

Flatbuffer

Tight Rope v0.6

26th 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*

FlatBufferMissionSequencerID : *SchedulableID*

ReaderID : *SchedulableID*

WriterID : *SchedulableID*

distinct \langle *nullSequencerId*, *nullSchedulableId*, *FlatBufferMissionSequencerIDID*,
ReaderID, *WriterID* \rangle

1.3 ThreadIds

section *ThreadId* **parents** *scj_prelude, GlobalTypes*

ReaderThreadID : *ThreadID*

WriterThreadID : *ThreadID*

distinct(*SafeletThreadId*, *nullThreadId*,
ReaderThreadID, *WriterThreadID*)

1.4 ObjectIds

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

FlatBufferObjectID : ObjectID

FlatBufferMissionObjectID : ObjectID

ReaderObjectID : ObjectID

WriterObjectID : ObjectID

distinct \langle *FlatBufferObjectID, FlatBufferMissionObjectID,*
ReaderObjectID, WriterObjectID \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* ==
 { *SafeltAppSync, MissionSequencerAppSync, MissionAppSync, MTAAppSync, 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* $\hat{=}$ **begin**

read_MethodBinder $\hat{=}$

$$\left(\begin{array}{l} \textit{binder_readCall} \\ \quad ? \textit{loc} : (\textit{loc} \in \textit{readLocs}) \\ \quad ? \textit{caller} : (\textit{caller} \in \textit{readCallers}) \longrightarrow \\ \textit{readCall} . \textit{loc} . \textit{caller} \longrightarrow \\ \textit{readRet} . \textit{loc} . \textit{caller} ? \textit{ret} \longrightarrow \\ \textit{binder_readRet} . \textit{loc} . \textit{caller} ! \textit{ret} \longrightarrow \\ \textit{read_MethodBinder} \end{array} \right)$$

write_MethodBinder $\hat{=}$

$$\left(\begin{array}{l} \textit{binder_writeCall} \\ \quad ? \textit{loc} : (\textit{loc} \in \textit{writeLocs}) \\ \quad ? \textit{caller} : (\textit{caller} \in \textit{writeCallers}) \times \mathbb{Z} \longrightarrow \\ \textit{writeCall} . \textit{loc} . \textit{caller} \times \mathbb{Z} \longrightarrow \\ \textit{writeRet} . \textit{loc} . \textit{caller} \longrightarrow \\ \textit{binder_writeRet} . \textit{loc} . \textit{caller} \longrightarrow \\ \textit{write_MethodBinder} \end{array} \right)$$

BinderActions $\hat{=}$

$$\left(\begin{array}{l} \textit{read_MethodBinder} \\ ||| \\ \textit{write_MethodBinder} \end{array} \right)$$

• *BinderActions* \triangle (*done_toplevel_sequencer* \longrightarrow **Skip**)

end

process *ApplicationB* $\hat{=}$ *Application* [*MethodCallBinderSync*] *MethodCallBinder*

2.3 Locking

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

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

process *Locking* $\hat{=}$ *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* $\hat{=}$

$$\left(\begin{array}{l} \text{SafeletFW} \\ \llbracket \text{ControlTierSync} \rrbracket \\ \text{TopLevelMissionSequencerFW}(\text{FlatBufferMissionSequencer}) \end{array} \right)$$

process *Tier0* $\hat{=}$

$$\left(\begin{array}{l} \text{MissionFW}(\text{FlatBufferMissionID}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{ManagedThreadFW}(\text{ReaderID}) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{ManagedThreadFW}(\text{WriterID}) \end{array} \right) \end{array} \right)$$

process *Framework* $\hat{=}$

$$\left(\begin{array}{l} \text{ControlTier} \\ \llbracket \text{TierSync} \rrbracket \\ (\text{Tier0}) \end{array} \right)$$

process *Application* $\hat{=}$

$$\left(\begin{array}{l} \text{FlatBufferApp} \\ ||| \\ \text{FlatBufferMissionSequencerApp} \\ ||| \\ \text{FlatBufferMissionApp} \\ ||| \\ \text{ReaderApp}(\text{FlatBufferMissionID}) \\ ||| \\ \text{WriterApp}(\text{FlatBufferMissionID}) \end{array} \right)$$

process *Program* $\hat{=}$ $(\text{Framework} \llbracket \text{AppSync} \rrbracket \text{ApplicationB}) \llbracket \text{LockingSync} \rrbracket \text{Locking}$

3 Safelet

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

process *FlatBufferApp* $\hat{=}$ **begin**

InitializeApplication $\hat{=}$
 $\left(\begin{array}{l} \textit{initializeApplicationCall} \longrightarrow \\ \textit{initializeApplicationRet} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

GetSequencer $\hat{=}$
 $\left(\begin{array}{l} \textit{getSequencerCall} \longrightarrow \\ \textit{getSequencerRet} ! \textit{FlatBufferMissionSequencerID} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

immortalMemorySizeMeth $\hat{=}$ **var** *ret* : \mathbb{Z} •
 $\left(\begin{array}{l} \textit{immortalMemorySizeCall} . \textit{FlatBuffer} \longrightarrow \\ (\textit{ret} := 1000000); \\ \textit{immortalMemorySizeRet} . \textit{FlatBuffer} ! \textit{ret} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$

Methods $\hat{=}$
 $\left(\begin{array}{l} \textit{GetSequencer} \\ \square \\ \textit{InitializeApplication} \\ \square \\ \textit{immortalMemorySizeMeth} \end{array} \right); \textit{Methods}$

• (*Methods*) \triangle (*end_safelet_app* \longrightarrow **Skip**)

end

4 Top Level Mission Sequencer

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

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

State
this : **ref** *FlatBufferMissionSequencerClass*

state *State*

Init
State'
this' = **new** *FlatBufferMissionSequencerClass*()

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

Methods $\hat{=}$
 $(\textit{GetNextMission}) ; \textit{Methods}$

• $(\textit{Init} ; \textit{Methods}) \triangle (\textit{end_sequencer_app} . \textit{FlatBufferMissionSequencer} \longrightarrow \mathbf{Skip})$

end

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

state <i>State</i> <i>returnedMission</i> : \mathbb{B}
--

state *State*

initial <i>Init</i> <i>State</i> '
<i>returnedMission</i> ' = <i>false</i>

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

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

• **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* $\hat{=}$ **begin**

State
this : **ref** *FlatBufferMissionClass*

state *State*

Init
State'

this' = **new** *FlatBufferMissionClass*()

InitializePhase $\hat{=}$
 $\left(\begin{array}{l} \textit{initializeCall} . \textit{FlatBufferMission} \longrightarrow \\ \textit{register} ! \textit{Reader} ! \textit{FlatBufferMission} \longrightarrow \\ \textit{register} ! \textit{Writer} ! \textit{FlatBufferMission} \longrightarrow \\ \textit{initializeRet} . \textit{FlatBufferMission} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

CleanupPhase $\hat{=}$
 $\left(\begin{array}{l} \textit{cleanupMissionCall} . \textit{FlatBufferMission} \longrightarrow \\ \textit{cleanupMissionRet} . \textit{FlatBufferMission} ! \textbf{True} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

bufferEmptyMeth $\hat{=}$ **var** *ret* : \mathbb{B} •
 $\left(\begin{array}{l} \textit{bufferEmptyCall} . \textit{FlatBufferMission} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{bufferEmpty}(); \\ \textit{bufferEmptyRet} . \textit{FlatBufferMission} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

cleanUpMeth $\hat{=}$ **var** *ret* : \mathbb{B} •
 $\left(\begin{array}{l} \textit{cleanUpCall} . \textit{FlatBufferMission} \longrightarrow \\ \textit{ret} := \textit{this} . \textit{cleanUp}(); \\ \textit{cleanUpRet} . \textit{FlatBufferMission} ! \textit{ret} \longrightarrow \\ \textbf{Skip} \end{array} \right)$

$$\begin{aligned}
\text{writeSyncMeth} \triangleq & \left(\begin{array}{l} \text{writeCall} . \text{FlatBufferMission} ? \text{thread} \longrightarrow \\ \left(\begin{array}{l} \text{startSyncMeth} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \left(\begin{array}{l} \mu X \bullet \\ \left(\begin{array}{l} \text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{bufferEmpty}()); \\ \text{if} (\text{loopVar}) \longrightarrow \\ \left(\begin{array}{l} \text{waitCall} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \\ \text{waitRet} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \end{array} \right) ; X \\ \text{Skip} \end{array} \right) \\ \square \neg (\text{loopVar}) \longrightarrow \text{Skip} \\ \text{fi} \end{array} \right) \end{array} \right) ; \\ \text{endSyncMeth} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \text{writeRet} . \text{FlatBufferMission} . \text{thread} \longrightarrow \\ \text{Skip} \end{array} \right)
\end{aligned}$$

$$\begin{aligned}
\text{readSyncMeth} \triangleq & \text{var ret} : \mathbb{Z} \bullet \\
& \left(\begin{array}{l} \text{readCall} . \text{FlatBufferMission} ? \text{thread} \longrightarrow \\ \left(\begin{array}{l} \text{startSyncMeth} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \text{lockAcquired} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \left(\begin{array}{l} \mu X \bullet \\ \left(\begin{array}{l} \text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := \text{bufferEmpty}(); \\ \text{if} (\text{loopVar}) \longrightarrow \\ \left(\begin{array}{l} \text{waitCall} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \\ \text{waitRet} . \text{FlatBufferMissionObject} ! \text{thread} \longrightarrow \end{array} \right) ; X \\ \text{Skip} \end{array} \right) \\ \square \neg (\text{loopVar}) \longrightarrow \text{Skip} \\ \text{fi} \end{array} \right) \end{array} \right) ; \\ \text{endSyncMeth} . \text{FlatBufferMissionObject} . \text{thread} \longrightarrow \\ \text{readRet} . \text{FlatBufferMission} ! \text{thread} ! \text{ret} \longrightarrow \\ \text{Skip} \end{array} \right)
\end{aligned}$$

$$\begin{aligned}
\text{Methods} \triangleq & \left(\begin{array}{l} \text{InitializePhase} \\ \square \\ \text{CleanupPhase} \\ \square \\ \text{bufferEmptyMeth} \\ \square \\ \text{cleanUpMeth} \\ \square \\ \text{writeSyncMeth} \\ \square \\ \text{readSyncMeth} \end{array} \right) ; \text{Methods}
\end{aligned}$$

$$\bullet (\text{Init} ; \text{Methods}) \triangle (\text{end_mission_app} . \text{FlatBufferMission} \longrightarrow \text{Skip})$$

end

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

state *State*

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

state *State*

initial *Init*

State'

buffer' = 0
t' = *testClass*

public *bufferEmpty* $\hat{=}$ **var** *ret* : \mathbb{B} •

$\left(\begin{array}{l} \text{if } (buffer = 0) \longrightarrow \\ \quad ret := \mathbf{True} \\ \quad \square \neg (buffer = 0) \longrightarrow \\ \quad \quad ret := \mathbf{False} \\ \text{fi} \end{array} \right)$

public *cleanUp* $\hat{=}$ **var** *ret* : \mathbb{B} •

(*ret* := **False**)

• **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 *ReaderIDApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*
MissionMethChan, *FlatBufferMissionMethChan*, *ObjectIds*, *ThreadIds*

process *ReaderIDApp* $\hat{=}$
fbMission : *MissionID* • **begin**

Run $\hat{=}$

$$\left(\begin{array}{l} \text{runCall} . \text{ReaderID} \longrightarrow \\ \left(\begin{array}{l} \mu X \bullet \\ \left(\begin{array}{l} \text{terminationPendingCall} . \text{fbMission} . \text{Reader} \longrightarrow \\ \text{terminationPendingRet} . \text{fbMission} . \text{Reader} ? \text{terminationPending} \longrightarrow \\ \text{var } \text{loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \text{if } (\text{loopVar}) \longrightarrow \\ \left(\begin{array}{l} \text{var } \text{result} : \mathbb{Z} \bullet \text{result} := 999; \\ \left(\begin{array}{l} \text{binder_readCall} . \text{fbMission} . \text{Reader} . \text{ReaderThread} \longrightarrow \\ \text{binder_readRet} . \text{fbMission} . \text{Reader} . \text{ReaderThread} ? \text{read} \longrightarrow \end{array} \right) \text{Skip} \\ \text{Skip} \end{array} \right) \\ \parallel \neg (\text{loopVar}) \longrightarrow \text{Skip} \\ \text{fi} \\ \text{Skip} \end{array} \right) ; X \end{array} \right) \end{array} \right) ; \end{array} \right)$$

Methods $\hat{=}$
 $(\text{Run}) ; \text{Methods}$

• $(\text{Methods}) \triangle (\text{end_managedThread_app} . \text{ReaderID} \longrightarrow \text{Skip})$

end

class *ReaderClass* $\hat{=}$ **begin**

state *State*

fbMission : *FlatBufferMission*

state *State*

initial *Init*

State'

• **Skip**

end

section *WriterIDApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*

MissionMethChan, *FlatBufferMissionMethChan*, *ObjectIds*, *ThreadIds*

process *WriterIDApp* $\hat{=}$
fbMission : *MissionID* • **begin**

Run $\hat{=}$

$$\left(\begin{array}{l} \text{runCall} . \text{WriterID} \longrightarrow \\ \left(\begin{array}{l} \left(\begin{array}{l} \mu X \bullet \\ \left(\begin{array}{l} \text{terminationPendingCall} . \text{fbMission} . \text{Writer} \longrightarrow \\ \text{terminationPendingRet} . \text{fbMission} . \text{Writer} ? \text{terminationPending} \longrightarrow \\ \text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{terminationPending}); \\ \text{if } (\text{loopVar}) \longrightarrow \\ \left(\begin{array}{l} \left(\begin{array}{l} \text{binder_writeCall} . \text{fbMission} . \text{Writer} . \text{WriterThread} ! i \longrightarrow \\ \text{binder_writeRet} . \text{fbMission} . \text{Writer} . \text{WriterThread} \longrightarrow \end{array} \right) ; \\ \text{Skip} \\ i := i + 1; \\ \text{var keepWriting} : \mathbb{B} \bullet \text{keepWriting} := (i \geq 5); \\ \text{if } (\neg \text{keepWriting} = \text{True}) \longrightarrow \\ \left(\begin{array}{l} \text{requestTerminationCall} . \text{fbMission} . \text{Writer} \longrightarrow \\ \text{requestTerminationRet} . \text{fbMission} . \text{Writer} ? \text{requestTermination} \longrightarrow \end{array} \right) ; X \\ \text{Skip} \\ \parallel \neg (\neg \text{keepWriting} = \text{True}) \longrightarrow \text{Skip} \\ \text{fi}; \\ \text{Skip} \end{array} \right) \\ \parallel \neg (\text{loopVar}) \longrightarrow \text{Skip} \\ \text{fi} \\ \text{Skip} \end{array} \right) \\ \text{runRet} . \text{WriterID} \longrightarrow \\ \text{Skip} \end{array} \right) \end{array} \right)$$

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

• (*Methods*) \triangle (*end_managedThread_app* . *WriterID* \longrightarrow **Skip**)

end

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

state *State*

fbMission : *FlatBufferMission*

i : \mathbb{Z}

state *State*

initial *Init*

State'

i' = 1

• **Skip**

end