Safety-Critical Java Level 2 Application Model : FlatBuffer

Matt Luckcuck

Department of Computer Science,
University of York, UK

ml881@york.ac.uk

11th January 2016

Contents

1 Introduction **2**

2 FlatBuffer Application Model **2**
2.1 Network . . . 2
2.2 ID Files . . . 4
2.2.1 SchedulablesIds . . . 4
2.2.2 ThreadIds . . . 4
2.2.3 ObjectIds . . . 4
2.3 Safelet . . . 6
2.4 Top-Level Mission Sequencer . . . 7
2.5 FlatBufferMission . . . 9
2.6 ReaderApp . . . 14
2.7 WriterApp . . . 15

3 Appendix **17**
3.1 Appendix A: Framework Specification . . . 17
3.1.1 GlobalTypes . . . 17
3.1.2 Priority . . . 18
3.1.3 Priority Queue . . . 19
3.1.4 Ids . . . 20
3.1.5 SafeletFW . . . 35
3.1.6 TopLevelMissionSequencerFW . . . 37
3.1.7 MissionFW . . . 39
3.1.8 SchedulableMissionSequencerFW . . . 45
3.1.9 Event Handlers . . . 48
3.1.10 ManagedThreadFW . . . 60
3.2 Appendix B: FlatBuffer Script . . . 62
3.2.1 FlatBuffer . . . 62
3.2.2 FlatBufferMissionSequencer . . . 63
3.2.3 FlatBufferMission . . . 64
3.2.4 Writer . . . 65
3.2.5 Reader . . . 66

1 Introduction

Safety-Critical Java (SCJ) [1] is a Java-based language for applications that must be certified. To aid certification efforts, SCJ is organised into three compliance levels. Level 0 applications are simple single-processor programs executed by a cyclic executive. By contrast, Level 2 applications are highly concurrent, potentially multi-processor, and make use of suspension and a variety of release patterns.

We model SCJ Level 2 applications using the state-rich process algebra *Circus* [3]. We approach this by splitting our models into a reusable *Framework* model, which captures the API behaviour of SCJ, and a specific *Application* model, which captures the application's behaviour.

The *FlatBuffer* application is a contrived example to illustrate some of the features of Level 2. The full script of the FlatBuffer program can be found in Appendix 3.2. FlatBuffer is an SCJ solution to the Readers-Writers Problem, using a one-place buffer. The program is named for its relatively 'flat' program hierarchy – SCJ Level 2 allows much more complex hierarchies than the FlatBuffer. The FlatBuffer is structurally simple but uses two of Level 2's unique features: managed threads and suspension. More interesting examples can be found in [2].

Here we present Application model that captures application-specific behaviour of the FlatBuffer application. It is intended that this Application model be combined with Framework model to produce a complete model of the FlatBuffer application, a summary of which can be found in Appendix 3.1.

2 FlatBuffer Application Model

2.1 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*

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

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

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

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

$$Locking \mathrel{\widehat{=}} \left(\begin{array}{l} \left(\begin{array}{l} ThreadFW(ReaderThread, MinPriority) \\ \quad [\![ThreadSync]\!] \\ ThreadFW(WriterThread, MinPriority) \end{array}\right) \\ ||| \\ \left(\begin{array}{l} ObjectFW(FlatBufferObject) \\ \quad [\![ObjectSync]\!] \\ ObjectFW(FlatBufferMissionObject) \\ \quad [\![ObjectSync]\!] \\ ObjectFW(ReaderObject) \\ \quad [\![ObjectSync]\!] \\ ObjectFW(WriterObject) \end{array}\right) \end{array}\right)$$

$$\textbf{process}\ Program \mathrel{\widehat{=}} Framework\ [\![\ AppSync\ ]\!]\ Application\ [\![\ LockingSync\ ]\!]\ Locking$$

2.2 ID Files

MissionIds

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

$$FlatBufferMission : MissionID$$

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

2.2.1 SchedulablesIds

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

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

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

2.2.2 ThreadIds

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

$$ReaderThread : ThreadID$$
$$WriterThread : ThreadID$$

$$distinct\langle SafeletThreadId, nullThreadId, ReaderThread, WriterThread\rangle$$

2.2.3 ObjectIds

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

$FlatBufferObject : ObjectID$
$FlatBufferMissionObject : ObjectID$
$ReaderObject : ObjectID$
$WriterObject : ObjectID$

$distinct\langle FlatBufferObject,$
$FlatBufferMissionObject,$
$ReaderObject,$
$WriterObject\rangle$

2.3 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

2.4 Top-Level Mission Sequencer

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

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

State

$this : \textbf{ref}\ FlatBufferMissionSequencerClass$

state *State*

Init

$State'$

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

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

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

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

end

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

state *State*

$$returnedMission : \mathbb{B}$$

state *State*

initial *Init*

$$State'$$

$$returnedMission' = false$$

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

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

• **Skip**

end

2.5 FlatBufferMission

section *FlatBufferMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *FlatBufferMissionClass*, *ObjectChan*, *ObjectIds*, *ThreadIds*, *FlatBufferMissionMethChan*

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

State

$this : \mathbf{ref}\ FlatBufferMissionClass$

state *State*

Init

$State'$

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

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

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

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

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

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

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

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

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

end

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

state *State*

$buffer : \mathbb{Z}$

state *State*

initial *Init*

$State'$

$buffer' = 0$

public *bufferEmpty* $\widehat{=}$ **var** $ret : \mathbb{B}$ •
$\left(\begin{array}{l} \textbf{if } (buffer = 0) \longrightarrow \\ \quad ret := \textbf{True} \\ [] \neg (buffer = 0) \longrightarrow \\ \quad ret := \textbf{False} \\ \textbf{fi} \end{array}\right)$

public *cleanUp* $\widehat{=}$ **var** $ret : \mathbb{B}$ •
$\big(ret := \textbf{False}\big)$

• **Skip**

end

section *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* : $MissionID \times ThreadID \times \mathbb{Z}$

2.6 ReaderApp

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

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

$$
\begin{array}{l}
Run \mathrel{\widehat=} \\
\left(\begin{array}{l}
runCall\,.\,Reader \longrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l}
\mu X \bullet \\
\left(\begin{array}{l}
terminationPendingCall\,.\,fbMission \longrightarrow \\
terminationPendingRet\,.\,fbMission\,?\,terminationPending \longrightarrow \\
\quad \textbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ terminationPending); \\
\textbf{if}\ (loopVar) \longrightarrow \\
\quad \left(\begin{array}{l}
\textbf{var}\ result : \mathbb{Z} \bullet result := 999; \\
\left(\begin{array}{l}
\left(\begin{array}{l}
readCall\,.\,fbMission\,.\,ReaderThread \longrightarrow \\
readRet\,.\,fbMission\,.\,ReaderThread\,?\,read \longrightarrow
\end{array}\right) \textbf{Skip} \\
\textbf{Skip}
\end{array}\right)
\end{array}\right) ;\ X \\
[]\ \neg\ (loopVar) \longrightarrow \textbf{Skip} \\
\textbf{fi}
\end{array}\right)
\end{array}\right) \\
\textbf{Skip}
\end{array}\right) \\
runRet\,.\,Reader \longrightarrow \\
\textbf{Skip}
\end{array}\right) ;
\end{array}
$$

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

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

end

2.7 WriterApp

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

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

State

$i : \mathbb{Z}$

state *State*

Init

$State'$

$i' = 1$

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

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

• $(Init\ ;\ \ Methods) \triangle (end\_managedThread\_app\ .\ Writer \longrightarrow \mathbf{Skip})$

end

3 Appendix

3.1 Appendix A: Framework Specification

3.1.1 GlobalTypes

section *GlobalTypes* **parents** *scj_prelude*, *SchedulableId*

$[ThreadID]$

$$SafeletThreadId : ThreadID$$
$$nullThreadId : ThreadID$$

$[ObjectID]$

$[totalThreads]$

$ThreadMap == ThreadID \pfun \mathbb{N}_1$

$ExceptionType ::= interruptedException \mid illegalMonitorStateException \mid illegalArgumentException \mid illegalThreadStateException \mid illegalStateException \mid ceilingViolationException$

$maxNanos == 999999$

$AperiodicType ::= aperiodic \mid aperiodicLong$

3.1.2 Priority

section *Priority* **parents** *scj_prelude*

$$
\begin{array}{|l}
MinPriority : \mathbb{N}_1 \\
MaxPriority : \mathbb{N}_1 \\
\hline
MaxPriority - MinPriority \geq 2
\end{array}
$$

$$PriorityLevel == MinPriority \mathinner{..} MaxPriority$$

3.1.3 Priority Queue

section *PriorityQueue* **parents** *scj_prelude*, *GlobalTypes*, *Priority*

$$PriorityQueue == PriorityLevel \to (\text{iseq}\ ThreadID)$$

$$\forall\, pq : PriorityQueue \bullet nullThreadId \notin \text{ran}(\bigcup(\text{ran}\, pq))$$

$$IsEmpty : PriorityQueue \to \mathbb{B}$$

$$\forall\, pq : PriorityQueue \mid (\bigcup(pq\ (\!|\ PriorityLevel\ |\!))) = \varnothing \bullet IsEmpty(pq) = \textbf{True}$$

$$AddToPriorityQueue : PriorityQueue \times ThreadID \times PriorityLevel \to PriorityQueue$$

$$\forall\, pq : PriorityQueue;\ t : ThreadID;\ p : PriorityLevel \mid t \neq nullThreadId \land t \notin \text{ran}(\bigcup(\text{ran}(pq))) \bullet AddToPriorityQueue(pq, t, p) = (pq \oplus \{p \mapsto pq(p) \frown \langle t \rangle\})$$

$$RemoveFromPriorityQueue : PriorityQueue \nrightarrow PriorityQueue \times ThreadID$$

$$(\forall\, pq : PriorityQueue \bullet (\exists\, t : ThreadID;\ p : PriorityLevel \mid p = max\,\{pl : PriorityLevel \mid pq(pl) \neq \langle\rangle\} \land t = head\, pq(p) \bullet RemoveFromPriorityQueue(pq) = (pq \oplus \{p \mapsto tail\, pq(p)\}, t)))$$

$$RemoveThreadFromPriorityQueue : PriorityQueue \times ThreadID \times PriorityLevel \to PriorityQueue$$

$$\forall\, pq : PriorityQueue;\ t : ThreadID;\ p : PriorityLevel \mid pq(p) \upharpoonright \{t\} \neq \langle\rangle \bullet RemoveThreadFromPriorityQueue(pq, t, p) = pq \oplus \{p \mapsto squash\,(pq(p) \rhd \{t\})\}$$

$$ElementsOf : PriorityQueue \to \mathbb{P}\ ThreadID$$

$$\forall\, pq : PriorityQueue \mid pq \neq \varnothing \bullet (\exists\, elems : \mathbb{P}\ ThreadID \mid elems = \bigcup(\text{ran}\ (\!|\ \text{ran}\, pq\ |\!)) \bullet ElementsOf(pq) = elems)$$

3.1.4 Ids

MissionId

section *MissionId*

[*MissionID*]

nullMissionId : *MissionID*

SchedulableId

section *SchedulableId*

[*SchedulableID*]

TopLevelSequencerId : *SchedulableID*
nullSequencerId : *SchedulableID*
nullSchedulableId : *SchedulableID*

SchedulableIds

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

$$
\begin{array}{|l}
InputHandlerId : SchedulableID \\
OutputHandlerId : SchedulableID \\
MainMissionSequencerId : SchedulableID \\
EchoMissionSequencerId : SchedulableID \\
InputMissionSequencerId : SchedulableID \\
OutputMissionSequencerId : SchedulableID \\
\hline
distinct\langle TopLevelSequencerId, InputHandlerId, OutputHandlerId, \\
\quad MainMissionSequencerId, EchoMissionSequencerId, \\
\quad InputMissionSequencerId, OutputMissionSequencerId, nullSequencerId, nullSchedulableId\rangle
\end{array}
$$

ObjectFW

section *Object* **parents** *scj_prelude*, *GlobalTypes*, *ObjectChan*, *MissionChan*, *SchedulableChan*, *SchedulableId*, *MissionId*, *MissionIds*, *TopLevelMissionSequencerChan*, *HandlerChan*, *SafeletMethChan*, *FrameworkChan*, *PriorityQueue*, *Priority*, *ThreadChan*

process $ObjectFW \mathrel{\widehat=} object : ObjectID \bullet$ **begin**

state *State*

$waitQueue : PriorityQueue$
$lockedBy : ThreadID$
$locks : \mathbb{N}$
$previousLocks : ThreadMap$
$queueForLock : PriorityQueue$
$ceilingPriority : PriorityLevel$
$waitForObjectThreads : \mathbb{P}\ ThreadID$

$locks > 0 \Leftrightarrow lockedBy \neq nullSchedulableID$
$lockedBy \notin \mathrm{dom}\ previousLocks$
$lockedBy \notin ElementsOf(waitQueue)$
$lockedBy \notin ElementsOf(queueForLock)$
$waitForObjectThreads \subseteq ElementsOf(waitQueue)$

Init

$State'$

$IsEmpty(queueForLock') = True$
$IsEmpty(waitQueue') = True$
$locks' = 0$
$previousLocks' = \varnothing$
$ceilingPriority' = MaxPriority$
$waitForObjectThreads' = \varnothing$

FullyUnlock

$\Delta State$
$lockedBy? : ThreadID$
$locks? : \mathbb{N}_1$

$previousLocks' = previousLocks \oplus \{lockedBy? \mapsto locks?\}$
$lockedBy' = nullSchedulableID$
$locks' = 0$
$waitQueue' = waitQueue$
$queueForLock' = queueForLock$
$ceilingPriority' = ceilingPriority$
$waitForObjectThreads' = waitForObjectThreads$

AddToQueueForLock

$\Delta State$
$someThread? : ThreadID$
$priorityLevel? : PriorityLevel$

$someThread? \neq nullSchedulableID$
$someThread? \notin ElementsOf(queueForLock)$
$queueForLock' = AddToPriorityQueue(queueForLock, someThread?, priorityLevel?)$
$lockedBy' = lockedBy$
$locks' = locks$
$previousLocks' = previousLocks$
$waitQueue' = waitQueue$
$ceilingPriority' = ceilingPriority$
$waitForObjectThreads' = waitForObjectThreads$

AssignEligible

$\Delta State$

$(queueForLock', lockedBy') = RemoveFromPriorityQueue(queueForLock)$
$lockedBy' \in \operatorname{dom} previousLocks \Rightarrow locks' = previousLocks(lockedBy')$
$lockedBy' \notin \operatorname{dom} previousLocks \Rightarrow locks' = 1$
$previousLocks' = \{lockedBy\} \lhd previousLocks$
$waitQueue' = waitQueue$
$ceilingPriority' = ceilingPriority$
$waitForObjectThreads' = waitForObjectThreads$

AddToWaitQueue

$\Delta State$
$someThread? : ThreadID$
$priorityLevel? : PriorityLevel$
$waitType? : WaitType$

$someThread? \neq nullSchedulableID$
$someThread? \notin ElementsOf(waitQueue)$
$waitQueue' = AddToPriorityQueue(waitQueue, someThread?, priorityLevel?)$
$lockedBy' = lockedBy$
$locks' = locks$
$previousLocks' = previousLocks$
$queueForLock' = queueForLock$
$ceilingPriority' = ceilingPriority$
$waitType? = waitForObject \Rightarrow waitForObjectThreads' = waitForObjectThreads \cup \{someThread?\}$
$waitType? = wait \Rightarrow waitForObjectThreads' = waitForObjectThreads$

RemoveThreadFromWaitQueue

$\Delta State$
$waitingThread? : ThreadID$
$priorityLevel? : PriorityLevel$

$waitingThread? \in \mathrm{ran}(waitQueue(priorityLevel?))$
$waitQueue' = RemoveThreadFromPriorityQueue(waitQueue, waitingThread?, priorityLevel?)$
$lockedBy' = lockedBy$
$locks' = locks$
$previousLocks' = previousLocks$
$ceilingPriority' = ceilingPriority$
$waitForObjectThreads' = waitForObjectThreads \setminus \{waitingThread?\}$

RemoveMostEligigbleFromWaitQueue

$\Delta State$
$notified! : ThreadID$
$waitType! : WaitType$

$(waitQueue', notified!) = RemoveFromPriorityQueue(waitQueue)$
$lockedBy' = lockedBy$
$locks' = locks$
$previousLocks' = previousLocks$
$queueForLock' = queueForLock$
$ceilingPriority' = ceilingPriority$
$notified! \in waitForObjectThreads \Rightarrow waitType! = waitForObject$
$notified! \notin waitForObjectThreads \Rightarrow waitType! = wait$
$waitForObjectThreads' = waitForObjectThreads \setminus \{notified!\}$

$$
\begin{array}{l}
Execute \mathrel{\widehat=} \\
\quad \textbf{var}\ interruptedThreads : \mathbb{P}\ ThreadID \bullet \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l}
Monitor \\
[\![\varnothing \mid \\
\quad MonitorSync \mid \\
\{waitQueue, waitForObjectThreads\}]\!]
\end{array}\right) \\
Synchronisation
\end{array}\right) \\
[\![\{waitQueue, waitForObjectThreads\} \mid \\
\quad MLCSync \mid \\
\{queueForLock, previousLocks, locks, lockedBy\}]\!] \\
MonitorLockController\ (interruptedThreads)
\end{array}\right) \\
\quad [\![\{waitQueue, waitForObjectThreads, queueForLock, previousLocks, locks, lockedBy\} \mid \\
\quad\quad CPCSync \mid \\
\quad \{ceilingPriority\}]\!] \\
\quad CeilingPriorityController
\end{array}
$$

$$
\begin{array}{l}
Monitor \mathrel{\widehat=} \\
\quad MonitorUnlocked
\end{array}
$$

$$
\begin{array}{l}
MonitorUnlocked \mathrel{\widehat=} \\
\quad \left(\begin{array}{l} startSynchMeth\,.\,object\,?\,someThread \longrightarrow \\ lock\_request\,.\,object\,!\,someThread \longrightarrow \\ MonitorUnlocked \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} lockAcquired\,.\,object\,?\,lockingThread \longrightarrow \\ get\_ceilingPriority\,.\,object\,?\,ceilingPriority \longrightarrow \\ \left(\begin{array}{l} \left(\begin{array}{l} get\_priorityLevel\,.\,lockingThread\,.\,object\,?\,priority : (priority \leq ceilingPriority) \longrightarrow \\ raise\_thread\_priority\,.\,lockingThread\,!\,ceilingPriority \longrightarrow \\ MonitorLocked(lockingThread) \end{array}\right) \\ \Box \\ \left(\begin{array}{l} get\_priorityLevel\,.\,lockingThread\,.\,object\,?\,priority : (priority > ceilingPriority) \longrightarrow \\ throw.ceilingViolationException \longrightarrow \\ \textbf{Chaos} \end{array}\right) \end{array}\right) \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
MonitorLocked \mathrel{\widehat=} \textbf{val}\, lockedBy : ThreadID \bullet \\
\quad \left(\begin{array}{l} startSynchMeth\,.\,object\,.\,lockedBy \longrightarrow \\ increment\_locks\,.\,object \longrightarrow \\ MonitorLocked(lockedBy) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} startSynchMeth\,.\,object\,?\,someThread : (someThread \neq lockedBy) \longrightarrow \\ lock\_request\,.\,object\,!\,someThread \longrightarrow \\ MonitorLocked(lockedBy) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} endSyncMeth\,.\,object\,.\,lockedBy \longrightarrow \\ \left(\begin{array}{l} \left(\begin{array}{l} decrement\_locks\,.\,object\,.\,0 \longrightarrow \\ lower\_thread\_priority\,.\,lockedBy \longrightarrow \end{array}\right) \\ MonitorUnlocked \\ \Box \\ \left(\begin{array}{l} decrement\_locks\,.\,object\,?\,l : (l \neq 0) \longrightarrow \\ MonitorLocked(lockedBy) \end{array}\right) \end{array}\right) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} unlock\_Monitor\,.\,object\,?\,unlockingThread \longrightarrow \\ fully\_unlock\,.\,object \longrightarrow \\ lower\_thread\_priority\,.\,unlockingThread \longrightarrow \\ MonitorUnlocked \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Synchronisation \mathrel{\widehat=} \\
\quad \left(\begin{array}{l} \left(\begin{array}{l} \left(\begin{array}{l} WaitActions \\ [\![\varnothing \mid WaitSync \mid \varnothing]\!] \\ NotifyActions \end{array}\right) \\ \quad [\![\varnothing \mid \\ \quad\quad WQSync \mid \\ \quad \{waitQueue, waitForObjectThreads\}]\!] \\ WaitQueueController \end{array}\right) \\ [\![\{waitQueue, waitForObjectThreads\} \mid InterruptSync \mid \varnothing]\!] \\ Interrupt \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
WaitActions \mathrel{\widehat{=}} \\
\quad \big(Wait \mid\mid\mid TimedWait \big) \mid\mid\mid WaitForObject
\end{array}
$$

$$
\begin{array}{l}
NotifyActions \mathrel{\widehat{=}} \\
\quad Notify \mid\mid\mid NotifyAll
\end{array}
$$

$$
\begin{array}{l}
Wait \mathrel{\widehat{=}} \\
\quad waitCall \,.\, object \,?\, someThread \longrightarrow \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
isInterruptedCall \,.\, someThread \longrightarrow \\
isInterruptedRet \,.\, someThread \,.\, False \longrightarrow \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
get\_lockedBy \,.\, object \,.\, someThread \longrightarrow \\
get\_priorityLevel \,.\, someThread \,.\, object \,?\, priorityLevel \longrightarrow \\
add\_to\_wait \,.\, object \,!\, someThread \,!\, priorityLevel \,!\, wait \longrightarrow \\
unlock\_Monitor \,.\, object \,!\, someThread \longrightarrow \\
Wait
\end{array} \right) \\
\quad \Box \\
\left(\begin{array}{l}
get\_lockedBy \,.\, object \,?\, lockedBy : (lockedBy \neq someThread) \longrightarrow \\
throw \,.\, illegalMonitorStateException \longrightarrow \\
\textbf{Chaos}
\end{array} \right)
\end{array} \right)
\end{array} \right) \\
\quad \Box \\
\left(\begin{array}{l}
isInterruptedCall \,.\, someThread \longrightarrow \\
isInterruptedRet \,.\, someThread \,.\, True \longrightarrow \\
throw.interruptedException \longrightarrow \\
\textbf{Chaos}
\end{array} \right)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
TimedWait \mathrel{\widehat{=}} \\
\quad TimedWaitHandler \\
\quad [\![\varnothing \mid \{\!| start\_timer |\!\} \mid \varnothing]\!] \\
\Big(\Big|\Big|\Big| \; t : ThreadID \bullet TimedWaitTimer(t) \Big)
\end{array}
$$

$$
\begin{array}{l}
TimedWaitHandler \mathrel{\widehat{=}} \\
\quad timedWaitCall \,.\, object \,?\, someThread \,?\, waitTime \longrightarrow \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
get\_lockedBy \,.\, object \,.\, someThread \longrightarrow \\
\textbf{if} (timeMillis(waitTime) < 0) \vee \\
\quad (timeNanos(waitTime) < 0 \wedge timeNanos(waitTime) > maxNanos)) \longrightarrow \\
\left(\begin{array}{l}
throw \,.\, illegalArgumentException \longrightarrow \\
\textbf{Chaos}
\end{array} \right) \\
[\!] (timeMillis(waitTime) > 0) \wedge \\
\quad (timeNanos(waitTime) > 0) \wedge (timeNanos(waitTime) \leq maxNanos)) \longrightarrow \\
\left(\begin{array}{l}
get\_priorityLevel \,.\, someThread \,.\, object \,?\, priorityLevel \longrightarrow \\
add\_to\_wait \,.\, object \,!\, someThread \,!\, priorityLevel \,!\, wait \longrightarrow \\
start\_timer \,.\, object \,!\, someThread \,!\, priorityLevel \,!\, waitTime \longrightarrow \\
unlock\_Monitor \,.\, object \,!\, someThread \longrightarrow \\
TimedWaitHandler
\end{array} \right) \\
\textbf{fi}
\end{array} \right) \\
\quad \Box \\
\left(\begin{array}{l}
get\_lockedBy \,.\, object \,?\, lockedBy : (lockedBy \neq someThread) \longrightarrow \\
throw \,.\, illegalMonitorStateException \longrightarrow \\
\textbf{Chaos}
\end{array} \right)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
TimedWaitTimer \mathrel{\widehat=} \mathbf{val}\ waitingThread : ThreadID \bullet \\
\quad \left(\begin{array}{l}
start\_timer\,.\,object\,.\,waitingThread\,?\,priorityLevel\,?\,waitTime \longrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l}
\mathbf{wait}\ valueOf(waitTime); \\
remove\_from\_wait\,.\,object\,!\,waitingThread\,!\,priorityLevel \longrightarrow \\
waitRet\,.\,object\,!\,waitingThread \longrightarrow \\
\mathbf{Skip}
\end{array} \right) \\
\Box \\
\left(\begin{array}{l}
cancel\_wait\_timer\,.\,object\,.\,waitingThread \longrightarrow \\
\mathbf{Skip}
\end{array} \right)
\end{array} \right) \\
; \\
relock\_this\,.\,object\,!\,waitingThread \longrightarrow \\
TimedWaitTimer(waitingThread)
\end{array} \right)
\end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
cancel\_wait\_timer\,.\,object\,.\,waitingThread \longrightarrow \\
TimedWaitTimer(waitingThread)
\end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
\big(waitRet\,.\,object\,.\,waitingThread \longrightarrow TimedWaitTimer(waitingThread) \big) \\
\Box \\
\big(waitForObjectRet\,.\,object\,.\,waitingThread\,?\,w \longrightarrow TimedWaitTimer(waitingThread) \big)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
WaitForObject \mathrel{\widehat=} \\
\quad WaitForObjectHandler \\
\quad\quad [\![\varnothing \mid \{\!|\ start\_waitForObject\_timer\ |\!\} \mid \varnothing]\!] \\
\quad \Big(\Big|\!\Big|\!\Big|\ t : ThreadID \bullet WaitForObjectTimer(t) \Big)
\end{array}
$$

$$
\begin{array}{l}
WaitForObjectHandler \mathrel{\widehat=} \\
\quad waitForObjectCall\,.\,object\,?\,someThread\,?\,waitTime \longrightarrow \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
get\_lockedBy\,.\,object\,.\,someThread \longrightarrow \\
\mathbf{if}((timeMillis(waitTime) < 0) \lor (timeNanos(waitTime) < 0)) \longrightarrow \\
\left(\begin{array}{l}
throw\,.\,illegalArgumentException \longrightarrow \\
\mathbf{Chaos}
\end{array} \right) \\
[\!]((timeMillis(waitTime) \geq 0) \land (timeNanos(waitTime) \geq 0)) \longrightarrow \\
\left(\begin{array}{l}
get\_priorityLevel\,.\,someThread\,.\,object\,?\,priorityLevel \longrightarrow \\
add\_to\_wait\,.\,object\,?\,someThread\,?\,priorityLevel\,!\,waitForObject \longrightarrow \\
start\_waitForObject\_timer\,.\,object\,!\,someThread\,!\,priorityLevel\,!\,waitTime \longrightarrow \\
unlock\_Monitor\,.\,object\,!\,someThread \longrightarrow \\
WaitForObjectHandler
\end{array} \right) \\
\mathbf{fi}
\end{array} \right) \\
\quad\quad \Box \\
\quad \left(\begin{array}{l}
get\_lockedBy\,.\,object\,?\,lockedBy : (lockedBy \neq someThread) \longrightarrow \\
throw\,.\,illegalMonitorStateException \longrightarrow \\
\mathbf{Chaos}
\end{array} \right)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
WaitForObjectTimer \mathrel{\widehat{=}} \mathbf{val}\ waitingThread : ThreadID \bullet \\
\left(\begin{array}{l}
start\_waitForObject\_timer\,.\,object\,?\,waitingThread\,?\,priorityLevel\,?\,waitTime \longrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l}
\mathbf{wait}\ valueOf(waitTime); \\
remove\_from\_wait\,.\,object\,!\,waitingThread\,!\,priorityLevel \longrightarrow \\
waitForObjectRet\,.\,object\,!\,waitingThread\,!\,\mathbf{False} \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
\Box \\
\left(\begin{array}{l}
cancel\_wait\_timer\,.\,object\,.\,waitingThread \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right) \\
relock\_this\,.\,object\,!\,waitingThread \longrightarrow \\
WaitForObjectTimer(waitingThread)
\end{array}\right) ; \\
\Box \\
\left(\begin{array}{l}
cancel\_wait\_timer\,.\,object\,.\,waitingThread \longrightarrow \\
WaitForObjectTimer(waitingThread)
\end{array}\right) \\
\Box \\
\left(\begin{array}{l}
\big(waitRet\,.\,object\,?\,n \longrightarrow WaitForObjectTimer(waitingThread)\big) \\
\Box \\
\big(waitForObjectRet\,.\,object\,?\,n\,?\,w \longrightarrow WaitForObjectTimer(waitingThread)\big)
\end{array}\right)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Notify \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
notify\,.\,object\,?\,someThread \longrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
get\_lockedBy\,.\,object\,.\,someThread \longrightarrow \\
\left(\begin{array}{l}
\mathbf{if}\ IsEmpty(waitQueue) = False \longrightarrow \\
\left(\begin{array}{l}
ResumeThread; \\
Notify
\end{array}\right) \\
[\!]\ IsEmpty(waitQueue) = True \longrightarrow \\
\quad Notify \\
\mathbf{fi}
\end{array}\right)
\end{array}\right) \\
\Box \\
\left(\begin{array}{l}
get\_lockedBy\,.\,object\,?\,lockedBy : (lockedBy \neq someThread) \longrightarrow \\
throw\,.\,illegalMonitorStateException \longrightarrow \\
\mathbf{Chaos}
\end{array}\right)
\end{array}\right)
\end{array}\right) \\
\Box \\
\left(\begin{array}{l}
\big(waitRet\,.\,object\,?\,n \longrightarrow Notify\big) \\
\Box \\
\big(waitForObjectRet\,.\,object\,?\,n\,?\,w \longrightarrow Notify\big)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
ResumeThread \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
removed\_thread\,.\,object\,?\,notified\,.\,wait \longrightarrow \\
cancel\_wait\_timer\,.\,object\,!\,notified \longrightarrow \\
relock\_this\,.\,object\,!\,notified \longrightarrow \\
waitRet\,.\,object\,!\,notified \longrightarrow \\
\mathbf{Skip}
\end{array}\right) \\
\Box \\
\left(\begin{array}{l}
removed\_thread\,.\,object\,?\,notified\,.\,waitForObject \longrightarrow \\
cancel\_wait\_timer\,.\,object\,!\,notified \longrightarrow \\
relock\_this\,.\,object\,!\,notified \longrightarrow \\
waitForObjectRet\,.\,object\,!\,notified\,!\,\mathbf{True} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
NotifyAll \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
notifyAll\,.\,object\,?\,someThread \longrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
get\_lockedBy\,.\,object\,.\,someThread \longrightarrow \\
NotifyAllHandler; \\
NotifyAll
\end{array}\right) \\
\Box \\
\left(\begin{array}{l}
get\_lockedBy\,.\,object\,?\,lockedBy : (lockedBy \neq someThread) \longrightarrow \\
throw\,.\,illegalMonitorStateException \longrightarrow \\
\textbf{Chaos}
\end{array}\right)
\end{array}\right)
\end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
\big(waitRet\,.\,object\,?\,n \longrightarrow NotifyAll\big) \\
\Box \\
\big(waitForObjectRet\,.\,object\,?\,n\,?\,w \longrightarrow NotifyAll\big)
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
NotifyAllHandler \mathrel{\widehat{=}} \textbf{var}\ notified : ThreadID \bullet \\
\quad \textbf{if}\ IsEmpty(waitQueue) = False \longrightarrow \\
\qquad \left(\begin{array}{l}
ResumeThread; \\
NotifyAllHandler
\end{array}\right) \\
\quad [\!]\ IsEmpty(waitQueue) = True \longrightarrow \\
\qquad \textbf{Skip} \\
\quad \textbf{fi}
\end{array}
$$

$$
\begin{array}{l}
WaitQueueController \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
add\_to\_wait\,.\,object\,?\,someThread\,?\,priorityLevel\,?\,waitType \longrightarrow \\
AddToWaitQueue; \\
WaitQueueController
\end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
remove\_from\_wait\,.\,object\,?\,waitingThread\,?\,priorityLevel \longrightarrow \\
RemoveThreadFromWaitQueue; \\
WaitQueueController
\end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
IsEmpty(waitQueue) = False \& \\
\textbf{var}\ notified : ThreadID \bullet \\
\textbf{var}\ waitType : WaitType \bullet \\
RemoveMostEligigbleFromWaitQueue; \\
removed\_thread\,.\,object\,!\,notified\,!\,waitType \longrightarrow \\
WaitQueueController
\end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
get\_waitQueue\,.\,object\,!\,waitQueue \longrightarrow \\
WaitQueueController
\end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
get\_waitForObjectThreads\,.\,object\,!\,waitForObjectThreads \longrightarrow \\
WaitQueueController
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Interrupt \mathrel{\widehat{=}} \\
\quad interrupt\,?\,waitingThread \longrightarrow \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
get\_waitQueue\,.\,object\,?\,retreivedWait : (waitingThread \in ElementsOf(retreivedWait)) \longrightarrow \\
cancel\_wait\_timer\,.\,object\,!\,waitingThread \longrightarrow \\
get\_priorityLevel\,.\,waitingThread\,.\,object\,?\,priorityLevel \longrightarrow \\
remove\_from\_wait\,.\,object\,!\,waitingThread\,!\,priorityLevel \longrightarrow \\
relock\_this\,.\,object\,!\,waitingThread \longrightarrow \\
\left(\begin{array}{l}
\left(\begin{array}{l}
get\_waitForObjectThreads\,.\,object\,?\,wfot : (waitingThread \notin wfot) \longrightarrow \\
waitRet\,.\,object\,!\,waitingThread \longrightarrow \\
\textbf{Skip}
\end{array} \right) \\
\Box \\
\left(\begin{array}{l}
get\_waitForObjectThreads\,.\,object\,?\,wfot : (waitingThread \in wfot) \longrightarrow \\
waitForObjectRet\,.\,object\,!\,waitingThread\,!\,\textbf{True} \longrightarrow \\
\textbf{Skip}
\end{array} \right)
\end{array} \right) ; \\
Interrupt
\end{array} \right) \\
\Box \\
\left(\begin{array}{l}
get\_waitQueue\,.\,object\,?\,retreivedWait : (waitingThread \notin ElementsOf(retreivedWait)) \longrightarrow \\
Interrupt
\end{array} \right)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
MonitorLockController \mathrel{\widehat{=}} \mathbf{val}\ interruptedThreads : \mathbb{P}\ ThreadID \bullet \\
\quad \left(\begin{array}{l} lock\_request\,.\,object\,?\,someThread \longrightarrow \\ get\_priorityLevel\,.\,someThread\,.\,object\,?\,priorityLevel \longrightarrow \\ AddToQueueForLock; \\ MonitorLockController(interruptedThreads) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} relock\_this\,.\,object\,?\,someThread \longrightarrow \\ get\_priorityLevel\,.\,someThread\,.\,object\,?\,priorityLevel \longrightarrow \\ \left(\begin{array}{l} AddToQueueForLock; \\ \left(\begin{array}{l} \left(\begin{array}{l} isInterruptedCall\,.\,someThread \longrightarrow \\ isInterruptedRet\,.\,someThread\,.\,False \longrightarrow \\ MonitorLockController(interruptedThreads) \end{array}\right) \\ \Box \\ \left(\begin{array}{l} isInterruptedCall\,.\,someThread \longrightarrow \\ isInterruptedRet\,.\,someThread\,.\,True \longrightarrow \\ interruptedThreads := interruptedThreads \cup \{someThread\}; \\ MonitorLockController(interruptedThreads) \end{array}\right) \end{array}\right) \end{array}\right) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} IsEmpty(queueForLock) = False \wedge lockedBy = nullSchedulableID \,\&\, \\ \left(\begin{array}{l} AssignEligible; \\ lockAcquired\,.\,object\,.\,lockedBy \longrightarrow \\ \mathbf{if}\ lockedBy \in interruptedThreads \longrightarrow \\ \left(\begin{array}{l} throw.interruptedException \longrightarrow \end{array}\right) \\ \mathbf{Chaos} \\ [\!]\, lockedBy \notin interruptedThreads \longrightarrow \left(MonitorLockController(interruptedThreads)\right) \\ \mathbf{fi} \end{array}\right) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} get\_lockedBy\,.\,object\,!\,lockedBy \longrightarrow \\ MonitorLockController(interruptedThreads) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} increment\_locks\,.\,object \longrightarrow \\ locks := locks + 1; \\ MonitorLockController(interruptedThreads) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} decrement\_locks\,.\,object\,!\,(locks - 1) \longrightarrow \\ \left(\begin{array}{l} locks := locks - 1; \\ \left(\begin{array}{l} \mathbf{if}\ locks = 0 \longrightarrow \\ \left(\begin{array}{l} lockedBy := nullSchedulableID; \\ MonitorLockController(interruptedThreads) \end{array}\right) \\ [\!]\, locks \neq 0 \longrightarrow \\ \quad MonitorLockController(interruptedThreads) \\ \mathbf{fi} \end{array}\right) \end{array}\right) \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} fully\_unlock\,.\,object \longrightarrow \\ FullyUnlock; \\ MonitorLockController(interruptedThreads) \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
CeilingPriorityController \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
setCeilingPriority\,?\,mission\,!\,object\,?\,priority \longrightarrow \\
ceilingPriority := priority; \\
\mu X \bullet \left(get\_ceilingPriority\,.\,object\,!\,ceilingPriority \longrightarrow X \right)
\end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
get\_ceilingPriority\,.\,object\,!\,ceilingPriority \longrightarrow \\
CeilingPriorityController
\end{array} \right)
\end{array}
$$

$$
\bullet \left(Init \;;\; Execute \right) \triangle \left(done\_toplevel\_sequencer \longrightarrow \mathbf{Skip} \right)
$$

end

ThreadFW

section *ThreadFW* **parents** *scj_prelude*, *GlobalTypes*, *ThreadChan*, *ObjectFWChan*, *FrameworkChan*, *Priority*

process $ThreadFW \mathrel{\widehat=} thread : ThreadID;\ basePriority : PriorityLevel \bullet$ **begin**

state *State*

$priorityStack : \mathrm{seq}_1\ PriorityLevel$
$activePriority : PriorityLevel$
$interrupted : \mathbb{B}$

$activePriority = last\ priorityStack$

Init

$\Delta State$

$priorityStack' = \langle basePriority \rangle$
$interrupted' = \mathbf{False}$

$$Execute \mathrel{\widehat=} \left(\left(\left(\begin{pmatrix} Priority \\ [\![\{basePriority\} \mid \{interrupted\}]\!] \\ Interrupts \end{pmatrix}\right)\right)\right) \\ \triangle \begin{pmatrix} done\_toplevel\_sequencer \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$Priority \mathrel{\widehat=}$
  if $priorityStack = \langle basePriority \rangle \longrightarrow$
    $IncreasePriority$
  $[]\ priorityStack \neq \langle basePriority \rangle \longrightarrow$
    $\begin{pmatrix} IncreasePriority \\ \Box DecreasePriority \end{pmatrix}$
  fi

$IncreasePriority \mathrel{\widehat=}$
  $raise\_thread\_priority\ .\ thread\ ?\ ceilingPriority \longrightarrow$
  $activePriority := ceilingPriority;$
  $IncreasePriority$

$DecreasePriority \mathrel{\widehat=}$
  $lower\_thread\_priority\ .\ thread \longrightarrow$
  $activePriority := basePriority;$
  $DecreasePriority$

$$
\begin{array}{l}
\mathit{Interrupts} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l} \mathit{Interrupt} \\ \quad [\![\varnothing \mid \varnothing]\!] \\ \mathit{IsInterrupted} \end{array} \right) \\
\quad [\![\varnothing \mid \varnothing]\!] \\
\mathit{Interrupted}
\end{array} \right) \\
\quad [\![\varnothing \mid \{\!| \mathit{set\_interrupted}, \mathit{get\_interrupted} |\!\} \mid \varnothing]\!] \\
\mathit{InterruptedController}
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Interrupt} \mathrel{\widehat{=}} \\
\quad \mathit{interrupt} \,.\, \mathit{thread} \longrightarrow \\
\quad \mathit{set\_interrupted} \,.\, \mathit{thread} \,!\, \mathit{True} \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
\mathit{IsInterrupted} \mathrel{\widehat{=}} \\
\quad \mathit{isInterruptedCall} \,.\, \mathit{thread} \longrightarrow \\
\quad \mathit{get\_interrupted} \,.\, \mathit{thread} \,?\, \mathit{interrupted} \longrightarrow \\
\quad \mathit{isInterruptedRet} \,.\, \mathit{thread} \,!\, \mathit{interrupted} \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
\mathit{Interrupted} \mathrel{\widehat{=}} \\
\quad \mathit{interruptedCall} \,.\, \mathit{thread} \longrightarrow \\
\quad \mathit{get\_interrupted} \,.\, \mathit{thread} \,?\, \mathit{interrupted} \longrightarrow \\
\quad \mathit{interruptedRet} \,.\, \mathit{thread} \,!\, \mathit{interrupted} \longrightarrow \\
\quad \mathit{set\_interrupted} \,.\, \mathit{thread} \,!\, \mathit{False} \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
\mathit{InterruptedController} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} \mathit{get\_interrupted} \,.\, \mathit{thread} \,!\, \mathit{interrupted} \longrightarrow \\ \mathit{InterruptedController} \end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l} \mathit{set\_interrupted} \,.\, \mathit{thread} \,?\, \mathit{newInterrupted} \longrightarrow \\ \mathit{interrupted} := \mathit{newInterrupted}; \\ \mathit{InterruptedController} \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{GetPriorityLevel} \mathrel{\widehat{=}} \\
\quad \mathit{get\_priorityLevel} \,.\, \mathit{thread} \,?\, \mathit{object} \,!\, \mathit{activePriority} \longrightarrow \\
\quad \mathit{GetPriorityLevel}
\end{array}
$$

$$
\bullet \; \big(\mathit{Init} \; ; \;\; \mathit{Execute} \big) \triangle \big(\mathit{done\_toplevel\_sequencer} \longrightarrow \mathbf{Skip} \big)
$$

end

3.1.5 SafeletFW

section *SafeletFW* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *TopLevelMissionSequencerChan*, *FrameworkChan*, *SchedulableChan*

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

State

$$globallyRegistered : \mathbb{F}\ SchedulableID$$
$$topLevelSequencer : SchedulableID$$

Init

$$State'$$

$$globallyRegistered' = \varnothing$$
$$topLevelSequencer' = nullSequencerId$$

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

$$Execute \mathrel{\widehat{=}} GetSequencerMeth;\ \mathbf{if}\ topLevelSequencer \neq nullSequencerId \longrightarrow \begin{pmatrix} start\_toplevel\_sequencer\ .\ topLevelSequencer \longrightarrow \\ Methods \end{pmatrix} [] topLevelSequencer = nullSequencerId \longrightarrow \mathbf{Skip}\ \mathbf{fi}$$

$$GetSequencerMeth \mathrel{\widehat{=}} getSequencerCall \longrightarrow\ getSequencerRet\ ?\ sequencer \longrightarrow\ topLevelSequencer := sequencer$$

$$Methods \mathrel{\widehat{=}} \left(\begin{pmatrix} Register; \\ Methods \end{pmatrix} \Box \begin{pmatrix} Deregister; \\ Methods \end{pmatrix} \Box \begin{pmatrix} done\_toplevel\_sequencer \longrightarrow \\ \mathbf{Skip} \end{pmatrix} \right)$$

$$
\begin{array}{l}
Register \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
register\,?\,schedulable : (schedulable \notin globallyRegistered)\,?\,mission \longrightarrow \\
\left(\begin{array}{l}
globallyRegistered := globallyRegistered \cup \{schedulable\}; \\
checkSchedulable\,.\,mission\,!\,\mathbf{True} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}\right) \\
\Box \\
\left(\begin{array}{l}
register\,?\,schedulable : (schedulable \in globallyRegistered)\,?\,mission \longrightarrow \\
checkSchedulable\,.\,mission\,!\,\mathbf{False} \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Deregister \mathrel{\widehat{=}} \\
\quad deregister\,?\,schedulables \longrightarrow \\
\quad globallyRegistered := (globallyRegistered \setminus schedulables); \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\bullet \left(Init\ ;\ InitializeApplication\ ;\ Execute\right)
$$

end

3.1.6 TopLevelMissionSequencerFW

section *TopLevelMissionSequencerFW* **parents** *TopLevelMissionSequencerChan*, *MissionId*, *MissionMethChan*, *SchedulableId*, *MissionFWChan*, *FrameworkChan*

process $TopLevelMissionSequencerFW \mathrel{\widehat=} sequencer : SchedulableID \bullet$ **begin**

State

$currentMission : MissionID$
$continue : \mathbb{B}$

Init

$State'$

$continue' = \mathbf{True}$
$currentMission' = nullMissionId$

$Start \mathrel{\widehat=}$
$start\_toplevel\_sequencer\,.\,sequencer \longrightarrow$
Skip

$Execute \mathrel{\widehat=}$
$\left(\left(\left(\begin{array}{l} RunMission; \\ end\_methods\,.\,sequencer \longrightarrow \\ \mathbf{Skip} \end{array}\right) \llbracket \{currentMission\} \mid \{\!|\, end\_methods \,|\!\} \mid \varnothing \rrbracket Methods\right) \llbracket \varnothing \mid CCSync \mid \{continue\} \rrbracket ContinueController\right)$

$RunMission \mathrel{\widehat=}$
$GetNextMission;$
$StartMission;$
$Continue$

$GetNextMission \mathrel{\widehat=}$
$getNextMissionCall\,.\,sequencer \longrightarrow$
$getNextMissionRet\,.\,sequencer\,?\,next \longrightarrow$
$currentMission := next$

$StartMission \mathrel{\widehat=}$
if $currentMission \neq nullMissionId \longrightarrow$
$\left(\begin{array}{l} start\_mission\,.\,currentMission\,.\,sequencer \longrightarrow \\ done\_mission\,.\,currentMission\,?\,returnedcontinue \longrightarrow \\ set\_continue\,.\,sequencer\,!\,returnedcontinue \longrightarrow \\ \mathbf{Skip} \end{array}\right)$
$[]\, currentMission = nullMissionId \longrightarrow$
$\left(\begin{array}{l} set\_continue\,.\,sequencer\,!\,\mathbf{False} \longrightarrow \\ \mathbf{Skip} \end{array}\right)$
fi

$$
\begin{array}{l}
\mathit{Continue} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} \mathit{get\_continue} \,.\, \mathit{sequencer} \,?\, \mathit{continue} : (\mathit{continue} = \mathbf{True}) \longrightarrow \\ \mathit{RunMission} \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} \mathit{get\_continue} \,.\, \mathit{sequencer} \,?\, \mathit{continue} : (\mathit{continue} = \mathbf{False}) \longrightarrow \\ \mathbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Methods} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} \mathit{SequenceTerminationPending}; \\ \mathit{Methods} \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} \mathit{end\_methods} \,.\, \mathit{sequencer} \longrightarrow \\ \mathbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{SequenceTerminationPending} \mathrel{\widehat{=}} \\
\quad \mathit{sequenceTerminationPendingCall} \,.\, \mathit{sequencer} \longrightarrow \\
\quad \mathit{get\_continue} \,.\, \mathit{sequencer} \,?\, \mathit{continue} \longrightarrow \\
\quad \mathit{sequenceTerminationPendingRet} \,.\, \mathit{sequencer} \,!\, \mathit{continue} \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
\mathit{ContinueController} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} \mathit{get\_continue} \,.\, \mathit{sequencer} \,!\, \mathit{continue} \longrightarrow \\ \mathit{ContinueController} \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} \mathit{set\_continue} \,.\, \mathit{sequencer} \,?\, \mathit{newContinue} \longrightarrow \\ \mathit{continue} := \mathit{newContinue}; \\ \mathit{ContinueController} \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} \mathit{end\_methods} \,.\, \mathit{sequencer} \longrightarrow \\ \mathbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Finish} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} \mathit{done\_toplevel\_sequencer} \longrightarrow \\ \mathit{end\_sequencer\_app} \,.\, \mathit{sequencer} \longrightarrow \\ \mathbf{Skip} \end{array}\right)
\end{array}
$$

- $\mathit{Init}\ ;\ \mathit{Start}\ ;\ \mathit{Execute}\ ;\ \mathit{Finish}$

end

3.1.7 MissionFW

section *MissionFW* **parents** *SafeletMethChan*, *MissionId*, *SchedulableId*, *MissionChan*, *SchedulableChan*, *FrameworkChan*, *ServicesChan*, *scj_prelude*

process $MissionFW \mathrel{\widehat=} mission : MissionID \bullet$ **begin**

State

$registeredSchedulables : \mathbb{F}\ SchedulableID$
$activeSchedulables : \mathbb{F}\ SchedulableID$
$missionTerminating : \mathbb{B}$
$applicationTerminating : \mathbb{B}$
$controllingSequencer : SchedulableID$

Init

$State'$

$registeredSchedulables' = \varnothing$
$activeSchedulables' = \varnothing$
$missionTerminating = \mathbf{False}$
$applicationTerminating = \mathbf{False}$
$controllingSequencer = nullSequencerId$

AddSchedulable

$\Delta State$
$s? : SchedulableID$

$s? \notin registeredSchedulables$
$registeredSchedulables' = registeredSchedulables \cup \{s?\}$
$activeSchedulables' = activeSchedulables$
$missionTerminating' = missionTerminating$
$applicationTerminating' = applicationTerminating$
$controllingSequencer' = controllingSequencer$

$Start \mathrel{\widehat=}$
$\left(\begin{array}{l} start\_mission\,.\,mission\,?\,mySequencer \longrightarrow \\ controllingSequencer := mySequencer \end{array}\right)$
$\Box$
$\left(\begin{array}{l} done\_toplevel\_sequencer \longrightarrow \\ applicationTerminating := \mathbf{True} \end{array}\right)$

$InitializePhase \mathrel{\widehat=}$
$initializeCall\,.\,mission \longrightarrow$
$Initialize$

$$
\begin{array}{l}
\mathit{Initialize} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l} \mathit{Register}; \\ \mathit{Initialize} \end{array} \right) \\
\Box \\
\left(\begin{array}{l} \mathit{SetCeilingPriority}; \\ \mathit{Initialize} \end{array} \right) \\
\Box \\
\left(\begin{array}{l} \mathit{initializeRet} \,.\, \mathit{mission} \longrightarrow \\ \mathbf{Skip} \end{array} \right)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Register} \mathrel{\widehat{=}} \\
\quad \mathit{register} \,?\, s \,!\, \mathit{mission} \longrightarrow \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l} \mathit{checkSchedulable} \,.\, \mathit{mission} \,?\, \mathit{check} : (\mathit{check} = \mathbf{True}) \longrightarrow \\ \mathit{AddSchedulable} \end{array} \right) \\
\Box \\
\left(\begin{array}{l} \mathit{checkSchedulable} \,.\, \mathit{mission} \,?\, \mathit{check} : (\mathit{check} = \mathbf{False}) \longrightarrow \\ \mathit{throw.illegalStateException} \longrightarrow \\ \mathbf{Chaos} \end{array} \right)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{RegisterException} \mathrel{\widehat{=}} \\
\quad \mathit{register} \,?\, s \,!\, \mathit{mission} \longrightarrow \\
\quad \mathit{throw.illegalStateException} \longrightarrow \\
\quad \mathbf{Chaos}
\end{array}
$$

$$
\begin{array}{l}
\mathit{SetCeilingPriority} \mathrel{\widehat{=}} \\
\quad \mathit{setCeilingPriority} \,.\, \mathit{mission} \,?\, o \,?\, p \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
\mathit{SetCeilingPriorityException} \mathrel{\widehat{=}} \\
\quad \mathit{setCeilingPriority} \,.\, \mathit{mission} \,?\, o \,?\, p \longrightarrow \\
\quad \mathit{throw.illegalStateException} \longrightarrow \\
\quad \mathbf{Chaos}
\end{array}
$$

$$
\begin{array}{l}
\mathit{MissionPhase} \mathrel{\widehat{=}} \\
\quad \mathit{Execute} \\
\qquad [\![\{ \mathit{registeredSchedulables}, \mathit{activeSchedulables}, \mathit{missionTerminating}, \\
\qquad\quad \mathit{applicationTerminating}, \mathit{controllingSequencer} \} \mid \{\!| \mathit{done\_schedulables} |\!\} \mid \varnothing]\!] \\
\quad \mathit{Exceptions}
\end{array}
$$

$$
\begin{array}{l}
Execute \mathrel{\widehat=} \\
\quad \left(\begin{array}{l}
\textbf{if } registeredSchedulables = \varnothing \longrightarrow \\
\quad \left(\begin{array}{l} done\_schedulables\,.\,mission \longrightarrow \\ \textbf{Skip} \end{array} \right) \\
[\!] registeredSchedulables \neq \varnothing \longrightarrow \\
\quad \left(\begin{array}{l}
activate\_schedulables\,.\,mission \longrightarrow \\
activeSchedulables := registeredSchedulables; \\
\left(\begin{array}{l}
TerminateAndDone \\
\quad [\![\{ activeSchedulables \} \mid \\
\quad\quad \{\!| stop\_schedulables, done\_schedulables |\!\} \\
\quad \mid \{ missionTerminating \}]\!] \\
Methods
\end{array} \right)
\end{array} \right) \\
\textbf{fi}
\end{array} \right) \\
\setminus \{\!| done\_schedulables |\!\}
\end{array}
$$

$$
\begin{array}{l}
TerminateAndDone \mathrel{\widehat=} \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
SignalTermination \\
\quad [\![\varnothing \mid TerminateSync \mid \{ activeSchedulables \}]\!]
\end{array} \right) ; \\
DoneSchedulables
\end{array} \right) \\
\quad done\_schedulables\,.\,mission \longrightarrow \\
\quad \textbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
SignalTermination \mathrel{\widehat=} \\
\quad \left(\begin{array}{l}
stop\_schedulables\,.\,mission \longrightarrow \\
get\_activeSchedulables\,.\,mission\,?\,schedulablesToStop \longrightarrow \\
StopSchedulables(schedulablesToStop); \\
schedulables\_stopped\,.\,mission \longrightarrow \\
\textbf{Skip}
\end{array} \right) \\
\quad \triangle (schedulables\_stopped\,.\,mission \longrightarrow \textbf{Skip})
\end{array}
$$

$$
\begin{array}{l}
StopSchedulables \mathrel{\widehat=} \textbf{val}\, schedulablesToStop : \mathbb{F}\, SchedulableID \bullet \\
\quad \left(\begin{array}{l}
|||\, s : schedulablesToStop \bullet \\
\quad\quad signalTerminationCall\,.\,s \longrightarrow \\
\quad\quad signalTerminationRet\,.\,s \longrightarrow \\
\quad\quad \textbf{Skip}
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
DoneSchedulables \mathrel{\widehat{=}} \\
\quad \left(\left(\begin{array}{l} \Box\, schedulable : activeSchedulables \bullet \\ done\_schedulable\,.\,schedulable \longrightarrow \\ activeSchedulables := activeSchedulables \setminus \{schedulable\}; \\ \mathbf{Skip} \end{array} \right) ; \right. \\
\quad \left. \begin{array}{l} \mathbf{if}\ activeSchedulables = \varnothing \longrightarrow \\ \quad \left(\begin{array}{l} schedulables\_stopped\,.\,mission \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\ [\!]\, activeSchedulables \neq \varnothing \longrightarrow \\ \quad DoneSchedulables \\ \mathbf{fi} \end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l} get\_activeSchedulables\,.\,mission\,!\,activeSchedulables \longrightarrow \\ DoneSchedulables \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
\left(\begin{array}{l} \left(\begin{array}{l} RequestTerminationMeth \\ [\![\varnothing \mid \{\!|\, end\_mission\_terminations \,|\!\} \mid \varnothing]\!] \\ TerminationPendingMeth \end{array} \right) \\ [\![\varnothing \mid MTCSync \mid \{missionTerminating\}]\!] \\ MissionTerminatingController \end{array} \right) \\
\quad [\![\{missionTerminating\} \mid \{\!|\, end\_mission\_terminations \,|\!\} \mid \varnothing]\!] \\
\left(\begin{array}{l} done\_schedulables\,.\,mission \longrightarrow \\ end\_mission\_terminations\,.\,mission \longrightarrow \\ \mathbf{Skip} \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
RequestTerminationMeth \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} end\_mission\_terminations\,.\,mission \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\
\quad \Box \\
\quad \left(\left(\begin{array}{l} \left(\begin{array}{l} \Box\, schedulable : registeredSchedulables \bullet requestTermination\,.\,mission\,.\,schedulable \longrightarrow \\ \mathbf{Skip} \end{array} \right) ; \\ \left(\left(\begin{array}{l} get\_missionTerminating\,.\,mission?missionTerminating : (missionTerminating = \mathbf{False}) \longrightarrow \\ set\_missionTerminating\,.\,mission\,!\,\mathbf{True} \longrightarrow \\ stop\_schedulables\,.\,mission \longrightarrow \\ RequestTerminationMeth \end{array} \right) \right. \\ \quad \Box \\ \quad \left. \left(\begin{array}{l} get\_missionTerminating\,.\,mission?missionTerminating : (missionTerminating = \mathbf{True}) \longrightarrow \\ RequestTerminationMeth \end{array} \right) \right) \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
TerminationPendingMeth \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} end\_mission\_terminations\,.\,mission \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l} terminationPendingCall\,.\,mission \longrightarrow \\ get\_missionTerminating\,.\,mission\,?\,missionTerminating \longrightarrow \\ terminationPendingRet\,.\,mission\,!\,missionTerminating \longrightarrow \\ TerminationPendingMeth \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
MissionTerminatingController \mathrel{\widehat=} \\
\quad \left(\begin{array}{l} get\_missionTerminating\,.\,mission\,!\,missionTerminating \longrightarrow \\ MissionTerminatingController \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} set\_missionTerminating\,.\,mission\,?\,newMissionTerminating \longrightarrow \\ missionTerminating := newMissionTerminating; \\ MissionTerminatingController \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} end\_mission\_terminations\,.\,mission \longrightarrow \\ \mathbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
CleanupPhase \mathrel{\widehat=} \\
\quad Cleanup \\
\qquad [\![\{ registeredSchedulables, activeSchedulables, missionTerminating, \\
\qquad\quad applicationTerminating, controllingSequencer \} \mid \{\!| \, done\_schedulables \, |\!\} \mid \varnothing]\!] \\
\quad Exceptions
\end{array}
$$

$$
\begin{array}{l}
Cleanup \mathrel{\widehat=} \\
\quad \left(\begin{array}{l} deregister!registeredSchedulables \longrightarrow \\ CleanupSchedulables; \\ cleanupMissionCall\,.\,mission \longrightarrow \\ cleanupMissionRet\,.\,mission\,?\,continueSequencer \longrightarrow \\ Finish(continueSequencer) \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
CleanupSchedulables \mathrel{\widehat=} \\
\quad \left(\begin{array}{l} |||\ s : registeredSchedulables \bullet \\ \quad cleanupSchedulableCall\,.\,s \longrightarrow \\ \quad cleanupSchedulableRet\,.\,s \longrightarrow \\ \quad \mathbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Finish \mathrel{\widehat=} \mathbf{val}\ continueSequencer : \mathbb{B} \bullet \\
\quad end\_mission\_app\,.\,mission \longrightarrow \\
\quad done\_mission\,.\,mission\,!\,continueSequencer \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
Exceptions \mathrel{\widehat=} \\
\quad \left(\begin{array}{l} RegisterException \\ ||| \\ SetCeilingPriorityException \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} done\_schedulables\,.\,mission \longrightarrow \\ \mathbf{Skip} \end{array}\right)
\end{array}
$$

$$
\bullet \left(\begin{array}{l} \mu X \bullet Init\ ;\ Start; \\ \left(\begin{array}{l} \mathbf{if}\ applicationTerminating = \mathbf{False} \longrightarrow \\ \quad (InitializePhase\ ;\ MissionPhase\ ;\ CleanupPhase\ ;\ X) \\ [\!]\ applicationTerminating = \mathbf{True} \longrightarrow \\ \quad \left(\begin{array}{l} end\_mission\_app\,.\,mission \longrightarrow \\ \mathbf{Skip} \end{array}\right) \\ \mathbf{fi} \end{array}\right) \end{array}\right)
$$

end

3.1.8 SchedulableMissionSequencerFW

section *SchedulableMissionSequencerFW* **parents** *SchedulableMissionSequencerChan*, *SchedulableChan*, *MissionIds*, *MissionChan*, *SchedulableId*, *scj_prelude*, *SafeletMethChan*, *FrameworkChan*

process $SchedulableMissionSequencerFW \mathrel{\widehat=} sequencer : SchedulableID \bullet$ **begin**

State

$currentMission : MissionID$
$continueAbove : \mathbb{B}$
$continueBelow : \mathbb{B}$
$controllingMission : MissionID$
$applicationTerminating : \mathbb{B}$

Init

$State'$

$continueAbove' = \mathbf{True}$
$continueBelow' = \mathbf{True}$
$applicationTerminating' = \mathbf{False}$
$currentMission' = nullMissionId$
$controllingMission' = nullMissionId$

GetContinue

$\Xi State$
$continue! : \mathbb{B}$

$continueAbove = \mathbf{True} \land continueBelow = \mathbf{True} \Rightarrow continue! = \mathbf{True}$

$Start \mathrel{\widehat=}$
$\quad \begin{pmatrix} Register; \\ Activate \end{pmatrix}$
$\quad \Box$
$\quad \begin{pmatrix} done\_toplevel\_sequencer \longrightarrow \\ applicationTerminating := \mathbf{True} \end{pmatrix}$
$\quad \Box$
$\quad \begin{pmatrix} activate\_schedulables\,?\,someMissionID \longrightarrow \\ Start \end{pmatrix}$

$Register \mathrel{\widehat=}$
$\quad register\,.\,sequencer\,?\,mID \longrightarrow$
$\quad controllingMission := mID$

$Activate \mathrel{\widehat=}$
$\quad activate\_schedulables.controllingMission \longrightarrow$
$\quad \mathbf{Skip}$

$$
\begin{array}{l}
\mathit{Execute} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l} \mathit{RunMission}; \\ \mathit{end\_methods} \,.\, \mathit{sequencer} \longrightarrow \end{array} \right) \\
\quad \mathbf{Skip}
\end{array} \right) \\
[\![\{ \mathit{currentMission} \} \mid \{\!| \mathit{end\_methods} |\!\} \mid \varnothing]\!] \\
\mathit{Methods}
\end{array} \right) \\
\quad [\![\varnothing \mid \mathit{CCSync} \mid \{ \mathit{continueAbove}, \mathit{continueBelow} \}]\!] \\
\quad \mathit{ContinueController}
\end{array} \right) ; \\
\quad \mathit{done\_schedulable} \,.\, \mathit{sequencer} \longrightarrow \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
\mathit{RunMission} \mathrel{\widehat{=}} \\
\quad \mathit{GetNextMission}; \\
\quad \mathit{StartMission}; \\
\quad \mathit{Continue}
\end{array}
$$

$$
\begin{array}{l}
\mathit{GetNextMission} \mathrel{\widehat{=}} \\
\quad \mathit{getNextMissionCall} \,.\, \mathit{sequencer} \longrightarrow \\
\quad \mathit{getNextMissionRet} \,.\, \mathit{sequencer} \,?\, \mathit{next} \longrightarrow \\
\quad \mathit{currentMission} := \mathit{next}
\end{array}
$$

$$
\begin{array}{l}
\mathit{StartMission} \mathrel{\widehat{=}} \\
\quad \mathbf{if} \;\; \mathit{currentMission} \neq \mathit{nullMissionId} \longrightarrow \\
\qquad \left(\begin{array}{l}
\mathit{start\_mission} \,.\, \mathit{currentMission} \,.\, \mathit{sequencer} \longrightarrow \\
\mathit{initializeRet} \,.\, \mathit{currentMission} \longrightarrow \\
\left(\begin{array}{l}
\mathit{SignalTermination} \\
\quad [\![\varnothing \mid \{\!| \mathit{end\_terminations} |\!\} \mid \varnothing]\!] \\
\left(\begin{array}{l}
\mathit{done\_mission} \,.\, \mathit{currentMission} \,?\, \mathit{continueReturn} \longrightarrow \\
\mathit{set\_continueBelow} \,.\, \mathit{sequencer} \,!\, \mathit{continueReturn} \longrightarrow \\
\mathit{end\_terminations} \,.\, \mathit{sequencer} \longrightarrow \\
\mathbf{Skip}
\end{array} \right)
\end{array} \right)
\end{array} \right) \\
\quad [\!] \; \mathit{currentMission} = \mathit{nullMissionId} \longrightarrow \\
\qquad \left(\begin{array}{l} \mathit{set\_continueBelow} \,.\, \mathit{sequencer} \,!\, \mathbf{False} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\
\quad \mathbf{fi}
\end{array}
$$

$$
\begin{array}{l}
\mathit{Continue} \mathrel{\widehat{=}} \\
\left(\begin{array}{l} \mathit{get\_continue} \,.\, \mathit{sequencer} \,?\, \mathit{continue} : (\mathit{continue} = \mathbf{True}) \longrightarrow \\ \mathit{RunMission} \end{array} \right) \\
\quad \Box \\
\left(\begin{array}{l} \mathit{get\_continue} \,.\, \mathit{sequencer} \,?\, \mathit{continue} : (\mathit{continue} = \mathbf{False}) \longrightarrow \\ \mathbf{Skip} \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{SignalTermination} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l} \mathit{end\_terminations} \,.\, \mathit{sequencer} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\
\Box \\
\left(\begin{array}{l}
\mathit{signalTerminationCall} \,.\, \mathit{sequencer} \longrightarrow \\
\mathit{set\_continueAbove} \,.\, \mathit{sequencer} \,!\, \mathbf{False} \longrightarrow \\
\mathit{requestTermination} \,.\, \mathit{currentMission} \,.\, \mathit{sequencer} \longrightarrow \\
\mathit{signalTerminationRet} \,.\, \mathit{sequencer} \longrightarrow \\
\mathbf{Skip}
\end{array} \right) ; \\
\mathit{end\_terminations} \,.\, \mathit{sequencer} \longrightarrow \\
\mathbf{Skip}
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat=} \\
\left(\begin{array}{l} SequenceTerminationPending; \\ Methods \end{array}\right) \\
\quad \Box \\
\left(\begin{array}{l} end\_methods\,.\,sequencer \longrightarrow \\ \textbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
SequenceTerminationPending \mathrel{\widehat=} \\
\quad sequenceTerminationPendingCall\,.\,sequencer \longrightarrow \\
\quad get\_continue\,.\,sequencer\,?\,continue \longrightarrow \\
\quad sequenceTerminationPendingRet\,.\,sequencer\,!\,continue \longrightarrow \\
\quad \textbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
ContinueController \mathrel{\widehat=} \textbf{var}\ continue : \mathbb{B} \bullet \\
\left(\begin{array}{l} GetContinue\ ;\ \ get\_continue\,.\,sequencer\,!\,continue \longrightarrow \\ ContinueController \end{array}\right) \\
\quad \Box \\
\left(\begin{array}{l} set\_continueBelow\,.\,sequencer\,?\,newContinueBelow \longrightarrow \\ continueBelow := newContinueBelow; \\ ContinueController \end{array}\right) \\
\quad \Box \\
\left(\begin{array}{l} set\_continueAbove\,.\,sequencer\,?\,newContinueAbove \longrightarrow \\ continueAbove := newContinueAbove; \\ ContinueController \end{array}\right) \\
\quad \Box \\
\left(\begin{array}{l} end\_methods\,.\,sequencer \longrightarrow \\ \textbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Cleanup \mathrel{\widehat=} \\
\quad cleanupSchedulableCall\,.\,sequencer \longrightarrow \\
\quad cleanupSchedulableRet\,.\,sequencer \longrightarrow \\
\quad Finish
\end{array}
$$

$$
\begin{array}{l}
Finish \mathrel{\widehat=} \\
\quad done\_schedulable\,.\,sequencer \longrightarrow \\
\quad \textbf{Skip}
\end{array}
$$

$$
\bullet \left(\begin{array}{l}
\mu X \bullet Init\ ;\ \ Start; \\
\left(\begin{array}{l}
\textbf{if}\ applicationTerminating = \textbf{False} \longrightarrow \\
\quad \left(Execute\ ;\ \ Cleanup\ ;\ \ X\right) \\
[]\, applicationTerminating = \textbf{True} \longrightarrow \\
\quad \left(\begin{array}{l} end\_sequencer\_app\,.\,sequencer \longrightarrow \\ \textbf{Skip} \end{array}\right) \\
\textbf{fi}
\end{array}\right)
\end{array}\right)
$$

end

3.1.9 Event Handlers

AperiodicEventHandlerFW

section *AperiodicEventHandlerFW* **parents** *MissionChan*, *SchedulableChan*, *SchedulableId*, *MissionId*, *MissionIds*, *TopLevelMissionSequencerChan*, *SafeletMethChan*, *FrameworkChan*, *AperiodicEventHandlerChan*, *AperiodicParameters*

process $AperiodicEventHandlerFW \mathrel{\widehat=}$
$schedulable : SchedulableID;\ aperiodicType : AperiodicType;$
$aperiodicParameters : AperiodicParameters \bullet$
begin

state *State*

$controllingMission : MissionID$
$applicationTerminating : \mathbb{B}$
$pending : \mathbb{B}$
$data : \mathbb{Z}$
$deadline : JTime$
$deadlineMissHandler : SchedulableID$

Init

$State'$

$controllingMission' = nullMissionId$
$applicationTerminating' = \mathbf{False}$
$pending' = \mathbf{False}$
$deadline' = deadlineOfAperiodic(aperiodicParameters)$
$deadlineMissHandler' = missHandlerOfAperiodic(aperiodicParameters)$

$Start \mathrel{\widehat=}$
$\begin{pmatrix} Register; \\ Activate \end{pmatrix}$
$\Box$
$\begin{pmatrix} activate\_schedulables?someMissionID \longrightarrow \\ Start \end{pmatrix}$
$\Box$
$\begin{pmatrix} done\_toplevel\_sequencer \longrightarrow \\ applicationTerminating := \mathbf{True} \end{pmatrix}$

$Register \mathrel{\widehat=}$
$register\,.\,schedulable\,?\,missionID \longrightarrow$
$controllingMission := missionID$

$Activate \mathrel{\widehat=}$
$activate\_schedulables\,.\,controllingMission \longrightarrow$
Skip

$$
\begin{array}{l}
Execute \mathrel{\widehat{=}} \\
\textbf{if } deadlineMissHandler! = nullSchedulableId \longrightarrow \\
\quad \left(\left(\left(\left(\begin{array}{l} \textbf{if } aperiodicType = aperiodic \longrightarrow \\ \quad Ready \\ [] aperiodicType = aperiodicLong \longrightarrow \\ \quad ReadyLong \\ \textbf{fi} \end{array} \right) \right) \right) \right. \\
\quad \quad [\![\{ pending, data \} \mid \{\!| end\_releases |\!\} \mid \varnothing]\!] \\
\quad \left. SignalTermination \right) \\
\quad [\![\{ pending, data \} \mid \\
\quad \quad DeadlineClockSync \cup \{\!| release\,.\,schedulable, releaseLong.schedulable |\!\} \mid \\
\quad \varnothing]\!] \\
\quad \left(\left(\left(\left(\begin{array}{l} release\,.\,schedulable \longrightarrow \textbf{Skip} \\ \Box \\ releaseLong\,.\,schedulable?data \longrightarrow \textbf{Skip} \end{array} \right) \right) \right) \right. \\
\quad \left. \begin{array}{l} ;\ DeadlineClock \\ \triangle \left(\begin{array}{l} end\_releases.schedulable \longrightarrow \\ \textbf{Skip} \end{array} \right) \end{array} \right) \\
\left. \right) \\
[] deadlineMissHandler == nullSchedulableId \longrightarrow \\
\quad \left(\left(\left(\left(\begin{array}{l} \textbf{if } aperiodicType = aperiodic \longrightarrow \\ \quad Ready \\ [] aperiodicType = aperiodicLong \longrightarrow \\ \quad ReadyLong \\ \textbf{fi} \end{array} \right) \right) \right) \right. \\
\quad \quad [\![\{ pending, data \} \mid \{\!| end\_releases |\!\} \mid \varnothing]\!] \\
\quad \left. SignalTermination \right) \\
\textbf{fi}
\end{array}
$$

$$
\begin{array}{l}
DeadlineClock \mathrel{\widehat{=}} \\
\quad \left(\left(\left(\left(\begin{array}{l} \textbf{wait } valueOf(deadline); \\ release\,.\,deadlineMissHandler \longrightarrow \\ DeadlineClock \end{array} \right) \right) \right) \right. \\
\quad \left. \begin{array}{l} \Box \\ \left(\begin{array}{l} release\_complete\,.\,schedulable \longrightarrow \\ DeadlineClock \end{array} \right) \end{array} \right) \\
\quad \triangle \left(\begin{array}{l} end\_releases\,.\,schedulable \longrightarrow \\ \textbf{Skip} \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
Ready \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} release.schedulable \longrightarrow \\ handleAsyncEventCall\,.\,schedulable \longrightarrow \\ Release \end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l} end\_releases\,.\,schedulable \longrightarrow \\ \textbf{Skip} \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{ReadyLong} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
\mathit{releaseLong}\,.\,\mathit{schedulable}\,?\,\mathit{longData} \longrightarrow \\
\mathit{data} := \mathit{longData}; \\
\mathit{handleAsyncLongEventCall}\,.\,\mathit{schedulable}\,.\,\mathit{data} \longrightarrow \\
\mathit{ReleaseLong}
\end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
\mathit{end\_releases}\,.\,\mathit{schedulable} \longrightarrow \\
\textbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{SignalTermination} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
\mathit{signalTerminationCall}\,.\,\mathit{schedulable} \longrightarrow \\
\mathit{end\_releases}\,.\,\mathit{schedulable} \longrightarrow \\
\mathit{signalTerminationRet}\,.\,\mathit{schedulable} \longrightarrow \\
\mathit{done\_schedulable}\,.\,\mathit{schedulable} \longrightarrow \\
\textbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Release} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l}
\mathit{release.schedulable} \longrightarrow \\
\mathit{pending} := \textbf{True}; \\
\mathit{Release}
\end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
\mathit{handleAsyncEventRet.schedulable} \longrightarrow \\
\textbf{if}\ \mathit{pending} = \textbf{True} \longrightarrow \\
\qquad \left(\begin{array}{l}
\mathit{pending} := \textbf{False}; \\
\mathit{release\_complete}\,.\,\mathit{schedulable} \longrightarrow \\
\mathit{handleAsyncEventCall.schedulable} \longrightarrow \\
\mathit{Release}
\end{array}\right) \\
[\!]\,\mathit{pending} = \textbf{False} \longrightarrow \\
\quad \mathit{Ready} \\
\textbf{fi}
\end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
\mathit{end\_releases.schedulable} \longrightarrow \\
\textbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{ReleaseLong} \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} \mathit{releaseLong.schedulable}\,?\,\mathit{longData} \longrightarrow \\ \mathit{data} := \mathit{longData}; \\ \mathit{pending} := \mathbf{True}; \\ \mathit{ReleaseLong} \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} \mathit{handleAsyncLongEventRet.schedulable} \longrightarrow \\ \mathbf{if}\ \mathit{pending} = \mathbf{True} \longrightarrow \\ \qquad \left(\begin{array}{l} \mathit{pending} := \mathbf{False}; \\ \mathit{release\_complete}\ .\ \mathit{schedulable} \longrightarrow \\ \mathit{handleAsyncLongEventCall.schedulable.data} \longrightarrow \\ \mathit{ReleaseLong} \end{array}\right) \\ [\!]\ \mathit{pending} = \mathbf{False} \longrightarrow \\ \quad \mathit{ReadyLong} \\ \mathbf{fi} \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} \mathit{end\_releases.schedulable} \longrightarrow \\ \mathbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Cleanup} \mathrel{\widehat{=}} \\
\quad \mathit{cleanupSchedulableCall}\ .\ \mathit{schedulable} \longrightarrow \\
\quad \mathit{cleanupSchedulableRet}\ .\ \mathit{schedulable} \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\bullet \left(\begin{array}{l} \\ \mu X \bullet \left(\begin{array}{l} \mathit{Init}\ ;\ \ \mathit{Start}; \\ \left(\begin{array}{l} \mathbf{if}\ \mathit{applicationTerminating} = \mathbf{False} \longrightarrow \\ \qquad \big(\mathit{Execute}\ ;\ \ \mathit{Cleanup}\ ;\ \ X\big) \\ [\!]\ \mathit{applicationTerminating} = \mathbf{True} \longrightarrow \\ \qquad \left(\begin{array}{l} \mathit{end\_aperiodic\_app}\ .\ \mathit{schedulable} \longrightarrow \\ \mathbf{Skip} \end{array}\right) \\ \mathbf{fi} \end{array}\right) \end{array}\right) \\ \\ \end{array}\right)
$$

end

PeriodicEventHandlerFW

section *PeriodicEventHandlerFW* **parents** *MissionChan*, *SchedulableChan*, *SchedulableId*, *MissionId*, *MissionIds*, *TopLevelMissionSequencerChan*, *PeriodicEventHandlerChan*, *SafeletMethChan*, *FrameworkChan*, *PeriodicParameters*

process $PeriodicEventHandlerFW \mathrel{\widehat=}$
$schedulable : SchedulableID;\ periodicParameters : PeriodicParameters \bullet$ **begin**

state *State*

$controllingMission : MissionID$
$applicationTerminating : \mathbb{B}$
$period : JTime$
$startTime : JTime$
$deadline : JTime$
$deadlineMissHandler : SchedulableID$
$missedReleases : \mathbb{N}$
$periodicTerminating : \mathbb{B}$

$valueOf(deadline) \leq valueOf(period)$

Init

$State'$

$controllingMission' = nullMissionId$
$applicationTerminating' = \mathbf{False}$
$periodicTerminating' = \mathbf{False}$
$period' = periodOf(periodicParameters)$
$startTimeOf(periodicParameters) = NULL \Rightarrow startTime' = time\,(0,0)$
$startTimeOf(periodicParameters) \neq NULL \Rightarrow$
$\quad startTime' = startTimeOf(periodicParameters)$
$deadlineOfPeriodic(periodicParameters) = NULL \Rightarrow$
$\quad deadline' = period'$
$deadlineOfPeriodic(periodicParameters) \neq NULL \Rightarrow$
$\quad deadline' = deadlineOfPeriodic(periodicParameters)$
$missedReleases' = 0$
$deadlineMissHandler' = missHandlerOfPeriodic(periodicParameters)$

$Start \mathrel{\widehat=}$
$\quad \begin{pmatrix} Register; \\ Activate \end{pmatrix}$
$\quad \Box$
$\quad \begin{pmatrix} activate\_schedulables?someMissionID \longrightarrow \\ Start \end{pmatrix}$
$\quad \Box$
$\quad \begin{pmatrix} done\_toplevel\_sequencer \longrightarrow \\ applicationTerminating := \mathbf{True} \end{pmatrix}$

$Register \mathrel{\widehat=}$
$\quad register\,.\,schedulable\,?\,missionID \longrightarrow$
$\quad controllingMission := missionID$

$$
\begin{array}{l}
Activate \mathrel{\widehat{=}} \\
\quad activate\_schedulables\,.\,controllingMission \longrightarrow \\
\quad \textbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
Execute \mathrel{\widehat{=}} \\
\quad \left(\left(\left(\begin{array}{l} \textbf{wait}\ valueOf(startTime); \\ \textbf{if}\ deadlineMissHandler \neq nullSchedulableId \longrightarrow \\ \quad RunningWithDeadlineDetection \\ [\!]\ deadlineMissHandler = nullSchedulableId \longrightarrow \\ \quad Running \\ \textbf{fi} \end{array} \right) \right) \right. \\
\qquad \Box \\
\qquad \left(\left(\begin{array}{l} end\_releases\,.\,schedulable \longrightarrow \\ \textbf{Skip} \end{array} \right) \right) \\
\qquad [\![\{ startTime \} \mid \{\!|\ stop\_period\ |\!\} \mid \varnothing]\!] \\
\quad \left. SignalTermination \right) \\
\quad [\![\{ startTime \} \mid PTCSYnc \mid \varnothing]\!] \\
PeriodicTerminatingController
\end{array}
$$

$$
\begin{array}{l}
Running \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} PeriodicClock \\ \quad [\![\varnothing \mid ReleaseSync \mid \{ missedReleases \}]\!] \\ Release(0) \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
RunningWithDeadlineDetection \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} Running \\ \quad [\![\{ missedReleases \} \mid ReleaseSync \mid \varnothing]\!] \\ DeadlineClock(0) \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
PeriodicClock \mathrel{\widehat{=}} \\
\quad release\,.\,schedulable \longrightarrow \\
\quad \left(\mu X \bullet \left(\begin{array}{l} \left(\begin{array}{l} \textbf{wait}\ valueOf(period); \\ release\,.\,schedulable \longrightarrow \\ X \end{array} \right) \\ \Box \\ \left(\begin{array}{l} end\_releases\,.\,schedulable \longrightarrow \\ \textbf{Skip} \end{array} \right) \end{array} \right) \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Release} \mathrel{\widehat{=}} \mathbf{val}\ \mathit{index} : \mathbb{N} \bullet \\
\mathbf{if}\ \mathit{missedReleases} = 0 \longrightarrow \\
\quad \left(\begin{array}{l} \mathit{release}\ .\ \mathit{schedulable} \longrightarrow \\ \mathit{handleAsyncEventCall}\ .\ \mathit{schedulable} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\
[\!]\ \mathit{missedReleases} \neq 0 \longrightarrow \\
\quad \left(\begin{array}{l} \mathit{handleAsyncEventCall}\ .\ \mathit{schedulable} \longrightarrow \\ \mathit{missedReleases} := \mathit{missedReleases} - 1; \\ \mathbf{Skip} \end{array} \right) \\
\mathbf{fi}\ ; \\
\left(\begin{array}{l}
\left(\begin{array}{l} \mathit{handleAsyncEventRet}\ .\ \mathit{schedulable} \longrightarrow \\ \mathit{periodic\_release\_complete}\ .\ \mathit{schedulable}\ .\ \mathit{index} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\
\quad [\![\varnothing \mid \{\!|\ \mathit{handleAsyncEventRet}\ |\!\} \mid \varnothing]\!] \\
\left(\begin{array}{l} \mu X \bullet \left(\begin{array}{l} \left(\begin{array}{l} \mathit{release}\ .\ \mathit{schedulable} \longrightarrow \\ \mathit{missedReleases} := \mathit{missedReleases} + 1; \\ X \end{array} \right) \\ \Box \\ \left(\begin{array}{l} \mathit{handleAsyncEventRet}\ .\ \mathit{schedulable} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \end{array} \right) \end{array} \right) \\
\left(\begin{array}{l} \mathit{get\_periodicTerminating}\ .\ \mathit{schedulable}\ ?\ \mathit{periodicTerminating} : (\mathit{periodicTerminating} = \mathbf{False}) \longrightarrow \\ \mathit{Release}(\mathit{index} + 1) \end{array} \right) \\
\Box \\
\left(\begin{array}{l} \mathit{get\_periodicTerminating}\ .\ \mathit{schedulable}\ ?\ \mathit{periodicTerminating} : (\mathit{periodicTerminating} = \mathbf{True}) \longrightarrow \\ \mathbf{Skip} \end{array} \right)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{DeadlineClock} \mathrel{\widehat{=}} \mathbf{val}\ \mathit{index} : \mathbb{N} \bullet \\
\left(\begin{array}{l}
\left(\begin{array}{l}
\left(\begin{array}{l} \mathbf{wait}\ \mathit{valueOf}(\mathit{deadline}); \\ \mathit{release}\ .\ \mathit{deadlineMissHandler} \longrightarrow \\ \mathit{periodic\_release\_complete}\ .\ \mathit{schedulable}\ .\ \mathit{index} \longrightarrow \\ \mathbf{Skip} \end{array} \right) \\
\Box \\
\left(\begin{array}{l} \mathit{periodic\_release\_complete}\ .\ \mathit{schedulable}\ .\ \mathit{index} \longrightarrow \\ \mathbf{Skip} \end{array} \right)
\end{array} \right) \\
\quad \triangle \left(\begin{array}{l} \mathit{end\_releases}\ .\ \mathit{schedulable} \longrightarrow \\ \mathit{periodic\_release\_complete}\ .\ \mathit{schedulable}\ ?\ \mathit{index} \longrightarrow \\ \mathbf{Skip} \end{array} \right)
\end{array} \right) \\
\qquad ||| \left(\left(\begin{array}{l} \mathbf{wait}\ \mathit{valueOf}(\mathit{period}); \\ \mathit{DeadlineClock}(\mathit{index} + 1) \end{array} \right) \right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{SignalTermination} \mathrel{\widehat{=}} \\
\quad \mathit{signalTerminationCall}\ .\ \mathit{schedulable} \longrightarrow \\
\quad \mathit{set\_periodicTerminating}\ .\ \mathit{schedulable}\ !\ \mathbf{True} \longrightarrow \\
\quad \mathit{end\_releases}\ .\ \mathit{schedulable} \longrightarrow \\
\quad \mathit{signalTerminationRet}\ .\ \mathit{schedulable} \longrightarrow \\
\quad \mathit{done\_schedulable}\ .\ \mathit{schedulable} \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
\mathit{Cleanup} \mathrel{\widehat{=}} \\
\quad \mathit{cleanupSchedulableCall}\ .\ \mathit{schedulable} \longrightarrow \\
\quad \mathit{cleanupSchedulableRet}\ .\ \mathit{schedulable} \longrightarrow \\
\quad \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
PeriodicTerminatingController \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} get\_periodicTerminating\,.\,schedulable\,!\,periodicTerminating \longrightarrow \\ PeriodicTerminatingController \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} set\_periodicTerminating\,.\,schedulable\,?\,newPeriodicTerminating \longrightarrow \\ periodicTerminating := newPeriodicTerminating; \\ PeriodicTerminatingController \end{array}\right) \\
\\
\bullet \left(\begin{array}{l} \\ \mu X \bullet \left(\begin{array}{l} Init\ ;\ \ Start; \\ \left(\begin{array}{l} \textbf{if}\ applicationTerminating = \textbf{False} \longrightarrow \\ \qquad \big(Execute\ ;\ \ Cleanup\ ;\ \ X\big) \\ [\!]\, applicationTerminating = \textbf{True} \longrightarrow \\ \qquad \left(\begin{array}{l} end\_periodic\_app\,.\,schedulable \longrightarrow \\ \textbf{Skip} \end{array}\right) \\ \textbf{fi} \end{array}\right) \end{array}\right) \\ \\ \end{array}\right) \\
\\
\textbf{end}
\end{array}
$$

OneShotEventHandlerFW

section *OneShotEventHandlerFW* **parents** *MissionChan*, *SchedulableChan*, *SchedulableId*, *MissionId*, *MissionIds*, *TopLevelMissionSequencerChan*, *OneShotEventHandlerChan*, *SafeletMethChan*, *FrameworkChan*, *AperiodicParameters*

process $OneShotEventHandlerFW \mathrel{\widehat=}$
$schedulable : SchedulableID;\ startTime : JTime;\ aperiodicParameters : AperiodicParameters \bullet$
begin

state *State*

$controllingMission : MissionID$
$applicationTerminating : \mathbb{B}$
$deadline : JTime$
$deadlineMissHandler : SchedulableID$

Init

$State'$

$controllingMission' = nullMissionId$
$applicationTerminating' = \mathbf{False}$
$deadline' = deadlineOfAperiodic(aperiodicParameters)$
$deadlineMissHandler' = missHandlerOfAperiodic(aperiodicParameters)$

$Start \mathrel{\widehat=}$
$\begin{pmatrix} Register; \\ Activate \end{pmatrix}$
$\Box$
$\begin{pmatrix} activate\_schedulables?someMissionID \longrightarrow \\ Start \end{pmatrix}$
$\Box$
$\begin{pmatrix} done\_toplevel\_sequencer \longrightarrow \\ applicationTerminating := \mathbf{True} \end{pmatrix}$

$Register \mathrel{\widehat=}$
$register\,.\,schedulable\,?\,mID \longrightarrow$
$controllingMission := mID$

$Activate \mathrel{\widehat=}$
$activate\_schedulables\,.\,controllingMission \longrightarrow$
Skip

$Execute \mathrel{\widehat=}$
$\left(\begin{pmatrix}\begin{pmatrix} Run \\ [\![\varnothing \mid MethodsSync \mid \varnothing]\!] \\ Methods \end{pmatrix} \\ [\![\varnothing \mid \{\!| end\_releases |\!\} \mid \varnothing]\!] \\ SignalTermination \end{pmatrix} \\ [\![\varnothing \mid STCSync \mid \{startTime\}]\!] \\ StartTimeController\right)$

$$
\begin{array}{l}
Run \mathrel{\widehat{=}} \\
\quad \textbf{if } deadlineMissHandler = nullSchedulableId \longrightarrow \\
\quad \left(\begin{array}{l} ScheduleOrWait \\ [\![\varnothing \mid ReleaseSync \mid \varnothing]\!] \\ Release \end{array}\right) \\
\quad [\!] deadlineMissHandler \neq nullSchedulableId \longrightarrow \\
\quad \left(\begin{array}{l} \left(\begin{array}{l} ScheduleOrWait \\ [\![\varnothing \mid ReleaseSync \mid \varnothing]\!] \\ Release \end{array}\right) [\![\varnothing \mid DeadlineSync \mid \varnothing]\!] \\ DeadlineClock \end{array}\right) \\
\quad \textbf{fi}
\end{array}
$$

$$
\begin{array}{l}
ScheduleOrWait \mathrel{\widehat{=}} \\
\quad get\_startTime\,.\,schedulable\,?\,startTime \longrightarrow \\
\quad \textbf{if } startTime! = NULL \longrightarrow \\
\quad\quad Scheduled \\
\quad [\!] startTime = NULL \longrightarrow \\
\quad\quad NotScheduled \\
\quad \textbf{fi}
\end{array}
$$

$$
\begin{array}{l}
Release \mathrel{\widehat{=}} \\
\quad \left(\begin{array}{l} handleAsyncEventCall\,.\,schedulable \longrightarrow \\ handleAsyncEventRet\,.\,schedulable \longrightarrow \\ release\_complete\,.\,schedulable \longrightarrow \\ Release \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} reschedule\_handler\,.\,schedulable\,?\,newStartTime \longrightarrow \\ set\_startTime\,.\,schedulable\,!\,newStartTime \longrightarrow \\ Release \end{array}\right) \\
\quad \Box \\
\quad \left(\begin{array}{l} end\_releases\,.\,schedulable \longrightarrow \\ stop\_release\,.\,schedulable \longrightarrow \\ \textbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
DeadlineClock \mathrel{\widehat{=}} \\
\quad release\,.\,schedulable \longrightarrow \\
\quad \left(\begin{array}{l} \left(\begin{array}{l} \left(\begin{array}{l} \textbf{wait } valueOf(deadline); \\ release\,.\,deadlineMissHandler \longrightarrow \\ DeadlineClock \end{array}\right) \\ \Box \\ \left(\begin{array}{l} release\_complete\,.\,schedulable \longrightarrow \\ DeadlineClock \end{array}\right) \Box \\ \left(\begin{array}{l} deschedule\_handler\,.\,schedulable \longrightarrow \\ DeadlineClock \end{array}\right) \end{array}\right) \\
\triangle \left(\begin{array}{l} end\_releases\,.\,schedulable \longrightarrow \\ \textbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Scheduled \mathrel{\widehat=} \\
\quad get\_startTime\,.\,schedulable\,?\,startTime \longrightarrow \\
\quad \left(\begin{array}{l}
\left(\begin{array}{l}
\textbf{wait}\ valueOf\,(startTime) \\
release\,.\,schedulable \longrightarrow \\
handleAsyncEventCall\,.\,schedulable \longrightarrow \\
NotScheduled
\end{array} \right) \\
\triangle \\
\left(\begin{array}{l}
\left(\begin{array}{l}
deschedule\_handler\,.\,schedulable \longrightarrow \\
NotScheduled
\end{array} \right) \\
\Box \\
\left(\begin{array}{l}
reschedule\_handler\,.\,schedulable\,?\,newStartTime \longrightarrow \\
set\_startTime\,.\,schedulable\,!\,newStartTime \longrightarrow \\
Scheduled
\end{array} \right)
\end{array} \right)
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
NotScheduled \mathrel{\widehat=} \\
\quad \left(\begin{array}{l}
deschedule\_handler\,.\,schedulable \longrightarrow \\
NotScheduled
\end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
reschedule\_handler\,.\,schedulable\,?\,newStartTime \longrightarrow \\
set\_startTime\,.\,schedulable\,!\,newStartTime \longrightarrow \\
Scheduled
\end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
end\_releases\,.\,schedulable \longrightarrow \\
\textbf{Skip}
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat=} \\
\quad \left(\begin{array}{l}
Deschedule; \\
Methods
\end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
GetNextReleaseTime; \\
Methods
\end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
ScheduleNextRelease; \\
Methods
\end{array} \right) \\
\quad \Box \\
\quad \left(\begin{array}{l}
end\_releases\,.\,schedulable \longrightarrow \\
\textbf{Skip}
\end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
Deschedule \mathrel{\widehat=} \textbf{var}\ wasScheduled : \mathbb{B} \bullet \\
\quad descheduleCall\,.\,schedulable \longrightarrow \\
\quad deschedule\_handler.schedulable \longrightarrow \\
\quad get\_startTime\,.\,schedulable\,?\,startTime \longrightarrow \\
\quad \left(\begin{array}{l}
\textbf{if}\ startTime = NULL \longrightarrow \\
\quad\quad wasScheduled := \textbf{False} \\
[]\, startTime \neq NULL \longrightarrow \\
\quad\quad wasScheduled := \textbf{True} \\
\textbf{fi}\,; \\
set\_startTime\,.\,schedulable\,!\,NULL \longrightarrow \\
descheduleRet\,.\,schedulable\,!\,wasScheduled \longrightarrow \\
\textbf{Skip}
\end{array} \right)
\end{array}
$$

$GetNextReleaseTime \mathrel{\widehat=}$
$\quad getNextReleaseTimeCall\,.\,schedulable \longrightarrow$
$\quad get\_startTime\,.\,schedulable\,?\,startTime \longrightarrow$
$\quad getNextReleaseTimeRet\,.\,schedulable\,!\,startTime \longrightarrow$
$\quad \textbf{Skip}$

$ScheduleNextRelease \mathrel{\widehat=}$
$\quad scheduleNextRelease\,.\,schedulable\,?\,newStartTime \longrightarrow$
$\quad set\_startTime\,.\,schedulable\,!\,newStartTime \longrightarrow$
$\quad \textbf{if}\ newStartTime = NULL \longrightarrow$
$\qquad \begin{pmatrix} deschedule\_handler.schedulable \longrightarrow \\ \textbf{Skip} \end{pmatrix}$
$\quad [] newStartTime \neq NULL \longrightarrow$
$\qquad \begin{pmatrix} reschedule\_handler\,!\,schedulable\,!\,newStartTime \longrightarrow \\ \textbf{Skip} \end{pmatrix}$
$\quad \textbf{fi}$

$SignalTermination \mathrel{\widehat=}$
$\quad signalTerminationCall\,.\,schedulable \longrightarrow$
$\quad end\_releases\,.\,schedulable \longrightarrow$
$\quad signalTerminationRet\,.\,schedulable \longrightarrow$
$\quad done\_schedulable\,.\,schedulable \longrightarrow$
$\quad \textbf{Skip}$

$StartTimeController \mathrel{\widehat=}$
$\quad \begin{pmatrix} get\_startTime\,.\,schedulable\,!\,startTime \longrightarrow \\ StartTimeController \end{pmatrix}$
$\quad \Box$
$\quad \begin{pmatrix} set\_startTime\,.\,schedulable\,?\,newStartTime \longrightarrow \\ StartTimeController \end{pmatrix}$

$Cleanup \mathrel{\widehat=}$
$\quad cleanupSchedulableCall\,.\,schedulable \longrightarrow$
$\quad cleanupSchedulableRet\,.\,schedulable \longrightarrow$
$\quad \textbf{Skip}$

$$\bullet \left(\begin{array}{l} \mu X \bullet \left(\begin{array}{l} Init\ ;\ \ Start; \\ \left(\begin{array}{l} \textbf{if}\ applicationTerminating = \textbf{False} \longrightarrow \\ \quad \big(Execute\ ;\ \ Cleanup\ ;\ \ X \big) \\ [] applicationTerminating = \textbf{True} \longrightarrow \\ \quad \begin{pmatrix} end\_oneShot\_app\,.\,schedulable \longrightarrow \\ \textbf{Skip} \end{pmatrix} \\ \textbf{fi} \end{array} \right) \end{array} \right) \end{array} \right)$$

end

3.1.10 ManagedThreadFW

section *ManagedThread* **parents** *ManagedThreadChan*, *SchedulableId*, *MissionId*, *MissionIds*, *TopLevelMissionSequencerChan*, *SchedulableChan*, *SafeletMethChan*, *FrameworkChan*

process $ManagedThreadFW \mathrel{\widehat=} schedulable : SchedulableID \bullet$ **begin**

State

$controllingMission : MissionID$
$applicationTerminating : \mathbb{B}$

Init

$State'$

$controllingMission' = nullMissionId$
$applicationTerminating' =$ **False**

$Start \mathrel{\widehat=}$
$\begin{pmatrix} Register; \\ Activate \end{pmatrix}$
$\Box$
$\begin{pmatrix} activate\_schedulables?someMissionID \longrightarrow \\ Start \end{pmatrix}$
$\Box$
$\begin{pmatrix} done\_toplevel\_sequencer \longrightarrow \\ applicationTerminating := \textbf{True} \end{pmatrix}$

$Register \mathrel{\widehat=}$
$register\,.\,schedulable\,?\,mID \longrightarrow$
$controllingMission := mID$

$Activate \mathrel{\widehat=}$
$activate\_schedulables\,.\,controllingMission \longrightarrow$
Skip

$Execute \mathrel{\widehat=}$
$Run \,[\![\, \varnothing \mid \{\!|\, runRet \,|\!\} \mid \varnothing \,]\!]\, Methods$

$Run \mathrel{\widehat=}$
$runCall\,.\,schedulable \longrightarrow$
$runRet\,.\,schedulable \longrightarrow$
$done\_schedulable\,.\,schedulable \longrightarrow$
Skip

$$
\begin{array}{l}
\mathit{Methods} \mathrel{\widehat{=}} \\
\quad \big(\mathit{SignalTerminationMeth}\ ;\ \ \mathit{Methods}\big) \\
\quad \Box \\
\quad \left(\begin{array}{l} \mathit{runRet}\,.\,\mathit{schedulable} \longrightarrow \\ \mathbf{Skip} \end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
\mathit{SignalTerminationMeth} \mathrel{\widehat{=}} \\
\quad \mathit{signalTerminationCall}\,.\,\mathit{schedulable} \longrightarrow \\
\quad \mathit{signalTerminationRet}\,.\,\mathit{schedulable} \longrightarrow \mathbf{Skip}
\end{array}
$$

$$
\begin{array}{l}
\mathit{Cleanup} \mathrel{\widehat{=}} \\
\quad \mathit{cleanupSchedulableCall}\,.\,\mathit{schedulable} \longrightarrow \\
\quad \mathit{cleanupSchedulableRet}\,.\,\mathit{schedulable} \longrightarrow \mathbf{Skip}
\end{array}
$$

$$
\bullet \left(\begin{array}{l}
\mu X \bullet \mathit{Init}\ ;\ \ \mathit{Start}; \\
\left(\begin{array}{l}
\mathbf{if}\ \mathit{applicationTerminating} = \mathbf{False} \longrightarrow \\
\qquad \big(\mathit{Execute}\ ;\ \ \mathit{Cleanup}\ ;\ \ X\big) \\
[]\, \mathit{applicationTerminating} = \mathbf{True} \longrightarrow \\
\qquad \left(\begin{array}{l} \mathit{end\_managedThread\_app}\,.\,\mathit{schedulable} \longrightarrow \\ \mathbf{Skip} \end{array}\right) \\
\mathbf{fi}
\end{array}\right)
\end{array}\right)
$$

end

3.2 Appendix B: FlatBuffer Script

3.2.1 FlatBuffer

```
1  public class FlatBuffer implements Safelet<Mission>{
2
3    public Level getLevel(){
4      return Level.LEVEL_2;
5    }
6
7    public MissionSequencer<Mission> getSequencer(){
8      // Create and return the main mission sequencer
9      StorageParameters storageParameters = new StorageParameters(
10         Const.OVERALL_BACKING_STORE_DEFAULT - 1000000,
11         new long[] { Const.HANDLER_STACK_SIZE },
12         Const.PRIVATE_MEM_DEFAULT, 10000 * 2, Const.MISSION_MEM_DEFAULT);
13
14     return new FlatBufferMissionSequencer(new PriorityParameters(5),
15         storageParameters);
16   }
17
18   @Override
19   public long immortalMemorySize(){
20     return Const.IMMORTAL_MEM_DEFAULT;
21   }
22
23   @Override
24   public void initializeApplication(){
25   }
26 }
```

3.2.2 FlatBufferMissionSequencer

```
1  public class FlatBufferMissionSequencer extends MissionSequencer<Mission>
2  {
3    private boolean returnedMission;
4
5    public FlatBufferMissionSequencer(PriorityParameters priorityParameters,
6        StorageParameters storageParameters){
7      super(priorityParameters, storageParameters);
8      returnedMission = false;
9    }
10
11   protected Mission getNextMission(){
12
13     if (!returnedMission){
14       returnedMission = true;
15       return new FlatBufferMission();
16     }
17     else{
18       return null;
19     }
20   }
21 }
```

3.2.3 FlatBufferMission

```
1  public class FlatBufferMission extends Mission{
2
3    private volatile int buffer;
4
5    public FlatBufferMission(){
6      Console.println("FlatBufferMission");
7      buffer = 0;
8      Services.setCeiling(this, 20);
9    }
10
11   protected void initialize(){
12     StorageParameters storageParameters = new StorageParameters(150 * 1000,
13         new long[] { Const.HANDLER_STACK_SIZE },
14         Const.PRIVATE_MEM_DEFAULT, Const.IMMORTAL_MEM_DEFAULT,
15         Const.MISSION_MEM_DEFAULT - 100 * 1000);
16
17     new Reader(new PriorityParameters(10), storageParameters, this).register();
18     new Writer(new PriorityParameters(10), storageParameters, this).register();
19   }
20
21   public boolean bufferEmpty(){
22
23     return buffer == 0;
24   }
25
26   public synchronized void write(int update) throws InterruptedException{
27
28     while (!bufferEmpty("Writer")){
29       this.wait();
30     }
31     buffer = update;
32     this.notify();
33   }
34
35   public synchronized int read() throws InterruptedException{
36     while(bufferEmpty("Reader")){
37       this.wait();
38     }
39     int out = buffer;
40     buffer = 0;
41     this.notify();
42     return out;
43   }
44
45   public boolean cleanUp(){
46     return false;
47   }
48
49   public long missionMemorySize(){
50     return 1048576;
51   }
52 }
```

3.2.4 Writer

```
1  public class Writer extends ManagedThread{
2
3    private final FlatBufferMission fbMission;
4    private int i = 1;
5
6    public Writer(PriorityParameters priority, StorageParameters storage,
7        FlatBufferMission fbMission){
8      super(priority, storage);
9      this.fbMission = fbMission;
10   }
11
12   public void run(){
13     while (!fbMission.terminationPending()){
14       try{
15         fbMission.write(i);
16       }
17       catch (InterruptedException e){
18         e.printStackTrace();
19       }
20
21       i++;
22
23       boolean keepWriting = i >= 5 ;
24       if(!keepWriting){
25         fbMission.requestTermination();
26       }
27     }
28   }
29 }
```

3.2.5 Reader

```
1  public class Reader extends ManagedThread{
2
3    private final FlatBufferMission fbMission;
4
5    public Reader(PriorityParameters priority, StorageParameters storage,
6        FlatBufferMission fbMission){
7      super(priority, storage);
8      this.fbMission = fbMission;
9    }
10
11   public void run(){
12     while (!fbMission.terminationPending())
13     {
14       int result=999;
15       try{
16         result = fbMission.read();
17       }
18       catch (InterruptedException e){
19         e.printStackTrace();
20       }
21       Console.println("Reader Read " + result + " from Buffer");
22     }
23   }
24 }
```

References

[1] The Open Group. Safety-Critical Java Technology Specification. Technical report, The Open Group, 27 December 2014.

[2] Andy Wellings, Matt Luckcuck, and Ana Cavalcanti. Safety-critical java level 2: motivations, example applications and issues. In *Proceedings of the 11th International Workshop on Java Technologies for Real-time and Embedded Systems*, JTRES '13, pages 48–57, New York, NY, USA, 9 October 2013. ACM.

[3] Jim Woodcock and Ana Cavalcanti. The Semantics of Circus. In Didier Bert, Jonathan P. Bowen, Martin C. Henson, and Ken Robinson, editors, *ZB 2002:Formal Specification and Development in Z and B*, volume 2272 of *Lecture Notes in Computer Science*, pages 184–203. Springer Berlin Heidelberg, 2002.