Flatbuffer

Tight Rope v0.6

October 24, 2015

1 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* ., *done_mission* .,
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 *ObjectSync* ==
{| |}

channelset *ThreadSync* ==
{| |}

channelset *LockingSync* ==
{| *lockAcquired*, *startSyncMeth*, *endSyncMeth*, *waitCall*, *waitRet*, *notify* |}

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

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

process *Tier0* $\widehat{=}$
$$\begin{pmatrix} MissionFW\,(FlatBufferMission) \\ \quad [\![MissionSync]\!] \\ \begin{pmatrix} ManagedThreadFW\,(Reader) \\ \quad [\![SchedulablesSync]\!] \\ ManagedThreadFW\,(Writer) \end{pmatrix} \end{pmatrix}$$

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

process *Application* $\widehat{=}$
$$\begin{pmatrix} FlatBufferApp \\ ||| \\ FlatBufferMissionSequencerApp \\ ||| \\ FlatBufferMissionApp(hijac.tools.tightrope.environments.VariableEnv \bullet 45ca843) \\ ||| \\ ReaderApp(FlatBufferMission) \\ ||| \\ WriterApp(FlatBufferMission) \end{pmatrix}$$

Locking $\widehat{=}$
$$\begin{pmatrix} \begin{pmatrix} ThreadFW\,(ReaderThread, MinPriority) \\ \quad [\![ThreadSync]\!] \\ ThreadFW\,(WriterThread, MinPriority) \end{pmatrix} \\ ||| \\ \begin{pmatrix} ObjectFW\,(FlatBufferObject) \\ \quad [\![ObjectSync]\!] \\ ObjectFW\,(FlatBufferMissionObject) \\ \quad [\![ObjectSync]\!] \\ ObjectFW\,(ReaderObject) \\ \quad [\![ObjectSync]\!] \\ ObjectFW\,(WriterObject) \end{pmatrix} \end{pmatrix}$$

process *Program* $\widehat{=}$ *Framework* $[\![$ *AppSync* $]\!]$ *Application* $[\![$ *LockingSync* $]\!]$ *Locking*

2 ID Files

2.1 MissionIds

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

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

2.2 SchedulablesIds

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

> *FlatBufferMissionSequencer* : *SchedulableID*
> *Reader* : *SchedulableID*
> *Writer* : *SchedulableID*
>
> ───────────
>
> *distinct⟨nullSequencerId, nullSchedulableId, Reader,*
> *Writer⟩*

2.3 ThreadIds

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

> *ReaderThread* : *ThreadID*
> *WriterThread* : *ThreadID*
>
> ───────────
>
> *distinct⟨SafeletThreadId, nullThreadId,*
> *ReaderThread,*
> *WriterThread⟩*

2.4 ObjectIds

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

> *FlatBufferObject* : *ObjectID*
> *FlatBufferMissionObject* : *ObjectID*
> *ReaderObject* : *ObjectID*
> *WriterObject* : *ObjectID*
>
> ───────────
>
> *distinct⟨FlatBufferObject,*
> *FlatBufferMissionObject,*
> *ReaderObject,*
> *WriterObject⟩*

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

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

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

end

4 Top Level Mission Sequencer

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

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

State
___

$\quad$ *this* : **ref** *FlatBufferMissionSequencerClass*
___

state *State*

Init
___

$\quad$ *State′*
___

$\quad$ *this′* = **new** *FlatBufferMissionSequencerClass*()
___

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

Methods $\widehat{=}$
$\begin{pmatrix} GetNextMission \end{pmatrix} ; \; Methods$

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

end

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

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

---

state *State*

---
initial *Init* _____
$\quad$ *State'*
$\quad$ —————
$\quad$ *returnedMission'* = *false*

---

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

$\bullet$ **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* $\widehat{=}$ *storageParameters* : *MissionID* $\bullet$ **begin**

__ *State* _____
$\quad$ *this* : **ref** *FlatBufferMissionClass*
_____

state *State*

__ *Init* _____
$\quad$ *State'*
$\quad$_____
$\quad$ *this'* = **new** *FlatBufferMissionClass*()
_____

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

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

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

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

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

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

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

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

$\mathbf{end}$

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

---
state *State* ————————————————————————————
  buffer : $\mathbb{Z}$
---

state *State*

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

public *bufferEmpty* $\widehat{=}$ **var** *ret* : $\mathbb{B}$ •
$$\begin{pmatrix} \textbf{if } (buffer = 0) \longrightarrow \\ \quad ret := \textbf{True} \\ [\!] \neg (buffer = 0) \longrightarrow \\ \quad ret := \textbf{False} \\ \textbf{fi} \end{pmatrix}$$

public *cleanUp* $\widehat{=}$ **var** *ret* : $\mathbb{B}$ •
$\big(ret := \textbf{False} \big)$

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

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

5.2 Schedulables of

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

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

$Run \;\widehat{=}$
$$
\begin{pmatrix}
runCall \, . \, Reader \longrightarrow \\
\begin{pmatrix}
\begin{pmatrix}
\begin{pmatrix}
\mu X \bullet \\
\begin{pmatrix}
terminationPendingCall \, . \, fbMission \longrightarrow \\
terminationPendingRet \, . \, fbMission \, ? \, terminationPending \longrightarrow \\
\mathbf{var} \; loopVar : \mathbb{B} \bullet loopVar := (\neg \; terminationPending); \\
\mathbf{if} \, (loopVar) \longrightarrow \\
\quad \begin{pmatrix}
\mathbf{var} \; result : \mathbb{Z} \bullet result := 999; \\
\begin{pmatrix}
readCall \, . \, fbMission \, . \, ReaderThread \longrightarrow \\
readRet \, . \, fbMission \, . \, ReaderThread \, ? \, read \longrightarrow \\
\mathbf{Skip}
\end{pmatrix} \mathbf{Skip} \\
\mathbf{Skip}
\end{pmatrix} \; ; \; X \\
\llbracket \; \neg \, (loopVar) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{pmatrix} \\
\mathbf{Skip}
\end{pmatrix} \; ; \\
runRet \, . \, Reader \longrightarrow \\
\mathbf{Skip}
\end{pmatrix}
\end{pmatrix}
$$

$Methods \;\widehat{=}$
$\begin{pmatrix} Run \end{pmatrix} \; ; \; Methods$

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

end

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

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


```
┌─ State ──────────────────────────────────────────────────
│ i : ℤ
└──────────────────────────────────────────────────────────
```


state *State*


```
┌─ Init ───────────────────────────────────────────────────
│ State′
├──────────────────────────────────────────────────────────
│ i′ = 1
└──────────────────────────────────────────────────────────
```


$Run \widehat{=}$
$$
\left(
\begin{array}{l}
runCall . Writer \longrightarrow \\
\left(
\left(
\begin{array}{l}
\mu X \bullet \\
\left(
\begin{array}{l}
terminationPendingCall . fbMission \longrightarrow \\
terminationPendingRet . fbMission ? terminationPending \longrightarrow \\
\mathbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ terminationPending); \\
\mathbf{if}\ (loopVar) \longrightarrow \\
\quad
\left(
\begin{array}{l}
\left(
\begin{array}{l}
writeCall . fbMission . WriterThread\ !\ i \longrightarrow \\
writeRet . fbMission . WriterThread \longrightarrow \\
\mathbf{Skip}
\end{array}
\right) ; \\
i := i + 1; \\
\mathbf{var}\ keepWriting : \mathbb{B} \bullet keepWriting := (i \geq 5); \\
\mathbf{if}\ (\neg\ keepWriting = \mathbf{True}) \longrightarrow \\
\quad
\left(
\begin{array}{l}
requestTerminationCall . fbMission \longrightarrow \\
requestTerminationRet . fbMission ? requestTermination \longrightarrow \\
\mathbf{Skip}
\end{array}
\right) \\
[\!]\ \neg\ (\neg\ keepWriting = \mathbf{True}) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}; \\
\mathbf{Skip}
\end{array}
\right) \\
[\!]\ \neg\ (loopVar) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{array}
\right) ; X
\end{array}
\right) \\
\mathbf{Skip}
\end{array}
\right) ; \\
runRet . Writer \longrightarrow \\
\mathbf{Skip}
\end{array}
\right)
$$

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

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

end