Flatbuffer

July 20, 2015

1 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* |}$\}$

section *Program* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *MissionFW*,
SafeletFW, *TopLevelMissionSequencerFW*, *NetworkChannels*, *ManagedThreadFW*,
SchedulableMissionSequencerFW, *PeriodicEventHandlerFW*, *OneShotEventHandlerFW*,
AperiodicEventHandlerFW, *FlatBufferApp*, *FlatBufferMissionSequencerApp*,
FlatBufferMissionApp, *ReaderApp*, *WriterApp*

process $ControlTier \mathrel{\widehat=}$
$$\left(\begin{array}{l} SafeletFW \\ \quad [\![ControlTierSync]\!] \\ TopLevelMissionSequencerFW\,(FlatBufferMissionSequencer) \end{array}\right)$$

process $Tier0 \mathrel{\widehat=}$
$$\left(\begin{array}{l} MissionFW\,(FlatBufferMission) \\ \quad [\![MissionSync]\!] \\ \left(\left(\begin{array}{l} ManagedThreadFW\,(Reader) \\ \quad [\![SchedulablesSync]\!] \\ ManagedThreadFW\,(Writer) \end{array}\right)\right) \\ \quad [\![SchedulablesSync]\!] \end{array}\right)$$

process $Framework \mathrel{\widehat=}$
$$\left(\begin{array}{l} ControlTier \\ \quad [\![TierSync]\!] \\ (\,Tier0\,) \end{array}\right)$$

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

process $Program \mathrel{\widehat=} Framework \;[\![\; AppSync \;]\!]\; Application$

2 ID Files

2.1 MissionIds

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

$$FlatBufferMission : MissionID$$

$$distinct\langle nullMissionId, FlatBufferMission\rangle$$

2.2 SchedulablesIds

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

$$FlatBufferMissionSequencer : SchedulableID$$
$$Reader : SchedulableID$$
$$Writer : SchedulableID$$

$$distinct\langle nullSequencerId, nullSchedulableId, Reader, Writer\rangle$$

3 Safelet

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

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

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

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

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

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

end

4 Top Level Mission Sequencer

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

process $FlatBufferMissionSequencerApp \mathrel{\widehat=} \textbf{begin}$

State

$returnedMission : boolean$

state *State*

Init

$State'$

$returnedMission' = false$

$$GetNextMission \mathrel{\widehat=} \left(\begin{array}{l} getNextMissionCall \,.\, FlatBufferMissionSequencer \longrightarrow \\ getNextMissionRet \,.\, FlatBufferMissionSequencer \,!\, FlatBufferMission \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$getNextMissionMeth \mathrel{\widehat=} \textbf{var}\ ret : MissionID \bullet \left(\begin{array}{l} getNextMissionCall \,.\, FlatBufferMissionSequencer \longrightarrow \\ \left(\begin{array}{l} \textbf{if}\ (\neg\ returnedMission = \textbf{True}) \longrightarrow \\ \quad \left(\begin{array}{l} returnedMission := \textbf{True}\,; \\ ret := FlatBufferMission \end{array}\right) \\ [\!]\ \neg\ (\neg\ returnedMission = \textbf{True}) \longrightarrow \\ \quad \big(ret := nullMissionId\big) \\ \textbf{fi} \end{array}\right); \\ getNextMissionRet \,.\, FlatBufferMissionSequencer \,!\, ret \longrightarrow \\ \textbf{Skip} \end{array}\right)$$

$$Methods \mathrel{\widehat=} \left(\begin{array}{l} GetNextMission \\ \Box \\ getNextMissionMeth \end{array}\right);\ Methods$$

$$\bullet\ (Methods) \triangle (end\_sequencer\_app \,.\, FlatBufferMissionSequencer \longrightarrow \textbf{Skip})$$

end

5 Missions

5.1 FlatBufferMission

section *FlatBufferMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*

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

State

$buffer : int$
$writer : Writer$
$reader : Reader$

state *State*

Init

$State'$

$buffer' = 0$
$writer' = init\_placeholder$
$reader' = init\_placeholder$

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

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

$bufferEmptyMeth \mathrel{\widehat=} \mathbf{var}\ ret : \mathbb{B} \bullet$
$\left(\begin{array}{l} bufferEmptyCall\,.\,FlatBufferMission\,?\,name \longrightarrow \\ \left(\begin{array}{l} \mathbf{if}\ (buffer = 0) \longrightarrow \\ \quad ret := \mathbf{True} \\ [] \neg\,(buffer = 0) \longrightarrow \\ \quad ret := \mathbf{False} \\ \mathbf{fi} \end{array}\right); \\ bufferEmptyRet\,.\,FlatBufferMission\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{array}\right)$

$writeSyncMeth \mathrel{\widehat=}$
$\left(\begin{array}{l} writeCall\,.\,FlatBufferMission\,?\,thread\,?\,update \longrightarrow \\ startSyncMeth\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ lockAcquired\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ \\ endSyncMeth\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ writeRet\,.\,FlatBufferMission\,.\,[]\,.\,thread \longrightarrow \\ \mathbf{Skip} \end{array}\right)$

$$readSyncMeth \mathrel{\widehat{=}} \left(\begin{array}{l} readCall\,.\,FlatBufferMission\,?\,thread \longrightarrow \\ startSyncMeth\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ lockAcquired\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ \\ endSyncMeth\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ readRet\,.\,FlatBufferMission\,.\,out\,.\,thread \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

$$waitOnMissionSyncMeth \mathrel{\widehat{=}} \left(\begin{array}{l} waitOnMissionCall\,.\,FlatBufferMission\,?\,thread\,?\,name \longrightarrow \\ startSyncMeth\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ lockAcquired\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ \\ endSyncMeth\,.\,FlatBufferMission\,.\,thread \longrightarrow \\ waitOnMissionRet\,.\,FlatBufferMission\,.\,[]\,.\,thread \longrightarrow \\ \mathbf{Skip} \end{array}\right)$$

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

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

end

5.2 Schedulables of FlatBufferMission

section *ReaderApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*

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

State

$$fbMission : FlatBufferMission$$

state *State*

Init

$$State'$$

$$fbMission' = fbMission$$

$$Run \mathrel{\widehat{=}} \begin{pmatrix} runCall\,.\,Reader \longrightarrow \\ runRet\,.\,Reader \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$Methods \mathrel{\widehat{=}} \begin{pmatrix} Run \\ \Box \\ runMeth \end{pmatrix} ;\ Methods$$

$$runMeth \mathrel{\widehat{=}} \mathbf{var}\ ret : null \bullet \begin{pmatrix} runCall\,.\,Reader \longrightarrow \\ \left(\left(\left(\begin{array}{l} \mu X \bullet \\ \left(\begin{array}{l} \mathbf{if}\ (\neg\ fbMission.terminationPending()) \longrightarrow \\ \quad ((?))\ ;\ X \\ [\!]\ \neg\ (\neg\ fbMission.terminationPending()) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{array}\right) \end{array}\right)\right)\right) ; \\ runRet\,.\,Reader\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$\bullet\ (Methods) \triangle (end\_managedThread\_app\,.\,Reader \longrightarrow \mathbf{Skip})$$

end

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

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

State

$fbMission : FlatBufferMission$
$i : int$

state *State*

Init

$State'$

$fbMission' = fbMission$
$i' = 1$

$$Run \mathrel{\widehat=} \begin{pmatrix} runCall\,.\,Writer \longrightarrow \\ runRet\,.\,Writer \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$Methods \mathrel{\widehat=} \begin{pmatrix} Run \\ \Box \\ runMeth \end{pmatrix} ;\ Methods$$

$$runMeth \mathrel{\widehat=} \mathbf{var}\ ret : null \bullet \begin{pmatrix} runCall\,.\,Writer \longrightarrow \\ \left(\left(\left(\begin{pmatrix} \mu X \bullet \\ \begin{pmatrix} \mathbf{if}\ (\neg\ fbMission.terminationPending()) \longrightarrow \\ \quad (([?])) ;\ X \\ [] \neg\ (\neg\ fbMission.terminationPending()) \longrightarrow \mathbf{Skip} \\ \mathbf{fi} \end{pmatrix} \end{pmatrix}\right)\right)\right) ; \\ runRet\,.\,Writer\,!\,ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$\bullet\ (Methods) \triangle (end\_managedThread\_app\,.\,Writer \longrightarrow \mathbf{Skip})$$

end