```
MODULE selector
EXTENDS $TLC$, $Sequences$, $SequencesExt$, $FiniteSets$, $Integers$

CONSTANTS $Timers$, $DeltaRange$, $Servers$, $Clients$, $Subscribers$

$Tasks \triangleq Subscribers \cup Servers \cup Clients$

  --algorithm selector
variables
     list for timer
     example: $\langle [delta \mapsto 3, name \mapsto \text{“timer1”}], [delta \mapsto 2, name \mapsto \text{“timer2”}] \rangle$
    $delta\_list = SetToSeq(\{[delta \mapsto random\_num(0, DeltaRange), name \mapsto x] : x \in Timers\})$ ;

     events
    $wait\_set = \{\}$ ;

     tasks
    $running = \{\}$ ;
    $waiting = Tasks$ ;

define
    $random\_num(min, max) \triangleq$ CHOOSE $i \in min \,..\, max$ : TRUE
    $pick\_task(set) \triangleq$ CHOOSE $x \in set$ : TRUE

    $starvation\_free \triangleq \forall x \in (Timers \cup Tasks)$ :
        LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$ IN
        $(((x \in delta\_set) \lor (x \in wait\_set)) \leadsto \Diamond(x \in running))$
    $running\_xor\_waiting \triangleq \forall x \in Tasks$ :
        $(x \in running \land x \notin waiting) \lor (x \notin running \land x \in waiting)$
    $running\_then\_not\_delta\_list \triangleq \forall x \in Timers$ :
        LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$ IN
        $x \in running \Rightarrow x \notin delta\_set$
    $type\_check \triangleq$
        LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$ IN
         $\land\ waiting \subseteq Tasks$
         $\land\ running \subseteq (Tasks \cup Timers)$
         $\land\ delta\_set \subseteq Timers$
end define

 To emulate incrementing clock, decrement the delta of the head of the $delta\_list$.
macro $increment\_clock()$
begin
    if $delta\_list \neq \langle\rangle \land delta\_list[1].delta > 0$ then
        $delta\_list[1].delta := delta\_list[1].delta - 1$ ;
    end if ;
end macro ;

 execute a $callback$ function
```

```
**procedure** $callback(name)$
**begin**
    $BeginCallback$:
        $increment\_clock()$ ;
        $running := running \cup \{name\}$ ;
        $waiting\ := waiting \setminus \{name\}$ ;

    $EndCallback$:
        $running := running \setminus \{name\}$ ;
        **if** $name \in Tasks$ **then**
            $waiting := waiting \cup\ \{name\}$
        **end if** ;
        **return** ;
**end procedure** ;

  reenable timer with at random delay
**procedure** $reload\_timer(name)$
**variables**
    $idx$ ;
    $delta$ ;
**begin**
    $BeginReloadTimer$:
        $increment\_clock()$ ;

          choose insertion point
        $idx := random\_num(1,\ Len(delta\_list) + 1)$ ;
        **if** $idx \leq Len(delta\_list)$ **then**
              insert to middle
            $delta := random\_num(0,\ delta\_list[idx].delta)$ ;

            $reload\_insert1$:
                  update delta and insert
                $delta\_list[idx].delta := delta\_list[idx].delta - delta$ ;

            $reload\_insert2$:
                $delta\_list := InsertAt(delta\_list,\ idx,\ [delta \mapsto delta,\ name \mapsto name])$ ;
        **else**
              insert to the end
            $delta := random\_num(0,\ DeltaRange)$ ;

            $reload\_insert\_end$:
                $delta\_list := Append(delta\_list,\ [delta \mapsto delta,\ name \mapsto name])$ ;
            **skip** ;
        **end if** ;

    $EndReloadTimer$:
        **return** ;
```

```
end procedure ;

  execute a task

  safe_drive:: selector:: notify
procedure notify(runnable)
variables
    task ;
begin
    BeginNotify:
        while $runnable \neq \{\}$ do
            $task := pick\_task(runnable)$ ;
            $runnable := runnable \setminus \{task\}$ ;
            call callback(task) ;
        end while ;

    EndNotify:
        return;
end procedure ;

  wait with timeout.

  safe_drive:: selector:: notify_timer
procedure notify_timer()
variables
    head ;
    $to\_be\_reloaded = \langle\rangle$ ;
begin
    BeginNotifyTimer:
        while $delta\_list \neq \langle\rangle \land delta\_list[1].delta = 0$ do
              pop front
            $head := Head(delta\_list)$ ;
            $delta\_list := Tail(delta\_list)$ ;

              call the callback function
            call callback(head.name) ;

              reenable timer later
            save_timer:
                $to\_be\_reloaded := Append(to\_be\_reloaded, head.name)$ ;
        end while ;

    ReladTimer:
          reenable timer
        while $to\_be\_reloaded \neq \langle\rangle$ do
            call reload_timer(to_be_reloaded[1]) ;
            $reload2$:
                $to\_be\_reloaded := Tail(to\_be\_reloaded)$ ;
        end while ;

    $EndNotifyTimer$:
        return ;
end procedure ;

 Emulate $ROS2$'s $rcl\_wait()$
procedure $rcl\_wait()$
begin
    $BeginRclWait$:
        while $delta\_list \neq \langle\rangle \land delta\_list[1].delta > 0 \land wait\_set = \{\}$ do
            $increment\_clock()$ ;
        end while ;

    $EndRclWait$:
        return ;
end procedure ;

 $safe\_drive::selector::wait\_timer$
procedure $wait\_timer()$
begin
    $BeginWaitTimer$:
        call $rcl\_wait()$ ;

    $EndWaitTimer$:
        return ;
end procedure ;

 $safe\_drive::selector::wait$
procedure $wait()$
begin
    $BeginWait$:
        call $wait\_timer()$ ;

    $NotifyTimer$:
        call $notify\_timer()$ ;

    $Notify$:
         pick $wait\_set$ tasks up
        with $tmp\_wait\_set = wait\_set$ do
            $wait\_set := \{\}$ ;
            call $notify(tmp\_wait\_set)$ ;
        end with ;

    $EndWait$:
        return ;
```

```
end procedure ;

fair process $trigger\_event \in Tasks$
begin
    $fire\_event$:
        while TRUE do
            $wait\_set := wait\_set \cup \{self\}$ ;
        end while ;
end process ;

fair +process $executor =$ "executor"
variables
    $head$ ;
    $to\_be\_reloaded = \langle\rangle$ ;
begin
    $BeginExecutor$:
        while TRUE do
            call $wait()$ ;
        end while ;
end process ;
end algorithm ;
 BEGIN TRANSLATION ($chksum(pcal) =$ "615b0ea5" $\land chksum(tla) =$ "7cf79cd9")
 Process variable head of process executor at line 203 col 5 changed to $head\_$
 Process variable $to\_be\_reloaded$ of process executor at line 204 col 5 changed to $to\_be\_reloaded\_$
 Parameter name of procedure $callback$ at line 49 col 20 changed to $name\_$
CONSTANT $defaultInitValue$
VARIABLES $delta\_list, wait\_set, running, waiting, pc, stack$

 define statement
$random\_num(min, max) \triangleq$ CHOOSE $i \in min \,..\, max$ : TRUE
$pick\_task(set) \triangleq$ CHOOSE $x \in set$ : TRUE

$starvation\_free \triangleq \forall x \in (Timers \cup Tasks)$ :
    LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$IN
    $(((x \in delta\_set) \lor (x \in wait\_set)) \leadsto \Diamond(x \in running))$
$running\_xor\_waiting \triangleq \forall x \in Tasks$ :
    $(x \in running \land x \notin waiting) \lor (x \notin running \land x \in waiting)$
$running\_then\_not\_delta\_list \triangleq \forall x \in Timers$ :
    LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$IN
    $x \in running \Rightarrow x \notin delta\_set$
$type\_check \triangleq$
    LET $delta\_set \triangleq \{y.name : y \in ToSet(delta\_list)\}$IN
     $\land waiting \subseteq Tasks$
     $\land running \subseteq (Tasks \cup Timers)$
     $\land delta\_set \subseteq Timers$
```

```
VARIABLES name_, name, idx, delta, runnable, task, head, to_be_reloaded,
          head_, to_be_reloaded_

vars ≜ ⟨delta_list, wait_set, running, waiting, pc, stack, name_, name,
        idx, delta, runnable, task, head, to_be_reloaded, head_,
        to_be_reloaded_⟩

ProcSet ≜ (Tasks) ∪ {"executor"}

Init ≜  Global variables
        ∧ delta_list = SetToSeq({[delta ↦ random_num(0, DeltaRange), name ↦ x] : x ∈ Timers})
        ∧ wait_set = {}
        ∧ running = {}
        ∧ waiting  = Tasks
         Procedure callback
        ∧ name_ = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure reload_timer
        ∧ name = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ idx = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ delta = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure notify
        ∧ runnable = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ task = [self ∈ ProcSet ↦ defaultInitValue]
         Procedure notify_timer
        ∧ head = [self ∈ ProcSet ↦ defaultInitValue]
        ∧ to_be_reloaded = [self ∈ ProcSet ↦ ⟨⟩]
         Process executor
        ∧ head_ = defaultInitValue
        ∧ to_be_reloaded_ = ⟨⟩
        ∧ stack = [self ∈ ProcSet ↦ ⟨⟩]
        ∧ pc = [self ∈ ProcSet ↦ CASE self ∈ Tasks → "fire_event"
                                    □   self = "executor" → "BeginExecutor"]

BeginCallback(self) ≜ ∧ pc[self] = "BeginCallback"
                      ∧ IF delta_list ≠ ⟨⟩ ∧ delta_list[1].delta > 0
                           THEN ∧ delta_list′ = [delta_list EXCEPT ![1].delta = delta_list[1].delta − 1]
                           ELSE ∧ TRUE
                                ∧ UNCHANGED delta_list
                      ∧ running′ = (running ∪ {name_[self]})
                      ∧ waiting′  = waiting \ {name_[self]}
                      ∧ pc′ = [pc EXCEPT ![self] = "EndCallback"]
                      ∧ UNCHANGED ⟨wait_set, stack, name_, name, idx,
                                   delta, runnable, task, head,
                                   to_be_reloaded, head_, to_be_reloaded_⟩

EndCallback(self) ≜ ∧ pc[self]  = "EndCallback"
```

```
        $\land\ running' = running \setminus \{name\_[self]\}$
        $\land$ IF $name\_[self] \in Tasks$
             THEN $\land\ waiting' = (waiting \cup \{name\_[self]\})$
             ELSE $\land$ TRUE
                  $\land$ UNCHANGED $waiting$
        $\land\ pc' = [pc$ EXCEPT $![self] = Head(stack[self]).pc]$
        $\land\ name\_' = [name\_$ EXCEPT $![self] = Head(stack[self]).name\_]$
        $\land\ stack' = [stack$ EXCEPT $![self] = Tail(stack[self])]$
        $\land$ UNCHANGED $\langle delta\_list, wait\_set, name, idx, delta,$
                      $runnable, task, head, to\_be\_reloaded,$
                      $head\_, to\_be\_reloaded\_\rangle$

$callback(self) \triangleq BeginCallback(self) \lor EndCallback(self)$

$BeginReloadTimer(self) \triangleq$ $\land\ pc[self] =$ "BeginReloadTimer"
        $\land$ IF $delta\_list \neq \langle\rangle \land delta\_list[1].delta > 0$
             THEN $\land\ delta\_list' = [delta\_list$ EXCEPT $![1].delta = delta\_list[1].delta -$
             ELSE $\land$ TRUE
                  $\land$ UNCHANGED $delta\_list$
        $\land\ idx' = [idx$ EXCEPT $![self] = random\_num(1, Len(delta\_list') + 1)]$
        $\land$ IF $idx'[self] \leq Len(delta\_list')$
             THEN $\land\ delta' = [delta$ EXCEPT $![self] = random\_num(0, delta\_list'[id$
                  $\land\ pc' = [pc$ EXCEPT $![self] =$ "reload_insert1"$]$
             ELSE $\land\ delta' = [delta$ EXCEPT $![self] = random\_num(0, DeltaRange)]$
                  $\land\ pc' = [pc$ EXCEPT $![self] =$ "reload_insert_end"$]$
        $\land$ UNCHANGED $\langle wait\_set, running, waiting, stack,$
                      $name\_, name, runnable, task, head,$
                      $to\_be\_reloaded, head\_,$
                      $to\_be\_reloaded\_\rangle$

$reload\_insert1(self) \triangleq$ $\land\ pc[self] =$ "reload_insert1"
        $\land\ delta\_list' = [delta\_list$ EXCEPT $![idx[self]].delta = delta\_list[idx[self]].delta - d$
        $\land\ pc' = [pc$ EXCEPT $![self] =$ "reload_insert2"$]$
        $\land$ UNCHANGED $\langle wait\_set, running, waiting, stack,$
                      $name\_, name, idx, delta, runnable,$
                      $task, head, to\_be\_reloaded, head\_,$
                      $to\_be\_reloaded\_\rangle$

$reload\_insert2(self) \triangleq$ $\land\ pc[self] =$ "reload_insert2"
        $\land\ delta\_list' = InsertAt(delta\_list, idx[self], [delta \mapsto delta[self], name \mapsto name[s$
        $\land\ pc' = [pc$ EXCEPT $![self] =$ "EndReloadTimer"$]$
        $\land$ UNCHANGED $\langle wait\_set, running, waiting, stack,$
                      $name\_, name, idx, delta, runnable,$
                      $task, head, to\_be\_reloaded, head\_,$
                      $to\_be\_reloaded\_\rangle$
```

```
$reload\_insert\_end(self) \triangleq$ $\land\ pc[self] = \text{“reload\_insert\_end”}$
                               $\land\ delta\_list' = Append(delta\_list, [delta \mapsto delta[self], name \mapsto name[self]])$
                               $\land\ \text{TRUE}$
                               $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“EndReloadTimer”}]$
                               $\land\ \text{UNCHANGED}\ \langle wait\_set, running, waiting, stack,$
                                         $name\_, name, idx, delta, runnable,$
                                         $task, head, to\_be\_reloaded, head\_,$
                                         $to\_be\_reloaded\_\rangle$

$EndReloadTimer(self) \triangleq$ $\land\ pc[self] = \text{“EndReloadTimer”}$
                             $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = Head(stack[self]).pc]$
                             $\land\ idx' = [idx\ \text{EXCEPT}\ ![self] = Head(stack[self]).idx]$
                             $\land\ delta' = [delta\ \text{EXCEPT}\ ![self] = Head(stack[self]).delta]$
                             $\land\ name' = [name\ \text{EXCEPT}\ ![self] = Head(stack[self]).name]$
                             $\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = Tail(stack[self])]$
                             $\land\ \text{UNCHANGED}\ \langle delta\_list, wait\_set, running, waiting,$
                                       $name\_, runnable, task, head,$
                                       $to\_be\_reloaded, head\_, to\_be\_reloaded\_\rangle$

$reload\_timer(self) \triangleq$ $BeginReloadTimer(self) \lor reload\_insert1(self)$
                         $\lor\ reload\_insert2(self) \lor reload\_insert\_end(self)$
                         $\lor\ EndReloadTimer(self)$

$BeginNotify(self) \triangleq$ $\land\ pc[self] = \text{“BeginNotify”}$
                        $\land\ \text{IF}\ runnable[self] \neq \{\}$
                              $\text{THEN}$ $\land\ task' = [task\ \text{EXCEPT}\ ![self] = pick\_task(runnable[self])]$
                                   $\land\ runnable' = [runnable\ \text{EXCEPT}\ ![self] = runnable[self] \setminus \{task'[self]\}$
                                   $\land\ \land\ name\_' = [name\_\ \text{EXCEPT}\ ![self] = task'[self]]$
                                      $\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = \langle[procedure \mapsto \text{“callback”},$
                                                                        $pc \mapsto \text{“BeginNotify”},$
                                                                        $name\_ \mapsto name\_[self]]\rangle$
                                                                        $\circ\ stack[self]]$
                                   $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“BeginCallback”}]$
                              $\text{ELSE}$ $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{“EndNotify”}]$
                                   $\land\ \text{UNCHANGED}\ \langle stack, name\_, runnable, task\rangle$
                        $\land\ \text{UNCHANGED}\ \langle delta\_list, wait\_set, running, waiting,$
                                  $name, idx, delta, head, to\_be\_reloaded,$
                                  $head\_, to\_be\_reloaded\_\rangle$

$EndNotify(self) \triangleq$ $\land\ pc[self] = \text{“EndNotify”}$
                      $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = Head(stack[self]).pc]$
                      $\land\ task' = [task\ \text{EXCEPT}\ ![self] = Head(stack[self]).task]$
                      $\land\ runnable' = [runnable\ \text{EXCEPT}\ ![self] = Head(stack[self]).runnable]$
                      $\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = Tail(stack[self])]$
                      $\land\ \text{UNCHANGED}\ \langle delta\_list, wait\_set, running, waiting,$
                                $name\_, name, idx, delta, head,$
```

```
                                    to_be_reloaded, head_, to_be_reloaded_⟩

notify(self) ≜ BeginNotify(self) ∨ EndNotify(self)

BeginNotifyTimer(self) ≜ ∧ pc[self] = "BeginNotifyTimer"
                         ∧ IF delta_list ≠ ⟨⟩ ∧ delta_list[1].delta = 0
                              THEN ∧ head' = [head EXCEPT ![self] = Head(delta_list)]
                                   ∧ delta_list' = Tail(delta_list)
                                   ∧ ∧ name_' = [name_ EXCEPT ![self] = head'[self].name]
                                     ∧ stack'  = [stack EXCEPT ![self]  = ⟨[procedure ↦  "callback
                                                                           pc         ↦ "save_tim
                                                                           name_      ↦  name_[s
                                                                           ∘ stack[self]]
                                   ∧ pc' = [pc EXCEPT ![self] = "BeginCallback"]
                              ELSE ∧ pc' = [pc EXCEPT ![self] = "ReladTimer"]
                                   ∧ UNCHANGED ⟨delta_list, stack, name_,
                                                head⟩
                         ∧ UNCHANGED ⟨wait_set, running, waiting, name,
                                      idx, delta, runnable, task,
                                      to_be_reloaded, head_,
                                      to_be_reloaded_⟩

save_timer(self) ≜ ∧ pc[self] = "save_timer"
                   ∧ to_be_reloaded' = [to_be_reloaded EXCEPT ![self] = Append(to_be_reloaded[self],
                   ∧ pc' = [pc EXCEPT ![self] = "BeginNotifyTimer"]
                   ∧ UNCHANGED ⟨delta_list, wait_set, running, waiting,
                                stack, name_, name, idx, delta, runnable,
                                task, head, head_, to_be_reloaded_⟩

ReladTimer(self) ≜ ∧ pc[self] = "ReladTimer"
                   ∧ IF to_be_reloaded[self] ≠ ⟨⟩
                        THEN ∧ ∧ name' = [name EXCEPT ![self] = to_be_reloaded[self][1]]
                               ∧ stack' = [stack EXCEPT ![self] = ⟨[procedure ↦  "reload_timer",
                                                                    pc        ↦  "reload2",
                                                                    idx       ↦  idx[self],
                                                                    delta     ↦  delta[self],
                                                                    name      ↦  name[self]]⟩
                                                                    ∘ stack[self]]
                             ∧ idx' = [idx EXCEPT ![self] = defaultInitValue]
                             ∧ delta' = [delta EXCEPT ![self] = defaultInitValue]
                             ∧ pc' = [pc EXCEPT ![self] = "BeginReloadTimer"]
                        ELSE ∧ pc' = [pc EXCEPT ![self] = "EndNotifyTimer"]
                             ∧ UNCHANGED ⟨stack, name, idx, delta⟩
                   ∧ UNCHANGED ⟨delta_list, wait_set, running, waiting,
                                name_, runnable, task, head,
                                to_be_reloaded, head_, to_be_reloaded_⟩
```

```
$reload2(self) \triangleq$ $\land pc[self] = \text{"reload2"}$
                 $\land to\_be\_reloaded' = [to\_be\_reloaded \text{ EXCEPT } ![self] = Tail(to\_be\_reloaded[self])]$
                 $\land pc' = [pc \text{ EXCEPT } ![self] = \text{"ReladTimer"}]$
                 $\land \text{UNCHANGED } \langle delta\_list, wait\_set, running, waiting, stack,$
                                   $name\_, name, idx, delta, runnable, task, head,$
                                   $head\_, to\_be\_reloaded\_\rangle$

$EndNotifyTimer(self) \triangleq$ $\land pc[self] = \text{"EndNotifyTimer"}$
                        $\land pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$
                        $\land head' = [head \text{ EXCEPT } ![self] = Head(stack[self]).head]$
                        $\land to\_be\_reloaded' = [to\_be\_reloaded \text{ EXCEPT } ![self] = Head(stack[self]).to\_be\_r$
                        $\land stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])]$
                        $\land \text{UNCHANGED } \langle delta\_list, wait\_set, running, waiting,$
                                          $name\_, name, idx, delta, runnable,$
                                          $task, head\_, to\_be\_reloaded\_\rangle$

$notify\_timer(self) \triangleq$ $BeginNotifyTimer(self) \lor save\_timer(self)$
                        $\lor ReladTimer(self) \lor reload2(self)$
                        $\lor EndNotifyTimer(self)$

$BeginRclWait(self) \triangleq$ $\land pc[self] = \text{"BeginRclWait"}$
                      $\land \text{IF } delta\_list \neq \langle\rangle \land delta\_list[1].delta > 0 \land wait\_set = \{\}$
                            $\text{THEN}$ $\land \text{IF } delta\_list \neq \langle\rangle \land delta\_list[1].delta > 0$
                                          $\text{THEN}$ $\land delta\_list' = [delta\_list \text{ EXCEPT } ![1].delta = delta\_list[1$
                                          $\text{ELSE}$ $\land \text{TRUE}$
                                                 $\land \text{UNCHANGED } delta\_list$
                                   $\land pc' = [pc \text{ EXCEPT } ![self] = \text{"BeginRclWait"}]$
                            $\text{ELSE}$ $\land pc' = [pc \text{ EXCEPT } ![self] = \text{"EndRclWait"}]$
                                   $\land \text{UNCHANGED } delta\_list$
                      $\land \text{UNCHANGED } \langle wait\_set, running, waiting, stack, name\_,$
                                        $name, idx, delta, runnable, task, head,$
                                        $to\_be\_reloaded, head\_, to\_be\_reloaded\_\rangle$

$EndRclWait(self) \triangleq$ $\land pc[self] = \text{"EndRclWait"}$
                    $\land pc' = [pc \text{ EXCEPT } ![self] = Head(stack[self]).pc]$
                    $\land stack' = [stack \text{ EXCEPT } ![self] = Tail(stack[self])]$
                    $\land \text{UNCHANGED } \langle delta\_list, wait\_set, running, waiting,$
                                      $name\_, name, idx, delta, runnable, task,$
                                      $head, to\_be\_reloaded, head\_,$
                                      $to\_be\_reloaded\_\rangle$

$rcl\_wait(self) \triangleq$ $BeginRclWait(self) \lor EndRclWait(self)$

$BeginWaitTimer(self) \triangleq$ $\land pc[self] = \text{"BeginWaitTimer"}$
                        $\land stack' = [stack \text{ EXCEPT } ![self] = \langle[procedure \mapsto \text{"rcl\_wait"},$
                                                                 $pc \qquad\quad \mapsto \text{"EndWaitTimer"}]\rangle$
                                                                 $\circ\, stack[self]]$
                                        $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{"BeginRclWait"}]$
                                        $\land\ \text{UNCHANGED}\ \langle delta\_list, wait\_set, running, waiting,$
                                                     $name\_, name, idx, delta, runnable,$
                                                     $task, head, to\_be\_reloaded, head\_,$
                                                     $to\_be\_reloaded\_\rangle$

$EndWaitTimer(self) \triangleq$ $\land\ pc[self] = \text{"EndWaitTimer"}$
                           $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = Head(stack[self]).pc]$
                           $\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = Tail(stack[self])]$
                           $\land\ \text{UNCHANGED}\ \langle delta\_list, wait\_set, running, waiting,$
                                        $name\_, name, idx, delta, runnable, task,$
                                        $head, to\_be\_reloaded, head\_,$
                                        $to\_be\_reloaded\_\rangle$

$wait\_timer(self) \triangleq BeginWaitTimer(self) \lor EndWaitTimer(self)$

$BeginWait(self) \triangleq$ $\land\ pc[self] = \text{"BeginWait"}$
                        $\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = \langle[procedure \mapsto \text{"wait\_timer"},$
                                                                  $pc \mapsto \text{"NotifyTimer"}]\rangle$
                                                                  $\circ\ stack[self]]$
                        $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{"BeginWaitTimer"}]$
                        $\land\ \text{UNCHANGED}\ \langle delta\_list, wait\_set, running, waiting,$
                                     $name\_, name, idx, delta, runnable, task,$
                                     $head, to\_be\_reloaded, head\_,$
                                     $to\_be\_reloaded\_\rangle$

$NotifyTimer(self) \triangleq$ $\land\ pc[self] = \text{"NotifyTimer"}$
                          $\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = \langle[procedure \mapsto \text{"notify\_timer"},$
                                                                    $pc \mapsto \text{"Notify"},$
                                                                    $head \mapsto head[self],$
                                                                    $to\_be\_reloaded \mapsto to\_be\_reloaded[self]]\rangle$
                                                                    $\circ\ stack[self]]$
                          $\land\ head' = [head\ \text{EXCEPT}\ ![self] = defaultInitValue]$
                          $\land\ to\_be\_reloaded' = [to\_be\_reloaded\ \text{EXCEPT}\ ![self] = \langle\rangle]$
                          $\land\ pc' = [pc\ \text{EXCEPT}\ ![self] = \text{"BeginNotifyTimer"}]$
                          $\land\ \text{UNCHANGED}\ \langle delta\_list, wait\_set, running, waiting,$
                                       $name\_, name, idx, delta, runnable, task,$
                                       $head\_, to\_be\_reloaded\_\rangle$

$Notify(self) \triangleq$ $\land\ pc[self] = \text{"Notify"}$
                    $\land\ \text{LET}\ tmp\_wait\_set \triangleq wait\_set\text{IN}$
                         $\land\ wait\_set' = \{\}$
                         $\land\ \land\ runnable' = [runnable\ \text{EXCEPT}\ ![self] = tmp\_wait\_set]$
                           $\land\ stack' = [stack\ \text{EXCEPT}\ ![self] = \langle[procedure \mapsto \text{"notify"},$
                                                                     $pc \mapsto \text{"EndWait"},$
                                                                     $task \mapsto task[self],$
```

$$\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{runnable} \mapsto \mathit{runnable}[\mathit{self}]]\rangle \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \circ\, \mathit{stack}[\mathit{self}]] \\
\qquad\qquad \land\ \mathit{task}' = [\mathit{task}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{defaultInitValue}] \\
\qquad\qquad \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \text{“BeginNotify”}] \\
\qquad \land\ \text{UNCHANGED}\ \langle \mathit{delta\_list}, \mathit{running}, \mathit{waiting}, \mathit{name\_}, \mathit{name}, \mathit{idx}, \\
\qquad\qquad\qquad \mathit{delta}, \mathit{head}, \mathit{to\_be\_reloaded}, \mathit{head\_}, \\
\qquad\qquad\qquad \mathit{to\_be\_reloaded\_}\rangle
\end{array}$$

$$\begin{array}{ll}
\mathit{EndWait}(\mathit{self}) \triangleq & \land\ \mathit{pc}[\mathit{self}] = \text{“EndWait”} \\
& \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Head}(\mathit{stack}[\mathit{self}]).\mathit{pc}] \\
& \land\ \mathit{stack}' = [\mathit{stack}\ \text{EXCEPT}\ ![\mathit{self}] = \mathit{Tail}(\mathit{stack}[\mathit{self}])] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{delta\_list}, \mathit{wait\_set}, \mathit{running}, \mathit{waiting}, \mathit{name\_}, \\
& \qquad \mathit{name}, \mathit{idx}, \mathit{delta}, \mathit{runnable}, \mathit{task}, \mathit{head}, \\
& \qquad \mathit{to\_be\_reloaded}, \mathit{head\_}, \mathit{to\_be\_reloaded\_}\rangle
\end{array}$$

$$\begin{array}{ll}
\mathit{wait}(\mathit{self}) \triangleq & \mathit{BeginWait}(\mathit{self}) \lor \mathit{NotifyTimer}(\mathit{self}) \lor \mathit{Notify}(\mathit{self}) \\
& \quad \lor\ \mathit{EndWait}(\mathit{self})
\end{array}$$

$$\begin{array}{ll}
\mathit{fire\_event}(\mathit{self}) \triangleq & \land\ \mathit{pc}[\mathit{self}] = \text{“fire\_event”} \\
& \land\ \mathit{wait\_set}' = (\mathit{wait\_set} \cup \{\mathit{self}\}) \\
& \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\mathit{self}] = \text{“fire\_event”}] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{delta\_list}, \mathit{running}, \mathit{waiting}, \mathit{stack}, \mathit{name\_}, \\
& \qquad \mathit{name}, \mathit{idx}, \mathit{delta}, \mathit{runnable}, \mathit{task}, \mathit{head}, \\
& \qquad \mathit{to\_be\_reloaded}, \mathit{head\_}, \mathit{to\_be\_reloaded\_}\rangle
\end{array}$$

$$\mathit{trigger\_event}(\mathit{self}) \triangleq \mathit{fire\_event}(\mathit{self})$$

$$\begin{array}{ll}
\mathit{BeginExecutor} \triangleq & \land\ \mathit{pc}[\text{“executor”}] = \text{“BeginExecutor”} \\
& \land\ \mathit{stack}' = [\mathit{stack}\ \text{EXCEPT}\ ![\text{“executor”}] = \langle[\mathit{procedure} \mapsto \text{“wait”}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{pc} \mapsto \text{“BeginExecutor”}]\rangle \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \circ\, \mathit{stack}[\text{“executor”}]] \\
& \land\ \mathit{pc}' = [\mathit{pc}\ \text{EXCEPT}\ ![\text{“executor”}] = \text{“BeginWait”}] \\
& \land\ \text{UNCHANGED}\ \langle \mathit{delta\_list}, \mathit{wait\_set}, \mathit{running}, \mathit{waiting}, \mathit{name\_}, \\
& \qquad \mathit{name}, \mathit{idx}, \mathit{delta}, \mathit{runnable}, \mathit{task}, \mathit{head}, \\
& \qquad \mathit{to\_be\_reloaded}, \mathit{head\_}, \mathit{to\_be\_reloaded\_}\rangle
\end{array}$$

$$\mathit{executor} \triangleq \mathit{BeginExecutor}$$

$$\begin{array}{ll}
\mathit{Next} \triangleq & \mathit{executor} \\
& \quad \lor\ (\exists\, \mathit{self} \in \mathit{ProcSet} : \ \lor\ \mathit{callback}(\mathit{self}) \lor \mathit{reload\_timer}(\mathit{self}) \\
& \qquad\qquad\qquad\qquad\qquad \lor\ \mathit{notify}(\mathit{self}) \lor \mathit{notify\_timer}(\mathit{self}) \\
& \qquad\qquad\qquad\qquad\qquad \lor\ \mathit{rcl\_wait}(\mathit{self}) \lor \mathit{wait\_timer}(\mathit{self}) \\
& \qquad\qquad\qquad\qquad\qquad \lor\ \mathit{wait}(\mathit{self})) \\
& \quad \lor\ (\exists\, \mathit{self} \in \mathit{Tasks} : \mathit{trigger\_event}(\mathit{self}))
\end{array}$$

$$\begin{array}{ll}
\mathit{Spec} \triangleq & \land\ \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}} \\
& \land\ \forall\, \mathit{self} \in \mathit{Tasks} : \mathrm{WF}_{\mathit{vars}}(\mathit{trigger\_event}(\mathit{self}))
\end{array}$$

```
$\land$  $\land \mathrm{SF}_{vars}(executor)$
   $\land \mathrm{SF}_{vars}(wait(\text{“executor”}))$
   $\land \mathrm{SF}_{vars}(callback(\text{“executor”}))$
   $\land \mathrm{SF}_{vars}(reload\_timer(\text{“executor”}))$
   $\land \mathrm{SF}_{vars}(notify(\text{“executor”}))$
   $\land \mathrm{SF}_{vars}(notify\_timer(\text{“executor”}))$
   $\land \mathrm{SF}_{vars}(rcl\_wait(\text{“executor”}))$
   $\land \mathrm{SF}_{vars}(wait\_timer(\text{“executor”}))$

END TRANSLATION
```