMID} \longrightarrow \\ \textit{register ! ReaderSID ! FlatBufferMissionMID} \longrightarrow \\ \textit{register ! WriterSID ! FlatBufferMissionMID} \longrightarrow \\ \textit{initializeRet . FlatBufferMissionMID} \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

CleanupPhase $\widehat{=}$
$\begin{pmatrix} \textit{cleanupMissionCall . FlatBufferMissionMID} \longrightarrow \\ \textit{cleanupMissionRet . FlatBufferMissionMID ! } \textbf{True} \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

bufferEmptyMeth $\widehat{=}$ **var** *ret* : $\mathbb{B}$ •
$\begin{pmatrix} \textit{bufferEmptyCall . FlatBufferMissionMID} \longrightarrow \\ \textit{ret} := \textit{this . bufferEmpty}(); \\ \textit{bufferEmptyRet . FlatBufferMissionMID ! ret} \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

cleanUpMeth $\widehat{=}$ **var** *ret* : $\mathbb{B}$ •
$\begin{pmatrix} \textit{cleanUpCall . FlatBufferMissionMID} \longrightarrow \\ \textit{ret} := \textit{this . cleanUp}(); \\ \textit{cleanUpRet . FlatBufferMissionMID ! ret} \longrightarrow \\ \textbf{Skip} \end{pmatrix}$

$writeSyncMeth \,\widehat{=}$
$$\left(\begin{array}{l} writeCall\,.\,FlatBufferMissionMID\,?\,caller\,?\,thread\,?\,update \longrightarrow \\ \left(\begin{array}{l} startSyncMeth\,.\,FlatBufferMissionOID\,.\,thread \longrightarrow \\ lockAcquired\,.\,FlatBufferMissionOID\,.\,thread \longrightarrow \\ \left(\begin{array}{l} \left(\begin{array}{l} \mu X \bullet \\ \left(\begin{array}{l} \mathbf{var}\,loopVar : \mathbb{B} \bullet loopVar := (\neg\,bufferEmpty()); \\ \mathbf{if}\,(loopVar = \mathbf{True}) \longrightarrow \\ \qquad \left(\begin{array}{l} waitCall\,.\,FlatBufferMissionOID\,.\,thread \longrightarrow \\ waitRet\,.\,FlatBufferMissionOID\,.\,thread \longrightarrow \\ \mathbf{Skip} \end{array}\right)\;;\;\;X \\ [\!] \,(loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array}\right) \\ ; \\ this\,.\,buffer := update; \\ notify\,.\,FlatBufferMissionOID\,!\,thread \longrightarrow \\ \mathbf{Skip} \end{array}\right)\;; \\ endSyncMeth\,.\,FlatBufferMissionOID\,.\,thread \longrightarrow \\ writeRet\,.\,FlatBufferMissionMID\,.\,caller\,.\,thread \longrightarrow \\ \mathbf{Skip} \end{array}\right) \end{array}\right)$$

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

$$Methods \,\widehat{=}\, \left(\begin{array}{l} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ bufferEmptyMeth \\ \Box \\ cleanUpMeth \\ \Box \\ writeSyncMeth \\ \Box \\ readSyncMeth \end{array}\right)\;;\;\;Methods$$

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

$\mathbf{end}$

section *FlatBufferMissionClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *MethodCallBindingCh*

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

---
state *State*
> $buffer : \mathbb{Z}$
> $t_T EST : testClass$
---

state *State*

---
initial *Init*
> $State'$
>
> $buffer' = 0$
---

public *bufferEmpty* $\widehat{=}$ **var** $ret : \mathbb{B} \bullet$

$$\begin{pmatrix} \textbf{if } (buffer = 0) \longrightarrow \\ \qquad ret := \textbf{True} \\ [\!] \neg (buffer = 0) \longrightarrow \\ \qquad ret := \textbf{False} \\ \textbf{fi} \end{pmatrix}$$

public *cleanUp* $\widehat{=}$ **var** $ret : \mathbb{B} \bullet$

$$\begin{pmatrix} ret := \textbf{False} \end{pmatrix}$$

$\bullet$ **Skip**

end

section *FlatBufferMissionMethChan* **parents** *scj_prelude, GlobalTypes, MissionId, SchedulableId*

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

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

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

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

5.2 Schedulables of FlatBufferMission

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

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

$Run \widehat{=}$
$$
\begin{pmatrix}
runCall \, . \, ReaderSID \longrightarrow \\
\begin{pmatrix}
\begin{pmatrix}
\mu X \bullet \\
\begin{pmatrix}
terminationPendingCall \, . \, fbMission \longrightarrow \\
terminationPendingRet \, . \, fbMission \, ? \, terminationPending \longrightarrow \\
\textbf{var } loopVar : \mathbb{B} \bullet loopVar := (\neg \, terminationPending); \\
\textbf{if } (loopVar = \textbf{True}) \longrightarrow \\
\quad \begin{pmatrix}
\textbf{var } result : \mathbb{Z} \bullet result := -1; \\
\begin{pmatrix}
binder\_readCall \, . \, fbMission \, . \, ReaderSID \, . \, ReaderTID \longrightarrow \\
binder\_readRet \, . \, fbMission \, . \, ReaderSID \, . \, ReaderTID \, ? \, read \longrightarrow
\end{pmatrix}; \\
\end{pmatrix}; \\
\llbracket (loopVar = \textbf{False}) \longrightarrow \textbf{Skip} \\
\textbf{fi}
\end{pmatrix} ; \; X
\end{pmatrix}
\end{pmatrix} ; \\
runRet \, . \, ReaderSID \longrightarrow \\
\textbf{Skip}
\end{pmatrix}
$$

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

• (*Methods*) $\triangle$ (*end_managedThread_app* . *ReaderSID* $\longrightarrow$ **Skip**)

end

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

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

┌─ **state** *State* ─────────────────────────────
│ *fbMission_T EST* : *FlatBufferMission*
└──────────────────────────────────────

state *State*

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

• **Skip**

end

section *WriterApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*
, *MissionMethChan*, *FlatBufferMissionMethChan*, *ObjectIds*, *ThreadIds*

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

$Run \widehat{=}$
$\left(\begin{array}{l} runCall \,.\, WriterSID \longrightarrow \\ \left(\begin{array}{l} \mathbf{var}\ i : \mathbb{Z} \bullet i := 1; \\ \left(\begin{array}{l} \mu X \bullet \\ \left(\begin{array}{l} terminationPendingCall \,.\, fbMission \longrightarrow \\ terminationPendingRet \,.\, fbMission\,?\,terminationPending \longrightarrow \\ \mathbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ terminationPending); \\ \mathbf{if}\ (loopVar = \mathbf{True}) \longrightarrow \\ \quad \left(\begin{array}{l} \left(\begin{array}{l} binder\_writeCall \,.\, fbMission \,.\, WriterSID \,.\, WriterTID\,!\,i \longrightarrow \\ binder\_writeRet \,.\, fbMission \,.\, WriterSID \,.\, WriterTID \longrightarrow \\ \mathbf{Skip} \end{array} \right); \\ i := i + 1; \\ \mathbf{if}\ (i \geq 5) \longrightarrow \\ \quad \left(\begin{array}{l} requestTerminationCall \,.\, fbMission \longrightarrow \\ requestTerminationRet \,.\, fbMission\,?\,requestTermination \longrightarrow \end{array} \right) \\ [\!] \neg\ (i \geq 5) \longrightarrow \mathbf{Skip} \\ \mathbf{fi}\,; \end{array} \right);\ X \\ [\!]\ (loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array} \right) \end{array} \right) \\ runRet \,.\, WriterSID \longrightarrow \\ \mathbf{Skip} \end{array} \right)\ ;$

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

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

end

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

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

___ **state** *State* _____
$\quad fbMission_T EST : FlatBufferMission$
_____

state *State*

___ **initial** *Init* _____
$\quad State'$
_____
_____

• **Skip**

end