Flatbuffer

Tight Rope v0.65

4th February 2016

1 ID Files

1.1 MissionIds

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

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

1.2 SchedulablesIds

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

FlatBufferMissionSequencerID : *SchedulableID*
ReaderID : *SchedulableID*
WriterID : *SchedulableID*

distinct⟨nullSequencerId, nullSchedulableId, FlatBufferMissionSequencerIDID,
ReaderID, WriterID⟩

1.3   ThreadIds

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

$ReaderThreadID : ThreadID$
$WriterThreadID : ThreadID$

$distinct\langle SafeletThreadId, nullThreadId,$
$ReaderThreadID, WriterThreadID\rangle$

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

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

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

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

BinderActions ≙
$$\begin{pmatrix} read\_MethodBinder \\ ||| \\ write\_MethodBinder \end{pmatrix}$$

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

end

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

2.3   Locking

process *Threads* $\widehat{=}$
$$\begin{pmatrix} \mathit{ThreadFW}\,(\mathit{ReaderThreadID}, 10) \\ ||| \\ \mathit{ThreadFW}\,(\mathit{WriterThreadID}, 10) \end{pmatrix}$$

process *Objects* $\widehat{=}$
$$\begin{pmatrix} \mathit{ObjectFW}\,(\mathit{FlatBufferObjectID}) \\ ||| \\ \mathit{ObjectFW}\,(\mathit{FlatBufferMissionObjectID}) \\ ||| \\ \mathit{ObjectFW}\,(\mathit{ReaderObjectID}) \\ ||| \\ \mathit{ObjectFW}\,(\mathit{WriterObjectID}) \end{pmatrix}$$

process *Locking* $\widehat{=}$ *Threads* $[\![$ *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*

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

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

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

$immortalMemorySizeMeth \widehat{=} \textbf{var}\ ret : \mathbb{Z} \bullet$
$$\begin{pmatrix} immortalMemorySizeCall\,.\,FlatBuffer \longrightarrow \\ \left(ret := 1000000 \right); \\ immortalMemorySizeRet\,.\,FlatBuffer\,!\,ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

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

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

end

4 Top Level Mission Sequencer

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

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

_____
State
  this : **ref** *FlatBufferMissionSequencerClass*
_____

state *State*

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

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

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

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

end

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

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

state *State*

---
initial *Init* _____
   State ′
   _____
   returnedMission ′ = *false*
_____

protected *getNextMission* $\widehat{=}$ **var** *ret* : *MissionID* •
$$\begin{pmatrix} \mathbf{if} \, (\neg \, returnedMission = \mathbf{True}) \longrightarrow \\ \quad \begin{pmatrix} this \, . \, returnedMission := true; \\ ret := FlatBufferMission \end{pmatrix} \\ [\!] \, (\neg \, returnedMission = \mathbf{True}) \longrightarrow \\ \quad \begin{pmatrix} ret := nullMissionId \end{pmatrix} \\ \mathbf{fi} \end{pmatrix}$$

• **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* $\widehat{=}$ **begin**

State
> *this* : **ref** *FlatBufferMissionClass*
---

state *State*

Init
> *State′*
>
> *this′* = **new** *FlatBufferMissionClass*()

InitializePhase $\widehat{=}$
$$\begin{pmatrix} initializeCall \, . \, FlatBufferMission \longrightarrow \\ register \, ! \, Reader \, ! \, FlatBufferMission \longrightarrow \\ register \, ! \, Writer \, ! \, FlatBufferMission \longrightarrow \\ initializeRet \, . \, FlatBufferMission \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

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

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

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

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

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

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

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

$\mathbf{end}$

13

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

---
state *State*
| $buffer : \mathbb{Z}$ |
| $t : testClass$ |
---

state *State*

---
initial *Init*
$State'$
$buffer' = 0$
$t' = testClass$
---

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

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

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

$\big(ret := \textbf{False} \big)$

$\bullet$ **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*

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

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

$Run \widehat{=}$
$\begin{pmatrix} runCall \, . \, Reader \longrightarrow \\ \begin{pmatrix} \begin{pmatrix} \mu X \bullet \\ \begin{pmatrix} terminationPendingCall \, . \, fbMission \, . \, Reader \longrightarrow \quad terminationPendingRet \, . \, fbMission \, . \, Reader \, ? \, terminationPend \\ \mathbf{var} \, loopVar : \mathbb{B} \bullet loopVar := (\neg \, terminationPending); \\ \mathbf{if} \, (loopVar = \mathbf{True}) \longrightarrow \\ \quad \begin{pmatrix} \mathbf{var} \, result : \mathbb{Z} \bullet result := 999; \\ (binder\_readCall \, . \, fbMission \, . \, Reader \, . \, ReaderThreadID \longrightarrow \quad binder\_readRet \, . \, fbMission \, . \, Reader \, . \, ReaderT \\ [\!] \, (loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{pmatrix} \\ \mathbf{Skip} \end{pmatrix} \\ runRet \, . \, Reader \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

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

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

end

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

    ┌─ **state** *State* ───
    │ *fbMission* : *FlatBufferMission*
    └───

state *State*

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

• **Skip**

end

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

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

$Run \widehat{=}$
$\Big($ *runCall . Writer* $\longrightarrow$
  $\Big($ $\Big($ μX •
      $\Big($ *terminationPendingCall . fbMission . Writer* $\longrightarrow$ *terminationPendingRet . fbMission . Writer* ? *terminationPend*
      var *loopVar* : $\mathbb{B}$ • *loopVar* := ($\neg$ *terminationPending*);
      if (*loopVar* = **True**) $\longrightarrow$
          $\Big($ (*binder_writeCall . fbMission . Writer . WriterThreadID* ! i $\longrightarrow$ *binder_writeRet . fbMission . Writer . Wri*
          $i := i + 1$;
          var *keepWriting* : $\mathbb{B}$ • *keepWriting* := *false*;
          if ($i \geq 5$) $\longrightarrow$
              $\big($ *this . keepWriting* := *true* $\big)$
          $[\!]$ ($i \geq 5$) $\longrightarrow$
              $\big($ *this . keepWriting* := *false* $\big)$
          fi ;
          if ($\neg$ *keepWriting* = **True**) $\longrightarrow$
              $\big($ *requestTerminationCall . fbMission . Writer* $\longrightarrow$ *requestTerminationRet . fbMission . Writer* ? *request*
          $[\!]$ ($\neg$ *keepWriting* = **True**) $\longrightarrow$ **Skip**
          fi ;
          Skip
      $[\!]$ (*loopVar* = **False**) $\longrightarrow$ **Skip**
      fi
  Skip
 runRet . Writer $\longrightarrow$
Skip

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

• ($Methods$) $\triangle$ (*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