Flatbuffer

Tight Rope v0.65

7th 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*

$$
\begin{array}{|l}
WriterTID : ThreadID \\
ReaderTID : ThreadID \\
\hline
distinct\langle SafeletTID, nullTID, \\
WriterTID, ReaderTID\rangle
\end{array}
$$

1.4 ObjectIds

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

$FlatBufferMissionOID : ObjectID$

$distinct\langle FlatBufferMissionOID\rangle$

2 Network

2.1 Network Channel Sets

$$\begin{array}{l}\mathbf{section}\ \mathit{NetworkChannels}\ \mathbf{parents}\ \mathit{scj\_prelude}, \mathit{MissionId}, \mathit{MissionIds},\\ \quad \mathit{SchedulableId}, \mathit{SchedulableIds}, \mathit{MissionChan}, \mathit{SchedulableChan}, \mathit{TopLevelMissionSequencerFWChan},\\ \quad \mathit{FrameworkChan}, \mathit{SafeletChan}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{TerminateSync} ==\\ \quad \{\!|\ \mathit{schedulables\_terminated}, \mathit{schedulables\_stopped}, \mathit{get\_activeSchedulables}\ |\!\}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{ControlTierSync} ==\\ \quad \{\!|\ \mathit{start\_toplevel\_sequencer}, \mathit{done\_toplevel\_sequencer}, \mathit{done\_safeletFW}\ |\!\}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{TierSync} ==\\ \quad \{\!|\ \mathit{start\_mission} . \mathit{FlatBufferMission}, \mathit{done\_mission} . \mathit{FlatBufferMission},\\ \quad \mathit{done\_safeletFW}, \mathit{done\_toplevel\_sequencer}\ |\!\}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{MissionSync} ==\\ \quad \{\!|\ \mathit{done\_safeletFW}, \mathit{done\_toplevel\_sequencer}, \mathit{register},\\ \mathit{signalTerminationCall}, \mathit{signalTerminationRet}, \mathit{activate\_schedulables}, \mathit{done\_schedulable},\\ \mathit{cleanupSchedulableCall}, \mathit{cleanupSchedulableRet}\ |\!\}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{SchedulablesSync} ==\\ \quad \{\!|\ \mathit{activate\_schedulables}, \mathit{done\_safeletFW}, \mathit{done\_toplevel\_sequencer}\ |\!\}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{ClusterSync} ==\\ \quad \{\!|\ \mathit{done\_toplevel\_sequencer}, \mathit{done\_safeletFW}\ |\!\}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{AppSync} ==\\ \quad \bigcup\{\mathit{SafeltAppSync}, \mathit{MissionSequencerAppSync}, \mathit{MissionAppSync},\\ \quad \mathit{MTAppSync}, \mathit{OSEHSync}, \mathit{APEHSync},\\ \quad \{\!|\ \mathit{getSequencer}, \mathit{end\_mission\_app}, \mathit{end\_managedThread\_app},\\ \quad \mathit{setCeilingPriority}, \mathit{requestTerminationCall}, \mathit{requestTerminationRet}, \mathit{terminationPendingCall},\\ \quad \mathit{terminationPendingRet}, \mathit{handleAsyncEventCall}, \mathit{handleAsyncEventRet}\ |\!\}\}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{ThreadSync} ==\\ \quad \{\!|\ \mathit{raise\_thread\_priority}, \mathit{lower\_thread\_priority}, \mathit{isInterruptedCall}, \mathit{isInterruptedRet}, \mathit{get\_priorityLevel}\ |\!\}\end{array}$$

$$\begin{array}{l}\mathbf{channelset}\ \mathit{LockingSync} ==\\ \quad \{\!|\ \mathit{lockAcquired}, \mathit{startSyncMeth}, \mathit{endSyncMeth}, \mathit{waitCall}, \mathit{waitRet}, \mathit{notify}, \mathit{isInterruptedCall}, \mathit{isInterruptedRet},\\ \quad \mathit{interruptedCall}, \mathit{interruptedRet}, \mathit{done\_toplevel\_sequencer}, \mathit{get\_priorityLevel}\ |\!\}\end{array}$$

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**

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

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

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

$\bullet\ BinderActions \bigtriangleup (done\_toplevel\_sequencer \longrightarrow \mathbf{Skip})$

end

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

2.3 Locking

$$\mathbf{process}\ Threads \mathrel{\widehat=} \\ \left(\begin{array}{l} ThreadFW(\,WriterTID, 10) \\ \interleave \\ ThreadFW(ReaderTID, 10) \end{array}\right)$$

$$\mathbf{process}\ Objects \mathrel{\widehat=} \\ \big(\,ObjectFW(FlatBufferMissionOID)\big)$$

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

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* $\widehat{=}$

$$\left(\begin{array}{l} SafeletFW \\ \quad [\![ControlTierSync]\!] \\ TopLevelMissionSequencerFW(FlatBufferMissionSequencer) \end{array}\right)$$

process *Tier0* $\widehat{=}$

$$\left(\begin{array}{l} MissionFW(FlatBufferMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} ManagedThreadFW(ReaderID) \\ \quad [\![SchedulablesSync]\!] \\ ManagedThreadFW(WriterID) \end{array}\right) \end{array}\right)$$

process *Framework* $\widehat{=}$

$$\left(\begin{array}{l} ControlTier \\ \quad [\![TierSync]\!] \\ (Tier0) \end{array}\right)$$

process *Application* $\widehat{=}$

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

process *Program* $\widehat{=} \left(Framework\ [\![\ AppSync\ ]\!]\ ApplicationB\right)\ [\![\ LockingSync\ ]\!]\ Locking$

3 Safelet

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

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

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

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

$$
immortalMemorySizeMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : \mathbb{Z} \bullet \\
\left(\begin{array}{l} immortalMemorySizeCall\,. \longrightarrow \\ \left(ret := 1000000\right); \\ immortalMemorySizeRet\,.\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)
$$

$$
Methods \mathrel{\widehat{=}} \\
\left(\begin{array}{l} GetSequencer \\ \Box \\ InitializeApplication \\ \Box \\ immortalMemorySizeMeth \end{array}\right);\ Methods
$$

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

end

4 Top Level Mission Sequencer

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

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

State

$this : \mathbf{ref}\ FlatBufferMissionSequencerClass$

state *State*

Init

$State\,'$

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

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

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

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

end

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

state *State*

$returnedMission : \mathbb{B}$

state *State*

initial *Init*

$State\,'$

$returnedMission' = false$

$$\mathbf{protected}\ getNextMission\ \widehat{=}\ \mathbf{var}\ ret : MissionID \bullet \left(\begin{array}{l}\mathbf{if}\ (\neg\ returnedMission = \mathbf{True}) \longrightarrow \\ \qquad \left(\begin{array}{l} this\,.\,returnedMission := true; \\ ret := FlatBufferMission\end{array}\right) \\ \talloblong\ (\neg\ returnedMission = \mathbf{True}) \longrightarrow \\ \qquad \left(ret := nullMissionId\right) \\ \mathbf{fi}\end{array}\right)$$

$\bullet$ **Skip**

end

5 Missions

5.1 FlatBufferMission

$\mathbf{section}\ \mathit{FlatBufferMissionApp}\ \mathbf{parents}\ \mathit{scj\_prelude}, \mathit{MissionId}, \mathit{MissionIds},$
$\mathit{SchedulableId}, \mathit{SchedulableIds}, \mathit{MissionChan}, \mathit{SchedulableMethChan}, \mathit{FlatBufferMissionMIDClass}$
$,$
$\mathit{ObjectChan}, \mathit{ObjectIds}, \mathit{ThreadIds}, \mathit{FlatBufferMissionMIDMethChan}$

$\mathbf{process}\ \mathit{FlatBufferMissionApp} \mathrel{\widehat=} \mathbf{begin}$

$$\begin{array}{|l}
\hline \mathit{State} \\
\mathit{this} : \mathbf{ref}\ \mathit{FlatBufferMissionClass} \\
\hline
\end{array}$$

$\mathbf{state}\ \mathit{State}$

$$\begin{array}{|l}
\hline \mathit{Init} \\
\mathit{State}' \\
\hline
\mathit{this}' = \mathbf{new}\ \mathit{FlatBufferMissionClass}() \\
\hline
\end{array}$$

$$\mathit{InitializePhase} \mathrel{\widehat=} \left(\begin{array}{l}
\mathit{initializeCall}\,.\,\mathit{FlatBufferMissionMID} \longrightarrow \\
\mathit{register}\,!\,\mathit{ReaderSID}\,!\,\mathit{FlatBufferMissionMID} \longrightarrow \\
\mathit{register}\,!\,\mathit{WriterSID}\,!\,\mathit{FlatBufferMissionMID} \longrightarrow \\
\mathit{initializeRet}\,.\,\mathit{FlatBufferMissionMID} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)$$

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

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

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

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

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

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

$$
\bullet\ (\mathit{Init}\ ;\ \mathit{Methods}) \bigtriangleup (\mathit{end\_mission\_app}\,.\,\mathit{FlatBufferMissionMID} \longrightarrow \mathbf{Skip})
$$

end

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

$$\begin{array}{|l} \textbf{state}\ State \\ \hline buffer : \mathbb{Z} \\ t : testClass \\ \hline \end{array}$$

state *State*

$$\begin{array}{|l} \textbf{initial}\ Init \\ \hline State' \\ \hline buffer' = 0 \\ t' = testClass \\ \hline \end{array}$$

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

$$\left(\begin{array}{l} \textbf{if}\ (buffer = 0) \longrightarrow \\ \quad ret := \textbf{True} \\ [\!]\ (buffer = 0) \longrightarrow \\ \quad ret := \textbf{False} \\ \textbf{fi} \end{array}\right)$$

public *cleanUp* $\widehat{=}$ **var** *ret* : $\mathbb{B}$ ●

$\big(ret := \textbf{False}\big)$

● **Skip**

end

section $\mathit{FlatBufferMissionMIDMethChan}$ **parents** $\mathit{scj\_prelude}, \mathit{GlobalTypes}, \mathit{MissionId}, \mathit{SchedulableId}$

channel $\mathit{bufferEmptyCall} : \mathit{MissionID}$
channel $\mathit{bufferEmptyRet} : \mathit{MissionID} \times \mathbb{B}$

channel $\mathit{cleanUpCall} : \mathit{MissionID}$
channel $\mathit{cleanUpRet} : \mathit{MissionID} \times \mathbb{B}$

channel $\mathit{writeCall} : \mathit{MissionID} \times \mathit{SchedulableID} \times \mathit{ThreadID} \times \mathbb{Z}$
channel $\mathit{writeRet} : \mathit{MissionID} \times \mathit{SchedulableID} \times \mathit{ThreadID}$

channel $\mathit{readCall} : \mathit{MissionID} \times \mathit{SchedulableID} \times \mathit{ThreadID}$
channel $\mathit{readRet} : \mathit{MissionID} \times \mathit{SchedulableID} \times \mathit{ThreadID} \times \mathbb{Z}$

5.2 Schedulables of FlatBufferMission

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

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

$$
\mathit{Run} \mathrel{\widehat{=}}
\left(\begin{array}{l}
\mathit{runCall} \,.\, \mathit{ReaderSID} \longrightarrow \\
\left(\left(\begin{array}{l}
\mu X \bullet \\
\left(\begin{array}{l}
\mathit{terminationPendingCall} \,.\, \mathit{fbMission} \,.\, \longrightarrow \\
\mathit{terminationPendingRet} \,.\, \mathit{fbMission} \,.\, ?\, \mathit{terminationPending} \longrightarrow \\
\ \mathbf{var}\ \mathit{loopVar} : \mathbb{B} \bullet \mathit{loopVar} := (\neg\ \mathit{terminationPending}); \\
\mathbf{if}\ (\mathit{loopVar} = \mathbf{True}) \longrightarrow \\
\qquad \left(\begin{array}{l}
\mathbf{var}\ \mathit{result} : \mathbb{Z} \bullet \mathit{result} := 999; \\
\left(\begin{array}{l}
\mathit{binder\_readCall} \,.\, \mathit{fbMission} \,.\,.\, \mathit{ReaderTID} \longrightarrow \\
\mathit{binder\_readRet} \,.\, \mathit{fbMission} \,.\,.\, \mathit{ReaderTID}\ ?\, \mathit{read} \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \mathbf{Skip}
\end{array}\right) ;\ X \\
[\!]\ (\mathit{loopVar} = \mathbf{False}) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{array}\right)
\end{array}\right) \\
\mathbf{Skip}
\end{array}\right) ; \\
\mathit{runRet} \,.\, \mathit{ReaderSID} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
$$

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

$\bullet$ $(\mathit{Methods}) \vartriangle (\mathit{end\_managedThread\_app} \,.\, \mathit{ReaderSID} \longrightarrow \mathbf{Skip})$

end

class $\mathit{ReaderClass} \mathrel{\widehat=}$ **begin**

state *State*

$\mathit{fbMission} : \mathit{FlatBufferMission}$

state *State*

initial *Init*

State'

$\bullet$ **Skip**

end

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

```
process WriterApp ≙
    fbMission : MissionID ● begin


Run ≙
(
 runCall . WriterSID ⟶
 (
  (
   μX ●
   (
    terminationPendingCall . fbMission . ⟶
    terminationPendingRet . fbMission . ? terminationPending ⟶
     var loopVar : 𝔹 ● loopVar := (¬ terminationPending);
    if (loopVar = True) ⟶
          (
           (
            binder_writeCall . fbMission . . WriterTID ! i ⟶
            binder_writeRet . fbMission . . WriterTID ⟶
            Skip
           ) ;
           i := i + 1;
           var keepWriting : 𝔹 ● keepWriting := false;
           if (i ≥ 5) ⟶
                 (this . keepWriting := true)
           ▯ (i ≥ 5) ⟶
                 (this . keepWriting := false)
           fi ;
           if (¬ keepWriting = True) ⟶
                 (
                  requestTerminationCall . fbMission . ⟶
                  requestTerminationRet . fbMission . ? requestTermination ⟶
                  Skip
                 )
           ▯ (¬ keepWriting = True) ⟶ Skip
           fi ;
           Skip
          ) ; X
    ▯ (loopVar = False) ⟶ Skip
    fi
   )
  )
 Skip
 ) ;
 runRet . WriterSID ⟶
 Skip
)


Methods ≙
(Run) ; Methods


● (Methods) △ (end_managedThread_app . WriterSID ⟶ Skip)


end
```

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

state State

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

state State

initial Init

State'

$i' = 1$

$\bullet$ **Skip**

end