Flatbuffer

Tight Rope v0.6

23rd January 2016

1 ID Files

1.1 MissionIds

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

$$FlatBufferMissionID : MissionID$$

$$distinct\langle nullMissionId, FlatBufferMissionID\rangle$$

1.2 SchedulablesIds

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

$$
\begin{array}{l}
FlatBufferMissionSequencerID : SchedulableID \\
ReaderID : SchedulableID \\
WriterID : SchedulableID \\
\hline
distinct\langle nullSequencerId, nullSchedulableId, FlatBufferMissionSequencerID, \\
ReaderID, WriterID\rangle
\end{array}
$$

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

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* |}

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

channelset $MethodCallBinderSync == \{\!|\ done\_toplevel\_sequencer,\ |\!\}$

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

$BinderActions \mathrel{\widehat{=}}$
)(

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

end

process $ApplicationB \mathrel{\widehat{=}} Application \ [\![\ MethodCallBinderSync\ ]\!]\ MethodCallBinder$

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

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

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

$$\textbf{process}\ \mathit{Program} \mathrel{\widehat{=}} \big(\mathit{Framework} \llbracket\ \mathit{AppSync}\ \rrbracket \mathit{ApplicationB}\big) \llbracket\ \mathit{LockingSync}\ \rrbracket \mathit{Locking}$$

3 Safelet

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

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

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

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

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

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

end

4 Top Level Mission Sequencer

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

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

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

• **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

$State'$

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

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

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

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

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

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

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

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

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

end

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

state *State*

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

state *State*

initial *Init*

$State'$

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

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

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

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

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

5.2 Schedulables of FlatBufferMission

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

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

$Run \mathrel{\widehat=}$
$\Big($ *runCall* . *Reader* $\longrightarrow$
$\Big($ $\Big($ $\Big($ $\mu X \bullet$
$\Big($ *terminationPendingCall* . *fbMission* . *Reader* $\longrightarrow$
terminationPendingRet . *fbMission* . *Reader* ? *terminationPending* $\longrightarrow$
var *loopVar* : $\mathbb{B} \bullet$ *loopVar* := $(\neg$ *terminationPending*);
if (*loopVar*) $\longrightarrow$
$\Big($ **var** *result* : $\mathbb{Z} \bullet$ *result* := 999;
$\Big($ $\Big($ *binder_readCall* . *fbMission* . *Reader* . *ReaderThread* $\longrightarrow$
binder_readRet . *fbMission* . *Reader* . *ReaderThread* ? *read* $\longrightarrow$ $\Big)$ **Skip** $\Big)$; X
$\Big)$
$[]$ $\neg$ (*loopVar*) $\longrightarrow$ **Skip**
fi
$\Big)$
Skip
runRet . *Reader* $\longrightarrow$
Skip
$\Big)$;

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

$\bullet$ (*Methods*) $\triangle$ (*end_managedThread_app* . *Reader* $\longrightarrow$ **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 $\widehat{=}$
$\Big($ *runCall* . *Writer* $\longrightarrow$
$\Big($ $\Big($ $\Big($ μX •
$\Big($ *terminationPendingCall* . *fbMission* . *Writer* $\longrightarrow$
terminationPendingRet . *fbMission* . *Writer* ? *terminationPending* $\longrightarrow$
var *loopVar* : $\mathbb{B}$ • *loopVar* := $(\neg$ *terminationPending*);
if $($*loopVar*$)$ $\longrightarrow$
$\Big($ $\Big($ *binder_writeCall* . *fbMission* . *Writer* . *WriterThread* ! *i* $\longrightarrow$ $\Big)$;
$\Big($ *binder_writeRet* . *fbMission* . *Writer* . *WriterThread* $\longrightarrow$ $\Big)$ $\Big)$;
Skip $\Big)$
i := *i* + 1;
var *keepWriting* : $\mathbb{B}$ • *keepWriting* := $(i \geq 5)$;
if $(\neg$ *keepWriting* = **True**$)$ $\longrightarrow$
$\Big($ $\Big($ *requestTerminationCall* . *fbMission* . *Writer* $\longrightarrow$
requestTerminationRet . *fbMission* . *Writer* ? *requestTermination* $\longrightarrow$ $\Big)$
Skip $\Big)$
$[]$ $\neg$ $(\neg$ *keepWriting* = **True**$)$ $\longrightarrow$ **Skip**
fi ;
Skip $\Big)$
$[]$ $\neg$ $($*loopVar*$)$ $\longrightarrow$ **Skip**
fi
Skip
runRet . *Writer* $\longrightarrow$
Skip $\Big)$

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

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

end

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

state *State*

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

state *State*

initial *Init*

$State'$

$i' = 1$

- **Skip**

end