Flatbuffer

Tight Rope v0.65

5th 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*

$$ReaderThreadID : ThreadID$$
$$WriterThreadID : ThreadID$$

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

1.4 ObjectIds

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

$$
\begin{array}{|l}
FlatBufferObjectID : ObjectID \\
FlatBufferMissionObjectID : ObjectID \\
ReaderObjectID : ObjectID \\
WriterObjectID : ObjectID \\
\hline
distinct\langle FlatBufferObjectID, FlatBufferMissionObjectID, \\
ReaderObjectID, WriterObjectID\rangle
\end{array}
$$

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

$readLocs == \{FlatBufferMission\}$
$readCallers == \{Reader\}$

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

$writeLocs == \{FlatBufferMission\}$
$writeCallers == \{Writer\}$

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

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

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

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

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

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

end

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

2.3 Locking

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

process $Objects \mathrel{\widehat{=}}$
$$\left(\begin{array}{l} ObjectFW(FlatBufferObjectID) \\ ||| \\ ObjectFW(FlatBufferMissionObjectID) \\ ||| \\ ObjectFW(ReaderObjectID) \\ ||| \\ ObjectFW(WriterObjectID) \end{array}\right)$$

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*

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

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

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

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

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

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

end

4 Top Level Mission Sequencer

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

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

State

$this : \mathbf{ref}\ FlatBufferMissionSequencerClass$

state *State*

Init

$State'$

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

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

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

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

end

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

state *State*

$$returnedMission : \mathbb{B}$$

state *State*

initial *Init*

$$State'$$

$$returnedMission' = false$$

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

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

• **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

$State'$

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

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

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

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

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

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

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

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

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

end

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

state *State*

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

state *State*

initial *Init*

$State'$

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

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

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

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

Run $\widehat{=}$
$\Big($ *runCall* . *ReaderSID* $\longrightarrow$
$\Big($ $\Big($ $\Big($ μX •
$\Big($ *terminationPendingCall* . *fbMission* . *Reader* $\longrightarrow$ *terminationPendingRet* . *fbMission* . *Reader* ? *terminationPend*
var *loopVar* : $\mathbb{B}$ • *loopVar* := $(\neg$ *terminationPending*$)$;
if $($*loopVar* = **True**$)$ $\longrightarrow$
$\Big($ **var** *result* : $\mathbb{Z}$ • *result* := 999;
$\Big($ *binder_readCall* . *fbMission* . *Reader* . *ReaderThreadID* $\longrightarrow$ *binder_readRet* . *fbMission* . *Reader* . *ReaderT*
$[]$ $($*loopVar* = **False**$)$ $\longrightarrow$ **Skip**
fi
Skip
runRet . *ReaderSID* $\longrightarrow$
Skip

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

• $($*Methods*$)$ $\triangle$ $($*end_managedThread_app* . *ReaderSID* $\longrightarrow$ **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 $\widehat{=}$
runCall . *WriterSID* $\longrightarrow$
μX •
terminationPendingCall . *fbMission* . *Writer* $\longrightarrow$ *terminationPendingRet* . *fbMission* . *Writer* ? *terminationPend*
var *loopVar* : $\mathbb{B}$ • *loopVar* := $(\neg$ *terminationPending*);
if (*loopVar* = **True**) $\longrightarrow$
(*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) $\longrightarrow$
(*this* . *keepWriting* := *true*)
$[]$ (*i* $\geq$ 5) $\longrightarrow$
(*this* . *keepWriting* := *false*)
fi ;
if ($\neg$ *keepWriting* = **True**) $\longrightarrow$
(*requestTerminationCall* . *fbMission* . *Writer* $\longrightarrow$ *requestTerminationRet* . *fbMission* . *Writer* ? *request*
$[]$ ($\neg$ *keepWriting* = **True**) $\longrightarrow$ **Skip**
fi ;
Skip
$[]$ (*loopVar* = **False**) $\longrightarrow$ **Skip**
fi
Skip
runRet . *WriterSID* $\longrightarrow$
Skip

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

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

end

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

state *State*

$fbMission : FlatBufferMission$
$i : \mathbb{Z}$

state *State*

initial *Init*

$State'$

$i' = 1$

- **Skip**

end