

Flatbuffer

Tight Rope v0.65

8th February 2016

1 ID Files

1.1 MissionIds

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

<i>FlatBufferMissionMID</i> : <i>MissionID</i>
--

<i>distinct</i> \langle <i>nullMissionId</i> , <i>FlatBufferMissionMID</i> \rangle
--

1.2 SchedulablesIds

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

FlatBufferMissionSequencerSID : *SchedulableID*

ReaderSID : *SchedulableID*

WriterSID : *SchedulableID*

distinct(*nullSequencerId*, *nullSchedulableId*, *FlatBufferMissionSequencerSID*,
ReaderSID, *WriterSID*)

1.3 ThreadIds

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

WriterTID : *ThreadID*

ReaderTID : *ThreadID*

distinct(*SafeletTID*, *nullTID*,
WriterTID, *ReaderTID*)

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* ==
 { *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* $\hat{=}$
 $\left(\begin{array}{c} \textit{ThreadFW}(\textit{WriterTID}, 10) \\ ||| \\ \textit{ThreadFW}(\textit{ReaderTID}, 10) \end{array} \right)$

process *Objects* $\hat{=}$
 $(\textit{ObjectFW}(\textit{FlatBufferMissionOID}))$

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

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

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*, *FlatBufferMissionSequencerSIDClass*

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

State
this : **ref** *FlatBufferMissionSequencerClass*

state *State*

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

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

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

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

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* •

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

• **Skip**

end

5 Missions

5.1 FlatBufferMission

section *FlatBufferMissionApp* **parents** *scj_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, FlatBufferMissionMIDClass, ObjectChan, ObjectIds, ThreadIds, FlatBufferMissionMIDMethChan*

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

State
this : **ref** *FlatBufferMissionClass*

state *State*

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

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

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

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

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

$$\begin{aligned}
& \text{writeSyncMeth} \triangleq \\
& \left(\begin{array}{l}
\text{writeCall} . \text{FlatBufferMissionMID} ? \text{thread} \rightarrow \\
\left(\begin{array}{l}
\text{startSyncMeth} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow \\
\text{lockAcquired} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow \\
\left(\begin{array}{l}
\mu X \bullet \\
\left(\begin{array}{l}
\text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := (\neg \text{bufferEmpty}()); \\
\text{if } (\text{loopVar} = \mathbf{True}) \rightarrow \\
\left(\begin{array}{l}
\text{waitCall} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow \\
\text{waitRet} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow
\end{array} \right) ; X \\
\mathbf{Skip}
\end{array} \right) \\
\sqcap (\text{loopVar} = \mathbf{False}) \rightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{array} \right) \\
; \\
\text{this} . \text{buffer} := \text{update}; \\
\text{notify} . \text{FlatBufferMissionOID} ! \text{thread} \rightarrow \\
\mathbf{Skip} \\
\text{endSyncMeth} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow \\
\text{writeRet} . \text{FlatBufferMissionMID} . \text{thread} \rightarrow \\
\mathbf{Skip}
\end{array} \right)
\end{array} \right)
\end{aligned}$$

$$\begin{aligned}
& \text{readSyncMeth} \triangleq \text{var ret} : \mathbb{Z} \bullet \\
& \left(\begin{array}{l}
\text{readCall} . \text{FlatBufferMissionMID} ? \text{thread} \rightarrow \\
\left(\begin{array}{l}
\text{startSyncMeth} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow \\
\text{lockAcquired} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow \\
\left(\begin{array}{l}
\mu X \bullet \\
\left(\begin{array}{l}
\text{var loopVar} : \mathbb{B} \bullet \text{loopVar} := \text{bufferEmpty}(); \\
\text{if } (\text{loopVar} = \mathbf{True}) \rightarrow \\
\left(\begin{array}{l}
\text{waitCall} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow \\
\text{waitRet} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow
\end{array} \right) ; X \\
\mathbf{Skip}
\end{array} \right) \\
\sqcap (\text{loopVar} = \mathbf{False}) \rightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{array} \right) \\
; \\
\text{var out} : \mathbb{Z} \bullet \text{out} := \text{buffer}; \\
\text{this} . \text{buffer} := 0; \\
\text{notify} . \text{FlatBufferMissionOID} ! \text{thread} \rightarrow \\
\mathbf{Skip}; \\
\text{ret} := \text{out} \\
\text{endSyncMeth} . \text{FlatBufferMissionOID} . \text{thread} \rightarrow \\
\text{readRet} . \text{FlatBufferMissionMID} . \text{thread} ! \text{ret} \rightarrow \\
\mathbf{Skip}
\end{array} \right)
\end{array} \right)
\end{aligned}$$

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

$$\bullet (\text{Init} ; \text{Methods}) \triangle (\text{end_mission_app} . \text{FlatBufferMissionMID} \rightarrow \mathbf{Skip})$$

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} •

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

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

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

• **Skip**

end

section *FlatBufferMissionMIDMethChan* **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* $\hat{=}$
fbMission : *MissionID* • **begin**

Run $\hat{=}$

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

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

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

end

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

state *State*

fbMission : *FlatBufferMission*

state *State*

initial *Init*

State'

• **Skip**

end

section *WriterApp* **parents** *ManagedThreadChan, SchedulableId, SchedulableIds*

, MissionMethChan, FlatBufferMissionMethChan, ObjectIds, ThreadIds

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

Run $\hat{=}$

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

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

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

end

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

state *State*

fbMission : *FlatBufferMission*

i : \mathbb{Z}

state *State*

initial *Init*

State'

i' = 1

• **Skip**

end