Flatbuffer

Tight Rope v0.6

September 27, 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 $ControlTier \mathrel{\widehat=}$
$$\left(\begin{array}{l} SafeletFW \\ \quad [\![ControlTierSync]\!] \\ TopLevelMissionSequencerFW(FlatBufferMissionSequencer) \end{array}\right)$$

process $Tier0 \mathrel{\widehat=}$
$$\left(\begin{array}{l} MissionFW(FlatBufferMission) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} ManagedThreadFW(Reader) \\ \quad [\![SchedulablesSync]\!] \\ ManagedThreadFW(Writer) \end{array}\right) \end{array}\right)$$

process $Framework \mathrel{\widehat=}$
$$\left(\begin{array}{l} ControlTier \\ \quad [\![TierSync]\!] \\ (\, Tier0) \end{array}\right)$$

process $Application \mathrel{\widehat=}$
$$\left(\begin{array}{l} FlatBufferApp \\ ||| \\ FlatBufferMissionSequencerApp \\ ||| \\ FlatBufferMissionApp \\ ||| \\ ReaderApp(FlatBufferMission) \\ ||| \\ WriterApp(FlatBufferMission) \end{array}\right)$$

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

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

2 ID Files

2.1 MissionIds

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

$$FlatBufferMission : MissionID$$

$$distinct\langle nullMissionId, FlatBufferMission\rangle$$

2.2 SchedulablesIds

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

$$FlatBufferMissionSequencer : SchedulableID$$
$$Reader : SchedulableID$$
$$Writer : SchedulableID$$

$$distinct\langle nullSequencerId, nullSchedulableId, Reader, Writer\rangle$$

2.3 ThreadIds

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

$$ReaderThread : ThreadID$$
$$WriterThread : ThreadID$$

$$distinct\langle SafeletThreadId, nullThreadId, ReaderThread, WriterThread\rangle$$

2.4 ObjectIds

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

$$FlatBufferObject : ObjectID$$
$$FlatBufferMissionObject : ObjectID$$
$$ReaderObject : ObjectID$$
$$WriterObject : ObjectID$$

$$distinct\langle FlatBufferObject, FlatBufferMissionObject, ReaderObject, WriterObject\rangle$$

3 Safelet

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

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

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

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

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

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

end

4 Top Level Mission Sequencer

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

process $FlatBufferMissionSequencerApp \mathrel{\widehat=} \textbf{begin}$

State

$this : \textbf{ref}\ FlatBufferMissionSequencerClass$

state *State*

Init

$State'$

$this' = \textbf{new}\ FlatBufferMissionSequencerClass()$

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

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

$\bullet\ (Init\ ;\ Methods) \triangle (end\_sequencer\_app\ .\ FlatBufferMissionSequencer \longrightarrow \textbf{Skip})$

end

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

state *State*

$$returnedMission : \mathbb{B}$$

state *State*

initial *Init*

$$State'$$

$$returnedMission' = false$$

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

$$\begin{pmatrix} \textbf{if } (\neg\ returnedMission = \textbf{True}) \longrightarrow \\ \quad \begin{pmatrix} this\ .\ returnedMission := true; \\ ret := FlatBufferMission \end{pmatrix} \\ [] \neg\ (\neg\ returnedMission = \textbf{True}) \longrightarrow \\ \quad \big(ret := nullMissionId\big) \\ \textbf{fi} \end{pmatrix}$$

• **Skip**

end

5 Missions

5.1 FlatBufferMission

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

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

State

$this : \mathbf{ref}\ FlatBufferMissionClass$

state *State*

Init

$State'$

$this' = \mathbf{new}\ FlatBufferMissionClass()$

$$InitializePhase \mathrel{\widehat=} \left(\begin{array}{l} initializeCall\,.\,FlatBufferMission \longrightarrow \\ register\,!\,Reader\,!\,FlatBufferMission \longrightarrow \\ register\,!\,Writer\,!\,FlatBufferMission \longrightarrow \\ initializeRet\,.\,FlatBufferMission \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$$CleanupPhase \mathrel{\widehat=} \left(\begin{array}{l} cleanupMissionCall\,.\,FlatBufferMission \longrightarrow \\ cleanupMissionRet\,.\,FlatBufferMission\,!\,\mathbf{False} \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

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

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

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

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

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

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

end

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

state *State*

$buffer : \mathbb{Z}$

state *State*

initial *Init*

$State'$

$buffer' = 0$

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

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

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

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

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

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

5.2 Schedulables of

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

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

$$
\begin{array}{l}
Run \mathrel{\widehat=} \\
\left(\begin{array}{l}
runCall\,.\,Reader \longrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
\mu X \bullet \\
\left(\begin{array}{l}
terminationPendingCall\,.\,fbMission \longrightarrow \\
terminationPendingRet\,.\,fbMission\,?\,terminationPending \longrightarrow \\
\quad \textbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ terminationPending); \\
\textbf{if}\ (loopVar) \longrightarrow \\
\qquad \left(\begin{array}{l}
\textbf{var}\ result : \mathbb{Z} \bullet result := 999; \\
\left(\begin{array}{l}
\left(\begin{array}{l}
readCall\,.\,fbMission\,.\,ReaderThread \longrightarrow \\
readRet\,.\,fbMission\,.\,ReaderThread\,?\,read \longrightarrow
\end{array}\right) \textbf{Skip} \\
\textbf{Skip}
\end{array}\right)
\end{array}\right) \ ;\ X \\
[\!]\ \neg\ (loopVar) \longrightarrow \textbf{Skip} \\
\textbf{fi}
\end{array}\right) \\
\textbf{Skip}
\end{array}\right) \\
runRet\,.\,Reader \longrightarrow \\
\textbf{Skip}
\end{array}\right) ;
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat=} \\
\left(Run\right) \ ;\ Methods
\end{array}
$$

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

end

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

channel $testCall : SchedulableID \times \mathbb{Z}$
channel $testRet : SchedulableID \times \mathbb{B}$

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

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

State

$i : \mathbb{Z}$

state *State*

Init

$State'$

$i' = 1$

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

$$
\begin{array}{l}
Methods \mathrel{\widehat=} \\
\left(Run\right);\ Methods
\end{array}
$$

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

end