Flatbuffer

Tight Rope v0.65

16th February 2016

1 ID Files

1.1 MissionIds

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

$$FlatBufferMissionMID : MissionID$$

$$distinct\langle nullMissionId, FlatBufferMissionMID\rangle$$

1.2 SchedulablesIds

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

$$
\begin{array}{l}
FlatBufferMissionSequencerSID : SchedulableID \\
ReaderSID : SchedulableID \\
WriterSID : SchedulableID \\
\hline
distinct\langle nullSequencerId, nullSchedulableId, FlatBufferMissionSequencerSID, \\
ReaderSID, WriterSID\rangle
\end{array}
$$

1.3 ThreadIds

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

$$WriterTID : ThreadID$$
$$ReaderTID : ThreadID$$

$$distinct\langle SafeletTID, nullTID, WriterTID, ReaderTID\rangle$$

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*, *FrameworkChan*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *ThreadIds*, *FrameworkChan*

channel $binder\_readCall : MissionID \times SchedulableID \times ThreadID$
channel $binder\_readRet : MissionID \times SchedulableID \times ThreadID \times \mathbb{Z}$

$readLocs == \{FlatBufferMissionMID\}$
$readCallers == \{ReaderSID\}$

channel $binder\_writeCall : MissionID \times SchedulableID \times ThreadID \times \mathbb{Z}$
channel $binder\_writeRet : MissionID \times SchedulableID \times ThreadID$

$writeLocs == \{FlatBufferMissionMID\}$
$writeCallers == \{WriterSID\}$

channelset $MethodCallBinderSync == \{\!|\ done\_toplevel\_sequencer,$
$binder\_readCall, binder\_readRet,$
$binder\_writeCall, binder\_writeRet\ |\!\}$

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

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

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

$$BinderActions \mathrel{\widehat=} \left(\begin{array}{l} read\_MethodBinder \\ |||\\ write\_MethodBinder \end{array}\right)$$

• $BinderActions \triangle (done\_toplevel\_sequencer \longrightarrow \mathbf{Skip})$

end

process $ApplicationB \mathrel{\widehat=} Application \llbracket\ MethodCallBinderSync\ \rrbracket\ MethodCallBinder$

2.3 Locking

process $Threads \mathrel{\widehat{=}}$
$\left(\begin{array}{l} ThreadFW(WriterTID, 10) \\ ||| \\ ThreadFW(ReaderTID, 10) \end{array}\right)$

process $Objects \mathrel{\widehat{=}}$
$\big(ObjectFW(FlatBufferMissionOID)\big)$

process $Locking \mathrel{\widehat{=}} Threads \llbracket ThreadSync \rrbracket Objects$

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

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

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

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

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

3 Safelet

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

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

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

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

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

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

end

4 Top Level Mission Sequencer

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

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

State

$this : \textbf{ref}\ FlatBufferMissionSequencerClass$

state *State*

Init

$State'$

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

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

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

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

end

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

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

state *State*

$$returnedMission : \mathbb{B}$$

state *State*

initial *Init*

$$State'$$

$$returnedMission' = \textbf{False}$$

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

$$\left(\begin{array}{l} \textbf{if } (\neg\ returnedMission = \textbf{True}) \longrightarrow \\ \quad \left(\begin{array}{l} this\,.\,returnedMission := \textbf{True}; \\ ret := FlatBufferMissionMID \end{array}\right) \\ [\!]\ \neg\ (\neg\ returnedMission = \textbf{True}) \longrightarrow \\ \quad \left(ret := nullMissionId\right) \\ \textbf{fi} \end{array}\right)$$

$\bullet$ **Skip**

end

5 Missions

5.1 FlatBufferMission

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

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

State

$this : \mathbf{ref}\ FlatBufferMissionClass$

state *State*

Init

$State'$

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

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

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

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

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

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

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

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

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

end

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

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

state *State*

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

state *State*

initial *Init*

$$State'$$

$$buffer' = 0$$
$$t' = testClass$$

public *bufferEmpty* $\widehat{=}$ **var** $ret : \mathbb{B} \bullet$
$$\left(\begin{array}{l} \textbf{if } (buffer = 0) \longrightarrow \\ \quad ret := \textbf{True} \\ [] \neg (buffer = 0) \longrightarrow \\ \quad ret := \textbf{False} \\ \textbf{fi} \end{array}\right)$$

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 $writeCall : MissionID \times SchedulableID \times ThreadID \times \mathbb{Z}$
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*, *MethodCallBindingChannels*, *MissionMethChan*, *FlatBufferMissionMethChan*, *ObjectIds*, *ThreadIds*

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

$$
\begin{array}{l}
Run \mathrel{\widehat=} \\
\left(\begin{array}{l}
runCall\,.\,ReaderSID \longrightarrow \\
\left(\begin{array}{l}
\mu X \bullet \\
\left(\begin{array}{l}
terminationPendingCall\,.\,fbMission \longrightarrow \\
terminationPendingRet\,.\,fbMission\,?\,terminationPending \longrightarrow \\
\textbf{Skip}; \\
\textbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ terminationPending); \\
\textbf{if}\ (loopVar = \textbf{True}) \longrightarrow \\
\qquad \left(\begin{array}{l}
\textbf{var}\ result : \mathbb{Z} \bullet result := 0; \\
binder\_readCall\,.\,fbMission\,.\,ReaderSID\,.\,ReaderTID \longrightarrow \\
binder\_readRet\,.\,fbMission\,.\,ReaderSID\,.\,ReaderTID\,?\,read \longrightarrow \\
\textbf{Skip};
\end{array}\right) ;\ X \\
[\!]\ (loopVar = \textbf{False}) \longrightarrow \textbf{Skip} \\
\textbf{fi}
\end{array}\right)
\end{array}\right) ; \\
runRet\,.\,ReaderSID \longrightarrow \\
\textbf{Skip}
\end{array}\right)
\end{array}
$$

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

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

end

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

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

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

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

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

end