

Flatbuffer

Tight Rope v0.6

October 19, 2015

1 Network

section *Program* **parents** *scj_prelude, MissionId, MissionIds, SchedulableId, SchedulableIds, MissionChan, SchedulableMethChan, MissionFW, SafeletFW, TopLevelMissionSequencerFW, NetworkChannels, ManagedThreadFW, SchedulableMissionSequencerFW, PeriodicEventHandlerFW, OneShotEventHandlerFW, AperiodicEventHandlerFW, FlatBufferApp, FlatBufferMissionSequencerApp, ObjectFW, ThreadFW, 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* $\hat{=}$
$$\left(\begin{array}{l} \text{MissionFW}(\text{FlatBufferMission}) \\ \llbracket \text{MissionSync} \rrbracket \\ \left(\begin{array}{l} \text{ManagedThreadFW}(\text{Reader}) \\ \llbracket \text{SchedulablesSync} \rrbracket \\ \text{ManagedThreadFW}(\text{Writer}) \end{array} \right) \end{array} \right)$$

process *Framework* $\hat{=}$
$$\left(\begin{array}{l} \text{ControlTier} \\ \llbracket \text{TierSync} \rrbracket \\ (\text{Tier0}) \end{array} \right)$$

process *Application* $\hat{=}$
$$\left(\begin{array}{l} \text{FlatBufferApp} \\ \llbracket \\ \text{FlatBufferMissionSequencerApp} \\ \llbracket \\ \text{FlatBufferMissionApp} \\ \llbracket \\ \text{ReaderApp}(\text{FlatBufferMission}) \\ \llbracket \\ \text{WriterApp}(\text{FlatBufferMission}) \end{array} \right)$$

Locking $\hat{=}$
$$\left(\begin{array}{l} \left(\begin{array}{l} \text{ThreadFW}(\text{ReaderThread}, \text{MinPriority}) \\ \llbracket \text{ThreadSync} \rrbracket \\ \text{ThreadFW}(\text{WriterThread}, \text{MinPriority}) \end{array} \right) \\ \llbracket \\ \left(\begin{array}{l} \text{ObjectFW}(\text{FlatBufferObject}) \\ \llbracket \text{ObjectSync} \rrbracket \\ \text{ObjectFW}(\text{FlatBufferMissionObject}) \\ \llbracket \text{ObjectSync} \rrbracket \\ \text{ObjectFW}(\text{ReaderObject}) \\ \llbracket \text{ObjectSync} \rrbracket \\ \text{ObjectFW}(\text{WriterObject}) \end{array} \right) \end{array} \right)$$

process $Program \hat{=} Framework \llbracket AppSync \rrbracket Application \llbracket LockingSync \rrbracket Locking$

2 ID Files

2.1 MissionIds

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

FlatBufferMission : *MissionID*

distinct(*nullMissionId*, *FlatBufferMission*)

2.2 SchedulablesIds

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

FlatBufferMissionSequencer : *SchedulableID*

Reader : *SchedulableID*

Writer : *SchedulableID*

distinct(*nullSequencerId*, *nullSchedulableId*, *Reader*,
Writer)

2.3 ThreadIds

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

ReaderThread : *ThreadID*

WriterThread : *ThreadID*

distinct(*SafeletThreadId*, *nullThreadId*,
ReaderThread,
WriterThread)

2.4 ObjectIds

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

FlatBufferObject : *ObjectID*

FlatBufferMissionObject : *ObjectID*

ReaderObject : *ObjectID*

WriterObject : *ObjectID*

distinct(*FlatBufferObject*,
FlatBufferMissionObject,
ReaderObject,
WriterObject)

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

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

• $(\textit{Methods}) \triangle (\textit{end_safelet_app} \longrightarrow \mathbf{Skip})$

end

4 Top Level Mission Sequencer

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

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

<i>State</i> <i>this</i> : ref <i>FlatBufferMissionSequencerClass</i>

state *State*

<i>Init</i> <i>State</i> '
<i>this</i> ' = new <i>FlatBufferMissionSequencerClass</i> ()

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

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

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

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

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

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

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

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

end

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

state <i>State</i> <i>buffer</i> : \mathbb{Z}

state *State*

initial <i>Init</i> <i>State</i> ' <hr/> <i>buffer</i> ' = 0

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

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

5.2 Schedulables of

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

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

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

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

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

end

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

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

<i>State</i> <i>i</i> : \mathbb{Z}

state *State*

<i>Init</i> <i>State'</i>
<i>i'</i> = 1

Run $\hat{=}$

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

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

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

end